On Data Clustering Analysis: Scalability, Constraints and Validation

Osmar R. Zaïane, Andrew Foss, Chi-Hoon Lee, and Weinan Wang

University of Alberta, Edmonton, Alberta, Canada

Summary. Clustering is the problem of grouping data based on similarity. While this problem has attracted the attention of many researchers for many years, we are witnessing a resurgence of interest in new clustering techniques. In this paper we discuss some very recent clustering approaches and recount our experience with some of these algorithms. We also present the problem of clustering in the presence of constraints and discuss the issue of clustering validation.

1 Introduction

Cluster analysis is the automatic identification of groups of similar objects. This analysis is achieved by maximizing inter-group similarity and minimizing intra-group similarity. Clustering is an unsupervised classification process that is fundamental to data mining. Many data mining queries are concerned either with how the data objects are grouped or which objects could be considered remote from natural groupings. There have been many works on cluster analysis, but we are now witnessing a significant resurgence of interest in new clustering techniques. Scalability and high dimensionality are not the only focus of the recent research in clustering analysis. Indeed, it is getting difficult to keep track of all the new clustering strategies, their advantages and shortcomings. The following are the typical requirements for a good clustering technique in data mining [10]:

- **Scalability:** The cluster method should be applicable to huge databases and performance should decrease linearly with data size increase.
- **Versatility:** Clustering objects could be of different types - numerical data, boolean data or categorical data. Ideally a clustering method should be suitable for all different types of data objects.
- **Ability to discover clusters with different shapes:** This is an important requirement for spatial data clustering. Many clustering algorithms can only discover clusters with spherical shapes.
- **Minimal input parameter:** The method should require a minimum amount of domain knowledge for correct clustering. However, most current clustering algorithms have several key parameters and they are thus not practical for use in real world applications.
- **Robust with regard to noise:** This is important because noise exists everywhere in practical problems. A good clustering algorithm should be able to perform successfully even in the presence of a great deal of noise.

- **Insensitive to the data input order:** The clustering method should give consistent results irrespective of the order the data is presented.
- **Scaleable to high dimensionality:** The ability to handle high dimensionality is very challenging but real data sets are often multidimensional.

Historically, there is no single algorithm that can fully satisfy all the above requirements. It is important to understand the characteristics of each algorithm so the proper algorithm can be selected for the clustering problem at hand. Recently, there are several new clustering techniques offering useful advances, possibly even complete solutions.

In the next section, we attempt to put various approaches to clustering in perspective and group them by their fundamental approach. We present their basic concepts and principles. In Section 3, we discuss clustering techniques for spatial data in the presence of physical constraints. Finally, in Section 4, we conclude and discuss methods for validating cluster quality.

2 Taxonomy on Clustering Techniques

There exist a large number of clustering algorithms. Generally speaking, these clustering algorithms can be clustered into four groups: partitioning methods, hierarchical methods, density-based methods and grid-based methods. This section gives a taxonomy analysis and an experimental study of representative methods in each group. In order to examine the clustering ability of clustering algorithms, we performed experimental evaluation upon k-means [12], CURE [21], ROCK [8], DBSCAN [2], CHAMELEON [14], WaveCluster [24] and CLIQUE [1]. The DBSCAN code came from its authors while CURE and ROCK codes were kindly supplied by the Department of Computer Science and Engineering, University of Minnesota. k-means, CHAMELEON, WaveCluster, and CLIQUE programs were locally implemented. We evaluate these algorithms by using two dimensional spatial data sets referenced and used in the CHAMELEON paper [14] and data sets referenced and used in the WaveCluster paper [24]. The reason for using two dimensional spatial data is because we can visually evaluate the quality of the clustering result. Often people can intuitively identify clusters on two dimensional spatial data, while this is usually very difficult for high dimensional data sets. We show the experimental results of each algorithm on the t7 data set from the CHAMELEON paper as shown in Figure 1. This data set is a good test because it has various cluster shapes including clusters within clusters and a great deal of noise.

