2005 한국경영과학회/대한산업공학회 춘계공동학술대회
2005 년 5 월 13 일~14 일, 충북대학교

An Optimization Approach to Data Clustering

Jumi Kim[a], Sigurdur Ólafsson[b]

[a]Entrue Consulting Partners, LG CNS
25F, Seoul Finance Center, 84 Taepyungro 1-ga,
Chung-gu, Seoul, Korea, 100-768
jumikim@lgcns.com

Abstract
Scalability of clustering algorithms is critical issues facing the data mining community. This is particularly true for computationally intense tasks such as data clustering. Random sampling of instances is one possible means of achieving scalability but a pervasive problem with this approach is how to deal with the noise that this introduces in the evaluation of the learning algorithm. This paper develops a new optimization based clustering approach using an algorithms specifically designed for noisy performance. Numerical results illustrate that with this algorithm substantial benefits can be achieved in terms of computational time without sacrificing solution quality.

1. Introduction

In recent years databases in modern enterprises have become massive and contain a wealth of important data. However, when traditional methods of analysis fall short in transforming this data into knowledge, exploratory techniques such as knowledge discovery in databases must be applied. This multidisciplinary field of data mining draws heavily on statistics and artificial intelligence, but numerous problems in data mining and knowledge discovery can also be formulated as optimization problems, and optimization techniques can therefore be used to solve large-scale data mining problems (Basu, 1998; Bradley et al., 1999).

As the importance of data mining has grown, one of the critical issues to emerge is how to scale data mining techniques to larger and larger databases (Bradley et al., 2002). This is particularly true for computationally intensive data mining tasks such as identifying natural clusters of instances (Kaufman and Rousseeuw, 1990). Several approaches to scalability enhancements have been studied at length in the literature (Provost and Kolluri, 1999), including using parallel mining algorithms (Forman and Zhang, 2000) and preprocessing the data by filtering out redundant or irrelevant features and thus reducing the dimensionality of the database (Ólafsson, 2003). Another approach to better scalability is using a selection of instance from a database rather than the entire database (Liu and Motoda, 2001). This paper deals with such instance selection and how it can be applied to data clustering within an optimization-based framework.

Perhaps the simplest approach to instance selection is random sampling (Kiven and Mannila, 1994). Numerous authors have studied this approach for specific data mining tasks such as clustering (Kaufman and Rousseeuw, 1990; Ng and Han, 1994, Ester, Kriegel, and Xu, 1995), association rule discovery (Toivonen, 1996), and decision tree induction (Chauchat and Rakotomalala, 2001). When implementing this approach, the most challenging issue is determining a sample size that improves performance of the algorithm without sacrificing solution quality. Bounds can be developed that allow for a prediction of sample effort needed, but such bounds usually require knowing certain problem parameters and typically overestimate the necessary sample size (Toivonen, 1996). On the other hand, too small a sample will lead to a bias and degeneration in performance. One possible solution is to use adaptive sampling (Domingo, Gavalde, and Watanabe, 2000; Provost, Jensen, and Oates, 1999).

In this paper we advocate an alternative approach that is based on a novel formulation of the clustering task as an optimization problem. We also take advantage of the fact that certain optimization techniques have been explicitly designed to account for noisy performance estimates, such as are common when performance is estimated using simulation. In particular, one such method is the nested partitions method that can be used to solve general global optimization problems (Shi and Olafsson, 2000) and specifically combinatorial type optimization problems with noisy performance (Olafsson, 1999). A defining characteristic of this method is that wrong moves made due to noise in performance estimates can be automatically corrected in a later move. In the scalable clustering context this means that nosier performance estimates, resulting from smaller samples of instances, may result in more steps taken by the algorithm but any bias will be automatically corrected. This eliminates the need to determine the exact sample size, although the computational performance of the algorithm may still depend to some extent on how it is selected.

