Metric Incremental Clustering of Nominal Data

Dan Simovici
University of Massachusetts at Boston,
Department of Computer Science,
Boston, Massachusetts 02125, USA,
dsim@cs.umb.edu

Namita Singla
University of Massachusetts at Boston,
Department of Computer Science,
Boston, Massachusetts 02125, USA,
namita@cs.umb.edu

Michael Kuperberg
Karlsruhe University,
Department of Computer Science,
Karlsruhe, Germany,
Michael.Kuperberg@informatik.uni-karlsruhe.de

Abstract

We present an algorithm for clustering nominal data that is based on a metric on the set of partitions of a finite set of objects; this metric is defined starting from a lower valuation of the lattice of partitions. The proposed algorithm seeks to determine a clustering partition such that the total distance between this partition and the partitions determined by the attributes of the objects has a local minimum. The resulting clustering is quite stable relative to the ordering of the objects.

1 Introduction

Clustering is an unsupervised learning process that partitions data such that similar data items are grouped together in sets referred to as clusters. This activity is important for condensing and identifying patterns in data. Despite the substantial effort invested in researching clustering algorithms by the data mining community, there are still many difficulties to overcome in building clustering algorithms. Indeed, as pointed in [16] “there is no clustering technique that is universally applicable in uncovering the variety of structures present in multidimensional data sets”. This situation has generated a variety of clustering techniques broadly divided into hierarchical and partitional; also, special clustering algorithms based on a variety of principles, ranging from neural networks and genetic algorithms, to tabu searches.

In this paper we focus on an incremental clustering algorithm that can be applied to nominal data, that is, to data whose attributes have no particular natural ordering. In general clustering, objects to be clustered are represented as points in an n-dimensional space $\mathbb{R}^n$ and standard distances, such as the Euclidean distance is used to evaluate similarity between objects. For objects whose attributes are nominal (e.g., color, shape, diagnostic, etc.), no such natural representation of objects is possible, which leaves only the Hamming distance as a dissimilarity measure, a poor choice for discriminating among multi-valued attributes of objects.

Incremental clustering has attracted a substantial amount of attention starting with Hartigan’s algorithm [15] implemented in [7]. A seminal paper by D. Fisher [14] contained COBWEB, an incremental clustering algorithm that involved restructurings of the clusters in addition to the incremental additions of objects. Incremental clustering related to dynamic aspects of databases were discussed in [5, 6]. It is also notable that incremental clustering has been used in a variety of applications [19, 20, 8, 11]. The interest in incremental clustering stems from the fact that the main memory usage is minimal since there is no need to keep in memory the mutual distances between objects and the algorithms are scalable with respect to the size of the set of objects and the number of attributes.

An *object system* is a pair $\mathcal{S} = (S, H)$, where S is set called the set of objects of $\mathcal{S}$, $H = \{A_1, \ldots, A_m\}$ is a set of mappings defined on S. We assume that for each mapping A_i (referred to as an attribute of $\mathcal{S}$) there exists a nonempty set E_i called the domain of A_i such that $A_i : S \longrightarrow E_i$ for $1 \leq i \leq m$. The value of an attribute A_i on an object t is denoted by $t[A_i]$. Our terminology is consistent with the terminology used in relational databases, where a table can be regarded as an object system; however, the notion of object system is more general because objects have an identity as members of the set S, instead of being regarded as just m-tuples of values. In this spirit, we shall refer to $t[A_i]$ as *projection of t on A_i*.

Let S be a set. A partition on S is a non-empty collection of subsets of S indexed by a set I, $\pi = \{B_i \mid i \in I\}$ such that $\bigcup_{i\in I} B_i = S$ and $i \neq j$ implies $B_i \cap B_j = \emptyset$. The sets B_i are commonly referred to as the *blocks of the partition* π. The set of partitions on S is denoted by $\mathsf{PART}(S)$.

An attribute A of an object system $\mathcal{S} = (S, H)$ generates a partition π^A of the set of objects S, where two objects belong to the same block of π^A if they have the same projection on A. We denote by B^A_a the block of π^A that consists of all tuples of S whose A-component is a. Note that for relational databases, π^A is the partition of the set of rows of a table that is obtained by using the **group by** A option of **select** in standard SQL.

A clustering of an object system $\mathcal{S} = (S, H)$ is defined as a partition κ of S. We shall seek to find clusterings starting from their relationships with partitions induced by attributes. As we shall see, this is a natural approach for nominal data.

