Survey of Clustering Data Mining Techniques

Pavel Berkhin
Accrue Software, Inc.

Clustering is a division of data into groups of similar objects. Representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification. It models data by its clusters. Data modeling puts clustering in a historical perspective rooted in mathematics, statistics, and numerical analysis. From a machine learning perspective clusters correspond to *hidden patterns*, the search for clusters is *unsupervised learning*, and the resulting system represents a *data concept*. 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.

Clustering is the subject of active research in several fields such as statistics, pattern recognition, and machine learning. This survey focuses on clustering in data mining. Data mining adds to clustering the complications of very large datasets with very many attributes of different types. This imposes unique computational requirements on relevant clustering algorithms. A variety of algorithms have recently emerged that meet these requirements and were successfully applied to real-life data mining problems. They are subject of the survey.

Categories and Subject Descriptors: I.2.6. [**Artificial Intelligence**]: Learning – *Concept learning*; I.4.6 [**Image Processing**]: Segmentation; I.5.1 [**Pattern Recognition**]: Models; I.5.3 [**Pattern Recognition**]: Clustering.

General Terms: Algorithms, Design

Additional Key Words and Phrases: Clustering, partitioning, data mining, unsupervised learning, descriptive learning, exploratory data analysis, hierarchical clustering, probabilistic clustering, k-means

Content:

Author’s address: Pavel Berkhin, Accrue Software, 1045 Forest Knoll Dr., San Jose, CA, 95129; e-mail: pavelb@accrue.com

1. Introduction

The goal of this survey is to provide a comprehensive review of different clustering techniques in data mining. *Clustering* is a division of data into groups of similar objects. Each group, called cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups. Representing data by fewer clusters necessarily loses certain fine details (akin to lossy data compression), but achieves simplification. It represents many data objects by few clusters, and hence, it models data by its clusters. Data modeling puts clustering in a historical perspective rooted in mathematics, statistics, and numerical analysis. From a machine learning perspective clusters correspond to *hidden patterns*, the search for clusters is *unsupervised learning*, and the resulting system represents a *data concept*. Therefore, clustering is unsupervised learning of a hidden data concept. Data mining deals with large databases that impose on clustering analysis additional severe computational requirements. These challenges led to the emergence of powerful broadly applicable data mining clustering methods surveyed below.

1.1. Notations

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*) $x_i = (x_{i1},...,x_{id}) \in A$ in attribute space A, where $i = 1{:}N$, and each component $x_{il} \in A_l$ is a numerical or nominal categorical *attribute* (or synonymously, *feature*, *variable*, *dimension*, *component*, *field*). For a discussion of attributes data types see [Han & Kamber 2001]. Such point-by-attribute data format conceptually corresponds to a $N \times d$ matrix and is used by the majority of algorithms reviewed below. However, data of other formats, such as variable length sequences and heterogeneous data, is becoming more and more popular. The simplest attribute space subset is a direct Cartesian product of sub-ranges $C = \prod C_l \subset A,\ C_l \subseteq A_l,\ l = 1:d,$ called a *segment* (also *cube, cell, region*). A *unit* is an elementary segment whose sub-ranges consist of a single category value, or of a small numerical bin. Describing the numbers of data points per every *unit* represents an extreme case of clustering, a *histogram*, where no actual clustering takes place. This is a very expensive representation, and not a very revealing one. User driven *segmentation* is another commonly used practice in data exploration that utilizes expert knowledge regarding the importance of certain sub-domains. We distinguish clustering from segmentation to emphasize the importance of the automatic learning process.

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

$$X = C_1\ 8 \ldots 8\ C_k\ \ 8\ C_{ouliers},\ C_{j_1}\ 1\ C_{j_2} = \phi\ .$$

1.2. Clustering Bibliography at Glance

General references regarding clustering include [Hartigan 1975; Spath 1980; Jain & Dubes 1988; Kaufman & Rousseeuw 1990; Dubes 1993; Everitt 1993; Mirkin 1996; Jain et al. 1999; Fasulo 1999; Kolatch 2001; Han et al. 2001; Ghosh 2002]. A very good introduction to contemporary data mining clustering techniques can be found in the textbook [Han & Kamber 2001].

There is a close relationship between clustering techniques and many other disciplines. Clustering has always been used in statistics [Arabie & Hubert 1996] and science [Massart & Kaufman 1983]. The classic introduction into pattern recognition framework is given in [Duda & Hart 1973]. Typical applications include *speech* and *character recognition*. Machine learning clustering algorithms were applied to *image segmentation* and *computer vision* [Jain & Flynn 1996]. For statistical approaches to pattern recognition see [Dempster et al. 1977] and [Fukunaga 1990]. Clustering can be viewed as a density estimation problem. This is the subject of traditional multivariate statistical estimation [Scott 1992]. Clustering is also widely used for data compression in image processing, which is also known as *vector quantization* [Gersho & Gray 1992]. Data fitting in numerical analysis provides still another venue in data modeling [Daniel & Wood 1980].

This survey's emphasis is on clustering in data mining. Such clustering is characterized by large datasets with many attributes of different types. Though we do not even try to review particular applications, many important ideas are related to the specific fields. Clustering in data mining was brought to life by intense developments in information retrieval and text mining [Cutting et al. 1992; Steinbach et al. 2000; Dhillon et al. 2001], spatial database applications, for example, GIS or astronomical data, [Xu et al. 1998; Sander et al. 1998; Ester et al. 2000], sequence and heterogeneous data analysis [Cadez et al. 2001], Web applications [Cooley et al. 1999; Heer & Chi 2001; Foss et al. 2001], DNA analysis in computational biology [Ben-Dor & Yakhini 1999], and many others. They resulted in a large amount of application-specific developments that are beyond our scope, but also in some general techniques. These techniques and classic clustering algorithms that relate to them surveyed below.

1.3. Classification of Clustering Algorithms

Categorization of clustering algorithms is neither straightforward, nor canonical. In reality, groups below overlap. For reader's convenience we provide a classification closely followed by this survey. Corresponding terms are explained below.

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
- Constraint-Based Clustering
- Clustering Algorithms Used in Machine Learning
  - Gradient Descent and Artificial Neural Networks
  - Evolutionary Methods
- Scalable Clustering Algorithms
- Algorithms For High Dimensional Data
  - Subspace Clustering
  - Projection Techniques
  - Co-Clustering Techniques

1.4. Plan of Further Presentation

Traditionally clustering techniques are broadly divided in *hierarchical* and *partitioning*. Hierarchical clustering is further subdivided into *agglomerative* and *divisive*. The basics of hierarchical clustering include Lance-Williams formula, idea of *conceptual clustering*, now classic algorithms SLINK, COBWEB, as well as newer algorithms CURE and CHAMELEON. We survey them in the section *Hierarchical 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. Algorithms of the first kind are surveyed in the section *Partitioning Relocation Methods*. They are further categorized into *probabilistic clustering* (EM framework, algorithms SNOB, AUTOCLASS, MCLUST), *k-medoids* methods (algorithms PAM, CLARA, CLARANS, and its extension), and *k-means* methods (different schemes, initialization, optimization, harmonic means, extensions). Such methods concentrate on how well points fit into their clusters and tend to build clusters of proper convex shapes.

Partitioning algorithms of the second type are surveyed in the section *Density-Based Partitioning*. They try to discover dense connected components of data, which are flexible in terms of their shape. Density-based connectivity is used in the algorithms DBSCAN, OPTICS, DBCLASD, while the algorithm DENCLUE exploits space density functions. These algorithms are 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 (algorithm GDBSCAN).

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. We discuss them in the section *Grid-Based Methods*. They frequently use hierarchical agglomeration as one phase of processing. Algorithms BANG, STING, WaveCluster, and an idea of fractal dimension are discussed in this section. Grid-based methods are fast and handle outliers well. Grid-based methodology is also used as an intermadiate step in many other algorithms (for example, CLIQUE, MAFIA).

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 algorithms ROCK, SNN, and CACTUS are surveyed in the section *Co-Occurrence of Categorical Data*. 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 and the algorithm STIRR exemplify this approach.

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. The boundary is blurred. In the section *Other Clustering Techniques* we discuss *relationship to supervised learning*, *gradient descent* and *ANN* (LKMA, SOM), *evolutionary methods* (simulated annealing, genetic algorithms (GA)), and the algorithm AMOEBA. We start, however, with the emerging field of *constraint-based clustering* that is influenced by requirements of real-world data mining applications.

Data Mining primarily works with large databases. Clustering large datasets presents scalability problems reviewed in the section *Scalability and VLDB Extensions*. Here we talk about algorithms like DIGNET, about BIRCH and other data squashing techniques, and about Hoffding or Chernoff bounds.

Another trait of real-life data is its high dimensionality. Corresponding developments are surveyed in the section *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 (DFT, PCA, wavelets). Another way to address the problem is through *subspace clustering* (algorithms CLIQUE, MAFIA, ENCLUS, OptiGrid, PROCLUS, ORCLUS). Still another approach clusters attributes in groups and uses their derived proxies to cluster objects. This double clustering is known as *co-clustering*.

Issues that are common to different clustering methods are overviewed in the section *General Algorithmic Issues*. We talk about *assessment of results*, determination of *appropriate number of clusters* to build, *data preprocessing* (attribute selection, data scaling, special data indices), *proximity measures*, and *handling outliers*.

1.5. Important Issues

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

- Type of attributes algorithm can handle
- Scalability to large datasets
- Ability to work with high dimensional data
- 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 of fuzzy)
- Reliance on a priori knowledge and user defined parameters
- Interpretability of results

While we try to keep these issues in mind, realistically, we mention only few with every algorithm we discuss. The above list is in no way exhaustive. For example, we also discuss such properties as ability to work in pre-defined memory buffer, ability to restart and ability to provide an intermediate solution.

2. Hierarchical Clustering

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) [Jain & Dubes 1988; Kaufman & Rousseeuw 1990]. 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 of hierarchical clustering include:

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

Disadvantages of hierarchical clustering are related to:

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

The classic approaches to hierarchical clustering are presented in the sub-section *Linkage Metrics*. Hierarchical clustering based on linkage metrics results in clusters of proper (convex) shapes. Active contemporary efforts to build cluster systems that incorporate our intuitive concept of clusters as connected components of arbitrary shape, including the algorithms CURE and CHAMELEON, are surveyed in the sub-section *Hierarchical Clusters of Arbitrary Shapes*. Divisive techniques based on binary taxonomies are presented in the sub-section *Binary Divisive Partitioning*. The sub-section *Other Developments* contains information related to incremental learning, model-based clustering, and cluster refinement.

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. Linkage metrics are constructed (see below) from elements of this matrix. 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. For example, nearest neighbor chains have decisive impact on memory consumption [Olson 1995]. A sparse matrix can be further used to represent intuitive concepts of closeness and connectivity. Notice that the way we process original (dis)similarity matrix and construct a linkage metric reflects our a priori ideas about the data model.

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.

One of the most striking developments in hierarchical clustering is the algorithm BIRCH. Since scalability is the major achievement of this blend strategy, this algorithm is discussed in the section *Scalable VLDB Extensions*. However, data squashing used by

BIRCH to achieve scalability, has independent importance. Hierarchical clustering of large datasets can be very sub-optimal, even if data fits in memory. Compressing data may improve performance of hierarchical algorithms.

2.1. 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(s) for merging/splitting depends on the (dis)similarity of cluster(s) 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. Other proximity measures are discussed in the section *General Algorithm Issues*.

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*. Major inter-cluster linkage metrics [Murtagh 1985, Olson 1995] include *single link*, *average link*, and *complete link*. The underlying dissimilarity measure (usually, distance) is computed for every pair of points with one point in the first set and another point in the second set. A specific operation such as minimum (single link), average (average link), or maximum (complete link) is applied to pair-wise dissimilarity measures:

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

Early examples include the algorithm SLINK [Sibson 1973], which implements single link, Voorhees' method [Voorhees 1986], which implements average link, and the algorithm CLINK [Defays 1977], which implements complete link. Of these SLINK is referenced the most. It is related to the problem of finding the Euclidean minimal spanning tree [Yao 1982] and has $O(N^2)$ complexity. The methods using inter-cluster distances defined in terms of pairs with points in two respective clusters (subsets) are called ***graph*** methods. They do not use any cluster representation other than a set of points. This name naturally relates to the connectivity graph $G = (X, E)$ introduced above, since every data partition corresponds to a graph partition. Such methods can be appended by so-called ***geometric*** methods in which a cluster is represented by its central point. It results in *centroid*, *median,* and *minimum variance* linkage metrics. Under the assumption of numerical attributes, the center point is defined as a centroid or an average of two cluster centroids subject to agglomeration.

All of the above linkage metrics can be derived as instances of the Lance-Williams updating formula [Lance & Williams 1967]

$$d(C_i \; 8 \; C_j, C_k) = a(i)d(C_i, C_k) + a(k)d(C_j, C_k) + bd(C_i, C_j) + c\left|d(C_i, C_k) - d(C_j, C_k)\right| .$$

Here *a,b,c* are coefficients corresponding to a particular linkage. This formula expresses a linkage metric between the union of the two clusters and the third cluster in terms of underlying components. The Lance-Williams formula has an utmost importance since it makes manipulation with dis(similarity) computationally feasible. Survey of linkage metrics can be found in [Murtagh 1983; Day & Edelsbrunner 1984]. When the base measure is distance, these methods capture inter-cluster closeness. However, a similarity-based view that results in intra-cluster connectivity considerations is also possible. This is how original average link agglomeration (Group-Average Method) [Jain & Dubes 1988] was introduced.

Linkage metrics-based hierarchical clustering suffers from time complexity. Under reasonable assumptions, such as *reducibility condition* (graph methods satisfy this condition), linkage metrics methods have $O(N^2)$ complexity [Olson 1995]. Despite the unfavorable time complexity, these algorithms are widely used. An example is algorithm AGNES (AGlomerative NESting) [Kaufman & Rousseeuw 1990] used in S-Plus.

When the connectivity $N \times N$ matrix is sparsified, graph methods directly dealing with the connectivity graph G can be used. In particular, hierarchical divisive MST (Minimum Spanning Tree) algorithm is based on graph partitioning [Jain & Dubes 1988].

2.2. Hierarchical Clusters of Arbitrary Shapes

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. The first one is data sampling (section *Scalability and VLDB Extensions*). 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. 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' aggregate closeness. 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. Shrinkage suppresses the affect of the outliers since outliers happen to be located further from the cluster centroid than the other scattered representatives. CURE is capable of finding clusters of different shapes and sizes, and it is insensitive to outliers. Since CURE uses sampling, estimation of its complexity is not straightforward. For low-dimensional data authors provide a complexity estimate of $O(N^2_{sample})$ defined in terms of sample size.

More exact bounds depend on input parameters: shrink factor α, number of representative points c, number of partitions, and sample size. Figure 1 illustrates agglomeration in Cure. Three clusters, each with three representatives, are shown before and after the merge and shrinkage. Two closest representatives are connected by arrow.

While the algorithm CURE works with numerical attributes (particularly low dimensional spatial data), the algorithm ROCK developed by the same researchers [Guha et al. 1999] targets hierarchical agglomerative clustering for categorical attributes. It is surveyed in the section *Co-Occurrence of Categorical Data*.

The hierarchical agglomerative algorithm CHAMELEON [Karypis et al. 1999a] utilizes dynamic modeling in cluster aggregation. It uses the connectivity graph G corresponding to the K-nearest neighbor model sparsification of the connectivity matrix: the edges of K most similar points to any given point are preserved, the rest are pruned. CHAMELEON has two stages. In the first stage small tight clusters are built to ignite the second stage. This involves a graph partitioning [Karypis & Kumar 1999]. In the second stage agglomerative process is performed. It utilizes measures of relative inter-connectivity $RI(C_i,C_j)$ and relative closeness $RC(C_i,C_j)$; both are locally normalized by quantities related to clusters C_i,C_j. In this sense the modeling is *dynamic*. Normalization involves certain non-obvious graph operations [Karypis & Kumar 1999]. CHAMELEON strongly relies on graph partitioning implemented in the library HMETIS (see the section *Co-Occurrence of Categorical Data*). Agglomerative process depends on user provided thresholds. A decision to merge is made based on the combination

$$RI(C_i,C_j)\cdot RC(C_i,C_j)^{\alpha}$$

of local relative measures. The algorithm does not depend on assumptions about the data model. This algorithm is proven to find clusters of different shapes, densities, and sizes in 2D (two-dimensional) space. It has a complexity of $O(Nm + N\log(N) + m^2\log(m))$, where m is number of sub-clusters built during first initialization phase. Figure 2 (analogous to the one in [Karypis & Kumar 1999]) presents a choice of four clusters (a)-(d) for a merge. While Cure would merge clusters (a) and (b), CHAMELEON makes intuitively better choice of merging (c) and (d).