2.1 Partitioning methods

Suppose there are n objects in the original data set, partitioning methods break the original data set into k partitions. The basic idea of partitioning is very intuitive, and the process of partitioning is typically to achieve certain optimal criterion iteratively. The most classical and popular partitioning

methods are k-means [12] and k-medoid [16], where each cluster is represented by the gravity centre of the cluster in k-means method or by one of the "central" objects of the cluster in k-medoid method. Once cluster representatives are selected, data points are assigned to these representatives, and iteratively, new better representatives are selected and points reassigned until no change is made. CLARANS [20] is an improved k-medoid algorithm. Another extension of k-means is the k-modes method [11], which is specially designed for clustering categorical data. Instead of a "mean" in k-means, k-modes defined a "mode" for each cluster. A new development on the standard k-means algorithm is bisecting k-means [26]. Starting with all data points in one cluster, the algorithm proceeds by selecting the largest cluster and splitting it into two using basic k-means. This iterates until the desired number of clusters k is reached. By the nature of the algorithm, bisecting k-means tends to produce clusters of similar sizes unlike k-means, which tends to result in lower entropy as large clusters will often have higher entropy.

All the partitioning methods have a similar clustering quality and the major dificulties with these methods include: (1) The number k of clusters to be found needs to be known prior to clustering requiring at least some domain knowledge which is often not available; (2) it is dificult to identify clusters with large variations in sizes (large genuine clusters tend to be split); (3) the method is only suitable for concave spherical clusters.

Because partitioning algorithms have similar clustering results, we only implemented k-means. K-mean's result on t7 is shown in Figure 1 (A). From here we see k-means tends indeed to find spherical clusters, and is unable to find arbitrary shaped clusters. This is actually a general problem for all the partition methods because they use only one gravity centre to represent a cluster, and clustering of all the other points are decided by their relative closeness to the gravity centres of clusters.

2.2 Hierarchical Methods

A hierarchical clustering algorithm produces a dendogram representing the nested grouping relationship among objects. If the clustering hierarchy is formed from bottom up, at the start each data object is a cluster by itself, then small clusters are merged into bigger clusters at each level of the hierarchy until at the top of the hierarchy all the data objects are in one cluster. This kind of hierarchical method is called agglomerative. The inverse process is called divisive hierarchical clustering. There are many new hierarchical algorithms that have appeared in the past few years. The major difference between all these hierarchical algorithms is how to measure the similarity between each pair of clusters.

BIRCH [33] introduced the concept of clustering features and the CF-tree. It first partitions objects hierarchically using the CF-tree structure. This CF-tree is used as a summarized hierarchical data structure which compresses data while trying to preserve the inherent clustering structure. After the