The reminder of this paper is organized as follows. In Section 2 we briefly review clustering techniques and in particular, focus on efforts in scalable clustering. In Section 3 we present the optimization-based clustering algorithm and demonstrate its effectiveness on a sample problems. In Section 4 we present some numerical results on the scalability of the algorithm with respect to the instance dimension, and Section 5 contains concluding remarks and suggestions for future research directions.

2. Scalable Clustering

Clustering has been an active area of research for several decades, and many clustering algorithms have been proposed in the literature (Kaufman and Rousseeuw, 1990; Grabmeier and Rudolph, 2002). In particular, considerable research has been devoted specifically to scalable clustering. We will start by briefly describing the various types of clustering algorithms and then mention some specific scalable methods.

Clustering algorithms can be roughly divided into two categories: hierarchical clustering and partitional clustering (Jain, Murty, and Flynn, 1999). In hierarchical clustering all of the instances are organized into a hierarchy that describes the degree of similarity between those instances (e.g., a dendrogram). Such representation may provide a great deal of information, but the scalability of this approach is questionable as the number of instances grows. Partitional clustering, on the other hand, simply creates one partition of the data where each instance falls into one cluster. Thus, less information is obtained but the ability to deal with a large number of instances is improved. Examples of the partitioning approach are the classic k-means and k-medoids clustering algorithms.

There are many other characteristics of clustering algorithms that must be considered to ensure scalability of the approach. For example, most clustering algorithms are polythetic, meaning that all features are consider simultaneously in tasks such as to determine the similarity of two instances. However, as the number of features becomes large this may pose scalability problems and it may be necessary to restrict attention to monothetic clustering algorithms that consider features one at a time. Most clustering algorithm are also non-incremental in the sense that all of the instances are considered simultaneously. However, there are a few algorithms that are incremental, which implies that they consider each instance separately. Such algorithms are particularly useful when the number of instances is large and keeping the entire set of instances in memory poses scalability problems.

Scalable clustering has received considerable attention in recent years, and here we will mention only a few of the methods that have been developed. For example, Zhang, Ramakrishnan, and Livny (1996) proposed BIRCH, a hierarchical algorithm for clustering. The key idea of this method is to summarize cluster representations using two innovative concepts, clustering feature and clustering feature tree.

Another approach to hierarchical clustering is the CURE algorithm developed by Guha, Rastogi, and Shim (1998). The steps of the CURE algorithm are to obtain a sample from the original database, partition the sample into a set of partitions and then cluster each partition, eliminate outliers and cluster the partial clusters. Finally, each data instance is labeled with the corresponding cluster.

Improved scalable versions of partitioning methods such as k-means and k-medoids have also been developed. The Clustering LARge Applications (CLARA) algorithm improves the scalability of the PAM k-medoids algorithm by applying PAM to multiple samples of the actual data and returns the best clustering (Kaufman and Rousseeuw, 1990). The CLARANS algorithm improves on this approach by obtaining random samples at each step, thus making the sampling dynamic (Ng and Han, 1994).

A single pass k-means clustering algorithm was proposed by Bradley, Fayyad, and Reina (1998), with the main idea being to use a buffer to save points from the database in a compressed form. This approach was simplified in the algorithm proposed by Farnstrom, Lewis, and Elkan (2000), in an effort to reduce the overhead that otherwise might cancel out any scalability improvements that might be achieved.

Yet another way of improving scalability is via distributed clustering, where instead of combining all data before clustering, data sets are operated on independently with minimum communication between the parallel clustering algorithms (Forman and Zhang, 2000).

The work presented in this paper is a partitional clustering algorithm that attempts to find cluster centers and uses random sampling to improve scalability. In that sense, it is the most similar to the CLARA and CLARANS algorithms, but its optimization-based approach sets it apart.

3. Optimization-Based Clustering

3.1. The NP-Method

The nested partitions (NP) method is an optimization method that has been suggested by Shi and Olafsson (2000) to solve general global optimization problems of the following form:

$$\min_{\boldsymbol{x}\in X} f(\boldsymbol{x}) \quad (1)$$

where $\boldsymbol{x}$ is a point in a n-dimensional space X and $f: X \rightarrow \boldsymbol{R}$ is a real-valued performance measure defined on this space. This performance may or may not be known deterministically. In our context, X is the space of all clusters and measures some quality of the clusters.