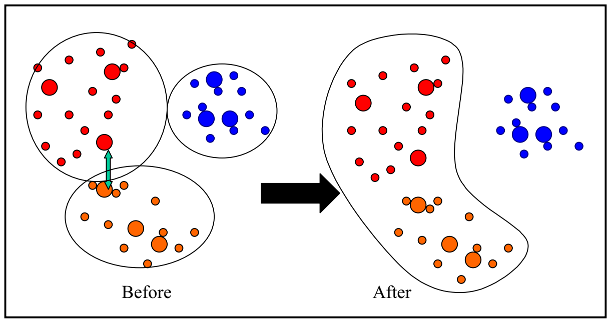


Figure 1: Agglomeration in Cure.

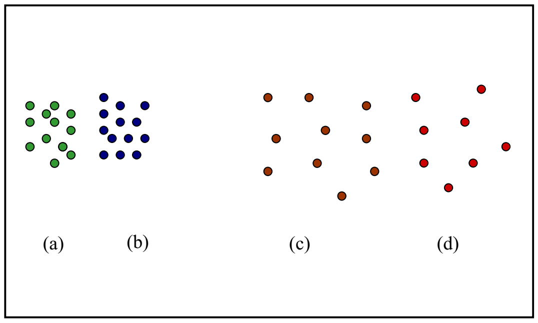


Figure 2: CHAMELEON merges (c) and (d).

2.3. Binary Divisive Partitioning

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 [Berry & Browne 1999]. SVD application to hierarchical divisive clustering of document collections resulted in the PDDP (Principal Direction Divisive Partitioning) algorithm [Boley 1998]. In our notations, object x is a document, l^{th} attribute corresponds to a word (*index term*), and matrix entry x_{il} is a measure (as TF-IDF) of l-term frequency in a document x. PDDP constructs SVD decomposition of the matrix

$$C = (X - e\bar{x}),\ \ \bar{x} = \tfrac{1}{N}\sum_{i=1:N} x_i,\ e = (1,...1)^T \in R^d .$$

This algorithm bisects data in Euclidean space by a hyperplane that passes through data centroid orthogonally to eigenvector with the largest singular value. The k-way splitting is also possible if the k largest singular values are considered. Bisecting is a good way to categorize documents and it results in a binary tree. When k-means (2-means) is used for bisecting, the dividing hyperplane is orthogonal to a line connecting two centroids. The comparative study of both approaches [Savaresi & Boley 2001] can be used for further references. Hierarchical divisive bisecting k-means was proven [Steinbach et al. 2000] to be preferable for document clustering.

While PDDP or 2-means are concerned with how to split a cluster, the problem of which cluster to split is also important. Casual strategies are: (1) split each node at a given level, (2) split the cluster with highest cardinality, and, (3) split the cluster with the largest intra-cluster variance. All three strategies have problems. For analysis regarding this subject and better alternatives, see [Savaresi et al. 2002].

2.4. Other Developments

Ward's method [Ward 1963] implements agglomerative clustering based not on linkage metric, but on an objective function used in k-means (sub-section *K-Means Methods*). The merger decision is viewed in terms of its effect on the objective function.

The popular hierarchical clustering algorithm for categorical data COBWEB [Fisher 1987] 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's dendrogram is called a *classification tree.* Each tree node C, a cluster, is associated with the conditional probabilities for categorical attribute-values pairs,

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

This easily can be recognized as a C-specific Naïve Bayes classifier. During the classification tree construction, every new point is descended along the tree and the tree is potentially updated (by an insert/split/merge/create operation). Decisions are based on an analysis of a ***category utility*** [Corter & Gluck 1992]

$$CU\{C_1,...,C_k\} = \left(\sum_{j=1:k} CU(C_j)\right)/k,$$
$$CU(C_j) = \sum_{l,p}(\Pr(x_l = v_{lp} \mid C_j)^2 - (\Pr(x_l = v_{lp})^2)$$

similar to GINI index. It rewards clusters C_j for increases in predictability of the categorical attribute values v_{lp}. Being incremental, COBWEB is fast with a complexity of $O(tN)$, though it depends non-linearly on tree characteristics packed into a constant *t*. There is the similar incremental hierarchical algorithm for all numerical attributes called CLASSIT [Gennari et al. 1989]. CLASSIT associates normal distributions with cluster nodes. Both algorithms can result in highly unbalanced trees.

Chiu et al. [2001] proposed another *conceptual* or *model-based* approach to hierarchical clustering. This development contains several different useful features, such as the extension of BIRCH-like preprocessing to categorical attributes, outliers handling, and a two-step strategy for monitoring the number of clusters including BIC (defined below). The model associated with a cluster covers both numerical and categorical attributes and constitutes a blend of Gaussian and multinomial models. Denote corresponding multivariate parameters by θ. With every cluster *C*, we associate a logarithm of its (classification) likelihood

$$l_C = \sum_{x_i \in C} \log p(x_i \mid \theta)$$

The algorithm uses *maximum likelihood estimates* for parameter θ. The distance between two clusters is defined (instead of linkage metric) as a decrease in log-likelihood

$$d(C_1, C_2) = l_{C_1} + l_{C_2} - l_{C_1 \cup C_2}$$

caused by merging of the two clusters under consideration. The agglomerative process continues until the stopping criterion is satisfied. As such, determination of the best *k* is automatic. This algorithm has the commercial implementation (in SPSS Clementine). The complexity of the algorithm is linear in *N* for the summarization phase.

Traditional hierarchical clustering is inflexible due to its greedy approach: after a merge or a split is selected it is not refined. Though COBWEB does reconsider its decisions, it is so inexpensive that the resulting classification tree can also have sub-par quality. Fisher [1996] studied iterative hierarchical cluster redistribution to improve once constructed dendrogram. Karypis et al. [1999b] also researched refinement for hierarchical clustering. In particular, they brought attention to a relation of such a refinement to a well-studied refinement of *k*-way graph partitioning [Kernighan & Lin 1970].

For references related to parallel implementation of hierarchical clustering see [Olson 1995].

3. Partitioning Relocation Clustering

In this section we survey data partitioning algorithms, which 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. Corresponding algorithms are described in the sub-section *Probabilistic Clustering.* 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* (see log-likelihood below).

Another approach starts with the definition of ***objective function*** depending on a partition. As we have seen (sub-section *Linkage Metrics*), pair-wise distances or similarities can be used to compute measures of iter- and 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 $k << N$). Depending on how representatives are constructed, iterative optimization partitioning algorithms are subdivided into ***k*-medoids** and ***k*-means** methods. *K*-medoid is the most appropriate data point within a cluster that represents it. Representation by *k*-medoids has two advantages. First, it presents no limitations on attributes types, and, second, the choice of medoids is dictated by the location of a predominant fraction of points inside a cluster and, therefore, it is lesser sensitive to the presence of outliers. In *k*-means case a cluster is represented by its centroid, which is a mean (usually weighted average) of points within a cluster. This works conveniently only with numerical attributes and can be negatively affected by a single outlier. On the other hand, centroids have the advantage of clear geometric and statistical meaning. The corresponding algorithms are reviewed in the sub-sections *K-Medoids Methods* and *K-Means Methods.*

3.1. Probabilistic Clustering

In the ***probabilistic*** approach, data is considered to be a sample independently drawn from a ***mixture model*** of several probability distributions [McLachlan & Basford 1988]. The main assumption is that data points are generated by, first, randomly picking a model *j* with probability $\tau_j, j = 1:k$, and, second, by drawing a point *x* from a corresponding distribution. The area around the mean of each (supposedly unimodal) 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 $\Pr(C_j \mid x)$ to *j*th model. The overall ***likelihood*** of the training data is its probability to be drawn from a given mixture model

$$L(X \mid C) = \prod\nolimits_{i=1:N} \sum\nolimits_{j=1:k} \tau_j \Pr(x_i \mid C_j)$$

Log-likelihood $\log(L(X \mid C))$ serves as an *objective function*, which gives rise to the ***Expectation-Maximization*** (EM) method. For a quick introduction to EM, see [Mitchell 1997]. Detailed descriptions and numerous references regarding this topic can be found in [Dempster et al. 1977; McLachlan & Krishnan 1997]. EM is a two-step iterative optimization. Step (E) estimates probabilities $\Pr(x \mid C_j)$, 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.

Restarting and other tricks are used to facilitate finding better local optimum. Moore [1999] suggested acceleration of EM method based on a special data index, KD-tree. 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) similar to BIRCH. 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

Because the mixture model has clear probabilistic foundation, the determination of the most suitable number of clusters *k* becomes a more tractable task. 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. See the sub-section "*How Many Clusters?*" for more details including terms MML and BIC used in the next paragraph.

The algorithm SNOB [Wallace & Dowe 1994] uses a mixture model in conjunction with the MML principle. Algorithm AUTOCLASS [Cheeseman & Stutz 1996] utilizes a mixture model and covers a broad variety of distributions, including Bernoulli, Poisson, Gaussian, and log-normal distributions. Beyond fitting a particular fixed mixture model, AUTOCLASS extends the search to different models and different *k*. To do this AUTOCLASS heavily relies on Bayesian methodology, in which a model complexity is reflected through certain coefficients (priors) in the expression for the likelihood previously dependent only on parameters' values. This algorithm has a history of industrial usage. The algorithm MCLUST [Fraley & Raftery 1999] is a software package (commercially linked with S-PLUS) for hierarchical, mixture model clustering, and discriminant analysis using BIC for estimation of goodness of fit. MCLUST uses Gaussian models with ellipsoids of different volumes, shapes, and orientations.

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) [Smyth 1999]. The dynamic data can consist of finite sequences subject to a first-order Markov model with a transition matrix dependent on a cluster. This framework also covers data objects consisting of *several* sequences, where number n_i of sequences per x_i is subject to geometric distribution [Cadez et al. 2000]. To emulate sessions of different lengths, finite-state Markov model (transitional probabilities between Web site pages) has to be augmented with a special "end" state. Cadez et al. [2001] used mixture model for customer profiling based on transactional information.

Model-based clustering is also used in a hierarchical framework: COBWEB, CLASSIT and development by Chiu et al. [2001] were already presented above. Another early example of conceptual clustering is algorithm CLUSTER/2 [Michalski & Stepp 1983].

3.2. *K*-Medoids Methods

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) [Kaufman & Rousseeuw 1990]. 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+2*k* 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.

Further progress is associated with Ng & Han [1994] who introduced the algorithm CLARANS (Clustering Large Applications based upon RANdomized Search) in the context of clustering in *spatial* databases. Authors considered a graph whose nodes are the sets of *k* medoids and an edge connects two nodes if they differ by exactly one medoid. While CLARA compares very few neighbors corresponding to a fixed small sample, CLARANS uses random search to generate neighbors by starting with an arbitrary node and randomly checking *maxneighbor* neighbors. If a neighbor represents a better partition, the process continues with this new node. Otherwise a local minimum is found, and the algorithm restarts until *numlocal* local minima are found (value *numlocal*=2 is recommended). The best node (set of medoids) is returned for the formation of a resulting partition. The complexity of CLARANS is $O(N^2)$ in terms of number of points. Ester et al. [1995] extended CLARANS to spatial VLDB. They used R*-trees [Beckmann 1990] to relax the original requirement that all the data resides in core memory, which allowed *focusing* exploration on the relevant part of the database that resides at a branch of the whole data tree.

3.3. *K*-Means Methods

The ***k-means*** algorithm [Hartigan 1975; 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

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

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 (see sub-section *Probabilistic Clustering*) [Mitchell 1997]. 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. For example, it coincides with pair-wise errors

$$E'(C) = \tfrac{1}{2} \sum_{j=1:k} \sum_{x_i, y_i \in C_j} \left\| x_i - y_i \right\|^2 ,$$

and with the difference between the total data variance and the inter-cluster variance. Therefore, the cluster separation is achieved simultaneously with the cluster tightness.

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's algorithm [Forgy 1965] and has many advantages:

- It easily works with any L_p-norm
- It allows straightforward parallelization [Dhillon & Modha 1999]
- It is insensitive with respect to data ordering.

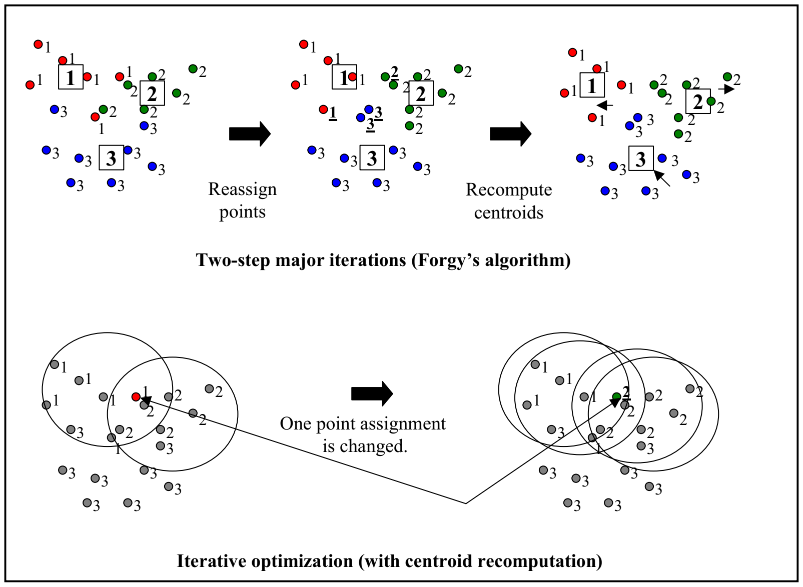
The second (classic in iterative optimization) version of *k*-means iterative optimization reassigns points based on more detailed analysis of effects on the objective function caused by moving a point from its current cluster to a potentially new one. If a move has a positive effect, the point is relocated 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. However, in L_2 case it is known [Duda & Hart 1973; Berkhin & Becher 2002] that all computations can be algebraically reduced to simply computing a single distance! Therefore, in this case both versions have the same computational complexity.

There is experimental evidence that compared with Forgy's algorithm, the second (classic) version frequently yields better results [Larsen & Aone 1999; Steinbach et al. 2000]. In particular, Dhillon et al. [2002] noticed that a Forgy's *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 3 illustrates both implementations.

Besides these two versions, there have been other attempts to find minimum of *k*-means objective function. For example, the early algorithm ISODATA [Ball & Hall 1965] used merges and splits of intermediate clusters.

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 cry 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's version, even empty)


Reassign points
Recompute centroids
Two-step major iterations (Forgy's algorithm)
One point assignment is changed.
Iterative optimization (with centroid recomputation)

A simple way to mitigate the affects of clusters initialization was suggested by Bradley & Fayyad [1998]. First, *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. Another interesting attempt [Babu & Murty 1993] is based on GA (see below). 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]. Zhang [2001] 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. The weights take into account how well a point fits into recipient clusters. This process involves so-called *harmonic means*.

We discuss scalability issues in the section *Scalability and VLDB Extensions*. For a comprehensive approach in relation to *k*-means see an excellent study [Bradley et al. 1998]. A generic method to achieve scalability is to preprocess or *squash* the data. Such preprocessing usually also takes care of outliers. Preprocessing has its drawbacks. It results in approximations that sometimes negatively affect final cluster quality. Pelleg & Moore [1999] suggested how to directly (without any squashing) accelerate *k*-means iterative process by utilizing KD-trees [Moore 1999]. The algorithm *X*-means [Pelleg &

Moore 2000] goes a step further: in addition to accelerating the iterative process it tries to incorporate a search for the best *k* in the process itself. While more comprehensive criteria discussed in the sub-section "*How Many Clusters?*" require running independent *k*-means and then comparing the results (costly experementation), *X*-means tries to split a part of already constructed cluster based on outcome of BIC criterion. This gives a much better initial guess for the next iteration and covers a user specified range of admissible *k*.

