Adaptive Clustering Algorithms

Alina Câmpan[1] and Gabriela Şerban[1]

[1] Department of Computer Science
"Babeş-Bolyai" University
1, M. Kogalniceanu Street, Cluj-Napoca, Romania
tel: +40.264.405327, fax: +40.264.591.960
{alina, gabis}@cs.ubbcluj.ro

Abstract. This paper proposes an adaptive clustering approach. We focus on re-clustering an object set, previously clustered, when the feature set characterizing the objects increases. We have developed adaptive extensions for two traditional clustering algorithms (*k-means* and *Hierarchical Agglomerative Clustering*). These extensions can be used for adjusting a clustering, that was established by applying the corresponding non-adaptive clustering algorithm before the feature set changed. We aim to reach the result more efficiently than applying the corresponding non-adaptive algorithm starting from the current clustering or from scratch. Experiments testing the method's efficiency are also reported.

1 Introduction

A large collection of clustering algorithms is available in the literature. The papers [9], [10] and [11] contain comprehensive overviews of the existing clustering techniques. Generally, these methods apply on a set of objects measured against a known set of features (attributes). But there are applications where the attribute set characterizing the objects evolves. For obtaining in these conditions a partition of the object set, the clustering algorithm can be, obviously, applied over and over again, beginning from scratch or from the current partition, each time when the attribute set changes. But this can be inefficient. We agree to call a clustering method *adaptive*, if it produces a clustering by adjusting an existing partition to attribute set extension.

We propose two adaptive clustering algorithms, named *Core Based Adaptive k-means (CBAk)* and *Hierarchical Core Based Adaptive Clustering (HCBAC)*. They follow a common approach, based on detecting stable structures (cores) inside the existing clusters and resuming the clustering process from these structures, when the attribute set increases. We aim to reach the result more efficiently than applying the corresponding non-adaptive algorithm starting from the current partition or from scratch.

2 Adaptive Core Based Clustering

2.1 Related Work

There are few approaches reported in the literature that refer to the problem of adapting the result of a clustering when the object feature set is extended. Early works treat the sequential use of features in the clustering process, one by one. An example of such a monothetic approach is mentioned in [11]. A more recent paper [16] analyzes the problem of adapting a clustering produced by the *DBSCAN* algorithm, using some additional structures and distance approximations in an Euclidian space. However, adapting a clustering resulted from a partitioning by relocation algorithm, or from a hierarchical agglomerative one has not been reported, to our knowledge.

2.2 Theoretical Model

Let $X = \{O_1, O_2, \ldots, O_n\}$ be the set of objects to be clustered. Each object is measured with respect to a set of m initial attributes and is therefore described by an m-dimensional vector $O_i = (O_{i1}, \ldots, O_{im}), O_{ik} \in \Re^+, 1 \leq i \leq n, 1 \leq k \leq m$. Usually, the attributes associated to objects are standardized, in order to ensure an equal weight to all of them [9].

In the following, we agree to denote by $\mathcal{A}$ one of the two non-adaptive traditional clustering algorithms, whom adaptive extensions we are studying in this paper: *k-means* and *Hierarchical Agglomerative Clustering Algorithm (HACA)*.

Let $\mathcal{K} = \{K_1, K_2, \ldots, K_p\}$ be the set of clusters discovered in data by applying $\mathcal{A}$. We mention that, in the case of *HACA*, the clustering process stops when p clusters are reached, and $\mathcal{K}$ represents the last attained partition. Each cluster is a set of objects, $K_j = \{O_1^j, O_2^j, \ldots, O_{n_j}^j\}, 1 \leq j \leq p$. The centroid (cluster mean) of the cluster K_j is denoted by f_j, where $f_j = \left(\frac{\sum_{k=1}^{n_j} O_{k1}^j}{n_j}, \ldots, \frac{\sum_{k=1}^{n_j} O_{km}^j}{n_j} \right)$. Even if it is not a typical concept of hierarchical clustering, we will make use in our approach of the centroid notion for *HACA*, also.