The intuitive idea of the NP method is quite simple. In each step, the method systematically partitions the feasible region into subsets and focus the computational effort in those subsets that are considered promising. The main components of the method are:

- **Partitioning**: at each iteration the feasible region is partitioned into subsets that cover the feasible region but concentrate the search in what is believed to be the most promising region.
- **Random sampling**: to evaluate each of the subsets, a random sample of solutions are obtained from each subset and used to estimate the performance of the region as a whole.

This method can be understood as an optimization framework that combines adaptive global sampling with local heuristic search. It uses a flexible partitioning method to divide the design space into regions that can be analyzed individually and then aggregates the results from each region to determine how to continue the search, that is, to concentrate the computational effort. Thus, the NP method adaptively samples from the entire design space and concentrates the sampling effort by systematic partitioning of the design space.

To implement the partitioning, the NP method maintains in the *k*th iteration what is called the most promising region, that is, a subregion $\boldsymbol{X}(k) \subseteq \boldsymbol{X}$ that is considered the most likely to contain the best solution. This most promising region is partitioned into a given number of subregions and what remains is aggregated into one region called the surrounding region. Thus, a disjoint collection of sets covering the entire feasible region is considered. The subregions and the surrounding region are sampled using random sampling, and the sampling information used to determine which region should be the most promising
region in the next iteration. If one of the subregions contains the best solution, this region is now selected as the new most promising region and is, in the next iteration, partitioned into smaller subregions. If the surrounding region contains the best solution this is taken as an indication that the last move might not have been

the best move, so the algorithm backtracks to what was the most promising region in the previous iteration. This partitioning creates a tree of subsets that we refer to as the partitioning tree.

3.2. Defining Clusters

The partitional clustering problem can be formulated as an optimization problem and thus solved within the NP framework. In particular, we have designed the NP method as a partitional clustering method for nominal data and incorporated k-means into the same framework. In this approach we assume that we want to partition a given data set into k clusters and that each clusters is defined by its center (each instance is assigned to the closest center). The decision variables are thus the ith coordinate of the cth cluster, where $i=1,2,K,n$, $c=1,2,K,k$. Therefore, this clustering problem reduces to locating the centers to optimize certain performance.

Selecting a performance measure to be optimized is very subjective, since determining what constitutes a good cluster is necessarily subjective and no single standard exists. We refer the reader to Estivill-Castro (2002) for a recent discussion of this issue and Grabmeyer and Rudolph (2002) for a more extensive survey. The most common measures are probably to maximize similarity within a cluster (that is, maximize homogeneity or compactness), and to minimize similarity between different clusters (that is, maximize separability between the clusters). A particular strength of the optimization-based framework is that any such measure, or combination of measures, can be adopted. Indeed, the function f can be defined as any measure of what is believed to indicate the quality of a cluster.

To minimize bias that may be introduced by analyzing very specific performance measures, we restrict ourselves here to a single measure of similarity within cluster, namely, its compactness:

$$f(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},K\,\boldsymbol{x}^{(k)}) = \sum_{y\in\Psi}\sum_{i=1}^{n}\left|y_i - x_i^{[y]}\right|^2 \qquad (2)$$

Here Ψ is the space of all instances, $y\in\Psi$ is a specific instance in this space, $\boldsymbol{x}^{[y]}$ is the cluster center to which the instance is assigned, and $\left|y_i - x_i^{[y]}\right|$ is the difference between the ith coordinate of the instance and the corresponding center.

We believe that by using such a simple measure we are better able to focus on the performance of the algorithm itself. For a particular application, however, this will without doubt be defined in a different fashion, but that will not change the implementation of the algorithm.

