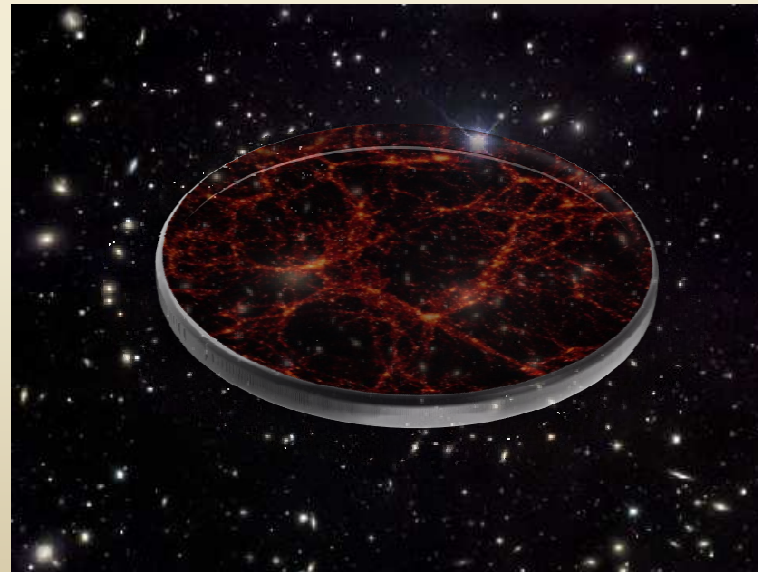




Review on Clustering in Data Mining

Massimo Brescia



Clustering at large

Clustering is a division of data into groups of similar objects. Representing the data by fewer clusters necessarily loses certain fine details (data compression), but achieves simplification.

From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system could represent a data concept in the KDD (Knowledge Discovery in Databases).

From a practical perspective clustering plays an outstanding role in data mining applications such as scientific data exploration, information retrieval and text mining, spatial database applications, Web analysis, CRM, marketing, medical diagnostics, computational biology, and many others.

Data mining in Astrophysics adds to clustering the complications of very large datasets with very many attributes of different types (high dimensionality). This imposes unique computational requirements on relevant clustering algorithms.



Notation Legend

To fix the context and to clarify prolific terminology, we consider a dataset X consisting of data points (or synonymously, objects, instances, cases, patterns, tuples, transactions):

$$x_i = (x_{i1}, \dots, x_{id}) \in A$$

A parameter space

$$i = 1 : N$$

$x_{il} \in A_l$ attribute (feature)

Such point-by-attribute data format conceptually corresponds to a $N \times d$ matrix.

The ultimate goal of clustering is to assign points to a finite system of k subsets, clusters. Usually subsets do not intersect (this assumption is sometimes violated), and their union is equal to a full dataset with possible exception of outliers:

$$X = C_1 \cdots C_k \cdots C_{outliers}, C_{j1} \cdots C_{j2} = \Phi$$

This notation choice could not match the one included in all examined documents. But it is required in order to explain further following considerations.



Clustering Requirements in Astrophysical Data Mining

What are the properties of clustering algorithms we are concerned with in data mining?

These properties include:

- Type of attributes that the algorithm can handle;
- Scalability to large datasets;
- Ability to work with high dimensional data (multi-D parameter space, multi-wavelength, multi-epoch etc...);
- Ability to find clusters of irregular shape;
- Handling outliers;
- Time complexity (when there is no confusion, we use the term complexity);
- Data order dependency;
- Labeling or assignment (hard or strict vs. soft or fuzzy);
- Reliance on a priori knowledge and user defined parameters;
- Interpretability of results;

We have to try to keep these issues in mind, realistically. The above list is in no way exhaustive. For example, we must deal also with implementation properties, such as ability to work in pre-defined memory buffer, ability to restart and ability to provide an intermediate solution.



Taxonomy of Clustering Algorithms

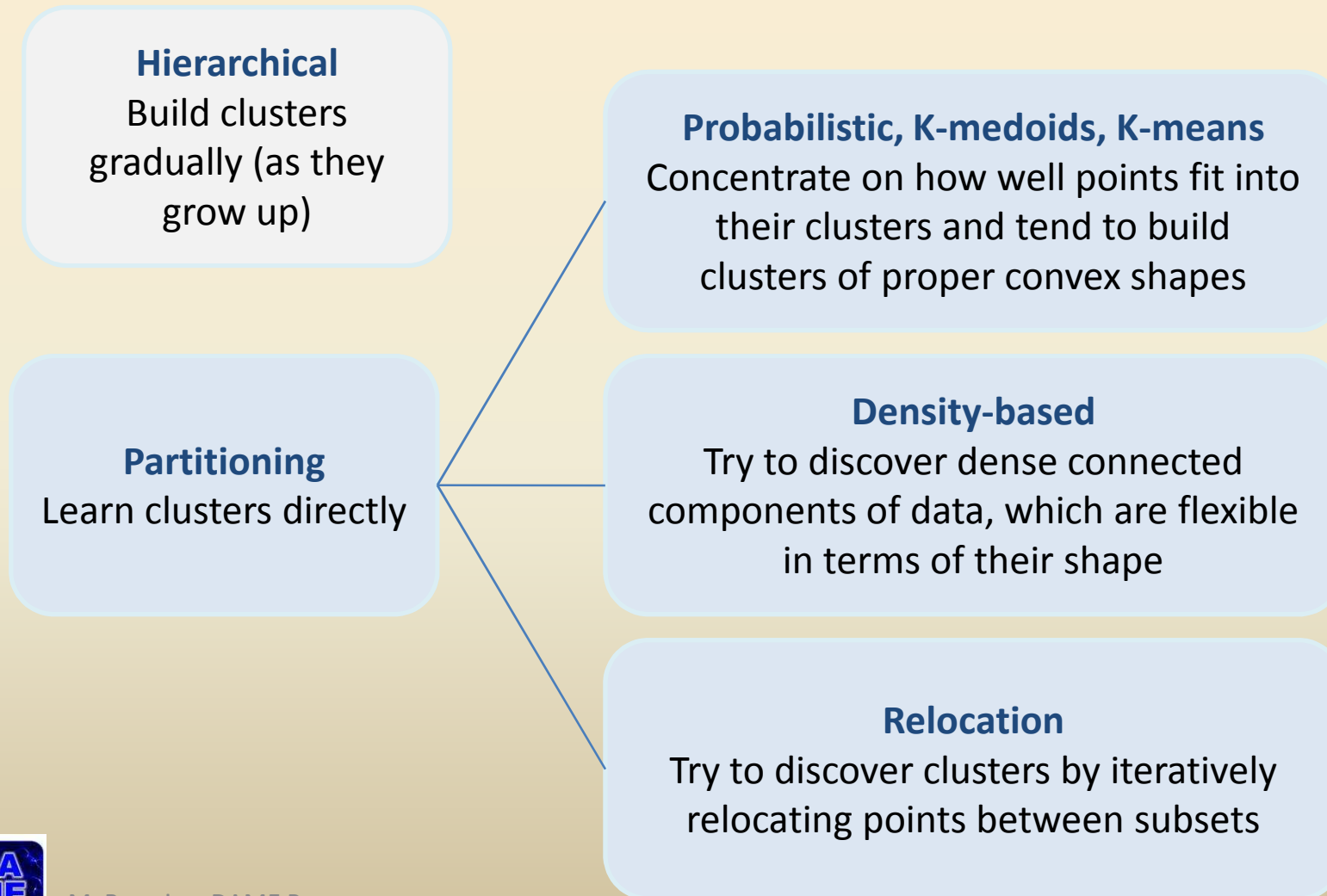
- Hierarchical Methods
 - Agglomerative Algorithms
 - Divisive Algorithms
- Partitioning Methods
 - Relocation Algorithms
 - Probabilistic Clustering
 - K-medoids Methods
 - K-means Methods
 - Density-Based Algorithms
 - Density-Based Connectivity Clustering
 - Density Functions Clustering
- Grid-Based Methods
- Methods Based on Co-Occurrence of Categorical Data
- Clustering Algorithms Used in Machine Learning
 - Artificial Neural Networks, Fuzzy and Hybrid Systems
 - Evolutionary Methods
 - Constraint-Based Clustering
- Algorithms For High Dimensional and Scalable Data
 - Subspace Clustering
 - Projection Techniques
 - Co-Clustering Techniques

Let's investigate them...



ABSTRACT - Hierarchical and Partitioning Clustering

While **hierarchical** algorithms build clusters gradually (as crystals are grown), **partitioning** algorithms learn clusters directly. In doing so, they either try to discover clusters by iteratively relocating points between subsets, or try to identify clusters as areas highly populated with data.



ABSTRACT - Density-based Clustering

Density-based

Try to discover dense connected components of data, which are flexible in terms of their shape

Density-based methods are basically less sensitive to outliers and can discover clusters of irregular shapes. They usually work with low-dimensional data of numerical attributes, known as spatial data. Spatial objects could include not only points, but also extended objects.

Density-Connectivity

Density-Functions



ABSTRACT - Grid-based Clustering

Some algorithms work with data indirectly by constructing summaries of data over the attribute space subsets. They perform space segmentation and then aggregate appropriate segments. They frequently use hierarchical agglomeration as one phase of processing. Grid-based methods are fast and handle outliers well. Grid-based methodology is also used as an intermediate step in many other algorithms.