The tremendous popularity of *k*-means algorithm has brought to life many other extensions and modifications. Mahalanobis distance can be used to cover hyper-ellipsoidal clusters [Mao & Jain 1996]. Maximum of intra-cluster variances, instead of the sum, can serve as an objective function [Gonzales 1985]. Generalizations that incorporate categorical attributes are known. Sometimes the term *k-prototypes* is used in this context [Huang 1998]. Modifications which constructs clusters of balanced size are discussed in the sub-section *Constrained-Based Clustering*.

4. Density-Based Partitioning

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 4 illustrates some cluster shapes that present a problem for partitioning relocation clustering (e.g., *k*-means), but are handled properly by density-based algorithms. They also have good scalability. These outstanding properties are tempered with certain inconveniencies.

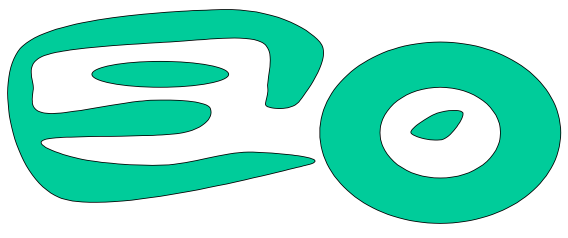


Figure 4. Irregular shapes difficult for *k*-means are

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. Another drawback is a lack of interpretability. An excellent introduction to density-based methods is contained in the textbook [Han & Kamber 2001].

Since density-based algorithms require a metric space, the natural setting for them is ***spatial*** data clustering [Han et al. 2001; Kolatch 2001]. To make computations feasible, some index of data is constructed (such as R*-tree). This is a topic of active research. Classic indices were effective only with reasonably low-dimensional data. The algorithm DENCLUE that, in fact, is a blend of a density-based clustering and a grid-based preprocessing is lesser affected by data dimensionality.

There are two major approaches for density-based methods. The first approach pins density to a training data point and is reviewed in the sub-section *Density-Based Connectivity*. Representative algorithms include DBSCAN, GDBSCAN, OPTICS, and

DBCLASD. 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.

4.1. Density-Based Connectivity

Crucial concepts of this section are ***density*** and ***connectivity*** both measured in terms of local distribution of nearest neighbors.

The algorithm DBSCAN (Density Based Spatial Clustering of Applications with Noise) [Ester et al. 1996] 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*** $N_\varepsilon(x) = \{y \in X \mid d(x,y) \le \varepsilon\}$ of the point *x*,
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).

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. Obviously, an effective computing of ε-neighborhoods presents a problem. However, in the case of low-dimensional ***spatial*** data, different effective indexation schemes exist (meaning $O(\log(N))$ rather than $O(N)$ fetches per search). DBSCAN relies on R*-tree indexation [Kriegel et al. 1990]. Therefore, on low-dimensional spatial data theoretical complexity of DBSCAN is $O(N\log(N))$. Experiments confirm slight super-linear runtime.

Notice that DBSCAN relies on Ů-neighborhoods and on frequency count within such neighborhoods to define a concept of a core object. Many spatial databases contain extended objects such as polygons instead of points. Any reflexive and symmetric predicate (for example, two polygons have a non-empty intersection) suffice to define a "neighborhood". Additional measures (as intensity of a point) can be used instead of a simple count as well. These two generalizations lead to the algorithm GDBSCAN [Sander et al. 1998], which uses the same two parameters as algorithm DBSCAN.

With regard to these two parameters ε and *MinPts*, there is no straightforward way to fit them to data. Moreover, different parts of data could require different parameters – the problem discussed earlier in conjunction with CHAMELEON. The algorithm OPTICS (Ordering Points To Identify the Clustering Structure) [Ankerst et al. 1999] adjusts DBSCAN to this challenge. It builds an augmented ordering of data which is consistent with DBSCAN, but goes a step further: keeping the same two parameters ε, *MinPts*, OPTICS covers a spectrum of all different $\varepsilon' \le \varepsilon$. The constructed ordering can be used automatically or interactively. With each point, OPTICS stores only two additional fields, the so-called core- and reachability-distances. For example, the core-distance is the distance to *MinPts*' nearest neighbor when it does not exceeds ε, or undefined otherwise. Experimentally, OPTICS exhibits runtime roughly equal to 1.6 of DBSCAN runtime.

While OPTICS can be considered as a DBSCAN extension in direction of different local densities, 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. Such *nonparametric* approach is implemented in the algorithm DBCLASD (Distribution Based Clustering of Large Spatial Databases) [Xu et al. 1998]. Assuming that points inside each cluster are uniformly distributed which may or may not be realistic, DBSCLAD defines a cluster as a non-empty arbitrary shape subset in X that has the expected distribution of distance to the nearest neighbor with a required confidence, and is the *maximal connected* set with this quality. This algorithm handles spatial data (minefield example is used). χ^2-test is used to check distribution requirement (standard consequence is a requirement for each cluster to have at least 30 points). Regarding connectivity, DBCLASD relies on *grid-based* approach to generate cluster-approximating polygons. The algorithm contains devices for handling real databases with noise and implements *incremental* unsupervised learning. Two venues are used. First, assignments are not final: points can change cluster membership. Second, certain points (noise) are not assigned, but are tried later. Therefore, once incrementally fetched points can be revisited internally. DBCLASD is known to run faster than CLARANS by a factor of 60 on some examples. In comparison with much more efficient DBSCAN, it can be 2-3 times slower. However, DBCLASD requires no user input, while empirical search for appropriate parameter requires several DBSCAN runs. In addition, DBCLASD discovers clusters of different densities.

4.2. Density Functions

Hinneburg & Keim [1998] shifted the emphasis from computing densities pinned to data points to computing density functions defined over the underlying attribute space. They proposed the algorithm DENCLUE (DENsity-based CLUstEring). Along with DBCLASD, it has a firm mathematical foundation. DENCLUE uses a ***density function***

$$f^{D}(x) = \text{—}_{y\in D} f(x,y)$$

that is the superposition of several ***influence functions***. When the f-term depends on $x - y$, the formula can be recognized as a convolution with a kernel. Examples include a *square wave function* $f(x,y)=\theta(\|x-y\|/\sigma)$ equal to 1, if distance between x and y is less than or equal to σ, and a Gaussian influence function $f(x,y)=e^{-\|x-y\|^{2}/2\sigma^{2}}$. This provides a high level of generality: the first example leads to DBSCAN, the second one to k-means clusters! Both examples depend on parameter σ. Restricting the summation to $D=\{y:\|x-y\|<k\sigma\}\subset X$ enables a practical implementation. DENCLUE concentrates on local maxima of density functions called *density-attractors* and uses a flavor of gradient hill-climbing technique for finding them. In addition to *center-defined* clusters,

arbitrary-shape clusters are defined as continuations along sequences of points whose local densities are no less than prescribed threshold ξ. The algorithm is stable with respect to outliers and authors show how to choose parameters σ and ξ. DENCLUE scales well, since at its initial stage it builds a *map* of hyper-rectangle cubes with edge length 2σ. For this reason, the algorithm can be classified as a *grid-based* method. Applications include high dimensional multimedia and molecular biology data. While no clustering algorithm could have less than $O(N)$ complexity, the runtime of DENCLUE scales with N sub-linearly! The explanation is that though all the points are fetched, the bulk of analysis (in clustering stage) involves only points in highly populated areas.

5. Grid-Based Methods

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 points' membership in segments resulted from space partitioning, while space partitioning is based on grid-characteristics accumulated from input data. One advantage of this indirect handling (data grid-data space-partitioning data-partitioning) is that accumulation of grid-data makes grid-based clustering techniques independent of data ordering. In contrast, relocation methods and all incremental algorithms are very sensitive with respect to data ordering. While density-based partitioning methods work best with numerical attributes, grid-based methods work with attributes of different types.

To some extent, 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, are presented in the section *Clustering High Dimensional Data*. In this section we survey algorithms that use grid-based technique as their major principle instrument.

BANG-clustering [Schikuta & Erhart 1997] improves the similar hierarchical algorithm GRIDCLUST [Schikuta 1996]. Grid-based segments are used to summarize data. The segments are stored in a special BANG-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.

The algorithm STING (STatistical INformation Grid-based method) [Wang et al. 97] works with numerical attributes (spatial data) and is designed to facilitate "region oriented" queries. In doing so, STING constructs data summaries in a way similar to BIRCH. It, however, assembles statistics in a hierarchical tree of nodes that are grid-cells. Figure 5 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). Only distribution type presents a problem – χ^2-test is used after bottom cell distribution types are handpicked. When the cell-tree is constructed (in $O(N)$ time), certain cells are identified and connected in clusters similar to DBSCAN. If the number of leaves is *K*, the cluster construction phase depends on *K* and not on *N*. This algorithm has a simple structure suitable for parallelization and allows for multi-resolution, though defining appropriate granularity is not straightforward. STING has been further enhanced to algorithm STING+ [Wang et al. 1999] that targets dynamically evolving spatial databases, and uses similar hierarchical cell organization as its predecessor. In addition, STING+ enables *active* data mining.

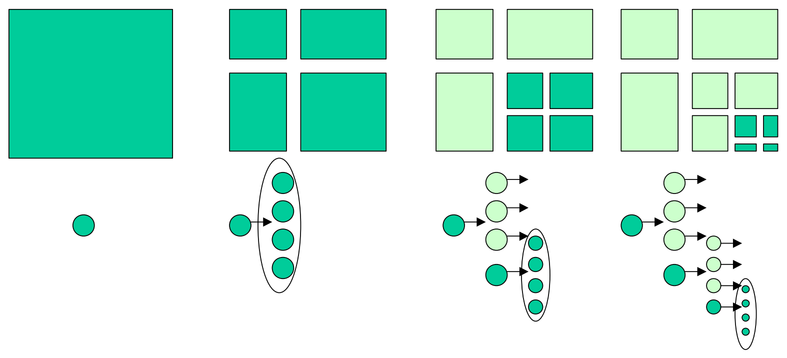


Figure 5. Cell generation and tree construction in STING.

To do so, it supports user defined *trigger conditions* (e.g., there is a region where at least 10 cellular phones are in use per square mile with total area of at least 10 square miles, or usage drops by 20% in a described region). The related measures, sub-triggers, are stored and updated over the hierarchical cell tree. They are suspended until the trigger *fires* with user-defined action. Four types of conditions are supported: absolute and relative conditions on regions (a set of adjacent cells), absolute and relative conditions on certain attributes.

The algorithm WaveCluster [Sheikholeslami et al. 1998] works with numerical attributes and has an advanced multi-resolution. It is also known for other outstanding properties:

- High quality of clusters
- Ability to work well in relatively high dimensional spatial data
- Successful handling of outliers
- $O(N)$ complexity

WaveCluster is based on ideas of signal processing. It applies wavelet transforms to filter the data. Notice that high-frequency parts of a signal correspond to boundaries, while low frequency high amplitude parts of a signal correspond to clusters' interiors. Wavelet transform provides us with useful filters. For example, hat-shape filter forces dense areas to serve as attractors and simultaneously suppresses lesser dense boundary areas. After getting back from signal to attribute space this makes clusters more sharp and eliminates outliers. WaveCluster goes in stages. It:

1) Bins every dimension and assigns points to corresponding units
2) Applies discrete Wavelet transform to so accumulated units
3) Finds connected components (clusters) in a transformed attribute space (corresponding to a certain level of resolution)
4) Assigns points

The algorithm's complexity is $O(N)$ for low dimensions, but exponentially grows with the dimension.

The hierarchy of grids allows definition of the *Hausdorff Fractal Dimension* (HFD) [Schalkoff 1991]. HFD of a set is the negative slope of a log-log plot of the number of cells $Cell(r)$ (occupied by a set) as a function of a grid size *r*. A fast algorithm (*box counting*) to compute HFD was introduced in [Liebovitch & Toth 1989]. The concept of HFD is fundamental to the FC (Fractal Clustering) algorithm [Barbara & Chen 2000] for numeric attributes, which works with several layers of grids (cardinality of each dimension is increased 4 times with each next layer). Although only occupied cells are kept to save memory, memory usage is still a significant problem. FC starts with initializing of *k* clusters. Initialization threshold and a data sample are used at this stage to come up with the appropriate *k*. Then FC scans full data incrementally. It tries to add an incoming point to each cluster that results in certain increase of HFD. If the smallest increase exceeds a threshold τ, a point is declared an outlier; otherwise a point is assigned so that HFD would be minimally impacted. The FC algorithm has few appealing properties:

Á Incremental structure (batches of data are fetched into core memory)
Á Suspendable nature always ready for on-line assignments
Á Ability to discover clusters of irregular shapes
Á $O(N)$ complexity

It also has a few problems:

Á Data order dependency
Á Strong dependency on clusters initialization
Á Dependency on parameters (threshold used in initialization, and τ)

6. Co-Occurrence of Categorical Data

In this section we talk about categorical data, which 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. This is why similarity (sub-section *Proximity Measures*) between them is usually measured by Jaccard coefficient $sim(T_1,T_2)=\left|T_1 \; 1 \; T_2\right| / \left|T_1 \; 8 \; T_2\right|$. 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.

The algorithm ROCK (Robust Clustering algorithm for Categorical Data) [Guha et al. 1999] deals with categorical data and has many common features with the algorithm CURE (section *Hierarchical Clustering*): (1) it is a hierarchical clustering, (2) agglomeration continues until specified number k of clusters is constructed, and (3) it uses data sampling in the same way as CURE does. ROCK defines a neighbor of a point x as a point y such that $sim(x,y) \geq \theta$ for some threshold θ, and proceeds to a definition of links *link*(*x,y*) between two points *x, y* equal to number of their common neighbors. Clusters consist of points with a high degree of connectivity – pair-wise points inside a cluster have on average a high number of links. ROCK utilizes the objective function

$$E = \text{—}_{j=1:k} \left|C_j\right| \cdot \text{—}_{x,y \in C_j} link(x,y) / \left|C_j\right|^{1+2f(\theta)},$$

where $f(\theta)$ is a data dependent function. E represents specifically normalized intra-connectivity measure.

To put this formula into perspective, notice that linkage metrics normalize the aggregate measures by the number of edges. For example, the average link metric is the sum of distances between each point C_i and each point in C_j divided by the factor $L = \left|C_i\right| \cdot \left|C_j\right|$. The value L can be rationalized on a more general level. If the expected number of edges per cluster is $\left|C\right|^{\beta}, \beta \in [1,2]$, then the aggregate inter-cluster similarity has to be normalized by the factor $\left(\left|C_i\right| + \left|C_j\right|\right)^{\beta} - \left|C_i\right|^{\beta} - \left|C_j\right|^{\beta}$ representing the number of inter-cluster edges. The average link normalization factor L corresponds to $\beta = 2$, the highest expected connectivity indeed. The ROCK objective function uses the same idea, but fits it with parameters. Whether a model fits particular data is an open question. Frequently, different regions of data have different properties, and therefore, global fit is impossible. ROCK relies on an input parameter θ and on a function $f(\theta)$ that have to fit data. It has a complexity of $O(c_m N_{sample} + N_{sample}^2 \log(N_{sample}))$, where coefficient c_m is a product of average and maximum number of neighbors.

The algorithm SNN (Shared Nearest Neighbors) [Ertoz et al. 2002] blends a density-based approach with the idea of ROCK. SNN sparsifies similarity matrix (therefore, unfortunately resulting in $O(N^2)$ complexity) by only keeping K-nearest neighbors, and thus derives the total strength of links for each x.

For this matter, the idea to use shared nearest neighbors in clustering was suggested by Jarvis & Patrick [1973] long ago. See also [Gowda & Krishna 1978].