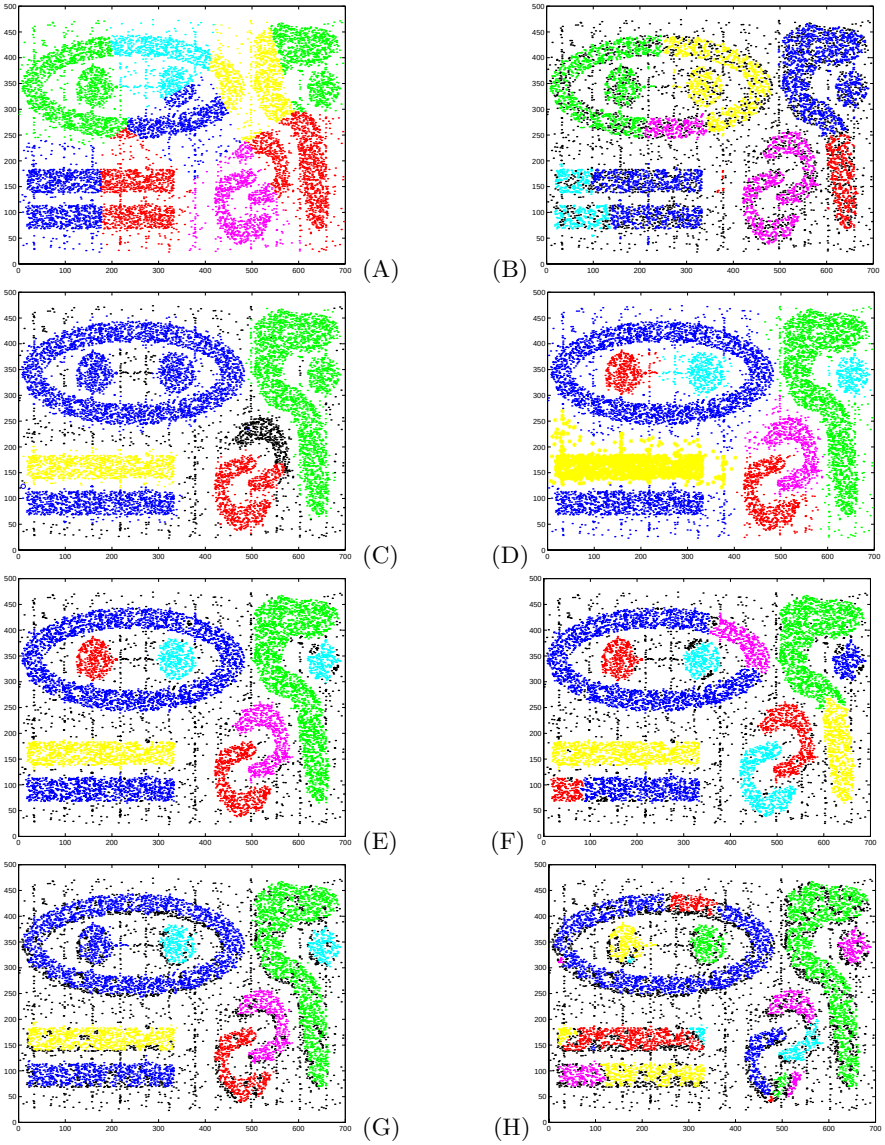

Fig. 1. Clustering results on t7.10k.dat. **(A)**: k-means with k=9; **(B)**: CURE with k=9, α=0.3, and 10 representative points per cluster; **(C)**: ROCK with θ=0.975 and k=1000; **(D)**: CHAMELEON with k-NN=10, MinSize=2.5%, and k=9; **(E)**: DBSCAN with ϵ=5.9 and MinPts=4; **(F)**: DBSCAN with ϵ=5.5 and MinPts=4; **(G)**: WaveCluster with *resolution*=5 and τ=1.5; **(H)**: WaveCluster with *resolution*=5 and τ=1.999397.

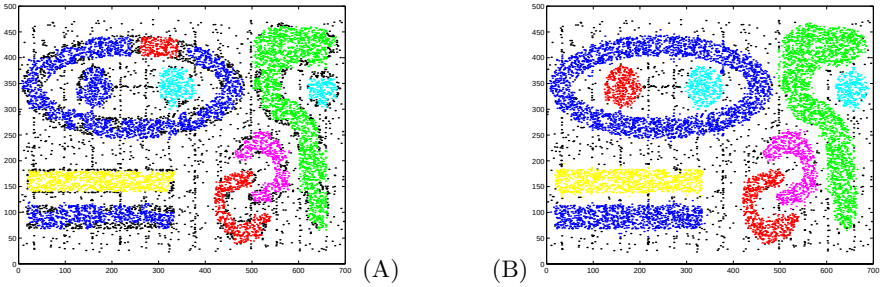

Fig. 2. Clustering results on t7.10k.dat. **(A)**: CLIQUE with $threshold = 0.18$ and $resolution = 20$; **(B)**: TURN* (no parameters needed).

building of the CF-tree, any clustering algorithm can be applied to the leaf nodes. BIRCH is particularly suitable for large data sets, however, it does not perform well if the clusters are not spherical in shape or there are big differences among cluster sizes.