Grid-based

Build clusters in
multi-step, by
creating a grid of the
space



ABSTRACT - Co-occurrence of Categorical Data Clustering

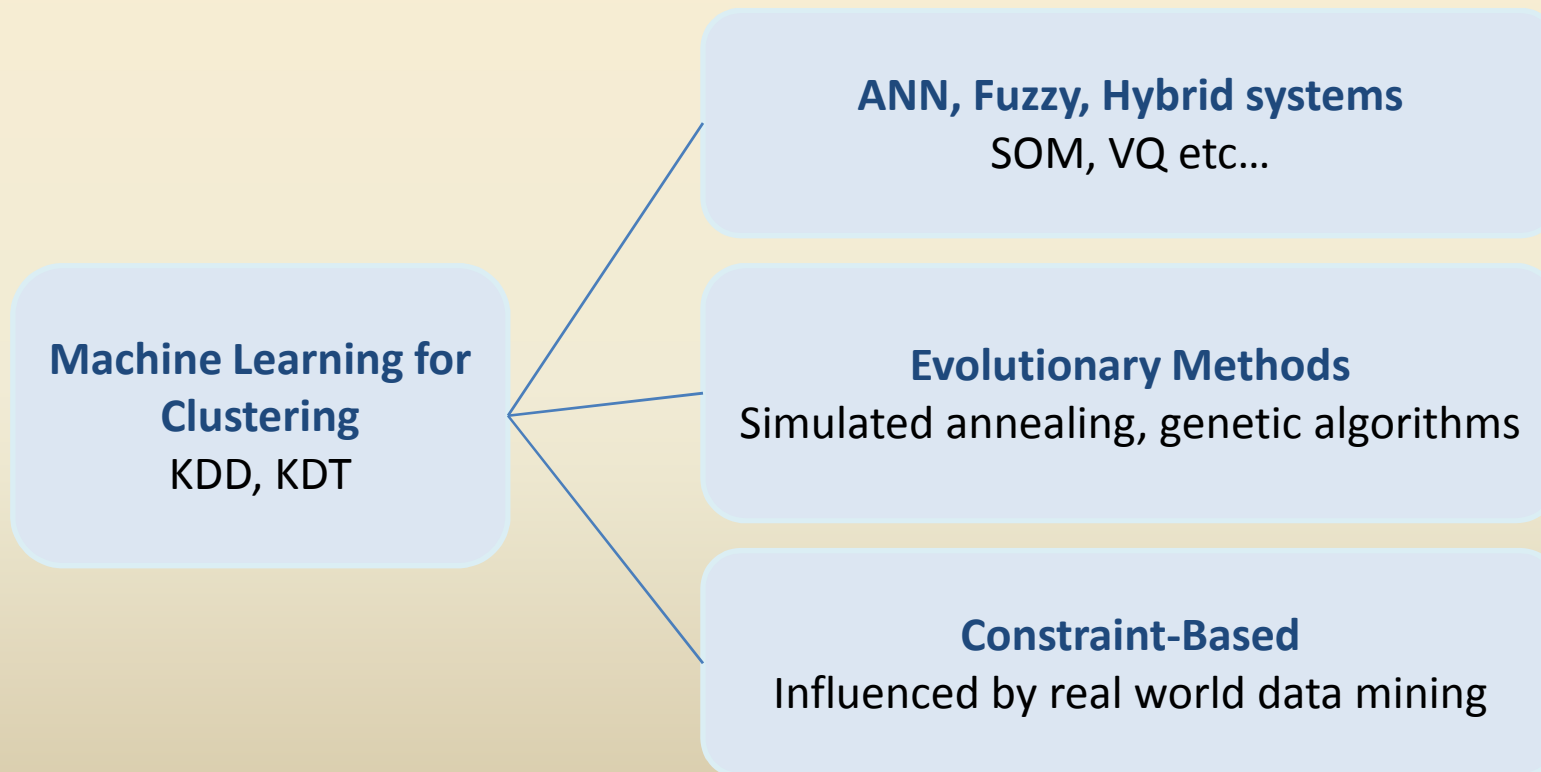
Categorical data is intimately connected with transactional databases. The concept of a similarity alone is not sufficient for clustering such data. The idea of categorical data co-occurrence comes to rescue. The situation gets even more aggravated with the growth of the number of items involved. To help with this problem an effort is shifted from data clustering to pre-clustering of items or categorical attribute values. Development based on hyper-graph partitioning exemplify this approach.

**Co-Occurrence of
Categorical Data**
Similarity and
categorical search in
transactional DB



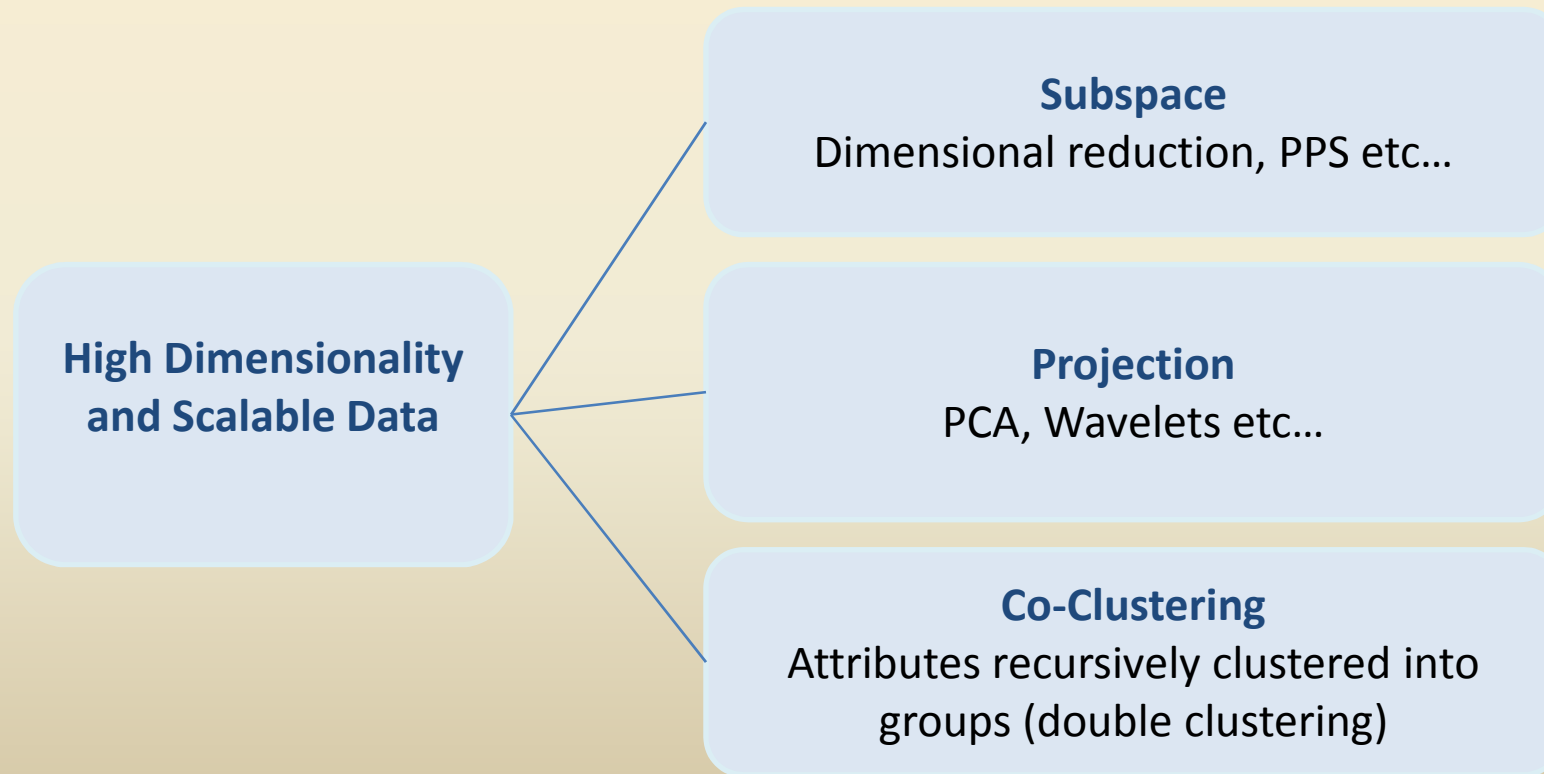
ABSTRACT - Machine Learning for Clustering

Many other clustering techniques are developed, primarily in machine learning, that either have theoretical significance, are used traditionally outside the data mining community, or do not fit in previously outlined categories. They are basically specialized for KDD (Knowledge Discovery in Databases) and KDT (Knowledge Discovery in Text). There are relationship to unsupervised learning (PPS, SOM), evolutionary methods (simulated annealing, genetic algorithms (GA)). There is however the emerging field of constraint-based clustering that is influenced by requirements of real world data mining applications.



ABSTRACT - High Dimensionality and Scalable Data Clustering

Data Mining primarily works with large databases (in this case we speak about scalability problems). But another trait of real-life data is its high dimensionality. Corresponding developments are surveyed in the Clustering High Dimensional Data. The trouble comes from a decrease in metric separation when the dimension grows. One approach to dimensionality reduction uses attributes transformations (PCA, wavelets). Another way to address the problem is through subspace clustering. Still another approach clusters attributes in groups and uses their derived proxies to cluster objects. This double clustering is known as co-clustering.



Taxonomy of Clustering Algorithms

- **Hierarchical Methods**
 - **Agglomerative Algorithms**
 - **Divisive Algorithms**
- Partitioning Methods
 - Relocation Algorithms
 - Probabilistic Clustering
 - K-medoids Methods
 - K-means Methods
 - Density-Based Algorithms
 - Density-Based Connectivity Clustering
 - Density Functions Clustering
- Grid-Based Methods
- Methods Based on Co-Occurrence of Categorical Data
- Clustering Algorithms Used in Machine Learning
 - Artificial Neural Networks, Fuzzy and Hybrid Systems
 - Evolutionary Methods
 - Constraint-Based Clustering
- Algorithms For High Dimensional and Scalable Data
 - Subspace Clustering
 - Projection Techniques
 - Co-Clustering Techniques



Hierarchical Clustering – General Issues

Hierarchical clustering builds a cluster hierarchy or, in other words, a tree of clusters, also known as a *dendrogram*. Every cluster node contains child clusters; sibling clusters partition the points covered by their common parent. Such an approach allows exploring data on different levels of granularity.