2 Metrics on Object Partitions

The set of partitions of a set can be naturally equipped with a partial order. For $\pi, \sigma \in \mathsf{PART}(S)$ we write $\pi \leq \sigma$ if every block B of π is included in a block of σ, or equivalently, if every block of σ is an exact union of blocks of π.

This partial order generates a lattice structure on $\mathsf{PART}(S)$; this means that for every two partitions $\pi, \pi' \in \mathsf{PART}(S)$ there is a least partition π_1 such that $\pi \leq \pi_1$ and $\pi' \leq \pi_1$ and there is a largest partition π_2 such that $\pi_2 \leq \pi$ and $\pi_2 \leq \pi'$. The first partition is denoted by $\pi \vee \pi'$, while the second is denoted by $\pi \wedge \pi'$.

To introduce a metric on the set of partitions of a finite set we define the mapping $v : \mathsf{PART}(S) \longrightarrow \mathbb{R}$ by $v(\pi) = \sum_{i=1}^n |B_i|^2$, where $\pi = \{B_1, \ldots, B_n\}$.

The mapping v is a lower valuation on $\mathsf{PART}(S)$, that is,

$$v(\pi \vee \sigma) + v(\pi \wedge \sigma) \geq v(\pi) + v(\sigma) \qquad (1)$$

for $\pi, \sigma \in \mathsf{PART}(S)$ (see Appendix A for a proof).

For every lower valuation v the mapping $d : (\mathsf{PART}(S))^2 \longrightarrow \mathbb{R}$ defined by $d(\pi, \sigma) = v(\pi) + v(\sigma) - 2v(\pi \wedge \sigma)$ is a metric on $\mathsf{PART}(S)$ (see [2, 1, 21]). A special property of this metric allows the formulation of an incremental clustering algorithm.

3 AMICA - A Metric Incremental Clustering Algorithm

Let $\mathcal{S} = (S, H)$ be an object system. We seek a clustering $\kappa = \{C_1, \ldots, C_n\} \in \mathsf{PART}(S)$ such that the total distance from κ to the partitions of the attributes:

$$D(\kappa) = \sum_{i=1}^{n} d(\kappa, \pi^{A_i})$$

is minimal.

The definition of d allows us to write:

$$d(\kappa, \pi^A) = \sum_{i=1}^{n} |C_i|^2 + \sum_{j=1}^{m_A} |B^A_{a_j}|^2 - 2 \sum_{i=1}^{n}\sum_{j=1}^{m_A} |C_i \cap B^A_{a_j}|^2,$$

Suppose now that t is a new object, $t \notin S$, and let $Z = S \cup \{t\}$. The following cases can occur:

1. the object t is added to an existing cluster C_k;
2. a new cluster, C_{n+1} is created that consists only of t.

Also, from the point of view of partition π^A, t is added to the block $B^A_{t[A]}$, which corresponds to the value $t[A]$ of the A-component of t.

In the first case let:

$$\begin{aligned} \kappa_{(k)} &= \{C_1, \ldots, C_{k-1}, C_k \cup \{t\}, C_{k+1}, \ldots, C_n\} \\ \pi^{A'} &= \{B^A_{a_1}, \ldots, B^A_{t[A]} \cup \{t\}, \ldots, B^A_{a_{m_A}}\} \end{aligned}$$

be the partitions of Z. Now, we have:

$$\begin{aligned} & d(\kappa_{(k)}, \pi^{A'}) - d(\kappa, \pi^A) \\ &\quad = (|C_k| + 1)^2 - |C_k|^2 + (|B^A_{t[A]}| + 1)^2 \\ &\qquad - |B^A_{t[A]}|^2 - 2(2|C_k \cap B^A_{t[A]}| + 1) \\ &\quad = 2|C_k| + 1 + 2|B^A_{t[A]}| + 1 - 4|C_k \cap B^A_{t[A]}| - 2 \\ &\quad = 2|C_k \oplus B^A_{t[A]}|. \end{aligned}$$

The minimal increase of $d(\kappa_{(k)}, \pi^{A'})$ is given by:

$$\min_k \sum_A 2|C_k \oplus B^A_{t[A]}|.$$

In the second case we deal with the partitions:

$$\begin{aligned} \kappa' &= \{C_1, \ldots, \ldots, C_n, \{t\}\} \\ \pi^{A'} &= \{B^A_{a_1}, \ldots, B^A_{t[A]} \cup \{t\}, \ldots, B^A_{a_{m_A}}\} \end{aligned}$$

and we have

$$d(\kappa', \pi^{A'}) - d(\kappa, \pi^A) = 2|B^A_{t[A]}|.$$

Consequently,

$$D(\kappa') - D(\kappa) = \begin{cases} \sum_A 2 \cdot |C_k \oplus B^A_{t[A]}| & \text{in Case 1} \\ \sum_A 2 \cdot |B^A_{t[A]}| & \text{in Case 2.} \end{cases}$$