CURE: Instead of using a single point to represent a cluster in centroid/medoid based methods, CURE [21] uses a set of points. This constant number of representative points of each cluster are selected so that they are well scattered and then shrunk towards the centroid of the cluster according to a shrinking factor. Interatively, clusters are merged based on their similarity. The similarity between two clusters is measured by the similarity of the closest pair of the representative points belonging to different clusters. With proper parameter selection, CURE partly remedies the problem of favouring clusters with spherical shape and similar sizes, and sensitivity to outliers. However, CURE's result are very sensitive to how the representative points are selected and the shrinking factor α. If α is large, CURE behaves like k-means, when small, CURE is sensitive to outliers. Through experiments we found that CURE has similar problems as k-means on complex data sets. In particular, it cannot find elongated clusters (Figure 1(B)).

ROCK [8] operates on a derived similarity graph, so it is not only suitable for numerical data, but also applicable for categorical data. Instead of using *distance* to measure similarity between data points, the concept of *links* is used. The basic idea is that data points are similar if they have enough common neighbours (i.e. links). The concept of *links* uses more global information of the cluster space compared with the distance similarity measure which only considers local distance between two points.

The problem of ROCK is that it is not successful in normalizing cluster links: it uses a fixed global parameter to normalize the total number of links. This fixed parameter actually reflects a fixed modeling of clusters, and it is not suitable for clusters with various densities. ROCK's clustering result is not good for complex clusters with various data densities. Also, it is very sensitive to the selection of parameters and sensitive to noise. After adjusting parameters for a long time, the best clustering result on t7 we could find is

illustrated in Figure 1(C). Notice that we set the number of clusters to be 1000, then among the resulting 1000 clusters, we got 5 big clusters, all the other 995 are just noise. This is because ROCK does not collect noise in its clustering process. For this data set, if we set cluster number to be 9, then most of the points, 9985 points, are in one cluster, and the other 15 points exists in the 8 noise clusters.

CHAMELEON [14] performs clustering through dynamic modeling: two clusters are merged only if the inter-connectivity and closeness between two clusters are comparable to the internal inter-connectivity and closeness within the clusters. CHAMELEON also operates on a derived similarity graph, so that this algorithm can be applied to both numerical data and categorical data. It operates on a sparse graph in which nodes represent data items, and weighted edges represent similarities among the data items. The sparse graph is formed by keeping *k-nearest neighbour* of each node. A graph partitioning method it used to pre-cluster objects in the sparse graph into a set of small clusters. These small clusters are then merged based on their relative interconnectivity and relative closeness. CHAMELEON has been found to be very effective in clustering, but has significant shortcomings: it cannot handle outliers and has too many parameters such as the number of nearest neighbours in the sparse graph, MINSIZE the stopping condition for the graph partitionning and α for adjusting relative closeness. CHAMELEON's result on the data set t7 is shown in Figure 1(D). Note that all the noise points are included inside neighbouring clusters.

The common disadvantage of hierarchical clustering algorithms is setting a termination condition which requires some domain knowledge for parameter setting. The problem of parameter setting makes them less useful for real world applications. Also typically, hierarchical clustering algorithms have high computational complexity.

2.3 Density-based Methods

The advantages of density-based methods are that they can discover clusters with arbitrary shapes and they do not need to preset the number of clusters.

DBSCAN [2] connects regions with sufficient high density into clusters. Each cluster is a maximum set of density-connected points. Points not connected are considered outliers. For each object of a cluster, the neighbourhood of a given radius (ϵ) has to contain at least a minimum number of points ($MinPts$). Methaphorically, it is like moving a flashlight of radius ϵ across the data set and connecting data points as long as $MinPts$ points are seen. This is the notion of density-reachability, and it is repeated until all points are labelled. In practice it works very well in spatial clustering. DBSCAN is very sensitive to the selection of ϵ and $MinPts$ and cannot identify clusters efficiently when cluster density varies considerably. When we apply DBSCAN to the data set t7, it gives very good results as illustrated in Figure 1(E). However, if we slightly change ϵ from 5.9 to 5.5, it gives bad results (see

Figure 1(F)). If we increase ϵ, the noise creates bridges that cause genuine clusters to merge.