Hierarchical clustering methods are categorized into **agglomerative** (bottom-up) and **divisive** (top-down).

An **agglomerative** clustering starts with one-point (singleton) clusters and recursively merges two or more most appropriate clusters.

A **divisive** clustering starts with one cluster of all data points and recursively splits the most appropriate cluster. The process continues until a stopping criterion (frequently, the requested number k of clusters) is achieved.

Advantages:

- Embedded flexibility regarding the level of granularity;
- Ease of handling of any forms of similarity or distance. Consequently, applicability to any attribute types;

Disadvantages:

- Vagueness of termination criteria;
- The fact that most hierarchical algorithms do not revisit once constructed (intermediate) clusters with the purpose of their improvement;



Hierarchical Clustering - Considerations

In hierarchical clustering our regular point-by-attribute data representation is sometimes of secondary importance. Instead, hierarchical clustering frequently deals with the $N \times N$ matrix of distances (dissimilarities) or similarities between training points. It is sometimes called connectivity matrix.

In case of huge datasets the requirement of keeping such a large matrix in memory is unrealistic. To relax this limitation different devices are used to introduce into the connectivity matrix some sparsity. This can be done by omitting entries smaller than a certain threshold, by using only a certain subset of data representatives, or by keeping with each point only a certain number of its nearest neighbors.

With the (sparsified) connectivity matrix we can associate the connectivity graph $G = (X, E)$, whose vertices X are data points, and edges E and their weights are pairs of points and the corresponding positive matrix entries. This establishes a connection between hierarchical clustering and graph partitioning.



Hierarchical Clustering – Linkage Metrics

Hierarchical clustering initializes a cluster system as a set of singleton clusters (agglomerative case) or a single cluster of all points (divisive case) and proceeds iteratively with merging or splitting of the most appropriate cluster(s) until the stopping criterion is achieved. The appropriateness of a cluster for merging/splitting depends on the (dis)similarity of cluster elements. This reflects a general presumption that clusters consist of similar points. An important example of dissimilarity between two points is the distance between them.

To merge or split subsets of points rather than individual points, the distance between individual points has to be generalized to the distance between subsets. Such derived proximity measure is called a linkage metric. The type of the linkage metric used significantly affects hierarchical algorithms, since it reflects the particular concept of closeness and connectivity.

A specific operation such as minimum (single link), average (average link), or maximum (complete link) is applied to pair-wise dissimilarity measures:

$$d(C_1, C_2) = \text{operation}\{d(x, y) \mid x \in C_1, y \in C_2\}$$

Linkage metrics-based hierarchical clustering suffers from time complexity.



Hierarchical Clustering - Arbitrary cluster shapes - 1

Linkage metrics based on Euclidean distance for hierarchical clustering of spatial data naturally predispose to clusters of proper convex shapes. Meanwhile, visual scanning of spatial images frequently attests clusters with curvy appearance.

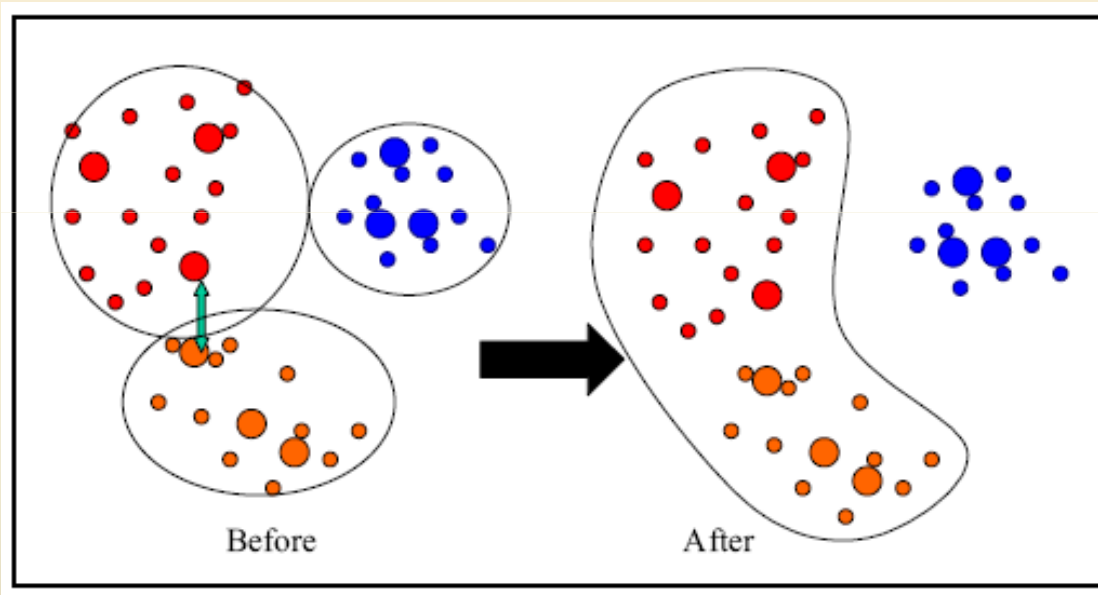
Guha et al. [1998] introduced the hierarchical agglomerative clustering algorithm CURE (Clustering Using REpresentatives). This algorithm has a number of novel features of general significance. It takes special care with outliers and with label assignment stage. It also uses two devices to achieve scalability [[See references in sdm01_07, wi07-bolelli-kmeans](#)].

The first one is data sampling. The second device is data partitioning in p partitions, so that fine granularity clusters are constructed in partitions first. A major feature of CURE is that it represents a cluster by a fixed number c of points scattered around it. The distance between two clusters used in the agglomerative process is equal to the minimum of distances between two scattered representatives.



Hierarchical Clustering - Arbitrary cluster shapes - 2

Therefore, CURE takes a middle-ground approach between the graph (all-points) methods and the geometric (one centroid) methods. Single and average link closeness is replaced by representatives. Selecting representatives scattered around a cluster makes it possible to cover non-spherical shapes. As before, agglomeration continues until requested number k of clusters is achieved. CURE employs one additional device: originally selected scattered points are shrunk to the geometric centroid of the cluster by user-specified factor α .



Three clusters, each with three representatives, are shown before and after the merge and shrinkage.

Two closest representatives are connected by arrow.

In the CURE algorithm more exact bounds depend on input parameters: shrink factor, number of representative points c , number of partitions, and sample size.



Hierarchical Clustering – Other techniques

Just for completeness purpose we mention Binary Divisive Partitioning that, in linguistics, information retrieval, and document clustering applications binary taxonomies are very useful. Linear algebra methods, based on Singular Value Decomposition (SVD) are used for this purpose in collaborative filtering and information retrieval. SVD application to hierarchical divisive clustering of document collections resulted in the PDDP (Principal Direction Divisive Partitioning) algorithm.

The popular hierarchical clustering algorithm for categorical data COBWEB [[see reference in parsons](#)] has two very important qualities. First, it utilizes incremental learning. Instead of following divisive or agglomerative approaches, it dynamically builds a dendrogram by processing one data point at a time. Second, COBWEB belongs to conceptual or model based learning. This means that each cluster is considered as a model that can be described intrinsically, rather than as a collection of points assigned to it.

COBWEB dendrogram is called a classification tree. Each tree node C , a cluster, is associated with the conditional probabilities for categorical attribute-values pairs:

$$\Pr(x_l = v_{lp} | C), l = 1 : d, p = 1 : |A_l|$$

This easily can be recognized as a specific Bayes classifier. During the classification tree construction, every new point is descended along the tree and the tree is potentially updated.



Taxonomy of Clustering Algorithms

- Hierarchical Methods
 - Agglomerative Algorithms
 - Divisive Algorithms
- **Partitioning Methods**
 - **Relocation Algorithms**
 - **Probabilistic Clustering**
 - **K-medoids Methods**
 - **K-means Methods**
 - **Density-Based Algorithms**
 - **Density-Based Connectivity Clustering**
 - **Density Functions Clustering**
- Grid-Based Methods
- Methods Based on Co-Occurrence of Categorical Data
- Clustering Algorithms Used in Machine Learning
 - Artificial Neural Networks, Fuzzy and Hybrid Systems
 - Evolutionary Methods
 - Constraint-Based Clustering
- Algorithms For High Dimensional and Scalable Data
 - Subspace Clustering
 - Projection Techniques
 - Co-Clustering Techniques



Partitioning Clustering

Partitioning algorithms divide data into several subsets. Because checking all possible subset systems is computationally infeasible, certain greedy heuristics are used in the form of iterative optimization. Specifically, this means different **relocation** schemes that iteratively reassign points between the k clusters. Unlike traditional hierarchical methods, in which clusters are not revisited after being constructed, relocation algorithms gradually improve clusters. With appropriate data, this results in high quality clusters.

One approach to data partitioning is to take a conceptual point of view that identifies the cluster with a certain model whose unknown parameters have to be found. More specifically, **probabilistic** models assume that the data comes from a mixture of several populations whose distributions and priors we want to find. One clear advantage of probabilistic methods is the interpretability of the constructed clusters. Having concise cluster representation also allows inexpensive computation of intra-clusters measures of fit that give rise to a global objective function (i.e. log-likelihood).

Another approach starts with the definition of objective function depending on a partition. As we have seen (Linkage Metrics), pair-wise distances or similarities can be used to compute measures of intra-cluster relations. In iterative improvements such pair-wise computations would be too expensive. Using unique cluster representatives resolves the problem: now computation of objective function becomes linear in N (and in a number of clusters). Depending on how representatives are constructed, iterative optimization partitioning algorithms are subdivided into **k-medoids** and **k-means** methods.



Partitioning Clustering – Probabilistic - 1

In the probabilistic approach, data is considered to be a sample independently drawn from a mixture model of several probability distributions.