The algorithm CACTUS (Clustering Categorical Data Using Summaries) [Ganti et al. 1999a] looks for hyper-rectangular clusters (called *interval regions*) in point-by-attribute data with categorical attributes. In our terminology such clusters are segments. CACTUS is based on the idea of co-occurrence for attribute-value pairs. (Implicitly uniform distribution within the range of values for each attribute is assumed). Two values *a, b* of two different attributes are *strongly connected* if the number of data points having both *a* and *b* is larger than the frequency expected under independency assumption by a user-defined margin α. This definition is extended to subsets *A*, *B* of two different attributes (each value pair $a \in A, b \in B$ has to be strongly connected), to segments (each 2D projection is strongly connected), and to the similarity of pair of values of a single attribute via connectivity to other attributes. The cluster is defined as the maximal strongly connected segment having at least α times more elements than expected from the segment under attributes independency assumption. CACTUS uses data summaries to generate all the strongly connected and similar attribute value pairs. As a second step, a heuristic is used to generate maximum segments. The complexity of the summarization phase is $O(cN)$, where the constant *c* depends on whether all the attribute-value summaries fit in memory (one data scan), or not (multiple data scans).

The situation with clustering transactional data becomes more aggravated when size of item universe grows. Here we have a classic case of low separation in high-dimensional space (section *Clustering High Dimensional Data*). With categorical data, the idea of auxiliary clustering of items, or more generally of categorical attribute values, gained popularity. It is very similar to the idea of co-clustering (sub-section *Co-Clustering*). This, formally speaking, preprocessing step becomes the major concern, while the following data clustering remains a lesser issue.

We start with the development of Han et al. [1997] that exemplifies this approach. After items are clustered (major step), a very simple method to cluster transactions themselves is used: each transaction *T* is assigned to a cluster C_j of items having most in common with *T*, as defined by a function $\left|T \cap C_j\right| / \left|C_j\right|$. Other choices come to mind, but again the primary objective is to find item groups. To achieve this *association rules* and *hyper-graph* machineries are used. First, frequent item-sets are generated from transactional data. A hyper-graph $H = (V, E)$ can be associated with item universe, so that vertices *V* are items. In a common graph, pairs of vertices are connected by edges, but in a hyper-graph several vertices are connected by hyper-edges. Hyper-edge $e \in E$ in *H* corresponds to a frequent item-set $\{v_1, ..., v_s\}$ and has a *weight* equal to an average of confidences among all association rules involving this item-set. A solution to the problem of *k*-way partitioning of a hyper-graph *H* is provided by algorithm HMETIS [Karypis et al. 1997].

The algorithm STIRR (Sieving Through Iterated Reinfircement) [Gibson et al. 1998] deals with co-occurrence for *d*-dimensional categorical objects, *tuples*. Extension to transactional data is obvious. It uses beautiful technique from functional analysis. Define *configurations* as weights $w = \{w_v\}$ over all different values *v* for all *d* attributes. Consider, for example, a value *v* of the first attribute. The tuples $x = (v, u_1, ..., u_{d-1})$

containing *v* result in a weight update $w_v' = \text{—}_{x, v \in x} z_x$, where terms $z_x = \Phi(w_{u_1}, ..., w_{u_{d-1}})$ depend on a *combining operator* Φ. An example of a combining operator is $\Phi(w_1, ..., w_{d-1}) = w_1 + ... + w_{d-1}$. So the weight is redistributed among different values. The major iteration scans the data X and results in the propagation of weights between different nodes $w_{new} = f(w)$ equal to a described update followed by the normalization of weights among the values of each attribute. Function *f* can be considered as a ***dynamic system*** (non-linear, if Φ is non-linear). STIRR relies on a deep analogy with the *spectral graph partitioning*. For linear dynamic system defined over the graph, a re-orthogonalization Gram-Schmidt process can be engaged to compute its eigenvectors that introduces negative weights. The few first non-principal eigenvectors (non-principle *basins*) define graph partitioning corresponding to positive/negative weights. The process works like this: few weights (configurations) $w^q = \{w_v^q\}$ are initialized. A major iteration updates them, $w_{new}^q = f(w^q)$, and new weights are re-orthogonalized. The process continues until *fixed point* of a dynamic system is achieved. Non-principle basins are analyzed. In STIRR a dynamic system instead of association rules formalizes co-occurrence. Additional references related to spectral graph partitioning can be found in [Gibson et al. 1998]. As the *convergence* of the process can cause a problem, the further progress is related to the modification of the dynamic system that guarantees it [Zhang et al. 2000].

7. Other Clustering Techniques

A number of other clustering algorithms have been developed. Some deal with the specific application requirements. *Constraint-based clustering* belongs to this category. Others have theoretical significance or are mostly used in other than data mining applications. We briefly discuss these developments in the sub-sections *Relation to Supervised Learning*, *Gradient Descent and ANN*, and *Evolutionary Methods*. Finally, in the sub-section *Other Developments* we very briefly mention developments that simply do not fit well in our classification.

7.1. Constraint-Based Clustering

In real-world applications customers are rarely interested in unconstrained solutions. Clusters are frequently subjected to some problem-specific limitations that make them suitable for particular business actions. Building of so conditioned cluster partitions is the subject of active research; for example, see survey [Han et al. 2001].

The framework for the constrained-based clustering is introduced in [Tung et al. 2001]. The taxonomy of clustering constraints includes constraints on individual objects (e.g., customer who recently purchased) 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 constrains 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 (i.e. frequent customers) in each cluster. Iterative optimization used in partitioning clustering relies on moving objects to their nearest cluster representatives. This may violate such constraint. A methodology of how to resolve this conflict is developed in [Tung et al. 2001].

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. The modification of the *k*-means objective function and of *k*-means updates that incorporate lower limits on cluster volumes is suggested in [Bradley et al. 2000]. This includes soft assignments of data points with coefficients subject to linear program requirements. Banerjee & Ghosh [2002] presented another modification to *k*-means algorithm. Their objective function corresponds to an isotropic Gaussian mixture with widths inversely proportional to numbers of points in the clusters. The result is the *frequency sensitive k*-means. Still another approach to building balanced clusters is to convert a task into a graph-partitioning problem [Strehl & Ghosh 2000].

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 [Tung et al. 2001] deals with this problem. It is best illustrated by the figure 6, showing the difference in constructing three clusters in absence of obstacle (left) and in presence of a river with a bridge (right).

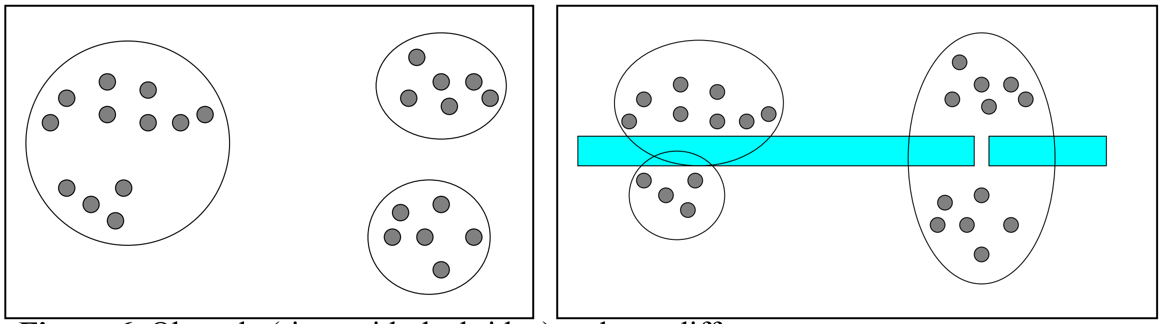


Figure 6. Obstacle (river with the bridge) makes a difference.

7.2. Relation to Supervised Learning

Both Forgy's *k*-means implementation and EM algorithms 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 [Kalton et al. 2001]. The model update is considered as the training of a predictive classifier based on current assignments serving as the target attribute values supervising the learning. Points' reassignments correspond to the forecasting using the recently trained classifier.

Liu et al. [2000] suggested another elegant connection to supervised learning. They 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.

7.3. Gradient Descent and Artificial Neural Networks

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:

$$E'(C) = \underline{\quad}_{i=1:N} \qquad {}_{j=1:k} \left\| x_i - c_j \right\|^2 \omega_{ij}^2$$

Exponential probabilities ɤ are defined based on Gaussian models. This makes the objective function differentiable with respect to means and allows application of general ***gradient decent*** method. Marroquin & Girosi [1993] presented a detailed introduction to this subject in the context of *vector quantization.* Gradient decent method in *k*-means is known as LKMA (Local K-Means Algorithm). At each iteration *t*, it modifies means c_j^t

$$c_j^{t+1} = c_j^t + a_t \underline{\quad}_{i=1:N} (x_i - c_j^t) w_{ij}^2 \quad \text{or} \quad c_j^{t+1} = c_j^t + a_t (x_i - c_j^t) w_{ij}^2$$

in the direction of gradient decent. In the second case one *x* is selected randomly. Scalars a_t satisfy certain monotone asymptotic behavior and converge to zero, coefficients *w* are defined trough ɤ [Bottou & Bengio 1995]. 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. Bibliography related to this dynamic field can be found in the monograph [Kohonen 2001]. We will not elaborate here about SOM except for two important features: (1) SOM uses incremental approach – 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. For further discussion see [Jain & Mao 1996].

7.4. Evolutionary Methods

Substantial information on ***simulated annealing*** in the context of partitioning (main focus) or hierarchical clustering is accumulated, including the algorithm SINICC (SImulation of Near-optima for Internal Clustering Criteria) [Brown & Huntley 1991]. The perturbation operator used in 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). SINICC also tries to address the interesting problem of choosing the most appropriate objective function. It has a real application – surveillance monitoring of ground-based entities by airborne and ground-based sensors. Similar to simulating annealing is the so-called ***tabu search*** [Al-Sultan 1995].

Genetic Algorithms (GA) [Goldberg 1989] are also used in cluster analysis. An example is the GGA (Genetically Guided Algorithm) for fuzzy and hard *k*-means [Hall et al.

1999]. This article can be used for further references. Sarafis et al. [2002] applied GA 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, a far cry from density-based methods. GA were also applied to clustering of categorical data using so-called generalized entropy to define the dissimilarity [Cristofor and Simovici 2002].

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 [Lee & Antonsson 2000] 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.

7.5. Other Developments

There are other developments that in terms of their performance qualify for data mining.

For 2D spatial data (for example, GIS database) the algorithm AMOEBA [Estivill-Castro & Lee 2000] uses Delaunay diagram (the dual of Voronoi diagram) to represent data proximity and has $O(N\log(N))$ complexity.

Harel & Koren [2001] suggested an approach related to agglomerative hierarchical graph methodology that they showed to successfully find local clusters in 2D. As above, consider a connectivity graph $G=(X,E)$. Using Delaunay diagram or keeping with any point only its *K*-nearest neighbors sparsifies the graph. The method relies on *random walk* to find separating edges *F* so that clusters become connected components of $G=(V,E-F)$.

8. Scalability and VLDB Extensions

Clustering algorithms face problems of scalability both in terms of computing time and memory requirements. In data mining reasonable runtime and ability to use certain limited core memory become especially important. There have been many interesting attempts to extend clustering to very large databases (VLDB), which can be divided into:

- Incremental mining,
- Data squashing,
- Reliable sampling.

The algorithm DIGNET [Thomopoulos et al. 1995; Wann & Thomopoulos 1997] (compare with "*the leader*" clustering algorithm in [Hartigan 1975]) is an example of incremental unsupervised learning. This means that it handles one data point at a time, and then discards it. DIGNET uses *k*-means cluster representation without iterative optimization. Centroids are instead *pushed* or *pulled* depending on whether they loose or win each next coming point. Such on-line clustering needs only one data pass, but strongly depends on data ordering, and it can result in sub-quality clusters. However, it handles outliers, clusters can be dynamically born or discarded, and the training process is resumable. This makes it very appealing for dynamic VLDB. Some further tools can be used to improve obtained clusters.

Data squashing techniques scan data to compute certain data summaries (*sufficient statistics*) [DuMouchel et al. 1999]. The obtained summaries are then used instead of the original data for further clustering. The pivotal role here belongs to the algorithm BIRCH (Balanced Iterative Reduction and Clustering using Hierarchies) [Zhang et al. 1996; Zhang et al. 1997]. This work had a significant impact on overall direction of scalability research in clustering. BIRCH creates a height-balanced tree of nodes that summarize data by accumulating its zero, first, and second moments. A node, called ***Cluster Feature*** (CF), is a tight small cluster of numerical data. The construction of a tree residing in core memory is controlled by some parameters. A new data point descends along the tree to the closest CF leaf. If it fits the leaf well and if the leaf is not overcrowded, CF statistics are incremented for all nodes from the leaf to the root. Otherwise a new CF is constructed. Since the maximum number of children per node (*branching factor*) is limited, one or several splits can happen. When the tree reaches the assigned memory size, it is rebuilt and a threshold controlling whether a new point is assigned to a leaf or starts a new leaf is updated to a coarser one. The outliers are sent to disk, and refitted gradually during tree rebuilds. The final leaves constitute input to any algorithm of choice. The fact that a CF-tree is balanced allows the log-efficient search. BIRCH depends on parameters that control CF tree construction (branching factor, maximum of points per leaf, leaf threshold), and it also depends on data ordering. When the tree is constructed (one data pass), it can be additionally condensed in the optional 2nd phase to further fit desired input cardinality of post-processing clustering algorithm. Next, in the 3rd phase a global clustering of CF (considered as individual points) happens. Finally, certain irregularities (for example, identical points getting to different CFs) can be resolved in an optional 4th phase. It makes one or more passes through data reassigning points to best possible clusters, as *k*-means does. The overall complexity is $O(N)$. Summarization phase of BIRCH was extended to mixed numerical and categorical attributes [Chiu et al. 2001].

A full interface between VLDB and relocation clustering (as *k*-means) includes following requirements [Bradley et al. 1998]. Algorithm has to:

- Take one (or less – early termination) data scan
- Provide on-line solution: some solution in-progress should always be available
- Be suspendable, stoppable, resumable
- Be able to incorporate additional data incrementally
- Be able to work in prescribed memory buffer
- Utilize different scanning modes (sequential, index, sample)
- Be able to operate in forward-only cursor over a view of database

The article suggests data compression that accumulates sufficient statistics like BIRCH does, but makes it in phases. Points that are compressed over the primary stage are discarded. They can be attributed to their clusters with very high confidence even if other points would shift. The rest is taken care of in the secondary phase, which tries to find dense subsets by *k*-means method with higher than requested *k*. Violators of this stage are still kept in retained set (RT) of singletons to be analyzed later.

BIRCH-like preprocessing substantially relies on vector-space operations. Meanwhile, in many applications, objects (for example, strings) belong to a metric space. In other words, all we can do with data points is to compute distances between them. Ganti et al. [1999b] proposed BIRCH-type data squashing BUBBLE for VLDB in metric spaces. Each leaf of the BUBBLE-tree is characterized by:

1) Number of its points
2) Medoid (called *clustroid*) that delivers a minimum to an error – a squared distance between it and all other points belonging to the leaf
3) Radius equal to the square root of an average error per a point

The problem to overcome is how to insert new points in the absence of a vector structure. BUBBLE uses a heuristic that relates to a distance preserving embedding of leaf points into a low-dimensional Euclidean vector space. Such embedding is known as isometric map in geometry and as multidimensional scaling in statistics. Certain analogy can also be made with embeddings used in support vector machines. While Euclidean distance (used in BIRCH) is cheap, the computation of a distance in a metric space (for example, edit distance for strings) can be expensive. Meanwhile, every insertion requires to compute distances to all the nodes descending to a leaf. The similar algorithm BUBBLE-FM handles this difficulty. It relaxes the computations by using *approximate* isometric embedding. This is possible due to the algorithm FastMap [Faloutsos & Lin 1995].

In the context of hierarchical density-based clustering in VLDB, Breunig et al. [2001] analyzed data reduction techniques such as sampling and BIRCH summarization, and noticed that they result in deterioration of cluster quality. To cure this, they approached data reduction through accumulation of *data bubbles* that are summaries of local information about distances and nearest neighbors. A data bubble contains an *extent*, the distance from a bubble's representative to most points in *X*, and the array of distances to each of *MinPts* nearest neighbors. Data bubbles are then used in conjunction with the algorithm OPTICS (see sub-section *Density-Based Connectivity*).

Grid-methods also generate data summaries, though their summarization phase relates to units and segments and not to CFs. Therefore, they are scalable.