Thus, the choice between adding an object to an existing cluster and creating a new cluster is based on comparing the numbers

$$\min_k \sum_A |C_k \oplus B^A_{t[A]}| \text{ and } \sum_A |B^A_{t[A]}|.$$

If the first number is smaller, we add t to a cluster C_k for which $\sum_A |C_k \oplus B^A_{t[A]}|$ is minimal; otherwise, we create a new one-object cluster.

Incremental clustering algorithms are affected, in general, by the order in which objects are processed by the clustering algorithm. Moreover, as pointed in [9], each such algorithm proceeds typically in a hill-climbing fashion that yields local minima rather than global ones. For some incremental clustering algorithms certain object orderings may result in rather poor clusterings. To diminish the ordering effect problem we expand the initial algorithm by adopting the "not-yet" technique introduced by Roure and Talavera in [23]. The basic idea is that a new cluster is created only when the inequality:

$$r(t) = \frac{\sum_A |B^A_{t[A]}|}{\min_k \sum_A |C_k \oplus B^A_{t[A]}|} < \alpha,$$

is satisfied, that is, only when the effect $r(t)$ of adding the object t on the total distance is significant enough. Here α is a parameter provided by the user, such that $\alpha <= 1$. Note that if $\alpha = 1$, we make no use of the NOT-YET buffer.

We formulate now a metric incremental clustering algorithm (referred to as AMICA – an acronym of the previous five words) that is using the properties of distance d. The variable nc denotes the current number of clusters.

If $\alpha < r(t) \leq 1$, then we place the object t in a NOT-YET buffer. If $r(t) \leq \alpha$ a new cluster that consists of the object $\{t\}$ is created. Otherwise, that is if $r(t) > 1$, the object t is placed in an existing cluster C_k that minimizes $\sum_A |C_k \oplus B^A_{t[A]}|$; this limits the number of new singleton clusters that would be otherwise created. After all objects of the set S have been examined, the objects contained by the NOT-YET buffer are processed with $\alpha = 1$. This prevents new insertions in the buffer and results in either placing these objects in existing clusters or in creating new clusters. The pseudocode of the algorithm is given next:

```
**Input:** data set $S$ and threshold $\alpha$
**Output:** clustering $C_1, \ldots, C_{nc}$
**Method:**
$nc = 0$;
$\ell = 1$;
while $S \neq \emptyset$ do
      select an object $t$;
      $S = S - \{t\}$;
      if $\sum_A |B^A_{t[A]}| \leq \alpha \min_{1 \leq k \leq nc} \sum_A |C_k \oplus B^A_{t[A]}|$
         then
            nc ++;
            create a new single-object
         cluster $C_{nc} = \{t\}$;
         else
            $r(t) = \frac{\sum_A |B^A_{t[A]}|}{\min_{1\leq k \leq nc} \sum_A |C_k \oplus B^A_{t[A]}|}$
      if $r(t) > 1$
         then
            $k =$ arg min$_k \sum_A |C_k \oplus B^A_{t[A]}|$
            add $t$ to cluster $C_k$;
         else /* this means $\alpha < r(t) \leq 1$ */
            place $t$ in NOT-YET buffer;
      end if;
endwhile;
process objects in the NOT-YET buffer
as above with $\alpha = 1$;
```

4 Experimental Results

We applied AMICA to synthetic data sets produced by an algorithm that generates clusters of objects having real-numbered components grouped around a specified number of centroids. The resulting tuples were discretized using a specified number of discretization intervals which allowed us to treat the data as nominal. The experiments were applied to several data sets with an increasing number of tuples and increased dimensionality and using several permutations of the set of objects. All experiments describe in this paper used $\alpha = 0.95$.

The stability of the obtained clusterings is quite remarkable. For example, in an experiment applied to a set that consists of 10,000 objects (grouped by the synthetic data algorithm around 6 centroids) a first pass of the algorithm produced 11 clusters; however, most objects (9895) are concentrated in the top 6 clusters, which approximate very well the "natural" clusters produced by the synthetic algorithm.

The next table compares the clusters produced by the first run of the algorithm with the cluster produced from a data set obtained by applying a random permutation.