The main assumption is that data points are generated by, first, randomly picking a model j with probability τ_j , $j = 1:K$, and, second, by drawing a point x from a corresponding distribution. The area around the mean of each distribution constitutes a natural cluster. So we associate the cluster with the corresponding distribution's parameters such as mean, variance, etc. Each data point carries not only its (observable) attributes, but also a (hidden) cluster ID (class in pattern recognition).

Each point x is assumed to belong to one and only one cluster, and we can estimate the probabilities of the assignment $\Pr(C_j | x)$ to j^{th} model. The overall **likelihood** of the training data is its probability to be drawn from a given mixture model:

$$L(X|C) = \prod_{\substack{i=1:N \\ j=1:k}} \tau_j \Pr(x_i | C_j)$$

Log-likelihood $\log(L(X|C))$ serves as an objective function, which gives rise to the Expectation-Maximization (EM) method [[see references in Murtagh-Clustering_in_massive_data_sets, tr439](#)]



Partitioning Clustering – Probabilistic - 2

EM is a two-step iterative optimization. Step (E) estimates probabilities $\Pr(x|C_j)$, which is equivalent to a soft (fuzzy) reassignment. Step (M) finds an approximation to a mixture model, given current soft assignments. This boils down to finding mixture model parameters that maximize log-likelihood. The process continues until log-likelihood convergence is achieved.

It was suggested acceleration of EM method based on a special data index, KD-tree [see references in fulltext, [Murtagh-Clustering_in_massive_data_sets, tr439](#)]. Data is divided at each node into two descendents by splitting the widest attribute at the center of its range. Each node stores sufficient statistics (including covariance matrix). Approximate computing over a pruned tree accelerates EM iterations.

Probabilistic clustering has some important features:

- It can be modified to handle recodes of complex structure;
- It can be stopped and resumed with consecutive batches of data, since clusters have representation totally different from sets of points;
- At any stage of iterative process the intermediate mixture model can be used to assign cases (on-line property);
- It results in easily interpretable cluster system;



Partitioning Clustering – Probabilistic - 3

From a data mining perspective, excessive parameter set causes overfitting, while from a probabilistic perspective, number of parameters can be addressed within the Bayesian framework.

An important property of probabilistic clustering is that mixture model can be naturally generalized to clustering heterogeneous data. This is important in practice, where an individual (data object) has multivariate static data (demographics) in combination with variable length dynamic data (customer profile). The dynamic data can consist of finite sequences with a transition matrix dependent on a cluster. This framework also covers data objects consisting of several sequences, where number n of sequences per is subject to geometric distribution.



Partitioning Clustering – K-Medoids

Medoids are representative objects of a data set or a cluster with a data set whose average dissimilarity to all the objects in the cluster is minimal. Medoids are similar in concept to means or centroids, but medoids are always members of the data set. Medoids are most commonly used on data when a mean or centroid cannot be defined such as 3-D trajectories or in the gene expression context.

In k-medoids methods a cluster is represented by one of its points. We have already mentioned that this is an easy solution since it covers any attribute types and that medoids have embedded resistance against outliers since peripheral cluster points do not affect them. When medoids are selected, clusters are defined as subsets of points close to respective medoids, and the objective function is defined as the averaged distance or another dissimilarity measure between a point and its medoid.

Two early versions of k-medoid methods are the algorithm PAM (Partitioning Around Medoids) and the algorithm CLARA (Clustering LARge Applications) [see reference [Kaufman & Rousseeuw 1990 in document publishbiocomp](#)]. PAM is iterative optimization that combines relocation of points between perspective clusters with re-nominating the points as potential medoids. The guiding principle for the process is the effect on an objective function, which, obviously, is a costly strategy. CLARA uses several (five) samples, each with $40+2k$ points, which are each subjected to PAM. The whole dataset is assigned to resulting medoids, the objective function is computed, and the best system of medoids is retained.



Partitioning Clustering – K-Means - 1

The k-means algorithm [Hartigan & Wong 1979] is by far the most popular clustering tool used in scientific and industrial applications. The name comes from representing each of k clusters C_j by the mean (or weighted average) c_j of its points, the so-called centroid. While this obviously does not work well with categorical attributes, it has the good geometric and statistical sense for numerical attributes. The sum of discrepancies between a point and its centroid expressed through appropriate distance is used as the objective function. For example, the L_2 -norm based objective function, the sum of the squares of errors between the points and the corresponding centroids, is equal to the total intra-cluster variance:

$$E(C) = \sum_{j=1:k, x_i \in C_j} \|x_i - c_j\|^2$$

The sum of the squares of errors can be rationalized as (a negative of) log-likelihood for normally distributed mixture model and is widely used in statistics (SSE). Therefore, k-means algorithm can be derived from general probabilistic framework

Note that only means are estimated. A simple modification would normalize individual errors by cluster radii (cluster standard deviation), which makes a lot of sense when clusters have different dispersions. An objective function based on L_2 -norm has many unique algebraic properties.



Partitioning Clustering – K-Means - 2

Two versions of k-means iterative optimization are known.

The first version is similar to EM algorithm and consists of two-step major iterations that (1) reassign all the points to their nearest centroids, and (2) recompute centroids of newly assembled groups. Iterations continue until a stopping criterion is achieved (for example, no reassignments happen). This version is known as Forgy algorithm [[see reference in document Murtagh-Clustering_in_massive_data_sets](#)] and has many advantages:

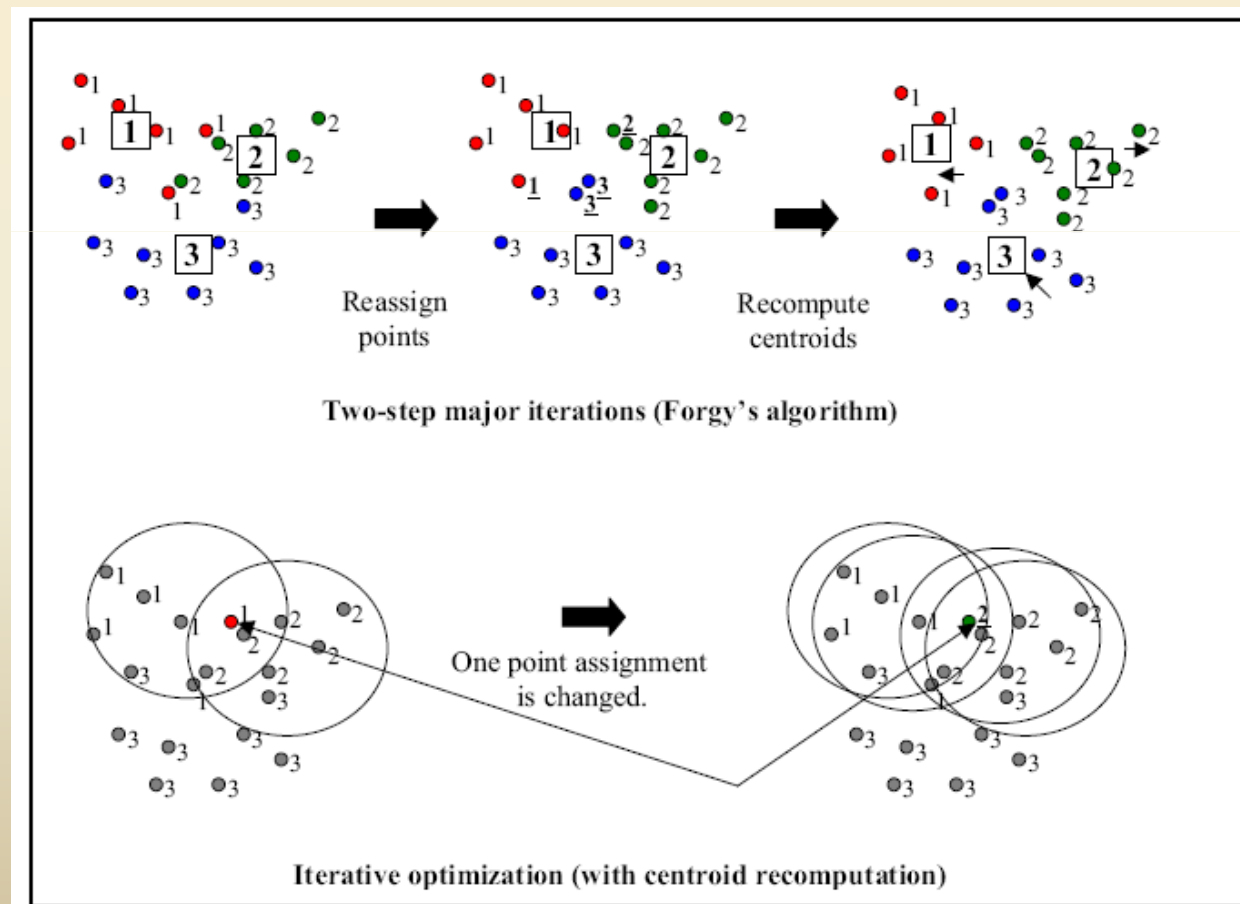
- It easily works with any L_p -norm;
- It allows straightforward parallelization;
- It is insensitive with respect to data ordering;

The second (classic in iterative optimization) version of k-means iterative optimization reassigns points based on more detailed analysis of effects on the objective function caused by moving a point from its current cluster to a potentially new one. If a move has a positive effect, the point is relocated and the two centroids are recomputed. **It is not clear that this version is computationally feasible, because the outlined analysis requires an inner loop over all member points of involved clusters affected by centroids shifts.**



Partitioning Clustering – K-Means - 3

There are experimental results demonstrating that compared with Forgy algorithm, the second (classic) version frequently yields better results. In particular it was noticed that a Forgy spherical k-means (using cosine similarity instead of Euclidean distance) has a tendency to get stuck when applied to document collections. They noticed that a version reassigning points and immediately recomputing centroids works much better. Figure illustrates both implementations.