The measure used for discriminating objects can be any *metric* or *semi-metric* function d. We used the *Euclidian distance*:

$$d(O_i, O_j) = d_E(O_i, O_j) = \sqrt{\sum_{l=1}^{m} (O_{il} - O_{jl})^2}.$$

The measured set of attributes is afterwards extended with s ($s \geq 1$) new attributes, numbered as $(m+1), (m+2), \ldots, (m+s)$. After the extension, the objects' vectors become $O_i' = (O_{i1}, \ldots, O_{im}, O_{i,m+1}, \ldots, O_{i,m+s}), 1 \leq i \leq n$.

We want to analyze the problem of recalculating the objects' grouping into clusters, after the attribute set extension. The new clusters can be, obviously, obtained by applying $\mathcal{A}$ on the set of extended objects starting:

- from scratch if $\mathcal{A}$ is *HACA*;

- from scratch or from the current partition $\mathcal{K}$ if $\mathcal{A}$ is *k-means*.

We try to avoid this process by replacing it with one less expensive but not less accurate. Therefore, we will try to efficiently adapt the current partition ($\mathcal{K}$), produced by $\mathcal{A}$.

We denote by $K'_j, 1 \leq j \leq p$, the set containing the same objects as K_j, after the extension. By $f'_j, 1 \leq j \leq p$, we denote the mean (center) of the set of K'_j. These sets $K'_j, 1 \leq j \leq p$, will not necessarily represent clusters after the attribute set extension. The newly arrived attributes can change the objects arrangement into clusters. But there is a chance, when adding one or few attributes to objects, that the old arrangement in clusters to be close to the actual one. With these being said, we agree, however, to continue to refer the sets K'_j as clusters.

We take as starting point the previous partition into clusters and study in which conditions an extended object $O_i^{j\prime}$ is still "correctly" placed into its cluster K'_j. We intuitively started from the fact that, at the end of the initial *k-means* clustering process, all objects are closer to the centroid of their cluster than to any other centroid. So, for any cluster j and any object $O_i^j \in K_j$, inequality (1) below holds.

$$d_E(O_i^j, f_j) \leq d_E(O_i^j, f_r), \forall j, r,\ 1 \leq j, r \leq p,\ r \neq j. \tag{1}$$

This inequality will not hold for every object in respect to the clusters produced by *HACA*. But as we used as linkage-metric in *HACA* average-link, it is likely, that a lot of objects will satisfy inequality (1) for this algorithm as well.

When attribute set extension happens, we will detect in each cluster a subset of objects (core) that could reach together in a cluster, if we would cluster the extended object set. We will use inequality (1), of objects closeness to the centers, as the stability condition for delimiting cores inside clusters. So, a core of cluster K_j will consist of those objects in K_j that have a considerable chance to remain stable in a cluster, and not to divide between more clusters as a result of the attribute set extension.

Definition 1.

a) *We denote by* $StrongCore_j = \{O_i^{j\prime} | O_i^{j\prime} \in K'_j, O_i^j$ *satisfies inequality (1)* ***before and after*** *attribute set extension*$\}$ *i.e. the set of all objects in* K'_j *closer,* ***before and after*** *extension, to the center of their cluster than to the center of any other cluster.*

b) *Let* $sat(O_i^{j\prime})$ *be the set of all clusters* $K'_r, \forall r,\ 1 \leq r \leq p,\ r \neq j$ *not containing* $O_i^{j\prime}$ *and for which object* $O_i^{j\prime}$ *satisfies inequality (1)* ***after*** *attribute set extension. We denote by* $WeakCore_j = \{O_i^{j\prime} | O_i^{j\prime} \in K'_j, O_i^j$ *satisfies inequality (1)* ***before*** *extension and* $|sat(O_i^{j\prime})| \geq \frac{\sum_{k=1}^{n_j} |sat(O_k^{j\prime})|}{n_j}\}$

c) $Core_j = StrongCore_j$ *iif* $StrongCore_j \neq \emptyset$*; otherwise,* $Core_j = WeakCore_j$. $OCore_j = K'_j \setminus Core_j$ *is the set of out-of-core objects in cluster* K'_j.

d) *We denote by $CORE$ the set $\{Core_j, 1 \leq j \leq p\}$ of all cluster cores and by $OCORE$ the set $\{OCore_j, 1 \leq j \leq p\}$.*

As we have already mentioned, for a partition produced by *k-means*, the inequality (1) holds, before the attribute set extension, for every object and its cluster.