3.3. Partitioning

The main implementation issue for applying NP is to define the partitioning. We suggest doing this by finding cluster centers for one feature at a time, that is, at each level of the partitioning tree the values for all the centers are limited to a given range for one feature. This defines the subsets or region that form the partitioning tree. Then, as for the generic NP method, random samples are obtained from each subset and to speed convergence the k-means algorithm is applied to those random samples and the resulting improved centers used to select the most promising region. This most promising region is the partitioned further, the surrounding region aggregated, and so forth.

Figure 1 demonstrates a partial partitioning tree where the first feature can take four different values, that is $x_1 \in \{1,2,3,4\}$, and the problem is to find the optimal location of k = 2 clusters (identified as *C1* and *C2*). This partitioning approach helps with the scalability of the method with respect to the feature dimension. It focuses on fixing one feature at a time and is in that sense monothetic, but not fully so as all features are randomly assigned values during the random sampling stage, and thus all the feature are used simultaneously to select subregions. This approach can thus be though of as having elements of both monothetic and polythetic clustering.

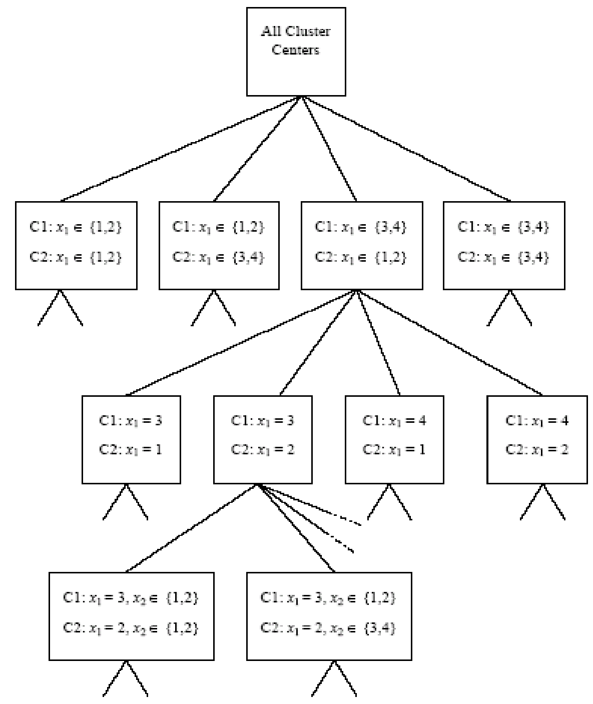
All Cluster Centers
C1: $x_1 \in \{1,2\}$ C2: $x_1 \in \{1,2\}$
C1: $x_1 \in \{1,2\}$ C2: $x_1 \in \{3,4\}$
C1: $x_1 \in \{3,4\}$ C2: $x_1 \in \{1,2\}$
C1: $x_1 \in \{3,4\}$ C2: $x_1 \in \{3,4\}$
C1: $x_1 = 3$ C2: $x_1 = 1$
C1: $x_1 = 3$ C2: $x_1 = 2$
C1: $x_1 = 4$ C2: $x_1 = 1$
C1: $x_1 = 4$ C2: $x_1 = 2$
C1: $x_1 = 3, x_2 \in \{1,2\}$ C2: $x_1 = 2, x_2 \in \{1,2\}$
C1: $x_1 = 3, x_2 \in \{1,2\}$ C2: $x_1 = 2, x_2 \in \{3,4\}$

Figure 1. Partial Partitioning Tree for Locating Two Clusters (Feature Fixed in Two Clusters)

It is also important to note that the partitioning tree imposes a structure on the space of all possible clusters, and thus determines the efficiency of the search through this space. Furthermore, the partitioning tree is simply determined by the order in which features are fixed, and investigating effective methods for ordering features for this purpose is an important future research topic.

3.4. Numerical Evaluation

To evaluate the effectiveness of the NP-based clustering approach, which we call algorithm NPCLUSTER, we compare it with the PAM algorithm, which is a variant of the k-medoids approach (Kaufman and Rousseeuw, 1990), and its more scalable variations of CLARA and CLARANS (Ng and Han, 1994). The motivation for the selection of these algorithms for comparison is that like NPCLUSTER, these algorithms use a partitional approach to identify cluster centers and employ a random sampling strategy to improve scalability.