Partitioning Clustering – K-Means - 4

The wide popularity of k-means algorithm is well deserved. It is simple, straightforward, and is based on the firm foundation of analysis of variances. The k-means algorithm also suffers from all the usual suspects:

- The result strongly depends on the initial guess of centroids (or assignments)
- Computed local optimum is known to be a far from the global one
- It is not obvious what is a good k to use
- The process is sensitive with respect to outliers
- The algorithm lacks scalability
- Only numerical attributes are covered
- Resulting clusters can be unbalanced (in Forgy version, even empty)

A simple way to mitigate the affects of clusters initialization was suggested starting from the consideration that k-means is performed on several small samples of data with a random initial guess. Each of these constructed systems is then used as a potential initialization for a union of all the samples. Centroids of the best system constructed this way are suggested as an intelligent initial guesses to ignite the k-means algorithm on the full data.



Partitioning Clustering – K-Means - 5

In the document [wi07-bolelli-kmeans](#) is presented KSVMMeans, a clustering algorithm for multi-type interrelated datasets that integrates the well known K-Means clustering with the highly popular Support Vector Machines. The experimental results on authorship analysis of two real world web-based datasets show that KSVMMeans can successfully discover topical clusters of documents and achieve better clustering solutions than homogeneous data clustering.

No initialization actually guarantees global minimum for k-means.

As is common to any combinatorial optimization, a logical attempt to cure this problem is to use simulated annealing [Brown & Huntley 1991] or Genetic Algorithms [Babu & Murty 1993]. It was also suggested another way to rectify optimization process by soft assignment of points to different clusters with appropriate weights (as EM does), rather than by moving them decisively from one cluster to another.



Partitioning Clustering - Density-based - 1

An open set in the Euclidean space can be divided into a set of its connected components. The implementation of this idea for partitioning of a finite set of points requires concepts of density, connectivity and boundary.

They are closely related to a point's nearest neighbors. A cluster, defined as a connected dense component, grows in any direction that density leads. Therefore, density-based algorithms are capable of discovering clusters of arbitrary shapes. Also this provides a natural protection against outliers. Figure below illustrates some cluster shapes that present a problem for partitioning relocation clustering (e.g., k-means), but are handled properly by density-based algorithms.



From a very general data description point of view, a single dense cluster consisting of two adjacent areas with significantly different densities (both higher than a threshold) is not very informative.

There are two major approaches for density-based methods.

The first approach pins density to a training data point and is known as **Density-Based Connectivity**. Representative algorithms include DBSCAN [see reference in documents p547-houle, p547-houle2, v1-27]. The second approach pins density to a point in the attribute space and is explained in the sub-section **Density Functions** It includes the algorithm DENCLUE (old).



Partitioning Clustering - Density-based - 2

Concerning **Density-Based Connectivity** methods, the algorithm DBSCAN (Density Based Spatial Clustering of Applications with Noise) targeting low-dimensional spatial data is the major representative in this category.

Two input parameters ε and *MinPts* are used to define:

- 1) An ε -neighborhood of the point x ,
$$N_\varepsilon(x) = \{y \in X \mid d(x, y) \leq \varepsilon\}$$
- 2) A core object (a point with a neighborhood consisting of more than *MinPts* points);
- 3) A concept of a point y density-reachable from a core object x (a finite sequence of core objects between x and y exists such that each next belongs to an ε -neighborhood of its predecessor);
- 4) A density-connectivity of two points x, y (they should be density-reachable from a common core object);



Partitioning Clustering - Density-based - 3

So defined density-connectivity is a symmetric relation and all the points reachable from core objects can be factorized into maximal connected components serving as clusters. The points that are not connected to any core point are declared to be outliers (they are not covered by any cluster). The non-core points inside a cluster represent its boundary. Finally, core objects are internal points. Processing is independent of data ordering. So far, nothing requires any limitations on the dimension or attribute types.

With regard to these two parameters and MinPts, there is no straightforward way to fit them to data. Moreover, different parts of data could require different parameters.

A more mathematically sound approach is to consider a random variable equal to the distance from a point to its nearest neighbor, and to learn its probability distribution. Instead of relying on user-defined parameters, a possible conjuncture is that each cluster has its own typical distance-to-nearest-neighbor scale. The goal is to discover such scales.



Partitioning Clustering - Density-based - 4

If one shifts the emphasis from computing densities pinned to data points to computing density functions defined over the underlying attribute space, the result is a **Density Function** algorithm DENCLUE (DENsity-based CLUstEring).

DENCLUE uses a density function $f(x, y)$ that is the superposition of several influence functions. When the f -term depends on x and y , the formula can be recognized as a convolution with a kernel. Examples include a square wave function:

$$f(x, y) = \theta\left(\frac{\|x - y\|}{\sigma}\right) = 1 \text{ if } \|x - y\| \leq \sigma \longrightarrow \text{DBSCAN}$$

and a gaussian influence function:

$$f(x, y) = e^{-\|x - y\|^2 / 2\sigma^2} \longrightarrow \text{K-means}$$



Taxonomy of Clustering Algorithms

- Hierarchical Methods
 - Agglomerative Algorithms
 - Divisive Algorithms
- Partitioning Methods
 - Relocation Algorithms
 - Probabilistic Clustering
 - K-medoids Methods
 - K-means Methods
 - Density-Based Algorithms
 - Density-Based Connectivity Clustering
 - Density Functions Clustering
- **Grid-Based Methods**
- Methods Based on Co-Occurrence of Categorical Data
- Clustering Algorithms Used in Machine Learning
 - Artificial Neural Networks, Fuzzy and Hybrid Systems
 - Evolutionary Methods
 - Constraint-Based Clustering
- Algorithms For High Dimensional and Scalable Data
 - Subspace Clustering
 - Projection Techniques
 - Co-Clustering Techniques



Grid-based Clustering - 1

In the previous section crucial concepts of density, connectivity, and boundary were used which required elaborate definitions. Another way of dealing with them is to inherit the topology from the underlying attribute space. To limit the search combinations, multi-rectangular segments are considered.

Recall that a **segment** (also cube, cell, region). is a direct Cartesian product of individual attribute sub-ranges (contiguous in case of numerical attributes). Since some binning is usually adopted for numerical attributes, methods partitioning space are frequently called grid-based methods. The elementary segment corresponding to single-bin or single-value sub-ranges is called a **unit**.

Overall, we shift our attention from data to space partitioning. **Data partitioning** is induced by point membership in segments resulted from space partitioning, while **space partitioning** is based on grid-characteristics accumulated from input data.

While density-based partitioning methods work best with numerical attributes, grid-based methods work with attributes of different types.



Grid-based Clustering - 2

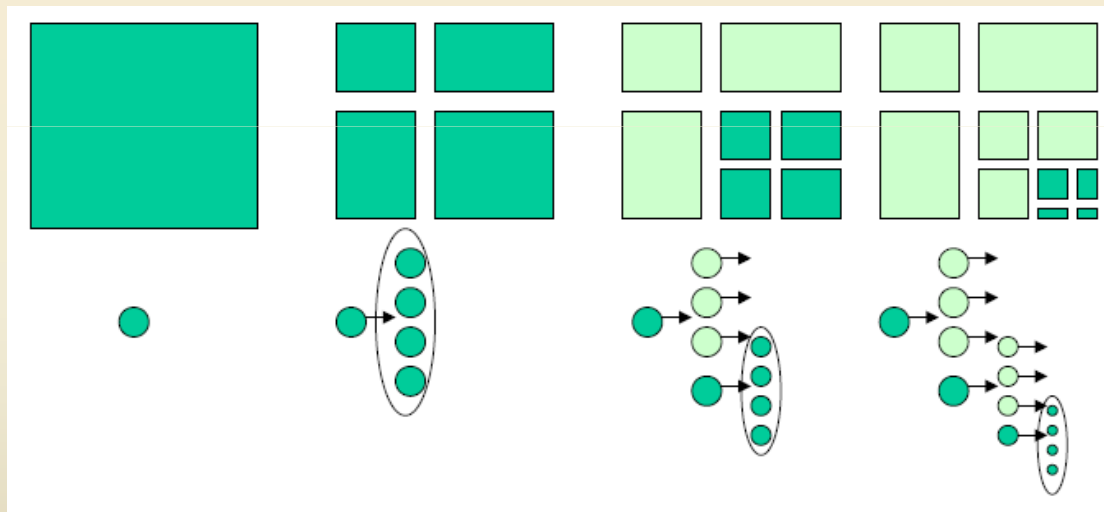
The grid-based methodology reflects a technical point of view. The category is eclectic: it contains both partitioning and hierarchical algorithms. The algorithm DENCLUE from the previous section uses grids at its initial stage. The very important grid-based algorithm CLIQUE and its descendent, algorithm MAFIA [[see reference in documents parsons, sdm01_07](#)] are examples but we will discuss later. Here we try to describe the grid-based methodology as much as possible in a general way.

Grid-based segments are used to summarize data. The segments are stored in a special structure that is a grid-directory incorporating different scales. Adjacent segments are neighbors. If a common face has maximum dimension they are called nearest neighbors. More generally, neighbors of degree between 0 and $d-1$ can be defined. The density of a segment is defined as a ratio between number of points in it and its volume. From the grid-directory, a dendrogram is directly calculated.



Grid-based Clustering - 3

In doing so, it however, assembles statistics in a hierarchical tree of nodes that are grid-cells. Figure below presents the proliferation of cells in 2-dimensional space and the construction of the corresponding tree. Each cell has four (default) children and stores a point count, and attribute-dependent measures: mean, standard deviation, minimum, maximum, and distribution type. Measures are accumulated starting from bottom level cells, and further propagate to higher-level cells (e.g., minimum is equal to a minimum among the children-minimums).