We have chosen the above cluster cores definition because of the following reason. In our algorithms, $Core_j$ will be the seed for cluster j in the adaptive process. But it is possible, especially in the case of *HACA*, that the $StrongCore$ of clusters to be empty. For managing this situation, when the $StrongCore$ of a cluster is detected to be empty, we weaken the core forming conditions. Correspondingly, we defined the $WeakCore$ of a cluster, which consists of the most stable objects in K'_j.

The cluster cores, chosen as we described, will serve as seed in the adaptive clustering process. All objects in $Core_j$ will surely remain together in the same group if clusters do not change. This will not be the case for all core objects, but for most of them, as we will see in the results section.

3 The *Core Based Adaptive k-means* Algorithm

We give next the *Core Based Adaptive k-means* algorithm. The algorithm starts by calculating the old clusters cores. The cores will be the new initial clusters from which the iterative processing begins. Next, the algorithm proceeds in the same manner as the classical *k-means* method does. We mention that the algorithm stops when the clusters from two consecutive iterations remain unchanged or the number of steps performed exceeds the maximum allowed number of iterations.

Algorithm Core Based Adaptive k-means is
Input: - the set $X = \{O_1, \ldots, O_n\}$ of m-dimensional previously clustered objects,
- the set $X' = \{O'_1, \ldots, O'_n\}$ of (m+s)-dimensional extended objects to be clustered; O'_i has the same first m components as O_i,
- the metric d_E between objects in a multi-dimensional space,
- p, the number of desired clusters,
- $K = \{K_1, \ldots, K_p\}$ the previous partition of objects in X,
- $noMaxIter$ the maximum number of iterations allowed.
Output: - the new partition $K' = \{K'_1, \ldots, K'_p\}$ for the objects in X'.
Begin
For all clusters $K_j \in K$
Calculate $Core_j = (StrongCore_j \neq \emptyset)?StrongCore_j : WeakCore_j$
$K'_j = Core_j$
Calculate f'_j as the mean of objects in K'_j
EndFor
While (K' changes between two consecutive steps) and

```
        (there were not performed noMaxIter iterations) do
      For all clusters K'_j do
        K'_j = {O'_i | d_E(O'_i, f'_j) ≤ d_E(O'_i, f'_r), ∀r, 1 ≤ r ≤ p, 1 ≤ i ≤ n}
      EndFor
      For all clusters K'_j do
        f'_j = the mean of objects in K'_j
      EndFor
   EndWhile
End.
```