Many algorithms use old-fashioned sampling with or without rigorous statistical reasoning. It is especially handy for different initializations as in CLARANS (sub-section *K-Medoids Methods*), Fractal Clustering (section *Grid-Based Methods*), or *k*-means [Bradley & Fayyad 98]. Notice that when clusters are constructed using whatever sample, assigning the whole data to the most appropriate clusters minimally adds the term $O(N)$ to the overall complexity.

Sampling has got a new life with the adoption by the data mining community of a special uniform check to control its adequacy. This check is based on *Hoffding* or *Chernoff* bounds [Motwani & Raghavan 1995] and says that, independent of the distribution of a real-valued random variable $Y,\ 0 \le Y \le R,$ the average of *n* independent observations lies within $\mathring{U}$ of the actual mean

$$\left|\overline{Y} - {}^{1}\!/_{n} \text{—}_{j=1:n} Y_j\right| \le \varepsilon$$

with probability 1-ŭ as soon as

$$\varepsilon = \sqrt{R^2 \ln(1/\delta)/2n} \ .$$

These bounds were used in the clustering algorithm CURE [Guha et al. 1998] and in the development of scalable decision trees in predictive mining [Hulten et al. 2001]. In the context of balanced clustering, a statistical estimation of a sample size is provided in [Banerjee & Ghosh 2002]. Due to their nonparametric nature, the bounds have a ubiquitous significance.

9. Clustering High Dimensional Data

The objects in 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 Naïve 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 [Beyer et al. 1999]. 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. For interesting insights into complications of high dimensional data, see [Aggarwal et al. 2000].

Basic exploratory data analysis (attribute selection) preceding the clustering step is the best way to address the first problem of irrelevant attributes. We consider this topic in the section *General Algorithmic Issues*. Below we present some techniques dealing with a situation when the number of already pre-selected attributes *d* is still high.

In the sub-section *Dimensionality Reduction* we talk briefly about traditional methods of dimensionality reduction. In the sub-section *Subspace Clustering* we review algorithms that try to circumvent high dimensionality by building clusters in appropriate subspaces of original attribute space. Such approach has a perfect sense in applications, since it is only better if we can describe data by fewer attributes. Still another approach that divides attributes into similar groups and comes up with good new derived attributes representing each group is discussed in the sub-section *Co-Clustering*.

Important source of high dimensional categorical data comes from transactional (market basket) analysis. Idea to group items very similar to co-clustering has already been discussed in the section *Co-Occurrence of Categorical Data*.

9.1. Dimensionality Reduction

When talking about high dimensionality, *how high is high*?

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.

How large is the gap? If we are dealing with a retail application, 52-weeks sales volumes represent a typical set of features, which is a special example of more general class of time series data. In customer profiling dozens of generalized item categories plus basic demographics result in at the least 50-100 attributes. Web clustering based on site contents results in 200-1000 attributes (pages/contents) for modest Web sites. Biology and genomic data can have dimensions that easily surpass 2000-5000 attributes. Finally, text mining and information retrieval also deal with many thousands of attributes (words or *index terms*). So, the gap *is significant*.

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. For sales profiles and OLAP-type data, roll-ups as sums or averages over time intervals (e.g., monthly volumes) can be used. Due to a fine seasonality of sales such brute force approaches rarely work. In multivariate statistics *principal components analysis* (PCA) is popular [Mardia et al. 1980; Joliffe 1986], but this approach is problematic since it leads to clusters with poor interpretability. 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 [Keogh et al. 2001].

Domain decomposition divides the data into subsets, *canopies*, [McCallum et al. 2000] 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.

9.2. Subspace Clustering

Some algorithms better adjust to high dimensions. For example, the algorithm CACTUS (section *Co-Occurrence of Categorical Data*) adjusts well since it defines a cluster only in terms of a cluster's 2D projections. In this section we cover techniques that are specifically designed to work with high dimensional data.

The algorithm CLIQUE (Clustering In QUEst) [Agrawal et al. 1998] for numerical attributes is fundamental in subspace clustering. It marries the ideas of:

- Density-based clustering

Á Grid-based clustering

Á Induction through dimensions similar to *Apriori* algorithm in association rules

Á MDL principle to select appropriate subspaces

Á Interpretability of clusters in terms of DNF representation

CLIQUE starts with the definition of a unit – elementary rectangular cell in a subspace. Only units whose densities exceed a threshold τ are retained. A bottom-up approach of finding such units is applied. First, 1-dimensional units are found by dividing intervals in ȝ equal-width bins (a grid). Both parameters τ and ȝ are the algorithm's inputs. 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 (Apriori-reasoning). All the subspaces are sorted by their coverage and lesser-covered subspaces are pruned. A cut point is selected based on MDL principle. A cluster is defined as a maximal set of connected dense units. It is represented by a DNF expression that is associated with a finite set of maximal segments (called *regions*) whose union is equal to a cluster. 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. If q is a highest subspace dimension selected, the complexity of dense units generations is $O(const^q + qN)$. Identification of clusters is a quadratic task in terms of units.

The algorithm ENCLUS (ENtropy-based CLUStering) [Cheng et al. 1999] follows in the footsteps of CLIQUE, but uses a different criterion for subspace selection. The criterion is derived from entropy related considerations: the subspace spanned by attributes $A_1,...,A_q$ with entropy $H(A_1,...,A_q)$ smaller than a threshold ɤ is considered good for clustering. Any subspace of a good subspace is also good, since

$$H(A_1,...,A_{q-1}) = H(A_1,...,A_q) - H(A_q \mid A_1,...,A_{q-1}) \le H(A_1,...,A_q) < \omega .$$

Low entropy subspace corresponds to a skewed distribution of unit densities. The computational costs of ENCLUS are high.

The algorithm MAFIA (Merging of Adaptive Finite Intervals) [Goil et al. 1999; Nagesh et al. 2001] 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 Ŭ, called cluster dominance factor, to select bins that are Ŭ-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 Ŭ (default value 1.5 works fine) presents no problem in comparison with global density threshold used in CLIQUE. Reporting for a range of Ŭ in a single run is supported. If q is a highest dimensionality of CDU, the complexity is $O(const^q + qN)$.

The algorithm OPTIGRID [Hinneburg & Keim 1999] uses data partitioning based on divisive recursion by multi-dimensional grids. Authors present a very good introduction into the effects of high-dimension geometry. Familiar concepts, as for example, uniform distribution, become blurred for large *d*. OPTIGRID uses density estimations in the same way the algorithm DENCLUE (by the same authors) does. It primarily focuses on separation of clusters by (hyper) planes that are not necessarily axes parallel. To find such planes consider a set of *contracting* linear projectors (functionals) $P_1,...,P_k, \|P_j\| \le 1$ of the attribute space *A* at a 1D line. For a density kernel of the form $K(x-y)$ (a tool of trade in DENCLUE) and a contracting projection, density induced after projection is more concentrated. A cutting plane is chosen so that it goes through the point of minimal density and discriminates two significantly dense half-spaces. Several cutting planes are chosen, and recursion continues with each subset of data.

The algorithm PROCLUS (PROjected CLUstering) [Aggarwal et al. 1999a] associates with a subset *C* a low-dimensional subspace such that the projection of *C* into the subspace is a tight cluster. The subset – subspace pair when exists constitutes a *projected cluster*. The number *k* of clusters and the average subspace dimension *l* are user inputs. The iterative phase of the algorithm deals with finding *k* good medoids, each associated with its subspace. A sample of data is used in a greedy hill-climbing technique. Manhattan distance divided by the subspace dimension is a useful normalized metric for searching among different dimensions. An additional data pass follows after iterative stage is finished to refine clusters including subspaces associated with the medoids.

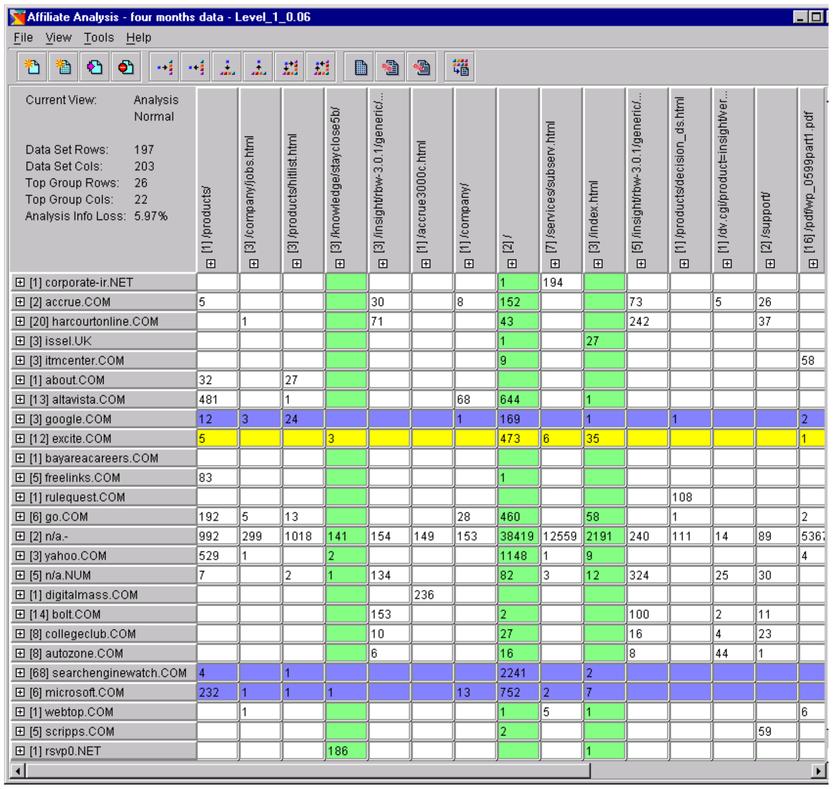
The algorithm ORCLUS (ORiented projected CLUSter generation) [Aggarwal & Yu 2000] uses a similar approach of projected clustering, but employs non-axes parallel subspaces of high dimensional space. In fact, both developments address a more generic issue: even in a low dimensional space, different portions of data could exhibit clustering tendency in different subspaces (consider several non-parallel non-intersecting cylinders in 3D space). If this is the case, any attribute selection is doomed. ORCLUS has a *k*-means-like transparent model that defines clusters as sets of points (partition) that have low sum-of-squares of errors (*energy*) in a certain subspace. More specifically, for $x \in C$, and directions $E = \{e_1,...,e_l\}$ (specific to *C*), the projection is defined as $\{x \cdot e_1,...,x \cdot e_l\}$. The projection only *decreases* energy. SVD diagonalization can be used to find directions (eigenvectors) corresponding to the lowest *l* eigenvalues of the covariance matrix. In reality, the algorithm results in *X* partitioning (the outliers excluded) into *k* clusters C_j together with their subspace directions E_j. The algorithm builds more than *k* clusters, with larger than *l*-dimensional *E* gradually fitting the optimal subspace and requested *k*. Though suggestion of picking a good parameter *l* is provided, uniform *l* is a certain liability.

Any other comparison aside, projected clusters provide data partitioning, while cluster systems resulted from CLIQUE overlap.

9.3. Co-Clustering

In OLAP attribute roll-ups can be viewed as representatives of the attribute groups. 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 [Anderberg 1973; Hartigan 1975] and is known under the names *simultaneous clustering*, *bi-dimensional clustering*, *block clustering*, *conjugate clustering*, *distributional clustering*, and *information bottleneck method*. The use of duality for analysis of categorical data (dual or multidimensional scaling) also has a long history in statistics [Nishisato 1980]. The similar approach of building groups of item was presented in the section *Co-Occurrence of Categorical Data*. In this section we turn to numerical attributes. Assume that the matrix *X* has non-negative elements. In this context it is known as *incidence*, *relational*, *frequency*, *contingency* matrix. In applications it can reflect intensity of a gene response in a tissue sample, frequency of visitation activity of a page, or the amount of a sale in a store per item category.



Current View: Analysis Normal

Data Set Rows: 197
Data Set Cols: 203
Top Group Rows: 26
Top Group Cols: 22
Analysis Info Loss: 5.97%

	[1] /products/	[3] /company/jobs.html	[3] /products/hitlist.html	[3] /knowledge/stayclose5b/	[3] /insight/rbw-3.0.1/generic/...	[1] /accrue3000c.html	[1] /company/	[2] /	[7] /services/subserv.html	[3] /index.html	[5] /insight/rbw-3.0.1/generic/...	[1] /products/decision_ds.html	[1] /dv.cgi/product=insight/ver...	[2] /support/	[16] /pdf/wp_0599part1.pdf
[1] corporate-ir.NET								1	194						
[2] accrue.COM	5				30		8	152			73		5	26	
[20] harcourtonline.COM		1			71			43			242			37	
[3] issel.UK								1		27					
[3] itmcenter.COM								9							58
[1] about.COM	32		27												
[13] altavista.COM	481		1				68	644		1					
[3] google.COM	12	3	24				1	169		1		1			2
[12] excite.COM	5			3				473	6	35					1
[1] bayareacareers.COM															
[5] freelinks.COM	83							1							
[1] rulequest.COM												108			
[6] go.COM	192	5	13				28	460		58		1			2
[2] n/a.-	992	299	1018	141	154	149	153	38419	12559	2191	240	111	14	89	536
[3] yahoo.COM	529	1		2				1148	1	9					4
[5] n/a.NUM	7		2	1	134			82	3	12	324		25	30	
[1] digitalmass.COM						236									
[14] bolt.COM					153			2			100		2	11	
[8] collegeclub.COM					10			27			16		4	23	
[8] autozone.COM					6			16			8		44	1	
[68] searchenginewatch.COM	4		1					2241		2					
[6] microsoft.COM	232	1	1	1			13	752	2	7					
[1] webtop.COM		1						1	5	1					6
[5] scripps.COM								2						59	
[1] rsvp0.NET				186						1					

Figure 7. Learning referring traffic on a Web site.

Govaert [1995] researched simultaneous block clustering of the rows and columns of contingency tables. He also reviewed an earlier work on the subject. An advanced algebraic approach to co-clustering based on bi-partite graphs and their minimal cuts in conjunction with text mining was proposed in [Dhillon 2001]. This paper contains an excellent introduction in relations between simultaneous clustering and graph partitioning, as well as in connection with SVD. A simple algorithm Ping-Pong [Oyanagi et al. 2001] was suggested to find populated areas in a sparse binary matrix. It redistributes influence of columns on rows and vice versa (compare with algorithm STIRR above) by transversal connection through matrix elements and provides an example of other than co-clustering, but a related development.

A series of publications deal 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. This process can be generalized. The grouping simplifies the original matrix *X* to the compressed form $\overline{X}$. Attribute clustering is productive when it minimally impacts information reduction $R = I(X) - I(\overline{X})$, where $I(X)$ is mutual information contained in *X* [Cover & Thomas 1990]. Such attribute grouping is intimately related to Naïve Bayes classification in predictive mining [Baker & McCallum 1998].

The outlined technique is very much relevant to grouping words in text mining. In this context the technique was explored under the name *information bottleneck method* [Tishby et al. 1999]. It was used to facilitate agglomerative co-clustering of words in document clustering [Slonim & Tishby 2000] and classification [Slonim & Tishby 2001].

Berkhin & Becher [2002] showed deep algebraic connection of distributional clustering to *k*-means. They used *k*-means adaptation to KL-distance as a major iterative step in the algorithm SIMPLIFYRELATION that gradually co-clusters points and attributes. This development has industrial application in Web analysis. Figure 7 shows how an original incidence matrix of Web site traffic between 197 referrers (rows) and 203 Web site pages (columns) is clustered into 26x22 matrix with 6% information loss. While KL-distance is not actually a distance, since it is not symmetric, it can be symmetrized to the so-called Jensen-Shanon divergence. Dhillon et al. [2002] used Jensen-Shanon divergence to cluster words in *k*-means fashion in text classification. Besides text and Web mining, the idea of co-clustering finds its way into other applications, as for example, clustering of gene microarrays [Busygin et al. 2002].

10. General Algorithmic Issues

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 ubiquitous that they are not even specific to unsupervised learning and can be considered as a part of overall data mining framework. Others are resolved in certain algorithms we presented. In fact, many algorithms were specifically designed for this reason. Now we overview common issues, and necessarily our coverage will be very fragmented.