We use a small but realistic data set from the UCI Repository of Machine Learning Databases (Blake and Merz, 1998), namely the breast cancer data set. We use two variants of this data set, which we refer to as 'large' and 'small,' with 699 and 286 instances, respectively. Both variants of the set have 9 features. The results for the large data are shown in Table 1 and those for the small data in Table 2.

Table 1. Comparison of Algorithms for Large Data Set

Algorithm	Similarity Value	Computation Time
NPCLUSTER	3292 ± 16	$3.7 \bullet 10^5$
PAM	3166 ± 0.5	$102.6 \bullet 10^5$
CLARA	5026 ± 54	$1.7 \bullet 10^5$
CLARANS	3620 ± 29	$5.2 \bullet 10^5$

Table 2. Comparison of Algorithms for Small Data Set

Algorithm	Similarity Value	Computation Time
NPCLUSTER	1128 ± 16	$6.9 \bullet 10^4$
PAM	978 ± 2	$192.4 \bullet 10^4$
CLARA	1971 ± 21	$1.8 \bullet 10^4$
CLARANS	1151 ± 9	$2.1 \bullet 10^4$

The quality of the clusters obtained is measured by the compactness or average similarity value (2) of the clusters. The smaller this value, the better the clustering. We note that although PAM has the best performance in terms of similarity values, it comes at a very high computation cost. By using sampling (NPCLUSTER, CLARA, CLARANS), the computation time can be reduced by two orders of magnitude. The CLARA algorithm, on the other hand, uses the least computation time for both data sets but the quality of the clusters is not satisfactory compared to the other methods. The NPCLUSTER and CLARANS seem to strike the best balance between speed and cluster quality and NPCLUSTER is particularly attractive for the larger data.

The ability of the NPCLUSTER method to use sampling and still obtain high quality solution stems from the fact that when an incorrect move is made in the partitioning tree, it can be corrected in the next (or a later) iteration, when a new sample of instances indicates that this was the wrong move. Thus, if there is large amount of noise in the performance estimates (i.e., a small sample of instance is used), then the algorithm may backtrack frequently. Frequent backtracking implies more iterations, and thus increases computation time so there is a tradeoff between fast computation in each iteration and needing more iterations when reducing the instance sample size. However, we note that the NPCLUSTER algorithm achieves this balance in an automated manner.

4. Scalability

As has been noted above, repeated calculation of cluster performance according to (2) is time consuming and a more scalable approach is to use an estimate

$$f(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\ldots \boldsymbol{x}^{(k)},\boldsymbol{I}) \quad (3)$$

that is calculated from a (small) subset $\boldsymbol{I}$ of the set of all instances. The key questions to be answered is how much savings in computation time can be achieved by using this estimate, what is the best sample size $|\boldsymbol{I}|$, and how sensitive the clustering algorithm performance is to this sample size.

To obtain some tentative answers to these questions and to demonstrate feasibility for the scalability improvements that are possible by using sampling, we apply the NP-based clustering method described above to the same two data sets as before. The numerical results are reported in Table 3, which shows the solution quality (similarity value), computation time, and average amount of backtracks for varying amounts of sampling. These results clearly indicate that random sampling can be effectively used to achieve substantial computational benefits without sacrificing solution quality. We recall that here solution quality is defined by equation (2) as being a measure of within cluster similarity or compactness, that is the sum of the deviation of instances from the cluster center of the cluster to which they are assigned.

For example, using 25% of the small data set reduces the computation time by 69% while the similarity only increases by 2% and is within two standard deviations of the value without sampling. Similarly for the large data set, by using 5% of the instances in each step, computational time can be reduce by 90% while similarity value is only increased by 2 percent.

Furthermore, the performance is not very sensitive to exact selection of amount of sampling. For example, for both problems the performance when 5% of instances is used is very similar to the performance when 25% of instances is used. Thus, the iterative nature of the NP-based clustering algorithm, and its automatic backtracking feature, allows us to achieve significant computational improvements without exact calibration of how many instances are needed.