4 The *Hierarchical Core Based Adaptive* Algorithm

We give next the *Hierarchical Core Based Adaptive* algorithm. The algorithm starts by calculating the old clusters cores. For each cluster j, the objects in $OCore_j$ will be extracted and distributed each one in its singleton. This is a divisive step. Clearly, from this cluster adjustment process will result a number p' of clusters, $p \leq p' \leq n$. In order to reach again the targeted number p of clusters, we proceed next to merge clusters in the same manner as the classical *HACA* does. But, as we do not generally start again from singletons, the number of steps will be significantly reduced. Also, as we will demonstrate by experiments, we do not significantly lose quality of clusters obtained by *HCBAC* compared to the quality of clusters provided by *HACA*. We mention that the algorithm stops when p clusters are obtained.

```
Algorithm Hierarchical Core Based Incremental Clustering is
Input: - the set X = {O_1, ..., O_n} of m-dimensional previously clustered
       objects,
       - the set X' = {O'_1, ..., O'_n} of (m+s)-dimensional extended objects
       to be clustered, O'_i has the same first m components as O_i,
       - the metric d_E between objects in a multi-dimensional space,
       - p, the number of desired clusters,
       - K = {K_1, ..., K_p} the previous partition of objects in X.
Output: - the re-partition K' = {K'_1, ..., K'_p} for the objects in X'.
Begin
   For all clusters K_j ∈ K do
      Calculate Core_j = (StrongCore_j ≠ ∅)?
                         StrongCore_j : WeakCore_j
      Calculate OCore_j = K_j \ Core_j
   EndFor
   C = ∅ // the current cluster set
   For i = 1 to p do
      If Core_i ≠ ∅
        C = C ∪ {Core_i}
      EndIf
```

```
        For all O ∈ OCore_i do
          C = C ∪ {O} //add a singleton to C
        EndFor
    EndFor
    While | C |> p do
        (C_u*, C_v*) := argmin_(C_u,C_v) d_E(C_u, C_v)
        C_new = C_u* ∪ C_v*
        C = C \ {C_u*, C_v*) ∪ {C_new}
    EndWhile
    K' = C
End.
```

As distance between two clusters $d_E(C_u, C_v)$ we considered the average-link metric:

$$d_E(C_u, C_v) = \frac{\sum_{a_i \in C_u} \sum_{b_j \in C_v} d_E(a_i, b_j)}{\mid C_u \mid \times \mid C_v \mid}.$$

This linkage metric leads to higher probability of well formed and stable cores than would lead the single-link metric, for example.

5 Experimental Evaluation

In this section we present some experimental results obtained by applying the *CBAk* and *HCBAC* algorithms described in section 3 and 4. We will compare each of two algorithms with its corresponding non-adaptive version (*CBAk* vs *k-means*, *HCBAC* vs *HACA*).

5.1 Quality Measures

Number of iterations. It determines the global calculus complexity and it is used for evaluating the performances of both *CBAk* and *HCBAC*.

The movement degree of the core objects and of the extra-core objects quantifies how the objects in either $Core_j \in CORE$, or $OCore_j \in OCORE$, remain together in clusters after the algorithm ends. It is measured for *CBAk*. It is not used for evaluating *HCBAC* because, once placed into a cluster, any set of objects will not be splitted anymore between clusters, in the agglomerative process.

As expected, more stable the core objects are and more they remain together in respect to the initial sets $Core_j$, better was the decision to choose them as seed for the adaptive clustering process.

We denote by $S = \{S_1, S_2, \ldots, S_p\}, S_i \subseteq K_i$, a set of clusters' subsets (as $CORE$ and $OCORE$ are). We express the *stability factor* of S as:

$$SF(S) = \frac{\sum_{j=1}^{p} \frac{|S_j|}{\text{no of clusters where the objects in } S_j \text{ ended}}}{\sum_{j=1}^{p} |S_j|} \tag{2}$$

The worst case is when each object in S_j ends in a different final cluster, and this happens for every set in S. The best case is when every S_j remains compact and it is found in a single final cluster. So, the limits between which $SF(CORE)$ varies are given below, where the higher the value of $SF(CORE)$ is, the better was the cores choice:

$$\frac{p}{\sum_{j=1}^{p} |Core_j|} \leq SF(CORE) \leq 1 \tag{3}$$

Squared sum error (SSE) is used for comparing the quality of the partitions produced by *CBAk* and by *k-means*. SSE of a partition $\mathcal{K}$ is defined as:

$$SSE(K) = \sum_{K_j \in K} \sum_{O_i \in K_j} (d(O_i, f_j))^2 \tag{4}$$

When comparing two partitions $\mathcal{K}_1$ and $\mathcal{K}_2$ for the same data set, we will say that $\mathcal{K}_1$ is better than $\mathcal{K}_2$ iff $SSE(\mathcal{K}_1) < SSE(\mathcal{K}_2)$.