Scalability for VLDB and high dimensional clustering were 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

10.1. 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 and conceptual algorithms, as COBWEB, 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. The review [Jain et al. 1999] extensively covers cluster validation, while a discussion of cluster visualization and related references can be found in [Kandogan 2001].

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. Simple criterion ***Rand*** serves this purpose [Rand 1971]. Computation of a Rand index (defined below) involves pairs of points that were assigned to the same and to the different clusters in each of two partitions. Hence it has $O(N^2)$ complexity and is not always feasible. ***Conditional entropy*** of a known label *s* given clustering partitioning [Cover & Thomas 1990]

$$H(S\,|\,J) = -\sum_j p_j \sum_s p_{s|j} \log(p_{s|j})$$

is another measure used. Here $p_j, p_{s|j}$ are probabilities of *j* cluster, and conditional probabilities of *s* given *j*. It has $O(N)$ complexity. Other measures are also used, for example, the ***F-measure*** [Larsen & Aone 1999].

10.2. How Many Clusters?

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*. Some industrial applications (SAS, NeoVista) report pseudo ***F-statistic***. This only makes sense for *k*-means clustering in context of ANOVA. Earlier publications on the subject analyzed cluster separation for different *k* [Engleman & Hartigan 1969; Milligan & Cooper 1985]. For instance, a distance between any two centroids (medoids) normalized by corresponding cluster's radii (standard deviations) and averaged (with cluster weights) is a reasonable choice of *coefficient of separation*. This coefficient has a very low $O(k^2)$ complexity. Another popular choice for separation measure is a ***Silhouette coefficient*** [Kaufman & Rousseeuw 1990]. For example, Silhouette coefficient is used in conjunction with CLARANS in [Ng & Han 1994]. It has $O(N^2)$ complexity. 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*

$$a(x) = \frac{1}{|C|-1} \sum_{y \in C, y \neq x} d(x,y),\ b(x) = \min_{G \neq C} \frac{1}{|G|} \sum_{y \in G} d(x,y)$$

The Silhouette coefficient of *x* is $s(x) = (b(x) - a(x)) / \max\{a(x), b(x)\}$, values close to +1 corresponding to a perfect and values below 0 to a bad clustering choice. The overall average of individual *s*(*x*) gives a good indication of cluster system appropriateness.

Still another approach to separation is to employ possible soft (or fuzzy) assignments. It has an intermediate *O*(*kN*) complexity. 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*** [Bezdek 1981] is equal to the sum of squares of the weights

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

(compare with GINI index). 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 strong probabilistic foundation of the *mixture model*, discussed in sub-section *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 (for example, AUTOCLASS [Cheeseman & Stutz 1995]) 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 including:

- *Minimum Description Length* (MDL) criterion [Rissanen 1978; Schwarz 1978; Rissanen 1989]
- *Minimum Message Length* (MML) criterion [Wallace & Freeman 87; Wallace & Dowe 94]
- *Bayesian Information Criterion* (BIC) [Schwarz 1978; Fraley & Raftery 1998]
- *Akaike's Information Criterion* (AIC) [Bozdogan 1983]
- *Non-coding Information Theoretic Criterion* (ICOMP) [Bozdogan 1994]

Á *Approximate Weight of Evidence* (AWE) criterion [Banfield & Raftery 1993]
Á *Bayes Factors* [Kass & Raftery 1995]

All these criteria are expressed through combinations of log-likelihood L, number of clusters k, number of parameters per cluster, total number of estimated parameters p, and different flavors of Fisher information matrix. For example,

$$MDL(k) = -L + p/2 - \log(p), \; k_{best} = \min\arg MDL(k),$$

$$BIC(k) = L - {}^{p}\!/_{2} \log(n), \; k_{best} = \max\arg BIC(k).$$

Further details and discussion can be found in [Bock 1996; Oliver et al. 1996; Fraley & Raftery 1998]. Few examples: MCLUST and X-means use BIC criterion, SNOB uses MML criterion, CLIQUE and evolutionary approach to k determination [Lee & Antonsson 2000] use MDL. Significant expertise is developed in estimation of goodness of fit based on the criteria above. For example, different ranges of BIC are suggested for weak, positive, and very strong evidence in favor of one clustering system versus another [Fraley & Raftery 1999]. Smyth [1998] suggested a *likelihood cross-validation* technique for determination the best k.

10.3. Data Preparation

Irrelevant attributes make chances of a successful clustering futile, because they negatively affect proximity measures and eliminate *clustering tendency*. Therefore, sound exploratory data analysis (EDA) is essential. An overall framework for EDA can be found in [Becher et al. 2000]. As its first order of business, EDA eliminates inappropriate attributes and reduces the cardinality of the retained categorical attributes. Next it provides attribute selection. Different attribute selection methods exist. Inconsistency rates are utilized in [Liu & Setiono 1996]. The idea of a Markov blanket is used in [Koller & Sahami 1996]. While there are others methods (for example, [Jebara & Jaakkola 2000]), most are used primarily for predictive and not descriptive mining and thus do not address general-purpose attribute selection for clustering. We conclude that cluster-specific attribute selection yet to be invented.

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 (especially, standardization) is widespread and allows constructing clusters of better shapes. To some extent *attribute scaling* can be viewed as the continuation of attribute selection.

In real-life applications it is crucial to handle attributes of different nature. For example, images are characterized by color, texture, shape, and location, resulting in four attribute subsets. Modha & Spangler [2002] suggested a very interesting approach for attribute scaling that pursues the objective of clustering in each attribute subset by computing weights (a simplex) that minimize the product of intra-cluster to inter-cluster ratios for the attribute subset projections (called *generalized Fisher ratio*).

In many applications data points have different significance. For example, in assortment planning, stores can be characterized by the profiles of sales of particular items in percentage. However, the overall sale volume gives additional weight to larger stores. Partitioning relocation algorithms easily handle weighted data, because centroids become centers of weights instead of means. The described practice is called *case scaling*.

Some algorithms depend on the effectiveness of data access. To facilitate this process data indices are constructed. Examples include the extension of the algorithm CLARANS [Ester et al. 1995] and the algorithm DBSCAN [Ester et al. 1996]. Index structures used for spatial data, include KD-trees [Friedman et al. 1977], R-trees [Guttman 1984], R*-trees [Kriegel et al. 1990]. A blend of attribute transformations (DFT, Polynomials) and indexing technique is presented in [Keogh et al. 2001a]. Other indices and numerous generalizations exist [Beckmann 1990; Faloutsos et al. 1994; Berchtold et al. 98; Wang et al. 1998; Karypis & Han 2000; Keogh et al. 2001b]. The major application of such data structures is in nearest neighbors search.

Preprocessing of multimedia data that is based on its embedding in Euclidean space (the algorithm FastMap) [Faloutsos & Lin 1995].

A fairly diverse range of preprocessing is used for variable length sequences. Instead of handling them directly (as in the sub-section *Probabilistic Clustering*), a fixed set of features to represent variable length sequences can be derived [Guralnik & Karypis 2001; Manilla & Rusakov 2001].

10.4. Proximity Measures

Both hierarchical and partitioning methods use different distances and similarity measures [Jain & Dubes 1988]. The usual L_p distance

$$d(x,y) = \|x-y\|_p, \ \|z\|_p = \left(\sum_{j=1:d} |z_j|^p\right)^{1/p}, \ \|z\| = \|z\|_2$$

is used for numerical data, $1 \le p < \infty$, 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. Manhattan distance corresponds to p=1. 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,

$$d(x,y) = \arccos\left(x^T y / \|x\| \cdot \|y\|\right).$$

It is used, in specific tools, as DIGNET (section Scalability and VLDB Extensions), and in particular applications, as text mining. All above distances assume attributes independence (diagonal covariance matrix S). Mahanalabonis distance

$$d(x,y) = \sqrt{(x-y)^T S^{-1} (x-y)}$$

[Mardia et al. 1980] is used in algorithms, as ORCLUS [Aggarwal & Yu 2000], that do not make this assumption.

Formula $s(x,y)=1/(1+d(x,y))$ defines similarity for numerical attributes. Other choices are *cosine, Dice coefficients* and distance *exponent*

$$s_{\cos}(x,y)=x^T y/\|x\|\|y\|,\ \ s_{Dice}=2x^T y/\left(\|x\|^2+\|y\|^2\right),\ s_{\exp}=\exp\left(-\|x-y\|^\alpha\right).$$

Now we shift our attention to categorical data. A number of similarity measures exist for categorical attributes [Dubes 1993; Everitt 1993]. Assuming binary attributes with values $\alpha,\beta=\pm$, let $d_{\alpha\beta}$ be a number of attributes having outcomes α in *x* and β in *y*. Then the *Rand* and *Jaccard* (also known as *Tanimoto*) indices *R*, *J* are equal to

$$R(x,y)=(d_{++}+d_{--})/(d_{++}+d_{+-}+d_{-+}+d_{--}),\ J(x,y)=(d_{++})/(d_{++}+d_{+-}+d_{-+})$$

Notice that Jaccard index treats positive and negative values asymmetrically, which makes it the measure of choice for transactional data, + meaning that an item is present. It is simply the fraction of common items of two transactions relative to the number of items in both transactions. It is also used in collaborative filtering, sequence analysis, text mining, and pattern recognition. *Extended Jaccard* coefficient is advocated in [Ghosh 2002]. For construction of similarity measures for market basket analysis see [Aggarwal et al. 1999b; Baeza-Yates 1992]. Similarity can also be constructed axiomatically based on information-theoretical considerations [Lin 1998]. The last two references contain material related to strings similarity (biology is one application). For strings over the same alphabet, *edit distance* is a frequent choice [Arslan & Egecioglu 2000]. It is based on the length of a sequence of transformations (such as insertion, deletion, transposition, etc.) that are necessary to transform one string into another. A classic Hamming distance [Cover & Thomas 1990] is also used. Further references can be found in the review [Jain et al. 1999]. Historically textual mining was a source of major inspirations for a concept of similarity [Resnik 1995].

Proximity measures between two clusters that can be derived from proximities between pairs of their points were discussed in the sub-section *Linkage Metrics*.

10.5. Handling Outliers

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 legitimate 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, so as not to pollute factual clusters.

There are multiple ways of how descriptive learning handles outliers. If a summarization or 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. Algorithms in the section *Scalability and VLDB Extensions* provide further examples. The algorithm BIRCH [Zhang et al. 1996; Zhang et al. 1997] revisits outliers during the major CF tree rebuilds, but in general handles them separately. This approach is shared by other similar systems [Chiu et al. 2001]. The framework of [Bradley et al. 1998] utilizes a multiphase approach to outliers.

Certain algorithms have specific features for outliers handling. The algorithm CURE [Guha et al. 1998] uses shrinkage of cluster's representives 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. The algorithm DBCSAN [Ester et al. 1996] uses concepts of internal (core), boundary (reachable), and outliers (non-reachable) points. The algorithm CLIQUE [Agrawal et al. 1998] goes a step further: it eliminates subspaces with low coverage. The algorithm WaveCluster [Sheikholeslami et al. 1998] is known to handle outliers very well through its filtering DSP foundation. The algorithm ORCLUS [Aggarwal & Yu 2000] produces a partition plus a set of outliers.

What is an outlier? Statistics defines an outlier as a point that does not fit a *probability distribution*. This approach has the problem with *discordance testing* for unknown multivariate distribution. Classic data analysis utilizes a concept of *depth* [Tukey 1977] and defines an outlier as a point of low depth. This concept becomes computationally infeasible for $d > 3$. Data mining is gradually develops its own definitions.

Consider two positive parameters *Ů*, *ŭ*. A point can be declared an outlier if its *Ů*-neighborhood contains less than 1-*ŭ* fraction of a whole dataset *X* [Knorr & Ng 1998]. Ramaswamy et al. [2000] noticed that this definition can be improved by eliminating parameter *Ů*. Rank all the points by their distance to the *K*-nearest neighbor and define the *ŭ* fraction of points with highest ranks as outliers. Both definitions are uniformly global. How to describe local outliers? In essence, different subsets of data 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 [Breunig et al. 2000]. The definition is based on the distance to the *k*-nearest neighbor. Knorr et al. [2001] addressed a related problem of how to eliminate outliers in order to compute an appropriate covariance matrix that describes a given locality. To do so, they utilized *Donoho-Stahel estimator* in two-dimensional space.

Crude handling of outliers works surprisingly well in many applications, because the simple truth is that many applications are concerned with systematic patterns. An example is customer segmentation with an objective of a direct mail campaign. On the other hand, philosophically outlier is a non-typical leftover after a regular clustering and, as such, it can easily attain a prominent significance. Therefore, in addition to eliminating negative effects of outliers on clusters construction, there is a separate reason driving interest in outlier detection. The reason is that in some applications, the outlier is the commodity of trade. This is so in medical diagnostics, fraud detection, network security, anomaly detection, and computer immunology. Some connections and further references can be found in [Forrest et al. 1997; Lee & Stolfo 1998; Ghosh et al. 1999]. In CRM, E-commerce, Web-site analytics outliers relate to a concept of *interesting* and *unexpected* [Piatetsky-Shapiro & Matheus 1994; Silberschatz & Tuzhilin 1996; Padmanabhan & Tuzhilin 1999; Padmanabhan & Tuzhilin 2000]. Most of the research in these applications is not directly related to clustering (but to pruning association rules).

Acknowledgements

Cooperation with Jonathan Becher was essential for the appearance of this text. It resulted in numerous discussions and various improvements. I am grateful to Jiawei Han for reading the text and his thoughtful remarks concerning the presentation of the material.

I am very much thankful to Sue Krouscup for her help with text preparation.

References

AGGARWAL, C.C., HINNEBURG, A., and KEIM, D.A. 2000. On the surprising behavior of distance metrics in high dimensional space. *IBM Research report*, RC 21739.

AGGARWAL, C.C., PROCOPIUC, C., WOLF, J.L., YU, P.S., and PARK, J.S. 1999a. Fast algorithms for projected clustering. In *Proceedings of the ACM SIGMOD Conference*, 61-72, Philadelphia, PA.

AGGARWAL, C.C., WOLF, J.L., and YU, P.S. 1999b. A new method for similarity indexing of market basket data. In *Proceedings of the ACM SIGMOD Conference*, 407-418, Philadelphia, PA.

AGGARWAL, C.C. and YU, P.S. 2000. Finding generalized projected clusters in high dimensional spaces. *Sigmod Record*, 29, 2, 70-92.

AGRAWAL, R., FALOUTSOS, C., and SWAMI, A. 1993. Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, Chicago, IL.

AGRAWAL, R., GEHRKE, J., GUNOPULOS, D., and RAGHAVAN, P. 1998. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD Conference*, 94-105, Seattle, WA.

AL-SULTAN, K. 1995. A Tabu search approach to the clustering problem. *Pattern Recognition*, 28, 9, 1443-1451.

ANDERBERG, M. 1973. *Cluster Analysis and Applications*. Academic Press, New York.

ANKERST, M., BREUNIG, M., KRIEGEL, H.-P., and SANDER, J. 1999. OPTICS: Ordering points to identify clustering structure. In *Proceedings of the ACM SIGMOD Conference*, 49-60, Philadelphia, PA.

ARABIE, P. and HUBERT, L.J. 1996. An overview of combinatorial data analysis, in: Arabie, P., Hubert, L.J., and Soete, G.D. (Eds.) *Clustering and Classification*, 5-63, World Scientific Publishing Co., NJ.

ARSLAN, A.N. and EGECIOGLU, O. 2000. Efficient algorithms for normalized edit distance. *Journal of Discrete Algorithms*, 1, 1.

BABU, G.P. and MURTY, M.N. 1993. A near-optimal initial seed value selection in K-means algorithm using a genetic algorithm. *Pattern Recogn. Lett.* 14, 10, 763-169.

BABU, G.P. and MARTY, M.N. 1994. Clustering with evolution strategies. *Pattern Recognition*, 27, 2, 321-329.

BAEZA-YATES, R. 1992. Introduction to data structures and algorithms related to information retrieval. In Frakes, W.B. and Baeza-Yates, R. (Eds.) *Information Retrieval, Data Structures and Algorithms*, 13-27, Prentice-Hall.