Table 3. Numerical Results for Different Percentage of instances Used

Data Set	Fraction	Similarity Value Avg. ± S.E.	Computation Time Avg. ± S.E.	Backtracking Avg. ± S.E.
Large	100%	4259.0 ± 46	394777 ± 6668	0.14 ± 0.05
	50%	4207.7 ± 53	101666 ± 1352	0.08 ± 0.04
	25%	4264.7 ± 51	43794 ± 498	0.08 ± 0.04
	5%	4363.4 ± 41	38966 ± 590	0.10 ± 0.05
	0.5%	4401.1 ± 49	44065 ± 775	0.14 ± 0.06
Small	100%	1302.3 ± 13	84115 ± 3166	0.72 ± 0.10
	50%	1276.6 ± 15	27295 ± 901	0.30 ± 0.07
	25%	1322.9 ± 12	25699 ± 689	0.56 ± 0.19
	5%	1363.1 ± 13	27698 ± 960	0.44 ± 0.12
	0.5%	1430.9 ± 12	33773 ± 1203	0.34 ± 0.08

Table 4. Estimated Coefficient of variation

Fraction	CV
100%	$16.9 \bullet 10^2$
50%	$1.33 \bullet 10^2$
25%	$1.14 \bullet 10^2$
5%	$1.51 \bullet 10^2$
0.5%	$1.76 \bullet 10^2$

We also note that the variability of the performance (as measured by the standard error reported in Table 3) is stable. This is somewhat surprising as one might expect that dealing with the noisier sets corresponding to a small sample of the original data might give rise to higher variability. The fact that such an increase is not observed is an indication that the NP-based clustering is very effective in dealing with such uncertainty.

There is, on the other hand, a significant observed change in the variability of the computing time. For both test problems, the variability is the least for when using 25% of the database, which also corresponds to the shortest computation time. The reduction in variability is not, however, solely explained by the shorter computation time as the estimated coefficient of variation (standard error divided by the average) shows a similar pattern. For example, for the larger problem the estimated coefficient of variation is shown in Table 4. Thus, we conclude that sampling does not only reduce the computation time, but the computation time is more stable when the sample size is selected appropriately.

5. Conclusions and Future Research

Clustering is one of the most important areas of knowledge discovery in databases, and the use of optimization techniques for clustering offers considerable promise. The scalability of such techniques is one of the key issues to be addressed as the field progresses. In particular, scalability with respect to increasing number of instances is critical as databases become ever larger. One way of dealing with this issue is to use a subset of all instances for the learning algorithm. The obvious tradeoff is between computational issues, where fewer instances imply faster learning, and solution quality, where using fewer instances may imply lower quality models.

We have designed an optimization bases approach to the partitional clustering problem where the algorithm is specifically designed to deal with noisy performance estimates, such as those that arise when only part of the data is used to create clusters. Numerical results show that considerable speedup can be achieved (up to 90% for the numerical examples) with no or minimal reduction in solution quality. Also, the algorithm is robust with respect to the amount of instances used so there is no need to carefully determine the fraction of the database that needs to be used.

There are numerous issues that should be addressed for further development of this methodology. For example, an extensive numerical evaluation on a variety of realistic and synthetic problems should be performed, relating the computational speedup to characteristics of the data, and developing heuristics for specifying amount of instances to be used and evaluating their robustness.

As noted in Section 3.3 above, determining intelligent ways of ordering the features is also a critical issue. The partitioning tree imposes a structure on the search space of all possible clusters and the order in which features are considered determines this structure. Thus, it an important future research topic is to investigate how the algorithm can be improved by determining a generic way of creating high quality partitioning for arbitrary clustering problems.

Finally, the ability of this approach to handle arbitrary performance functions opens up some interesting possibilities. In this paper we only considered a measure of cluster compactness, but as noted in Section 3.2, any measure can be used. Thus, it is of interest to apply the new algorithm with various measures of cluster performance and compare qualities of the resulting cluster. In other words, by using the same optimization methodology, but different measures of what makes a good cluster, and analyzing the resulting clusters, we believe insights into data clustering in general could be obtained.