The degree of compactness of a partition is the measure equivalent to *SSE* for *HCBAC*. The degree of compactness, or the *dispersion* (*DISP*) of a partition $\mathcal{K}$ is defined as follows:

$$DISP(K) = \frac{\sum_{k=1}^{p} \frac{\sum_{O_i, O_j \in K_k, i>j} d(O_i, O_j)}{C^2_{|K_k|}}}{p} \tag{5}$$

where $K = \{K_1, \ldots, K_p\}$ is the cluster set obtained after applying a clustering algorithm. *DISP* expresses the average distance between objects in a cluster, for all clusters and $C^2_{|K_k|}$ represents the number of combinations of 2 elements from the set K_k.

As expected, the smaller the dispersion is, more compact clusters we have obtained and better was the cores choice at the beginning of the adaptive clustering process.

Clustering tendency. For measuring the clustering tendency of a data set, we use the Hopkins statistics, H [14], an approach that uses statistical tests for spatial randomness. H takes values between 0 and 1, and a value near 1 indicates

that the data is highly clustered. Usually, for a data set with clustering tendency, we expect for H values greater than 0.5.

Information gain. For comparing the informational relevance of the attributes we used the *information gain (IG)* measure ([12]).

As case studies, for experimenting our theoretical results and for evaluating the performance of the algorithms, we consider some experiments that are briefly described in the following subsections.

We have to mention that the data were taken from the website "http://www.cormactech.com/neunet".

5.2 Experiment 1. Cancer

The breast cancer database was obtained from the University of Wisconsin Hospitals, Madison, Dr. William H. Wolberg.

The objects to be clusterized in this experiment are patients: each patient is identified by 9 attributes ([15]). The attributes have been used to represent instances and each one takes integer values between 1 and 10. Each instance has one of 2 possible classes: benign or malignant. In this experiment there are 457 patients (objects).

5.3 Experiment 2. Dermatology

The objects to be clusterized in this experiment are also patients: each patient is identified by 34 attributes, 33 of which are linear valued and one of them is nominal. There are 1617 objects (patients).

The aim of the clustering process is to determine the type of Eryhemato-Squamous Disease ([8]).

In the dataset constructed for this domain, the family history feature has the value 1 if any of these diseases has been observed in the family, and 0 otherwise. The age feature simply represents the age of the patient. Every other feature (clinical and histopathological) was given a degree in the range of 0 to 3. Here, 0 indicates that the feature was not present, 3 indicates the largest amount possible, and 1, 2 indicate the relative intermediate values.

5.4 Experiment 3. Wine

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines ([1]).

The objects to be clusterized in this experiment are wine instances: each is identified by 13 attributes. There are 178 objects (wine instances).

We have to mention that all attributes in this experiment are continuous.

5.5 *CBAk* Results

In this section we comparatively present the results obtained by applying the *CBAk* algorithm and *k-means*, for the experimental data. We mention that the results are calculated in average, for several executions. We considered two variants for *k-means*: resuming from the current partition (denoted by v1) and starting from scratch (denoted by v2).

Table 1. The comparative results for *k-means* and *CBAk*

Experiment	Cancer	Dermatology	Wine
No of objects	457	366	178
No of attributes (m+s)	9	34	13
No of new attributes (s)	4	3	4
No of clusters	2	6	3
No of k-means iterations for m attributes	5.66	10.43	9.26
No of k-means (v1) iterations for (m+s) attributes	4	1	4.94
No of k-means (v2) iterations for (m+s) attributes	7	10.33	6.85
No of CBAk iterations for (m+s) attributes	4	1	4.66
No of objects StrongCore/WeakCore (% from no of objects) - CBAk	96.4/0	100/0	91.66/0
No of objects Core/OutOfCore (% from no of objects) - CBAk	96.4/3.6	100/0	91.66/8.34
SF(CORE) - CBAk	0.67	1	0.587
SF(OCORE) - CBAk	0.84	-	0.77
k-means (v1) SSE for (m+s) attributes	13808.78	12740.23	49.73
k-means (v2) SSE for (m+s) attributes	13808.78	13049.22	49.021
CBAk SSE for (m+s) attributes	13808.78	12796.65	50.17
H for s attributes	0.666	0.68122	0.7018
H for m+s attributes	0.7148	0.6865	0.7094