Taxonomy of Clustering Algorithms

- Hierarchical Methods
 - Agglomerative Algorithms
 - Divisive Algorithms
- Partitioning Methods
 - Relocation Algorithms
 - Probabilistic Clustering
 - K-medoids Methods
 - K-means Methods
 - Density-Based Algorithms
 - Density-Based Connectivity Clustering
 - Density Functions Clustering
- Grid-Based Methods
- **Methods Based on Co-Occurrence of Categorical Data**
- Clustering Algorithms Used in Machine Learning
 - Artificial Neural Networks, Fuzzy and Hybrid Systems
 - Evolutionary Methods
 - Constraint-Based Clustering
- Algorithms For High Dimensional and Scalable Data
 - Subspace Clustering
 - Projection Techniques
 - Co-Clustering Techniques



Co-occurrence of Categorical Data Clustering

Categorical data frequently relates to the concept of a variable size transaction that is a finite set of elements called items from a common item universe. For example, market basket data has this form. Every transaction can be presented in a point-by-attribute format, by enumerating all items j , and by associating with a transaction the binary attributes that indicate whether j -items belong to a transaction or not. Such representation is sparse and two random transactions have very few items in common.

Common to this and others examples of point-by-attribute format for categorical data, is high dimensionality, significant amount of zero values, and small number of common values between two objects. Conventional clustering methods, based on similarity measures, do not work well. Since categorical/transactional data is important in customer profiling, assortment planning, Web analysis, and other applications, different clustering methods founded on the idea of co-occurrence of categorical data have been developed.

This clustering technique is not directly involved in Astrophysical Clustering problems



Taxonomy of Clustering Algorithms

- Hierarchical Methods
 - Agglomerative Algorithms
 - Divisive Algorithms
- Partitioning Methods
 - Relocation Algorithms
 - Probabilistic Clustering
 - K-medoids Methods
 - K-means Methods
 - Density-Based Algorithms
 - Density-Based Connectivity Clustering
 - Density Functions Clustering
- Grid-Based Methods
- Methods Based on Co-Occurrence of Categorical Data
- **Clustering Algorithms Used in Machine Learning**
 - **Artificial Neural Networks, Fuzzy and Hybrid Systems**
 - **Evolutionary Methods**
 - **Constraint-Based Clustering**
- Algorithms For High Dimensional and Scalable Data
 - Subspace Clustering
 - Projection Techniques
 - Co-Clustering Techniques

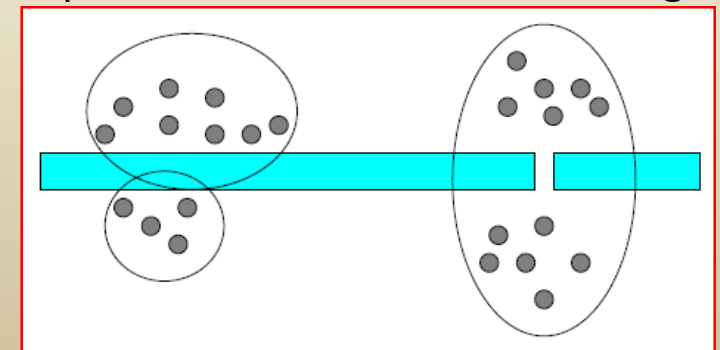
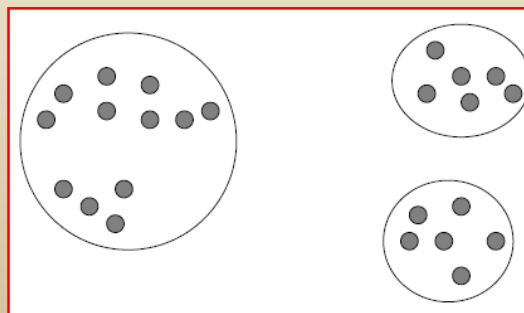


Machine Learning for Clustering – Constraint-based

The taxonomy of clustering constraints includes constraints on individual objects and parameter constraints (e.g., number of clusters) that can be addressed through preprocessing or external cluster parameters. The taxonomy also includes constraints on individual clusters that can be described in terms of bounds on aggregate functions (min, avg, etc.) over each cluster. These constraints are essential, since they require a new methodology. In particular, an existential constraint is a bound from below on a count of objects of a certain subset in each cluster. Iterative optimization used in partitioning clustering relies on moving objects to their nearest cluster representatives.

The most frequent requirement is to bound number of cluster points from below. Unfortunately, k-means algorithm, which is used most frequently, sometimes provides a number of very small (in certain implementations empty) clusters.

Important constraint-based clustering application is to cluster 2D spatial data in the presence of obstacles. Instead of regular Euclidean distance, a length of the shortest path between two points can be used as an obstacle distance. The COD (Clustering with Obstructed Distance) algorithm deals with this problem. It is best illustrated by the figure showing the difference in constructing three clusters in absence of obstacle (left) and in presence of a river with a bridge (right).



Machine Learning for Clustering – Supervised Learning

It could sound like a mistake to introduce supervised learning in clustering problems. But it is not!

Both Forgy k-means implementation and EM algorithms, seen previously, are iterative optimizations. Both initialize k models and then engage in a series of two-step iterations that: (1) reassign (hard or soft) data points, (2) update a combined model.

This process can be generalized to a framework relating clustering with predictive mining. The model update is considered as the training of a predictive classifier based on current assignments servicing as the target attribute values supervising the learning. Point reassignments correspond to the forecasting using the recently trained classifier.

I founded another elegant connection to supervised learning. It considered binary target attribute defined as Yes on points subject to clustering, and defined as No on non-existent artificial points uniformly distributed in a whole attribute space. A decision tree classifier is applied to the full synthetic data. Yes-labeled leaves correspond to clusters of input data. The new technique CLTree (CLustering based on decision Trees) resolves the challenges of populating the input data with artificial No-points such as: (1) adding points gradually following the tree construction; (2) making this process virtual (without physical additions to input data); (3) problems with uniform distribution in higher dimensions. [[see reference in document parsons](#)]



Machine Learning for Clustering – ANNs

Soft reassignments make a lot of sense, if k-means objective function is slightly modified to incorporate (similar to EM) “fuzzy errors”, that is if it accounts for distances not only to the closest, but also to the less fit centroids:

$$E(C) = \sum_{i=1:N, j=1:k} \|x_i - c_j\|^2 \cdot \omega_{ij}^2$$

Exponential probabilities are defined based on Gaussian models. This makes the objective function differentiable with respect to means and allows application of general **gradient descent** method.

Gradient descent method in k-means is known as LKMA (Local K-Means Algorithm). At each iteration t , it modifies means c_j in the direction of gradient descent (x is selected randomly):

$$c_j^{t+1} = c_j^t + a_t (x_i - c_j^t) \omega_{ij}^2$$

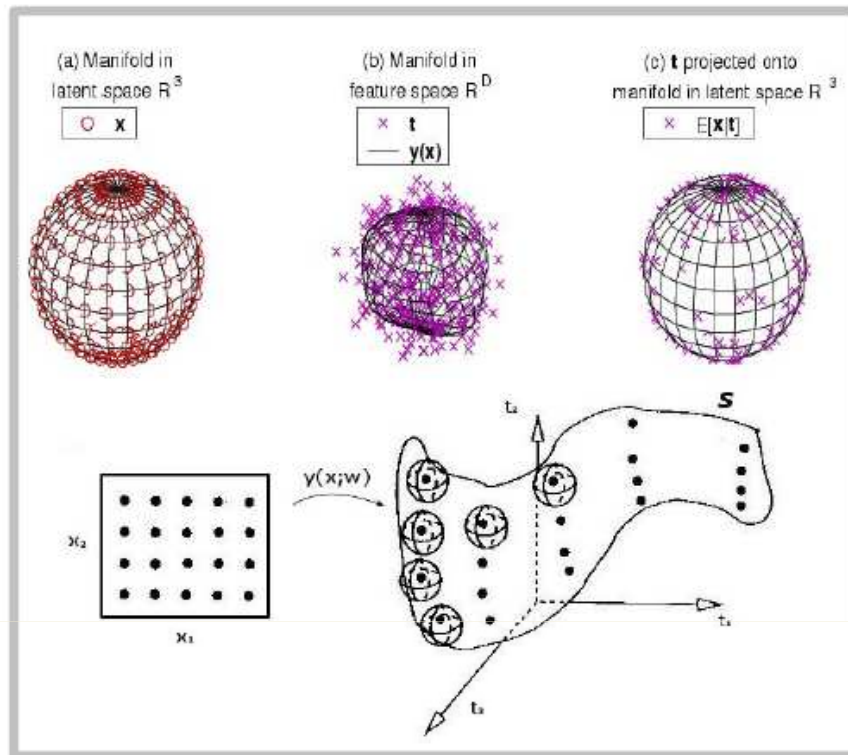
Such updates are also used in a different context of artificial neural network (ANN) clustering, namely SOM (Self-Organized Map) [Kohonen 1990]. SOM is popular in vector quantization. We will not elaborate here about SOM (it should be well known) except for two important features:

- (1) SOM uses incremental approach and points (patterns) are processed one-by-one;
- (2) SOM allows to map centroids into 2D plane that provides for a straightforward visualization.

In addition to SOM, other ANN developments, such as adaptive resonance theory [Carpenter et al. 1991], have relation to clustering.



Machine Learning for Clustering – PPS & NEC



PPS: the Beauty of Spheres

The original m -dimensional data space is mapped in a lower n -dimensional space, called “latent space”. **Visualization ease** as a spherical manifold is fitted to the data, then projected into the manifold in R^3 and plotted as points on the sphere surface. Each latent variable on the sphere is responsible for a number of projected points, which form a “cluster”.