BAKER, L.D. and MCCALLUM, A. K. 1998. Distributional clustering of words for text classification. In *Proceedings of the 21st ACM SIGIR Conference*, Melbourne, Australia.

BALL, G. and HALL, D. 1965. ISODATA, a novel method of data analysis and classification. *Technical Report AD-699616*, SRI, Stanford, CA.

BANERJEE, A. and GHOSH, J. 2002. On scaling up balanced clustering algorithms. In *Proceedings of the 2nd SIAM ICDM*, 333-349, Arlington, VA.

BANFIELD, J. and RAFTERY, A. 1993. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49, 803-821.

BARBARA, D. and CHEN, P. 2000. Using the fractal dimension to cluster datasets. In *Proceedings of the 6th ACM SIGKDD*, 260-264, Boston, MA.

BECHER, J., BERKHIN, P., and FREEMAN, E. 2000. Automating exploratory data analysis for efficient data mining. In *Proceedings of the 6th ACM SIGKDD*, 424-429, Boston, MA.

BECKMANN, N., KRIEGEL, H-P., SCHNEIDER, R., and SEEGER, B. 1990. The R*-tree: An efficient access method for points and rectangles. In *Proceedings of International Conference on Geographic Information Systems*, Ottawa, Canada.

BERCHTOLD, S., BÖHM, C., and KRIEGEL, H-P. 1998. The Pyramid-technique: towards breaking the curse of dimensionality. In *Proceedings of the ACM SIGMOD Conference*, 142-153, Seattle, WA.

BERKHIN, P. and BECHER, J. 2002. Learning Simple Relations: Theory and Applications. In *Proceedings of the 2nd SIAM ICDM*, 420-436, Arlington, VA.

BEN-DOR, A. and YAKHINI, Z. 1999. Clustering gene expression patterns. In *Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB 99)*, 11-14, Lyon, France.

BERRY, M.W. and BROWNE, M. 1999. *Understanding Search Engines: Mathematical Modeling and Text Retrieval*. SIAM.

BERRY, M., DUMAIS, S., LANDAUER, T., and O'BRIEN, G. 1995. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37, 4, 573-595.

BOTTOU, L. and BENGIO, Y. 1995. Convergence properties of the K-means algorithms. In Tesauro, G. and Touretzky, D. (Eds.) *Advances in Neural Information Processing Systems 7*, 585-592, The MIT Press, Cambridge, MA.

BEYER, K., GOLDSTEIN, J., RAMAKRISHNAN, R., and SHAFT, U. 1999. When is nearest neighbor meaningful? In *Proceedings of the* 7th ICDT, Jerusalem, Israel.

BEZDEK, D. 1981. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, NY.

BOCK, H.H. 1996. Probability models in partitional cluster analysis. In Ferligoj, A. and Kramberger, A. (Eds.) *Developments in Data Analysis*, 3-25, Slovenia.

BOLEY, D.L. 1998. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2, 4, 325-344.

BOZDOGAN, H. 1983. Determining the number of component clusters in the standard multivariate normal mixture model using model-selection criteria. *TR UIC/DQM/A83-1*, Quantitative Methods Department, University of Illinois, Chicago, IL.

BOZDOGAN, H. 1994. Mixture-model cluster analysis using model selection criteria and a new information measure of complexity. In *Proceedings of the 1st US/Japan Conference on the Frontiers of Statistical Modeling: An Informational Approach*, 69-113, Dordrecht, Netherlands.

BRADLEY, P. S., BENNETT, K. P., and DEMIRIZ, A. 2000. Constrained k-means clustering. *Technical Report MSR-TR-2000-65*. Microsoft Research, Redmond, WA.

BRADLEY, P. and FAYYAD, U. 1998. Refining initial points for k-means clustering. In *Proceedings of the 15th ICML*, 91-99, Madison, WI.

BRADLEY, P., FAYYAD, U., and REINA, C. 1998. Scaling clustering algorithms to large databases. In *Proceedings of the 4th ACM SIGKDD*, 9-15, New York, NY.

BREUNIG, M., KRIEGEL, H-P., KROGER, P., and SANDER, J. 2001. Data Bubbles: quality preserving performance boosting for hierarchical clustering. In *Proceedings of the ACM SIGMOD Conference*, Santa Barbara, CA.

BREUNIG, M.M., KRIEGEL, H.-P., NG, R.T., and SANDER, J. 2000. LOF: identifying density-based local outliers. In *Proceedings of the ACM SIGMOD Conference*, 93-104, Dallas, TX.

BROWN, D. and HUNTLEY, C. 1991. A practical application of simulated annealing to clustering. *Technical report IPC-TR-91-003*, University of Virginia.

BUSYGIN, S., JACOBSEN, G., and KRÄMER, E. 2002. Double conjugated clustering applied to leukemia microarray data, *2nd SIAM ICDM, Workshop on clustering high dimensional data*, Arlington, VA.

CADEZ, I., GAFFNEY, S., and SMYTH, P. 2000. A general probabilistic framework for clustering individuals. *Technical Report UCI-ICS 00-09*.

CADEZ, I., SMYTH, P., and MANNILA, H. 2001. Probabilistic modeling of transactional data with applications to profiling, Visualization, and Prediction, In *Proceedings of the 7th ACM SIGKDD*, 37-46, San Francisco, CA.

Carpenter, G.A., Grossberg, S., and Rosen, D.B. 1991. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks, 4, 759-771.

CHEESEMAN, P. and STUTZ, J. 1996. Bayesian Classification (AutoClass): Theory and Results. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy , R. (Eds.) *Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press.

CHENG, C., FU, A., and ZHANG, Y. 1999. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the 5th ACM SIGKDD*, 84-93, San Diego, CA.

CHIU, T., FANG, D., CHEN, J., and Wang, Y. 2001. A Robust and scalable clustering algorithm for mixed type attributes in large database environments. In *Proceedings of the 7th ACM SIGKDD*, 263-268, San Francisco, CA.

COOLEY, R., MOBASHER, B., and SRIVASTAVA, J. 1999. Data preparation for mining world wide web browsing. *Journal of Knowledge Information Systems*, 1, 1, 5-32.

CORTER, J. and GLUCK, M. 1992. Explaining basic categories: feature predictability and information. *Psychological Bulletin*, 111, 291-303.

COVER, T.M. and THOMAS, J.A. 1990. *Elements of Information Theory*. John Wiley & Sons, New York, NY.

CRISTOFOR, D. and SIMOVICI, D.A. 2002. An information-theoretical approach to clustering categorical databases using genetic algorithms. *2nd SIAM ICDM, Workshop on clustering high dimensional data*, Arlington, VA.

CUTTING, D., KARGER, D., PEDERSEN, J., and TUKEY, J. 1992. Scatter/gather: a cluster-based approach to browsing large document collection. In *Proceedings of the 15th ACM SIGIR Conference*, 318-329, Copenhagen, Denmark.

DANIEL, C. and WOOD, F.C. 1980. *Fitting Equations To Data: Computer Analysis of Multifactor Data*. John Wiley & Sons, New York, NY.

DAY, W. and EDELSBRUNNER, H. 1984. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1, 7, 7-24.

DEFAYS, D. 1977. An efficient algorithm for a complete link method. *The Computer Journal*, 20, 364-366.

DEMPSTER, A., LAIRD, N., and RUBIN, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B, 39, 1, 1-38.

DHILLON, I. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th ACM SIGKDD*, 269-274, San Francisco, CA.

DHILLON, I., FAN, J., and GUAN, Y. 2001. Efficient clustering of very large document collections. In Grossman, R.L., Kamath, C., Kegelmeyer, P., Kumar, V., and Namburu, R.R. (Eds.) *Data Mining for Scientific and Engineering Applications*, Kluwer Academic Publishers.

DHILLON, I., GUAN, Y., and KOGAN, J. 2002. Refining clusters in high dimensional data. *2nd SIAM ICDM, Workshop on clustering high dimensional data*, Arlington, VA.

DHILLON, I., MALLELA, S., and KUMAR, R. 2002. Enhanced Word Clustering for Hierarchical Text Classification, In *Proceedings of the 8th ACM SIGKDD*, 191-200, Edmonton, Canada.

DHILLON, I. and MODHA, D. 1999. A data clustering algorithm on distributed memory multiprocessors. *5th ACM SIGKDD*, *Large-scale Parallel KDD Systems Workshop*, 245-260, San Diego, CA.

DUBES, R.C. 1993. Cluster Analysis and Related Issues. In Chen, C.H., Pau, L.F., and Wang, P.S. (Eds.) *Handbook of Pattern Recognition and Computer Vision*, 3-32, World Scientific Publishing Co., River Edge, NJ.

DUDA, R. and HART, P. 1973. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, NY.

DUMOUCHEL, W., VOLINSKY, C., JOHNSON, T., CORTES, C., and PREGIBON, D. 1999. Squashing flat files flatter. In *Proceedings of the 5th ACM SIGKDD*, 6-15, San Diego, CA.

ENGLEMAN, L. and HARTIGAN, J. 1969. Percentage points of a test for clusters. *Journal of the American Statistical Association*, 64, 1647-1648.

ERTOZ, L., STEINBACH, M., and KUMAR, V. 2002. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data, *Technical Report*.

ESTER M., FROMMELT A., KRIEGEL H.-P., and SANDER J. 2000. Spatial data mining: database primitives, algorithms and efficient DBMS support. *Data Mining and Knowledge Discovery*, Kluwer Academic Publishers, 4, 2-3, 193-216.

ESTER, M., KRIEGEL, H-P., SANDER, J. and XU, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd ACM SIGKDD*, 226-231, Portland, Oregon.

ESTER, M., KRIEGEL, H-P., and XU, X. 1995. A database interface for clustering in large spatial databases. In *Proceedings of the 1st ACM SIGKDD*, 94-99, Montreal, Canada.

ESTIVILL-CASTRO, V. and LEE, I. 2000. AMOEBA: Hierarchical Clustering Based on Spatial Proximity Using Delaunay Diagram. In *Proceedings of the 9th International Symposium on Spatial Data Handling*. Beijing, China.

EVERITT, B. 1993. *Cluster Analysis* (3rd ed.). Edward Arnold, London, UK.

FALOUTSOS, C. and LIN, K. 1995. Fastmap: A fast algorithm for indexing, data mining and visualization of traditional and multimedia. In *Proceedings of the ACM SIGMOD Conference*, 163-174, San Jose, CA.

FALOUTSOS, C., RANGANATHAN, M., and MANOLOPOULOS, Y. 1994. Fast subsequence matching in time-series databases. In *Proceedings of the ACM SIGMOD Conference*, 419-429, Minneapolis, MN.

FASULO, D. 1999. An analysis of recent work on clustering algorithms. *Technical Report UW-CSE01 -03-02, University of Washington*.

FISHER, D. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2, 139-172.

FISHER, D. 1996. Iterative optimization and simplification of hierarchical clustering. *Journal of Artificial Intelligence Research*, 4, 147-179.

FORGY, E. 1965. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21, 768-780.

FORREST, S., HOFMEYR, S.A., and SOMAYAJI, A. 1997. Computer immunology. *Communications of the ACM*, 40, 88-96.

FOSS, A., WANG, W., and ZAANE, O. 2001. A non-parametric approach to Web log analysis. *1st SIAM ICDM, Workshop on Web Mining,* 41-50, Chicago, IL.

FRALEY, C. and RAFTERY, A. 1999. MCLUST: Software for model-based cluster and discriminant analysis, *Tech Report 342*, Dept. Statistics, Univ. of Washington.

FRALEY, C. and RAFTERY, A. How many clusters? 1998. Which clustering method? Answers via model-based cluster analysis. *The Computer Journal*, 41, 8, 578-588.

FRIEDMAN, J.H., BENTLEY, J.L., and FINKEL, R.A. 1977. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3, 3, 209-226.

FUKUNAGA, K. 1990. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA.

GANTI, V., GEHRKE, J. and RAMAKRISHNAN, R. 1999a. CACTUS-Clustering Categorical Data Using Summaries. In *Proceedings of the 5th ACM SIGKDD*, 73-83, San Diego, CA.

GANTI, V., RAMAKRISHNAN, R., GEHRKE, J., POWELL, A., and FRENCH, J. 1999b. Clustering large datasets in arbitrary metric spaces. In *Proceedings of the 15th ICDE*, 502-511, Sydney, Australia.

GENNARI, J., LANGLEY, P., and FISHER, D. 1989. Models of incremental concept formation. *Artificial Intelligence*, 40, 11-61.

GERSHO, A. and GRAY, R. M. 1992. *Vector Quantization and Signal Compression. Communications and Information Theory*. Kluwer Academic Publishers, Norwell, MA.

GHOSH, J., 2002. Scalable Clustering Methods for Data Mining. In Nong Ye (Ed.) *Handbook of Data Mining*, Lawrence Erlbaum, to appear.

GHOSH, A.K., SCHWARTZBARD, A., and SCHATZ. M. 1999. Learning program behavior profiles for intrusion detection. In *Proceedings of the SANS Conference and Workshop on Intrusion Detection and Response*, San Francisco, CA.

GIBSON, D., KLEINBERG, J., and RAGHAVAN, P. 1998. Clustering categorical data: An approach based on dynamic systems. In *Proceedings of the 24th International Conference on Very Large Databases*, 311-323, New York, NY.

GOIL, S., NAGESH, H., and CHOUDHARY, A. 1999. MAFIA: Efficient and scalable subspace clustering for very large data sets. *Technical Report CPDC-TR-9906-010*, Northwestern University.

GOLDBERG, D. 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

GONZALEZ, T.F. 1985. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38, 293-306.

GOVAERT, G. 1995. Simultaneous clustering of rows and columns. *Control and Cybernetics*, 24, 437-458.

GOWDA, K.C. and KRISHNA, G. 1978. Agglomerative clustering using the concept of mutual nearest neighborhood. *Pattern Recognition*, 10, 105-112.

GUHA, S., RASTOGI, R., and SHIM, K. 1998. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD Conference*, 73-84, Seattle, WA.

GUHA, S., RASTOGI, R., and SHIM, K. 1999. ROCK: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th ICDE*, 512-521, Sydney, Australia.

GURALNIK, V. and KARYPIS, G. 2001. A Scalable algorithm for clustering sequential data. *IEEE ICDM 2001*, Silicon Valley, CA.

GUTTMAN, A. 1984. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD Conference*, 47-57, Boston, MA.

HALL, L.O., OZYURT, B., and BEZDEK, J.C. 1999. Clustering with a genetically optimized approach. *IEEE Trans. on Evolutionary Computation*, 3, 2, 103-112.

HAN, E-H., KARYPIS, G., KUMAR, V., and MOBASHER, B. 1997. Clustering based on association rule hypergraphs. *ACM SIGMOD Conference, Data Mining Workshop* (DMKD'97), Tucson, Arizona.

HAN, J. and KAMBER, M. 2001. *Data Mining*. Morgan Kaufmann Publishers.

HAN, J., KAMBER, M., and TUNG, A. K. H. 2001. Spatial clustering methods in data mining: A survey. In Miller, H. and Han, J. (Eds.) *Geographic Data Mining and Knowledge Discovery*, Taylor and Francis.

HAREL, D. and KOREN, Y. 2001. Clustering spatial data using random walks, In *Proceedings of the 7th ACM SIGKDD*, 281-286. San Francisco, CA.

HARTIGAN, J. 1975. *Clustering Algorithms*. John Wiley & Sons, New York, NY.

HARTIGAN, J. and WONG, M. 1979. Algorithm AS136: A k-means clustering algorithm. *Applied Statistics*, 28, 100-108.

HEER, J. and CHI, E. 2001. Identification of Web user traffic composition using multi-modal clustering and information scent. *1st SIAM ICDM*, *Workshop on Web Mining*, 51-58, Chicago, IL.

HINNEBURG, A. and KEIM, D. 1998. An efficient approach to clustering large multimedia databases with noise. In *Proceedings of the 4th ACM SIGKDD*, 58-65, New York, NY.

HINNEBURG, A. and KEIM, D. 1999. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proceedings of the 25th Conference on VLDB*, 506-517, Edinburgh, Scotland.

HUANG, Z. 1998. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2, 3, 283-304.