From Table 1 we observe that using the *CBAk* algorithm the number of iterations for finding the solution is less than or at most equal to the number of *k-means* iterations, for both variants (v1 and v2). The cores' stability factor, $SF(CORE)$, is high, taking into account that most of the objects are contained in cores. We mention that for every run of each experiment, SSE(CBAk) has been roughly equal to SSE(k-means), both for v1 and v2.

In Table 2 we present, for each experiment, the attributes in decreasing order of their information gain (IG) - the new attributes are emphasized.

From Table 2 it results that the importance of the added attributes influences the number of iterations performed by the *CBAk* algorithm for finding the solution.

A problem with the *k-means* algorithm is that it is sensitive to the selection of the initial partition (centroids) and may converge to a local minimum of the

Table 2. The decreasing order of attributes in respect to the information gain measure

Experiment	Order of attributes	IG of new attributes / IG of old attributes (%)
Cancer	2 3 **6 7** 5 4 **8** 1 **9**	64,7%
Dermatology	22 21 23 1 **34** 30 28 13 26 7 17 9 29 10 16 11 25 15 6 27 4 20 **32** 8 5 24 3 31 12 2 19 18 14 **33**	7,6%
Wine	7 **10 12 13** 6 1 2 **11** 9 4 5 3 8	57%

squared error value if the initial partition is not properly chosen. In order to evaluate properly our algorithm, we considered the same initial centroids when running *k-means* for the initial and feature-extended object set (m and $m + s$ number of attributes). It would be interesting to analyze how a good initial centroids choice affects the results.

5.6 *HCBAC* Results

In this section we comparatively present the results obtained by applying the *HCBAC* and *HACA* algorithms, for the experimental data.

Table 3. The comparative results for *HACA* and *HCBAC*

Experiment	Cancer	Dermatology	Wine
No of objects	457	366	178
No of attributes (m+s)	9	34	13
No of new attributes (s)	4	3	4
No of clusters	2	6	3
No of HACA iterations for m attributes	455	360	175
No of HACA iterations for (m+s) attributes (N1)	455	360	175
No of HCBAC iterations for (m+s) attributes (N2)	22	27	2
Reduction of the no of iterations (N1-N2)/N1(%)	95.16 %	92.5 %	98.8 %
DISP(HACA) for m attributes	5.3507	8.0207	0.83
DISP(HACA) for (m+s) attributes	7.6505	7.9284	0.9871
DISP(HCBAC) for (m+s) attributes	7.702	8.1697	0.8337
No of objects StrongCore/WeakCore (% from no of objects) HCBAC	95/0	92.6/0	98.8/0

From Table 3 we observe that using the *HCBAC* the number of iterations for finding the solution is smaller than in the case of *HACA*. Also, the clusters obtained by *HCBAC* are roughly equally dispersed as those given by *HACA*. So, the clusters quality remains at about the same level, but the clustering process is more efficient.

5.7 Adaptive Horizontal Fragmentation in Object Oriented Databases

A practical problem, where the proposed methods can be efficiently used, is the adaptive horizontal fragmentation of object oriented databases.

A horizontal fragmentation approach that uses data mining clustering methods for partitioning object instances into fragments has been presented in [4], [5], [6], [7]. Essentially, that approach takes full advantage of existing data, where statistics are already present, and develops fragmentation around user applications (queries) that are to be optimized by the obtained fragmentation. But real databases applications evolve in time, and consequently they require re-fragmentation in order to deal with new applications entering the system and others leaving. Obviously, for obtaining the fragmentation that fits the new user applications set, the original fragmentation scheme can be applied from scratch. However, this process can be inefficient.