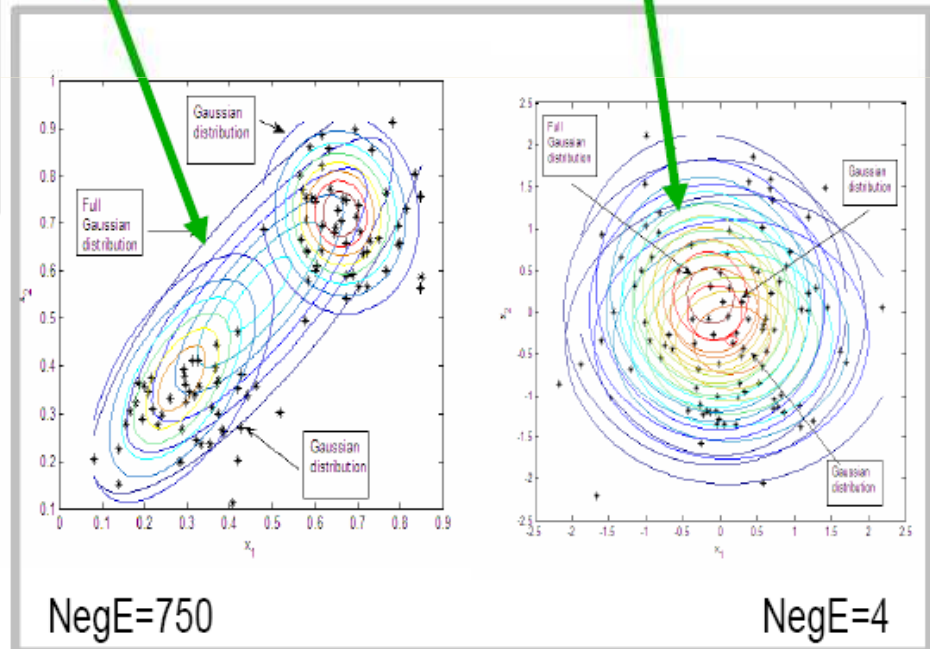
NEC: a matter of Gaussians

Clustering method based on the “neg-entropy” NegE, a measure of non gaussianity of a variable. If A is gaussian, then $\text{NegE}(A) = 0$. Given a threshold d :

If $\text{NegE}(A \cup B) < d$, then clusters A and B are replaced by cluster $A \cup B$

Not replaced!

Replaced!



Machine Learning for Clustering – Evolutionary Clustering

Here substantial information on simulated annealing in the context of partitioning (main focus) or hierarchical clustering is reported. General annealing has a simple meaning in clustering: it amounts to a relocation of a point from its current to a new randomly chosen cluster (very similar to k-means scheme).

Genetic Algorithms (GA) [Goldberg 1989] are also used in cluster analysis, especially in the context of k-means objective function. A population is a set of k-means systems represented by grid segments instead of centroids. Every segment is described by d rules (genes), one per attribute range. The population is improved through mutation and crossover specifically devised for these rules. Unlike in normal k-means, clusters can have different size and elongation; however, shapes are restricted to segments, far from density-based methods. GA were also applied to clustering of categorical data using so-called generalized entropy to define the dissimilarity.

Evolutionary techniques rely on certain parameters to empirically fit data and have high computational costs that limit their application in data mining. However, usage of combined strategies (e.g., generation of initial guess for k-means) has been attempted [Babu & Murty 1993; Babu & Murty 1994]. Usage of GA with variable length genome to simultaneously improve k-means centroids and k itself also has a merit in comparison with running multiple k-means to determine a k , since changes in k happen before full convergence is achieved.



Taxonomy of Clustering Algorithms

- Hierarchical Methods
 - Agglomerative Algorithms
 - Divisive Algorithms
- Partitioning Methods
 - Relocation Algorithms
 - Probabilistic Clustering
 - K-medoids Methods
 - K-means Methods
 - Density-Based Algorithms
 - Density-Based Connectivity Clustering
 - Density Functions Clustering
- Grid-Based Methods
- Methods Based on Co-Occurrence of Categorical Data
- Clustering Algorithms Used in Machine Learning
 - Artificial Neural Networks, Fuzzy and Hybrid Systems
 - Evolutionary Methods
 - Constraint-Based Clustering
- **Algorithms For High Dimensional and Scalable Data**
 - **Subspace Clustering**
 - **Projection Techniques**
 - **Co-Clustering Techniques**



High Dimensionality Clustering – General Intro

The objects in astrophysical data mining could have hundreds of attributes. Clustering in such **high dimensional** spaces presents tremendous difficulty, much more so than in predictive learning. In decision trees, for example, irrelevant attributes simply will not be picked for node splitting, and it is known that they do not affect Bayes as well. In clustering, however, high dimensionality presents a dual problem. First, under whatever definition of similarity, the presence of irrelevant attributes eliminates any hope on clustering tendency. After all, searching for clusters where there are no clusters is a hopeless enterprise. While this could also happen with low dimensional data, the likelihood of presence and number of irrelevant attributes grows with dimension.

The second problem is the **dimensionality curse** that is a loose way of speaking about a lack of data separation in high dimensional space. Mathematically, nearest neighbor query becomes unstable: the distance to the nearest neighbor becomes indistinguishable from the distance to the majority of points. This effect starts to be severe for dimensions greater than 15. Therefore, construction of clusters founded on the concept of proximity is doubtful in such situations.

Basic exploratory data analysis (attribute selection) preceding the clustering step is the best way to address the first problem of irrelevant attributes.



High Dimensionality Clustering – Dimensionality Reduction

Many spatial clustering algorithms depend on indices in spatial datasets (sub-section Data Preparation) to facilitate quick search of the nearest neighbors. Therefore, indices can serve as good proxies with respect to dimensionality curse performance impact. Indices used in clustering algorithms are known to work effectively for dimensions below 16. For a dimension $d > 20$ their performance degrades to the level of sequential search (though newer indices achieve significantly higher limits). Therefore, we can arguably claim that data with more than 16 attributes is high dimensional.

Two general purpose techniques are used to fight high dimensionality:

(1) Attributes transformations and (2) domain decomposition

Attribute transformations are simple functions of existent attributes. In multivariate statistics principal components analysis (PCA) is popular, but this approach is problematic since it leads to clusters with poor interpretability [see reference in document [jcc1659](#)]. Singular value decomposition (SVD) technique is used to reduce dimensionality in information retrieval [Berry et al. 1995; Berry & Browne 1999] and statistics [Fukunaga 1990]. Low-frequency Fourier harmonics in conjunction with Parseval's theorem are successfully used in analysis of time series [Agrawal et al. 1993], as well as wavelets and other transformations.

Domain decomposition divides the data into subsets using some inexpensive similarity measure, so that the high dimensional computation happens over smaller datasets. Dimension stays the same, but the costs are reduced. This approach targets the situation of high dimension, large data, and many clusters.



High Dimensionality Clustering – Subspace - 1

The algorithm CLIQUE (Clustering In QUEst) [[see reference in documents parsons, sdm01_07](#)] for numerical attributes is fundamental in subspace clustering. It marries the ideas of:

- Density-based clustering;
- Grid-based clustering;
- Induction through dimensions similar to a priori algorithm in association rules;

CLIQUE starts with the definition of a unit elementary rectangular cell in a subspace. Only units whose densities exceed a threshold are retained. A bottom-up approach of finding such units is applied. First, 1-dimensional units are found by dividing intervals in equal-width bins (a grid). The recursive step from $q-1$ -dimensional units to q -dimensional units involves self-join of $q-1$ units having first common $q-2$ dimensions (a priori-reasoning). All the subspaces are sorted by their coverage and lesser-covered subspaces are pruned. A cut point is selected. A cluster is defined as a maximal set of connected dense units.

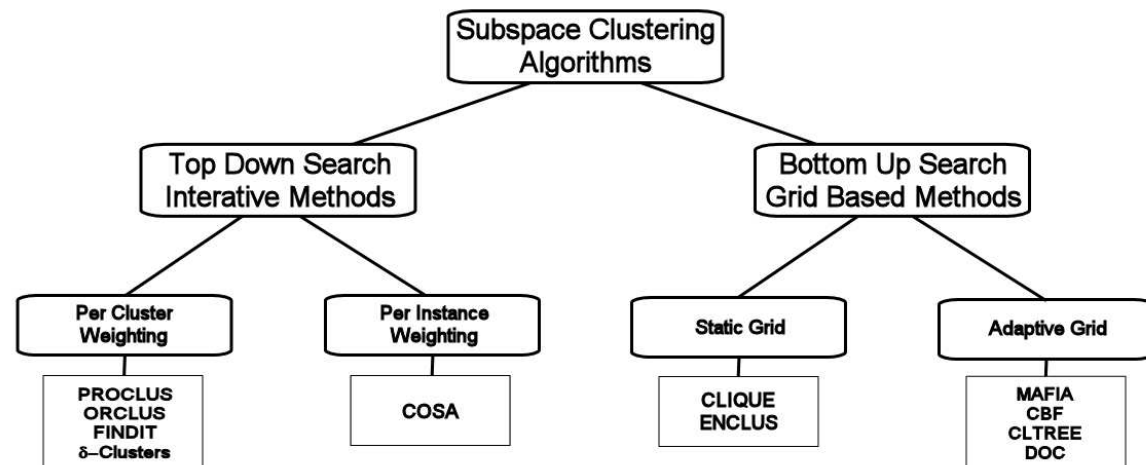
Effectively, CLIQUE results in attribute selection (it selects several subspaces) and produces a view of data from different perspectives. The result is a series of cluster systems in different subspaces. This versatility goes more in vein with data description rather than with data partitioning: different clusters overlap.



High Dimensionality Clustering – Subspace - 2

The algorithm MAFIA (Merging of Adaptive Finite Intervals) [see reference in documents [parsons](#), [sdm01_07](#)] significantly modifies CLIQUE. It starts with one data pass to construct adaptive grids in each dimension. Many (1000) bins are used to compute histograms by reading blocks of data in core memory, which are then merged together to come up with a smaller number of variable-size bins than CLIQUE does. The algorithm uses a parameter, called cluster dominance factor, to select bins that are x -times more densely populated relative to their volume than on average. These are $q=1$ candidate dense units (CDUs).