By the same authors, OPTICS [18] is an extension to DBSCAN. Instead of producing one set of clustering results with one pre-setting radius (ϵ), OPTICS produces an augmented ordering of the database representing its density-based clustering structure. This cluster-ordering actually contains the information about every clustering level of the data set. Restrictions of OPTICS are that it is still more suitable for numerical data, and also the user still needs to set one parameter, $MinPts$.

TURN* [6] consists of an overall algorithm and two component algorithms, one, an efficient resolution dependent clustering algorithm TURN-RES which returns both a clustering result and certain global statistics (cluster features) from that result and two, TurnCut, an automatic method for finding the important or "optimum" resolutions from a set of resolution results from TURN-RES. TurnCut uses the core of the TURN algorithm [5] to detect a change in the third differential of a series to identify important areas in a cluster feature across resolution series built by repeated calls to TURN-RES by the *TURN** algorithm. This change is the "*turning*" point in the overall trend of the curve - acceleration or reversal of the rate of change of the clustering feature studied. This is very similar to the concept of finding the "knee" in the cluster feature graph [17] used for cluster validation. Unlike other approaches such as grid-based clustering, a resolution is simply a scale by which all data point values are multiplied, thus all data points are used in the process. At a given resolution, TURN-RES computes how tightly packed the points are around each point and marks points with a high value as "internal". At the same time, those neighbours of each point that are "close" are also marked. These definitions are resolution dependent. Clustering involves combining all close neighbours to an internal point into its cluster, and iterating for all of those points that are internal. Applied to the data set t7, TURN* gives very good results without the need of inputting parameters. Results are illustrated in Figure 2(B).

2.4 Grid-based Methods

Grid-based methods first quantize the clustering space into a finite number of cells, and then perform clustering on the gridded cells. The main advantage of grid-based methods is that their speed only depends on the resolution of griding, but not on the size of the data set. Grid-based methods are more suitable for high density data sets with a huge number of data objects in limited space.

WaveCluster [24] is a novel clustering approach based on wavelet transforms. WaveCluster first summarizes the data by applying a multi-resolution grid structure on the data space, then the original two-dimensional data space is considered as two-dimensional signals and signal processing techniques, wavelet transforms, are applied to convert the spatial data into the

frequency domain. After wavelet transform, the natural clusters in the data become distinguishable in a transformed frequency space. Dense regions, i.e. clusters, are easily captured in the frequency domain. The process of clustering is reduced to finding connected strong signal components in the low pass digital filter along the horizontal and vertical dimensions. Because Wavelet transforms can filter out noise points, clustering in this fequency domain is usually much simpler than clustering in the original 2-dimensional space. In the process of WaveCluster, there are two main parameters to be selected: one is the grid resolution; the other one is the signal threshold τ for deciding whether a cell is a significant cell in the low pass digital filter of both dimensions. WaveCluster's clustering result on t7 is shown in Figure 1(G). Notice that WaveCluster can not separate the two clusters connected by a "bridge". This is because in the convolved and down-sampled image with the low pass filter, the bridge connecting the two clusters is still very a strong signal. To separate the two clusters, other genuine clusters have to be separated as well. Figure 1(H) shows another cluster result of WaveCluster by adjusting signal threshold τ. Now it separates the bridge-connected clusters but breaks other genuine clusters.