We have applied the *CBAk* method in the case when new user applications arrive in the system and the current fragments must be accordingly adapted ([2]). The obtained results were good. The adaptive fragmentation keeps the fragmentation quality around the non-adaptive one and the processing time is improved, as the incremental method performs, generally, in less time than the full fragmentation process.

6 Conclusions and Future Work

In this paper we proposed a new approach for adapting the result of a clustering when the attribute set describing the objects increases. Two traditional algorithms, *k-means* and *HACA*, were adapted to follow this approach. The experiments on different data sets prove that, in most cases, the results are reached more efficiently using the proposed adaptive methods than running the corresponding classical algorithms, on the feature-extended object set. But there are some situations when it is better to resort to a non-adaptive clustering of the feature-extended object set, than using the proposed algorithms. Intuitively, such situations can be: the addition of a large number of features or the addition of new features with large information gain and contradictory information with respect to the old feature set.

Further work could be done in the following directions:

- to make experiments that would cover a range of additional cases like: varied number of added features, discrete or continuous features and to consider a more substantial number of features and objects;
- to isolate conditions to decide when it is more effective to adapt the result of a clustering of the feature-extended object set than to resume or restart the clustering using *k-means* or *HACA*;
- to study how the information brought into the system by the newly added attributes, their correlation with the initial ones, influences the performance of the adaptive algorithms;

– to apply the adaptive algorithms on precise problems, from where the need of such algorithms originated.

References

1. Aeberhard, S., Coomans, D., de Vel, O.: THE CLASSIFICATION PERFORMANCE OF RDA. Tech. Rep. **92–01**, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland (1992)
2. Campan, A., Darabant, A.S., Serban, G., Clustering Techniques for Adaptive Horizontal Fragmentation in Object Oriented Databases. In Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics ICTAMI 2005, Alba-Iulia, Romania, (2005) (to appear).
3. CorMac Technologies Inc, Canada: Discover the Patterns in Your Data. http://www.cormactech.com/neunet (2006)
4. Darabant, A.S., Campan, A., Semi-supervised learning techniques: k-means clustering in OODB Fragmentation. In pRoceedings of IEEE International Conference on Computational Cybernetics ICCC 2004, Vienna University of Technology, Austria, August 30 - September 1, (2004) 333–338
5. Darabant, A.S., Campan, A., Hierarchical AI Clustering for Horizontal Object Fragmentation. In Proceedings of Int. Conf. of Computers and Communications, Oradea, May, (2004) 117–122
6. Darabant, A.S., Campan, A., AI Clustering Techniques: a New Approach to Object Oriented Database Fragmentation. In Proceedings of the 8th IEEE International Conference on Intelligent Engineering Systems, Cluj Napoca, (2004) 73–78
7. Darabant, A.S., Campan, A., Cret, O., Hierarchical Clustering in Object Oriented Data Models with Complex Class Relationships. In Proceedings of the 8th IEEE International Conference on Intelligent Engineering Systems, Cluj Napoca, (2004) 307–312
8. Demiroz, G., Govenir, H. A., Ilter, N.: Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals. Artificial Intelligence in Medicine
9. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers (2001)
10. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs, New Jersey (1998)
11. Jain, A., Murty, M. N., Flynn, P.: Data clustering: A review. ACM Computing Surveys, **31(3)** (1999) 264-323
12. Quinlan, J. R.: C4.5: Programs for Machine Learning, Morgan Kaufmann. San Mateo, California (1993)
13. Şerban, G., Câmpan, A.: Core Based Incremental Clustering, Studia Universitatis "Babeş-Bolyai", Informatica, **L(1)** (2005) 89–96
14. Tan, P.-N., Steinbach, M., Kumar, V., Introduction to Data Mining. Addison Wesley, chapters 8,9, (2005)
15. Wolberg, W., Mangasarian, O. L.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Proceedings of the National Academy of Sciences, U.S.A., Volume **87**, December (1990) 9193–9196
16. Wu, F., Gardarin, G., Gradual Clustering Algorithms. Proceedings of the 7th International Conference on Database Systems for Advanced Applications (DASFAA'01), (2001) 48–57