Then MAFIA proceeds recursively to higher dimensions (every time a data scan is involved). The difference between MAFIA and CLIQUE is that to construct a new q -CDU, MAFIA tries two $q-1$ -CDUs as soon as they share any (not only first dimensions) $q-2$ -face. This creates an order of magnitude more candidates. Adjacent CDUs are merged into clusters and clusters that are proper subsets of the higher dimension clusters are eliminated. The parameter (default value 1.5 works fine) presents no problem in comparison with global density threshold used in CLIQUE.



High Dimensionality Clustering – Co-Clustering

An interesting general idea of producing attribute groups in conjunction with clustering of points themselves leads to the concept of co-clustering. Co-clustering is a simultaneous clustering of both points and their attributes. This approach reverses the struggle: to improve clustering of points based on their attributes, it tries to cluster attributes based on the points. So far we were concerned with grouping only rows of a matrix X . Now we are talking about grouping its columns as well. This utilizes a canonical duality contained in the point-by-attribute data representation.

The idea of co-clustering of data points and attributes is old and is known under the names *simultaneous clustering*, *bi-dimensional clustering*, *block clustering*, *conjugate clustering*, *distributional clustering*, and *information bottleneck method*.

The co-clustering deals with distributional clustering of attributes based on the informational measures of attribute similarity. Two attributes (two columns in matrix X) with exactly the same probability distributions are identical for the purpose of data mining, and so, one can be deleted. Attributes that have probability distributions that are close in terms of their Kullback-Leibler (KL) distance [Kullback & Leibler 1951] can still be grouped together without much of an impact. In addition, a natural derived attribute, the mixed distribution (a normalized sum of two columns) is now available to represent the group.



Clustering – General Summary

We have presented many different clustering techniques. However, there are common issues that must be addressed to make any clustering algorithm successful. Some are so general that they are not even specific to unsupervised learning and can be considered as a part of overall data mining framework. Others, more connected with our research topic (data mining of massive datasets in Astronomy) are partially resolved in certain algorithms we presented.

Now we overview common issues, and necessarily our coverage will be very fragmented. In particular high dimensional clustering (one of our most significant targets) was already surveyed above, but several others significant issues are discussed below:

- Assessment of results
- Choice of appropriate number of clusters
- Data preparation
- Proximity measures
- Handling outliers



Clustering – Assessment of results

The data mining clustering process starts with the assessment of whether any cluster tendency has a place at all, and correspondingly includes, appropriate attribute selection, and in many cases feature construction. It finishes with the validation and evaluation of the resulting clustering system. The clustering system can be assessed by an expert, or by a particular automated procedure. Traditionally, the first type of assessment relates to two issues: (1) cluster interpretability, (2) cluster visualization. Interpretability depends on the technique used. Model-based probabilistic algorithms have better scores in this regard. K-means and k-medoid methods generate clusters that are interpreted as dense areas around centroids or medoids and, therefore, also score well.

Regarding automatic procedures, when two partitions are constructed (with the same or different number of subsets k), the first order of business is to compare them. Sometimes the actual class label s of one partition is known. Still clustering is performed generating another label j . The situation is similar to testing a classifier in predictive mining when the actual target is known. Comparison of s and j labels is similar to computing an error, confusion matrix, etc., in predictive mining.



Clustering – Choice of appropriate number of clusters - 1

In many methods number k of clusters to construct is an input user parameter. Running an algorithm several times leads to a sequence of clustering systems. Each system consists of more granular and less-separated clusters. In the case of k -means, the objective function is monotone decreasing. Therefore, the answer to the question of which system is preferable is not trivial.

Many criteria have been introduced to find an optimal k . For instance, a distance between any two centroids (medoids) normalized by corresponding cluster radii (standard deviations) and averaged (with cluster weights) is a reasonable choice of coefficient of separation.

Another choice for separation measure is the following:

Consider average distance between the point x of cluster C and other points within C and compare it with averaged distance to the best fitting cluster G other than C :

$$\begin{cases} a(x) = \frac{1}{|C|-1} \cdot d(x, y)_{y \in C, y \neq x} \\ b(x) = \min_{G \neq C} \frac{1}{|G|} \cdot d(x, y)_{y \in G} \end{cases} \Rightarrow s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

The coefficient of x is $s(x)$, values close to +1 corresponding to a perfect clustering choice and values below 0 to a bad clustering choice.

The overall average of individual $s(x)$ gives a good indication of cluster system appropriateness.



Clustering – Choice of appropriate number of clusters - 2

Still another approach to separation is to employ possible soft (or fuzzy) assignments. Remember that assignment of a point to a particular cluster may frequently involve certain arbitrariness. Depending on how well a point fits a particular cluster C , different probabilities or weights $w(x,C)$ can be introduced so that a hard (strict) assignment is defined as:

$$C(x) = \min \arg_C w(x, C)$$

A Partition coefficient is equal to the sum of squares of the weights:

$$X = \frac{1}{N} \sum_{x \in X} w(x, C(x))^2$$

Each of the discussed measures can be plotted as a function of k and the graph can be used to choose the best k .

The Probabilistic Clustering allows viewing a choice of optimal k as a problem of fitting the data by the best model. The question is whether adding new parameters results in a better model. In Bayesian learning the likelihood of the model is directly affected (through priors) by the model complexity (number of parameters is proportional to k). Several criteria were suggested.

All these criteria are expressed through combinations of log-likelihood L , number of clusters k , number of parameters per cluster and total number of estimated parameters.



Clustering – Data Preparation

Attributes transformation and clustering have already been discussed in the context of dimensionality reduction. The practice of assigning different weights to attributes and/or scaling of their values is widespread and allows constructing clusters of better shapes.

In real-life applications it is crucial to handle attributes of different nature. For example, astronomical images are characterized by color, wavelength, texture, shape, and location, resulting in five attribute subsets.

Some algorithms depend on the effectiveness of data access. To facilitate this process data indices are constructed. Index structures used for spatial data, include KD-trees. A blend of attribute transformations (DFT, Polynomials) and indexing technique is present in many methods. The major application of such data structures is in nearest neighbors search.



Clustering – Proximity Measures

Both hierarchical and partitioning methods use different distances and similarity measures. The usual distance L_p :

$$L_p = d(x, y) = \|x - y\|_{1 \leq p \leq \infty}$$

in which lower p corresponds to a more robust estimation (therefore, less affected by outliers). Euclidean ($p=2$) distance is by far the most popular choice used in k-means objective function (sum of squares of distances between points and centroids) that has a clear statistical meaning of total inter-clusters variance. The distance that returns the maximum of absolute difference in coordinates is also used and corresponds to $p = \infty$.

In many applications (profile analyses) points are scaled to have a unit norm, so that the proximity measure is an angle between the points:

$$d(x, y) = \arccos\left(\frac{x^T y}{\|x\| \cdot \|y\|}\right)$$



Clustering – Handling Outliers - 1

Applications that derive their data from measurements have an associated amount of noise, which can be viewed as outliers. Alternately, outliers can be viewed as records having abnormal behavior. In general, clustering techniques do not distinguish between the two: neither noise nor abnormalities fit into clusters. Correspondingly, the preferable way to deal with outliers in partitioning the data is to keep one extra set of outliers.

There are multiple ways of how descriptive learning handles outliers. If a data preprocessing phase is present, it usually takes care of outliers. For example, this is the case with grid-based methods. They simply rely on input thresholds to eliminate low-populated cells. Other algorithms revisit outliers during the decision tree rebuilds, but in general handle them separately, by producing a partition plus a set of outliers.

Certain algorithms have specific features for outliers handling. Some of them use shrinkage of cluster representatives to suppress the effects of outliers. K-medoids methods are generally more robust than k-means methods with respect to outliers: medoids do not “feel” outliers. Others (such as DBSCAN) use concepts of internal (core), boundary (reachable), and outliers (non-reachable) points. The algorithm CLIQUE goes a step further: it eliminates subspaces with low coverage.



Clustering – Handling Outliers - 2

What is exactly an outlier? Statistics defines an outlier as a point that does not fit a probability distribution. Classic data analysis utilizes a concept of depth and defines an outlier as a point of low depth. This concept becomes computationally infeasible for $d > 3$ (human limits). Data mining is gradually developing its own definitions.

Consider two positive parameters x, y . A point can be declared an outlier if its neighborhood contains less than 1 -fraction of a whole dataset X . Rank all the points by their distance to the K -nearest neighbor and define the fraction of points with highest ranks as outliers.

How to describe local outliers? Different subsets of data can have different densities and may be governed by different distributions. A point close to a tight cluster can be a more probable outlier than a point that is further away from a more dispersed cluster. The concept of local outlier factor (LOF) that specifies a degree of outlier-ness comes to rescue. The definition is simply based on the distance to the k -nearest neighbor.



Clustering – Suggestions to improve this review

The sort of review done above is not exhaustive, nor it wants to solve the clustering investigation. It has been done in order to introduce the problem scenario and to drive reader towards to evaluate best choice.

Remember that we have to find and implement the better clustering method for a specific problem: self-adaptive data mining of astronomical massive data sets with unsupervised clustering techniques (possibly including fuzziness).

In particular fuzzy-based models have intentionally left out of this review, in order to avoid any biased concept to be investigated in the current thesis design.

Anyway, I suggest the reader to learn basic clustering theory from this presentation and by further integrate it with the deep analysis of the following papers:

(file v1-27) IMDC: An Image-Mapped Data Clustering Technique for Large Datasets

(file Murtagh-Clustering_in_massive_data_sets) Clustering in Massive Data Sets

(file jcc1659) Nonlinear Mapping of Massive Data Sets by Fuzzy Clustering and Neural Networks

(file 2002-EuroPar-Terence) Parallel Fuzzy c-Means Clustering for Large Data Sets

(file parsons) Subspace Clustering for High Dimensional Data: A Review

(file fulltext) Cluster analysis of massive datasets in astronomy