HULTEN, G., SPENCER, L., and DOMINGOS, P. 2001. Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD*, 97-106, San Francisco, CA.

JAIN, A. and DUBES, R. 1988. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ.

JAIN, A.K. and FLYNN, P.J. 1966. Image segmentation using clustering. In *Advances in Image Understanding: A Festschrift for Azriel Rosenfeld*, IEEE Press, 65-83.

JAIN, A.K. and MAO, J. 1996. Artificial neural networks: A tutorial. *IEEE Computer*, 29, 3, 31-44.

JAIN, A.K, MURTY, M.N., and FLYNN P.J. 1999. Data clustering: a review. *ACM Computing Surveys*, 31, 3, 264-323.

JARVIS, R.A. and PATRICK, E.A. 1973. Clustering using a similarity measure based on shared nearest neighbors. *IEEE Transactions on Computers*, C-22, 11.

JEBARA, T. and JAAKKOLA, T. 2000. Feature selection and dualities in maximum entropy discrimination. In *Proceedings of the 16th UIA Conference*, Stanford, CA.

JOLIFFE, I. 1986. *Principal Component Analysis*. Springer-Verlag, New York, NY.

KALTON, A., LANGLEY, P., WAGSTAFF, K., and YOO, J. 2001. Generalized clustering, supervised learning, and data assignment. In *Proceedings of the 7th ACM SIGKDD*, 299-304, San Francisco, CA.

KANDOGAN, E. 2001. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *Proceedings of the 7th ACM SIGKDD*, 107-116, San Francisco, CA.

KARYPIS, G., AGGARWAL, R., KUMAR, V., and SHEKHAR, S. 1997. Multilevel hypergraph partitioning: application in VLSI domain, In *Proceedings ACM/IEEE Design Automation Conference.*

KARYPIS, G., HAN, E.-H., and KUMAR, V. 1999a. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling, *COMPUTER*, 32, 68-75.

KARYPIS, G., HAN, E.-H., and KUMAR, V. 1999b. Multilevel refinement for hierarchical clustering. *Technical Report # 99-020*.

KARYPIS, G. and HAN, E.-H. 2000. Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval & categorization. *Technical Report TR-00-016*, Department of Computer Science, University of Minnesota, Minneapolis.

KARYPIS, G. and KUMAR, V., 1999. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20, 1.

KASS, R. and RAFTERY, A. 1995. Bayes factors. *Journal of Amer. Statistical Association*, 90, 773-795.

KAUFMAN, L. and ROUSSEEUW, P. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, New York, NY.

KEOGH, E., CHAKRABARTI, K., MEHROTRA, S., and PAZZANI, M. 2001. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proceedings of the ACM SIGMOD Conference*, Santa Barbara, CA.

KEOGH, E., CHAKRABARTI, K., PAZZANI M., and MEHROTRA, S. 2001a. Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*, 3, 3.

KEOGH, E, CHU, S., and PAZZANI, M. 2001b. Ensemble-index: A new approach to indexing large databases. In *Proceedings of the 7th ACM SIGKDD*, 117-125, San Francisco, CA.

KERNIGHAN, B.W. and LIN, S. 1970. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49, 2, 291-307.

KOHONEN, T. 1990. The self-organizing map. *Proceedings of the IEEE*, 9, 1464-1479.

KOHONEN, T. 2001. *Self-Organizing Maps*. Springer Series in Information Sciences, 30, Springer.

KOLATCH, E. 2001. Clustering Algorithms for Spatial Databases: A Survey. PDF is available on the Web.

KOLLER, D. and SAHAMI, M. 1996. Toward optimal feature selection. In *Proceedings of the 13th ICML*, 284-292, Bari, Italy.

KNORR E. and NG R. 1998. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24h Conference on VLDB*, 392-403, New York, NY.

KNORR, E.M., NG, R.T., and ZAMAR, R.H. 2001. Robust Space Transformations for distance-based operations. In *Proceedings of the 7th ACM SIGKDD*, 126-135, San Francisco, CA.

KRIEGEL H.-P., SEEGER B., SCHNEIDER R., and BECKMANN N. 1990. The R*-tree: an efficient access method for geographic information systems. In *Proceedings International Conference on Geographic Information Systems*, Ottawa, Canada.

KULLBACK, S. and LEIBLER, R.A. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22, 76-86.

LANCE, G. and WILLIAMS W. 1967. A general theory of classification sorting strategies. *Computer Journal*, 9, 373-386.

LARSEN, B. and AONE, C. 1999. Fast and effective text mining using linear-time document clustering. In *Proceedings of the 5th ACM SIGKDD*, 16-22, San Diego, CA.

LEE, C-Y. and ANTONSSON, E.K. 2000. Dynamic partitional clustering using evolution strategies. In *Proceedings of the 3rd Asia-Pacific Conference on Simulated Evolution and Learning*, Nagoya, Japan.

LEE, W. and STOLFO, S. 1998. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX.

LIEBOVITCH, L. and TOTH, T. 1989. A fast algorithm to determine fractal dimensions by box counting. *Physics Letters*, 141A, 8.

LIN, D. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th ICML*, 296-304, Madison, WI.

LIU, B., XIA, Y., and YU, P.S. 2000. Clustering through decision tree construction. In *SIGMOD 2000.*

LIU, H. and SETIONO, R. 1996. A probabilistic approach to feature selection - a filter solution. In *Proceedings of the 13th ICML*, 319-327, Bari, Italy.

MANILLA, H. and RUSAKOV, D. 2001. Decomposition of event sequences into independent components. In *Proceedings of the 1st SIAM ICDM*, Chicago, IL.

MAO, J. and JAIN, A.K. 1996. A Self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks*, 7, 1, 16-29.

MARDIA, K., KENT, J. and BIBBY, J. 1980. *Multivariate Analysis*. Academic Press.

MARROQUIN, J.L. and GIROSI, F. 1993. Some extensions of the k-means algorithm for image segmentation and pattern classification. *A.I. Memo 1390*. MIT, Cambridge, MA.

MASSART, D. and KAUFMAN, L. 1983. The Interpretation of Analytical Chemical Data by the Use of Cluster Analysis. John Wiley & Sons, New York, NY.

MCCALLUM, A., NIGAM, K., and UNGAR, L.H. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the 6th ACM SIGKDD*, 169-178, Boston, MA.

MCLACHLAN, G. and BASFORD, K. 1988. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, NY.

MCLACHLAN, G. and KRISHNAN, T. 1997. *The EM Algorithm and Extensions*. John Wiley & Sons, New York, NY.

MICHALSKI, R.S. and STEPP, R. 1983. Learning from observations: conceptual clustering. In *Machine Learning: An Artificial Intelligence Approach*. San Mateo, CA, Morgan Kaufmann.

MILLIGAN, G. and COOPER, M. 1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50, 159-179.

MIRKIN, B. 1996. *Mathematic Classification and Clustering*. Kluwer Academic Publishers.

MITCHELL, T. 1997. *Machine Learning*. McGraw-Hill, New York, NY.

MODHA, D. and SPANGLER, W. 2002. Feature weighting in k-means clustering. *Machine Learning*, 47.

MOORE, A. 1999. Very fast EM-based mixture model clustering using multiresolution kd-trees. *Advances in Neural Information Processing Systems*, 11.

MOTWANI, R. and RAGHAVAN, P. 1995. *Randomized Algorithms*. Cambridge University Press.

MURTAGH, F. 1983. A survey of recent advances in hierarchical clustering algorithms. *Computer Journal*, 26, 4, 354-359.

MURTAGH, F. 1985. *Multidimensional Clustering Algorithms*. Physica-Verlag, Vienna.

NAGESH, H., GOIL, S., and CHOUDHARY, A. 2001. Adaptive grids for clustering massive data sets, In *Proceedings of the 1st SIAM ICDM*, Chicago, IL.

NG, R. and HAN, J. 1994. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th Conference on VLDB*, 144-155, Santiago, Chile.

NISHISATO, S. 1980. *Analysis of Categorical Data: Dual Scaling and Its Applications*. University of Toronto.

OLIVER, J., BAXTER, R. and WALLACE, C. 1996. Unsupervised learning using MML. In *Proceedings of the 13th ICML*, Bari, Italy.

OLSON, C. 1995. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21, 1313-1325.

OYANAGI, S., KUBOTA, K., and NAKASE, A. 2001. Application of matrix clustering to Web log analysis and access prediction. *7th ACM SIGKDD, WEBKDD Workshop*, San Francisco, CA.

PADMANABHAN, B. and TUZHILIN, A. 1999. Unexpectedness as a measure of interestingness in knowledge discovery. *Decision Support Systems Journal*, 27, 3, 303-318.

PADMANABHAN, B. and TUZHILIN, A. 2000. Small is beautiful: discovering the minimal set of unexpected patterns. In *Proceedings of the 6th ACM SIGKDD*, 54-63, Boston, MA.

PELLEG, D. and MOORE, A. 1999. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the 5th ACM SIGKDD*, 277-281, San Diego, CA.

PELLEG, D. and MOORE, A. 2000. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *Proceedings 17th ICML*, Stanford University.

PIATETSKY-SHAPIRO, G. and MATHEUS, C.J. 1994. The interestingness of deviations. In *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*.

RAMASWAMY, S., RASTOGI, R., and SHIM, K. 2000. Efficient algorithms for mining outliers from large data sets, *Sigmoid Record*, 29, 2, 427-438.

RAND, W.M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Assoc*, 66, 846-850.

RESNIK, P. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI*-95, 448-453, Montreal, Canada.

RISSANEN, J. 1978. Modeling by shortest data description. *Automatica*, 14, 465-471.

RISSANEN J. 1989. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Co., Singapore.

SANDER, J., ESTER, M., KRIEGEL, H.-P., and XU, X. 1998. Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. In *Data Mining and Knowledge Discovery*, 2, 2, 169-194.

SARAFIS, I., ZALZALA, A.M.S., and TRINDER, P.W. 2002. A genetic rule-based data clustering toolkit. To be published in *Congress on Evolutionary Computation (CEC)*, Honolulu, USA.

SAVARESI, S. and BOLEY, D. 2001. On performance of bisecting k-means and PDDP. In *Proceedings of the 1st SIAM ICDM*, Chicago, IL.

SAVARESI, S.M., BOLEY, D.L., BITTANTI, S., and GAZZANIGA, G. 2002. Cluster Selection in divisive clustering algorithms. In *Proceedings of the 2nd SIAM ICDM*, 299-314, Arlington, VA.

SCHALKOFF, R. 1991. *Pattern Recognition. Statistical, Structural and Neural Approaches*. John Wiley & Sons, New York, NY.

SCHIKUTA, E. 1996. Grid-clustering: a fast hierarchical clustering method for very large data sets. In *Proceedings 13th International Conference on Pattern Recognition,* 2, 101-105.

SCHIKUTA, E., ERHART, M. 1997. The BANG-clustering system: grid-based data analysis. In *Proceeding of Advances in Intelligent Data Analysis, Reasoning about Data, 2nd International Symposium,* 513-524, London, UK.

SCHWARZ, G. 1978. Estimating the dimension of a model. *The Annals of Statistics*, 6, 461-464.

SCOTT, D.W. 1992. *Multivariate Density Estimation*. Wiley, New York, NY.

SHEIKHOLESLAMI, G., CHATTERJEE, S., and ZHANG, A. 1998. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the 24th Conference on VLDB*, 428-439, New York, NY.

SIBSON, R. 1973. SLINK: An optimally efficient algorithm for the single link cluster method. *Computer Journal*, 16, 30-34.

SILBERSCHATZ, A. and TUZHILIN, A. 1996. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. on Knowledge and Data Eng.*, 8, 6.

SLONIM, N. and TISHBY, N. 2000. Document clustering using word clusters via the Information Bottleneck Method. In *Proceedings SIGIR*, 208-215.

SLONIM, N. and TISHBY, N. 2001. The power of word clusters for text classification. In *23rd European Colloquium on Information Retrieval Research*.

SMYTH, P. 1998. Model selection for probabilistic clustering using cross-validated likelihood. *ICS Tech Report 98-09*, Statistics and Computing.

SMYTH, P. 1999. Probabilistic model-based clustering of multivariate and sequential data. In *Proceedings of the 7th International Workshop on AI and Statistics*, 299-304.

SPATH H. 1980. *Cluster Analysis Algorithms*. Ellis Horwood, Chichester, England.

STEINBACH, M., KARYPIS, G., and KUMAR, V. 2000. A comparison of document clustering techniques. *6th ACM SIGKDD, World Text Mining Conference*, Boston, MA.

STREHL, A. and GHOSH, J. 2000. A scalable approach to balanced, high-dimensional clustering of market baskets, In *Proceedings of 17th International Conference on High Performance Computing,* Springer LNCS, 525-536, Bangalore, India.

THOMOPOULOS, S., BOUGOULIAS, D., and WANN, C-D. 1995. Dignet: An unsupervised-learning clustering algorithm for clustering and data fusion. *IEEE Trans. on Aerospace and Electr. Systems*, 31, 1, 2,1-38.

TISHBY, N., PEREIRA, F.C., and BIALEK, W. 1999. The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, 368-377.

TUKEY, J.W. 1977. *Exploratory Data Analysis*. Addison-Wesley.

TUNG, A.K.H., HOU, J., and HAN, J. 2001. Spatial clustering in the presence of obstacles. In *Proceedings of the 17th ICDE*, 359-367, Heidelberg, Germany.

TUNG, A.K.H., NG, R.T., LAKSHMANAN, L.V.S., and HAN, J. 2001. Constraint-Based Clustering in Large Databases, In *Proceedings of the* 8th ICDT, London, UK.

VOORHEES, E.M. 1986. Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. *Information Processing and Management*, 22, 6, 465-476.

WALLACE, C. and DOWE, D. 1994. Intrinsic classification by MML – the Snob program. In the *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*, 37-44, UNE, World Scientific Publishing Co., Armidale, Australia.

WALLACE, C. and FREEMAN, P. 1987. Estimation and inference by compact coding. *Journal of the Royal Statistical Society,* Series B, 49, 3, 240-265.

XU, X., ESTER, M., KRIEGEL, H.-P., and SANDER, J. 1998. A distribution-based clustering algorithm for mining large spatial datasets. In *Proceedings of the 14th ICDE*, 324-331, Orlando, FL.

WANN C.-D. and THOMOPOULOS, S.A. 1997. A Comparative study of self-organizing clustering algorithms Dignet and ART2. *Neural Networks*, 10, 4, 737-743.

WARD, J.H. 1963. Hierarchical grouping to optimize an objective function. *Journal Amer. Stat, Assoc.*, 58, 301, 235-244.

WANG, W., YANG, J., and MUNTZ, R. 1997. STING: a statistical information grid approach to spatialdata mining. In *Proceedings of the 23rd Conference on VLDB*, 186-195, Athens, Greece.

WANG, W., YANG, J., and MUNTZ, R.R. 1998. PK-tree: a spatial index structure for high dimensional point data. In *Proceedings of the 5th International Conference of Foundations of Data Organization*.

WANG, W., YANG, J., and MUNTZ, R.R. 1999. STING+: An approach to active spatial data mining. In *Proceedings 15th ICDE*, 116-125, Sydney, Australia.

XU, X., ESTER, M., KRIEGEL, H.-P., and SANDER, J. 1998. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the 14th ICDE*, 324-331, Orlando, FL.

YAO, A. 1982. On constructing minimum spanning trees in k-dimensional space and related problems. *SIAM Journal on Computing*, 11, 4, 721-736.

ZHANG, B. 2001. Generalized k-harmonic means – dynamic weighting of data in unsupervised learning. In *Proceedings of the 1st SIAM ICDM*, Chicago, IL.

ZHANG, T., RAMAKRISHNAN, R. and LIVNY, M. 1996. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD Conference*, 103-114, Montreal, Canada.

ZHANG, T., Ramakrishnan, R., and LIVNY, M. 1997. BIRCH: A new data clustering algorithm and its applications. *Journal of Data Mining and Knowledge Discovery*, 1, 2, 141-182.

ZHANG, Y., FU, A.W., CAI, C.H., and Heng. P.-A. 2000. Clustering categorical data. In *Proceedings of the 16th ICDE*, 305, San Diego, CA.