CLIQUE [1] is specifically designed for finding subspace clusters in sparce high dimensional data. CLIQUE's clustering process starts with the lower dimensional space. When clustering for the k-dimensional space, CLIQUE makes use of information of the (k-1)-dimension which is already available. For instance, potentially dence cells of the grid in the 2-dimensional space are identified by dense regions in the 1-dimentional space. All other cells are simply disregarded. This is similar to the *a-priori* principle used in mining frequent itemsets for association rules. CLIQUE is, however, unsuitable for very noisy or dense spaces. CLIQUE's clustering result on t7 is shown in Figure 2(A). CLIQUE is not ideal for clustering 2-dimensional data. Its sensitivity to noise makes it merge genuine clusters at high resolution and disregard edge points when the resolution is not high enough.

3 Clustering spatial data in presence of constraints

So far we have seen algorithms that focus on the efficiency and effectiveness of clustering data. However, none of the algorithms consider possible constraints to the clustering. Examples of this are: (1) Constraints on individual objects; (2) obstacle objects as constraints; (3) clustering parameters as "constraints"; and (4) contraints imposed on each individual cluster [29]. In spatial data in particular, where data clustering has many applications in geograhic information systems, there are physical constaints such as obstacles (rivers, highways, mountain ranges, etc.) and crossings (bridges, pedways, etc.) that can hinder or signicantly alter the clustering process.

3.1 Constraint-based Clustering

In this section we present three algorithms recently devised that deal with clustering spatial data in the presence of physical constraints: COD-CLARANS [28], AUTOCLUST+ [3] and DBCluC [32].

COD-CLARANS [28] has been derived from CLARANS [20], a variant of the k-medoids approach. COD-CLARANS takes into account the obstacles by integrating an optimization scheme into the distance-error function in the course of the CLARANS algorithm. Obstacles are modeled with *a visibility graph VG = (V, E)* such that each vertex of the obstacle has a corresponding node in V, and two nodes v_1 and v_2 in V are connected by an edge in E if and only if the corresponding vertices they represent are visible to each other. The visibility graph is pre-processesed in order to compute the obstructed distance between two data objects in a given planar space. The obstructed distance is a detoured distance between two data objects with consideration of the visibility graph. Once the visibility graph is constructed, *Micro-clustering* is applied as pre-processing to group some data points from presumably a same cluster in order to minimize the number of data points to consider during COD-CLARANS clustering and fit the data set into main memory. Along with using the CLARANS clustering algorithm, COD-CLARANS uses a pruning function to reduce the search space by minimizing distance errors when selecting cluster representatives. Unfortunately, COD-CLARANS inherits the problems from CLARANS, a partitioning clustering method. It is sensitive to noise and assumes significant pre-processing of the data space to deal with the obstacles.

AUTOCLUST+ [3] is based on an extension to the clustering algorithm AUTOCLUST [4], a graph partitioning algorithm. The algorithm uses a Delaunay Diagram where all data points are represented and linked by edges based on mean and standard deviations of distances between points. Points linked by short edges, indicating closness, are clustered together. Other edges represent relations between clusters, and between clusters and noise. AUTOCLUST proceeds by eliminating edges from the graph to isolate data points that form clusters. In AUTOCLUST+, an obstacle is modeled by a set of line segments obstructing the edges from the Delaunay Diagram. AUTOCLUST+ then removes the edges from the Delaunay Diagram if they are impeded by a line segment from an obstacle. A removed edge is replaced by a detour path defined as the shortest path between the two data objects.

However, reconstructing the diagram after the removal of edges eventually degrades the performance of the algorithm since the algorithm needs to find a detour path for every intersected line segment with the Delaunay diagram.

DBCluC [32] is a density-based algorithm derived from DBSCAN. It discovers clusters of arbitrary shapes and isolates noise while considering not only disconnectivity dictated by obstacles but also connectivity forced by bridges. Obstacles and bridges are modeled by polygons where bridge-polygons have special edges considered as entry points. The DBCluC algo-

rithm extends the density reachability notion in DBSCAN to take into account visibility spaces created by the polygon edges. To enhance the clustering performance, polygons are reduced to a minimal set of obstruction lines that preserve the integrity of the visibility spaces. These lines stop the propagation of point neighbourhood in DBSCAN when they are from obstruction-polygons, or extend the neighbourhood when they are bridge-polygons.