References

[1] Basu, A. (1998). Perspectives on operations research in data and knowledge management. European Journal of Operational Research, 111, 1-14.

[2] Blake, C.L. and C.J. Merz (1998). UCI Repository of Machine Learning Databases [http://www.ici.uci.edu/mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine, CA.

[3] Bradley, P., U. Fayyad, and C. Reina (1998). Scaling clustering algorithms to large databases. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, 9-15.

[4] Bradley, P., J. Gehrke, R. Ramakrishnan, and R. Srikant (2002). Scaling mining algorithms to large databases. Communications of the ACM 45(8), 38-43.

[5] Bradley, P.S., Mangasarian, O.L., and Street, W.N. (1998). Feature selection via mathematical programming, INFORMS Journal on Computing, 10, 209-217.

[6] Chauchat, J-H. and R. Rakotomalala (2001). Sampling strategies for building decision trees from very large databases comprising many continuous attributes. In Liu and Motada (eds.) Instance Selection and Construction for Data Mining, Kluwer.

[7] Domingo, C. Gavalda R., and Watanabe, R. (2000). Adaptive Sampling Methods for Scaling Up Knowledge discovery. In Journal of Knowledge Discovery and Data Mining.

[8] Estevill-Castro, V. (2002). Why so many clustering algorithms - a position paper. SIGKDD Explorations 4(1), 65-75.

[9] Ester, M., H.-P. Kriegel, and X. Xu (1995). Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. In Proc. 4th Int. Symp. Large Spatial Databases, 67-82.

[10] Farnstrom, F., J. Lewis, and C. Elkan (2000). Scalability for clustering algorithms revisited. SIGKDD Explorations 2(1),51-57.

[11] Forman, G. and B. Zhang (2000). Distributed data clustering can be efficient and exact. SIGKDD Explorations 2(2), 34-38.

[12] Guha, S. R. Rastogi, and K Shim (1998). CURE: An efficient clustering algorithm for large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 73-84.

[13] Grabmeier, J. and A. Rudolph (2002). Techniques of cluster algorithms in data mining. Data Mining and Knowledge Discovery 6, 303-360.

[14] Jain, A.K., Murty, M.N. and Flynn, P.J. (1999). Data clustering: a review. ACM Computing Surveys 31, 264 - 323.

[15] John, G. and P. Langley (1996). Static versus dynamic sampling for data mining. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 367-370.

[16] Kaufman, L. and P.J. Rousseeuw (1990). Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, New York.

[17] Kiven, J. and H. Mannila (1994). The power of sampling in knowledge discovery. In ACM Symposium on Principles of Database Theory, 77-85.

[18] Liu, H. and H. Motoda (2001). Instance Selection and Construction for Data Mining, Kluwer.

[19] Ng, R.T. and J. Han (1994). Efficient and Effective Clustering Methods for Spatial Data Mining. In Proceedings of 20th International Conference on Very Large Data Bases.

[20] Olafsson, S. (2003). Improving scalability of e-commerce systems with knowledge discovery. In Prabu, Kumara and Kamath (eds.) Scalable Enterprise System - An Introduction to Recent Advances, Kluwer.

[21] Provost, F., D. Jenson, and T. Oates (1999). Efficient progressive sampling. In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, 23-32.

[22] Provost, F. and V. Kolluri (1999). A survey of methods for scaling up inductive algorithms. Data Mining and Knowledge Discovery 3: 131-169.

[23] Shi, L. and S. Olafsson (2000). Nested partitions method for global optimization. Operations Research 48, 390-407.

[24] Toivonen, H. (1996). Sampling large databases for association rules. In Proceedings of the 22nd International Conference on Very Large Databases, 134-145.

[25] Zhang, T., R. Ramakrishnan, and M. Livny (1996). BIRCH: An efficient data clustering method for very large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 103-114.