4 Conclusion: Clustering Validation

One of the main difficulties with clustering algorithms is that, after clustering, how can one assess the quality of the clusters returned? Many of the popular clustering algorithms are known to perform poorly on many types of data sets. In addition, virtually all current clustering algorithms require their parameters to be tweaked for the best results, but this is impossible if one cannot assess the quality of the output. While 2D spatial data allows for assessment by visual inspection, the result is dependent on the resolution presented to the inspector and most clustering tasks are not 2D or even spatial. One solution [22,13,15] is to reduce any output to a 2D spatial presentation. Other solutions are based on 1) external, 2) internal and 3) relative criteria [17]. The External and Internal Criteria approaches use Monte Carlo methods [27] to evaluate whether the clustering is significantly different from chance.

In the External approach, the clustering result C can be compared to an independent partition of the data P built according to our intuition of the structure of the data set or the proximity matrix P is compared to P. The Internal Criteria approach uses some quantities or features inherent in the data set to evaluate the result. If the clustering is hierarchical, a matrix P_c, representing the proximity level at which two vectors are found in the same cluster for the first time, can be compared to P. This is repeated for many synthetic data sets to determine significance. For non-hierarchical methods a cluster membership matrix is compared to P using the same Monte Carlo method to determine significance.

These methods are clearly very expensive in processing time and only tell us that the clustering result is not pure chance. The Relative Criteria does not involve statistical tests but attempts to evaluate among several results arising from different parameter settings. The challenge is to characterize the clustering result in a way that tells us the quality of the clustering. Naturally, there is a grey line between measures used by clustering algorithms to determine where to join or split clusters and indices, e.g. [30,23,19,9], proposed to determine if that was good. Like many clustering algorithms, these indices suffer from problems especially inability to handle non-spherical clusters. Another approach computes several indices such as the Root-Mean-Square Standard Deviation, a measure of homogeneity, and plots them against k [23]. Whatever the index, having created a graph, this is inspected visually for either

a minima/maxima or a "knee", being the greatest jump of the index with a change in k. There is, however, no rigorous way of ensuring that this "knee" identifies the correct k. There are different indices defined for evaluating fuzzy clustering, e.g. [7,30]. An evaluation [17] of a number of indices on data that contained only concave but not always circular clusters, found different indices were better on different data sets showing their shape-dependence. In a few cases, Clustering Validation approaches have been integrated into clustering algorithms giving a relatively automatic clustering process. Smyth presented MCCV [25], the Monte Carlo Cross-Validation algorithm though this is intended for data sets where a likelihood function such as Gaussian mixture models can be defined. We have developed TURN [5] and TURN* [6] which handle arbitrary shapes, noise, and very large data sets in a fast and efficient way. TURN is intended for categorical data while TURN* is density based for spatial data.

An extended version of this paper is available in [31].

References

1. Agrawal R., Gehrke J., Gunopulos D. and Raghavan P. (1998) Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pp 94–105.
2. 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 *Proc. ACM-SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp 226–231.
3. Estivill-Castro V. and Lee I. (2000) Autoclust+: Automatic clustering of point-data sets in the presence of obstacles. In *Int. Workshop on Temporal and Spatio-Temporal Data Mining*, pp 133–146.
4. Estivill-Castro V. and Lee I. (2000) Autoclust: Automatic clustering via boundary extraction for mining massive point-data sets. In *Proc. 5th International Conference on Geocomputation*.
5. Foss A., Wang W. and Zaïane O. R. (2001) A non-parametric approach to web log analysis. In *Proc. of Workshop on Web Mining in First International SIAM Conference on Data Mining*, pp 41–50.
6. Foss A. and Zaïane O. R. (2002) TURN* unsupervised clustering of spatial data. submitted to *ACM-SIKDD Intl. Conf. on Knowledge Discovery and Data Mining*, July 2002.
7. Gath I. and Geva A. (1989) Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7).
8. Guha S., Rastogi R. and Shim K. (1999) ROCK: a robust clustering algorithm for categorical attributes. In *15th ICDE Int'l Conf. on Data Engineering*.
9. Halkidi M., Vazirgiannis M. and Batistakis I. (2000) Quality scheme assessment in the clustering process. In *Proc. of PKDD, Lyon, France*.
10. Han J. and Kamber M. (2000) *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers.
11. Huang Z. (1998) Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, v2 pp283–304.

12. MacQueen J. (1967) Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley Symp. Math. Statist. Prob.*.
13. Sammon J. Jr. (1969) A non-linear mapping for data structure analysis. *IEEE Trans. Computers*, v18 pp401–409.
14. Karypis G., Han E.-H. and Kumar V. (1999) Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8) pp68–75.
15. Kohonen T. (1995) *Self-Organizing Maps*, Springer-Verlag.
16. Kaufman L. and Rousseeuw P. J. (1990) *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons.
17. Halkidi M., Batistakis Y. and Vazirgiannis M. (2001) On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3) pp 107–145.
18. Ankerst M., Breunig M, Kriegel H.-P. and Sander J. (1999) Optics: Ordering points to identify the clustering structure. In *Proc. ACM-SIGMOD Conf. on Management of Data*, pp 49–60.
19. Pal N. R. and Biswas J. (1997) Cluster validation using graph theoretic concepts. *Pattern Recognition*, 30(6).
20. Ng R. and Han J. (1994) Efficient and effective clustering method for spatial data mining. In *Proc. Conf. on Very Large Data Bases* , pp 144–155.
21. Guha S., Rastogi R. and Shim K. (1998) CURE: An efficient clustering algorithm for large databases. In *Proc. ACM-SIGMOD Conf. on Management of Data*.
22. Schwenker F., Kestler H. and Palm G. (2000) An algorithm for adaptive clustering and visualisation of highdimensional data sets. In H.-J. L. G. della Riccia, R. Kruse, editor, *Computational Intelligence in Data Mining*, pp 127–140. Springer, Wien, New York.
23. Sharma S. (1996) *Applied Multivariate Techniques*. John Willey & Sons.
24. Sheikholeslami G., Chatterjee S. and Zhang A. (1998) Wavecluster: a multiresolution clustering approach for very large spatial databases. In *Proc. 24th Conf. on Very Large Data Bases*.
25. Smyth P. (1996) Clustering using monte carlo cross-validation. *Proc. ACM-SIGKDD Int. Conf. Knowledge Discovery and Data Mining*.
26. Steinbach M., Karypis G. and Kumar V. (2000) A comparison of document clustering techniques. In *SIGKDD Workshop on Text Mining*.
27. Theodoridis S. and Koutroubas K. (1999) *Pattern recognition*, Academic Press.
28. Tung A. K. H., Hou J. and Han J. (2001) Spatial clustering in the presence of obstacles. In *Proc. ICDE Int. Conf. On Data Engineering*.
29. Tung A. K. H., Ng R., Lakshmanan L. V. S. and Han J. (2001) Constraint-based clustering in large databases. In *Proc. ICDT*, pp 405–419.
30. Xie X. and Beni G. (1991) A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4).
31. Zaïane O. R., Foss A., Lee C.-H. and Wang W. (2002) Data clustering analysis - from simple groupings to scalable clustering with constraints. Technical Report, TR02-03, Department of Computing Science, University of Alberta.
32. Zaïane O. R. and Lee C.-H. (2002) Clustering spatial data in the presence of obstacles and crossings: a density-based approach. submitted to *IDEAS Intl. Database Engineering and Applications Symposium*.
33. Zhang T., Ramakrishnan R. and Livny M. (1996) BIRCH: an efficient data clustering method for very large databases. In *Proc. ACM-SIGKDD Int. Conf. Managament of Data*, pp 103–114.