Initial Run		Random Permutation		
Cluster	Size	Cluster	Size	Distribution (Original cluster)
1	1548	1	1692	1692 (2)
2	1693	2	1552	1548 (1), 3 (3), 1 (2)
3	1655	3	1672	1672 (5)
4	1711	4	1711	1711 (4)
5	1672	5	1652	1652 (3)
6	1616	6	1616	1616 (6)
7	1	7	85	85 (8)
8	85	8	10	10 (9)
9	10	9	8	8 (10)
10	8	10	1	1 (11)
11	1	11	1	1 (7)

Note that the clusters are stable; they remain almost invariant with the exception of their numbering. Similar results were obtained for other random permutations and collections of objects.

As expected with incremental clustering algorithms, the time requirements scale up very well with the number of tuples. On an IBM T20 system equipped with a 700 MHz Pentium III and with a 256 MB RAM, we obtained the following results for three randomly chosen permutations of each set of objects.

Number of objects	Time for 3 permutations (ms)			Average time (ms)
2000	131	140	154	141.7
5000	410	381	432	407.7
10000	782	761	831	794.7
20000	1103	1148	1061	1104

Another series of experiments involved the application of the algorithm to databases that contain nominal data. We applied AMICA to the mushroom data set from the standard UCI data mining collection (see [4]). The data set contains 8124 mushroom records and is typically used as test set for classification algorithms. In classification experiments the task is to construct a classifier that is able to predict the poisonous/edible character of the mushrooms based on the values of the attributes of the mushrooms.

We discarded the class attribute (poisonous/edible) and applied AMICA to the remaining data set. Then, we identified the edible/poisonous character of mushrooms that are grouped together in the same cluster. This yields the clusters $C_1, \ldots, C_9$:

Cl. num.	Poisonous/Edible	Total	Percentage of dominant group
1	825/2752	3577	76.9%
2	8/1050	1058	99.2%
3	1304/0	1304	100%
4	0/163	163	100%
5	1735/28	1763	98.4%
6	0/7	7	100%
7	0/192	192	100%
8	36/16	52	69%
9	8/0	8	100%

Note that in almost all resulting clusters there is a dominant character, and for five out of the total of nine clusters there is complete homogeneity.

A study of the stability of the clusters similar to the one performed for synthetic data shows the same stability relative to input orderings as follows from the next two tables that describe clusterings obtained under two randomly chosen permutations:

C_i	Computed Clusters First Random Permutation									
	C'_1	C'_2	C'_3	C'_4	C'_5	C'_6	C'_7	C'_8	C'_9	C'_{10}
	3540	1797	1095	192	1296	8	36	7	137	16
3577	3540	0	37	0	0	0	0	0	0	0
1058	0	0	1058	0	0	0	0	0	0	0
1304	0	8	0	0	1296	0	0	0	0	0
163	0	26	0	0	0	0	0	0	137	0
1763	0	1763	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	7	0	0
192	0	0	0	192	0	0	0	0	0	0
52	0	0	0	0	0	0	36	0	0	16
8	0	0	0	0	0	8	0	0	0	0

and

C_i	Computed Clusters Second Random Permutation								
	C'_1	C'_2	C'_3	C'_4	C'_5	C'_6	C'_7	C'_8	C'_9
	3548	1809	1052	192	1296	165	52	8	2
3577	3548	29	0	0	0	0	0	0	0
1058	0	6	1052	0	0	0	0	0	0
1304	0	8	0	0	1296	0	0	0	0
163	0	0	0	0	0	163	0	0	0
1763	0	1763	0	0	0	0	0	0	0
7	0	3	0	0	0	2	0	0	2
192	0	0	0	192	0	0	0	0	0
52	0	0	0	0	0	0	52	0	0
8	0	0	0	0	0	0	0	8	0

Note that the previous tables contains mostly zeros. This shows that the clusters remain essentially stable under input data permutations (with the exception of the order in which they are created).

5 Conclusion and Future Work

AMICA provides good quality, stable clusterings for nominal data, an area of clustering that is less explored than the standard clustering algorithms that act on ordinal data. Clusterings produced by the algorithm show a rather low sensitivity to input orderings.

Further investigations in the behavior of the algorithm are warranted. For example, we ran AMICA with a rather high value of the threshold $\alpha = 0.95$. Future work will include an examination of the dependency of the maximal size of the `NOT-YET` buffer for various values of α.

AMICA could be combined with special discretization algorithms such as fixed k-interval discretization [10], metric discretization [24], fuzzy discretization (see [17, 18]), Shannon-entropy discretization [12, 13], proportional k-interval discretization (see [25, 26]), or techniques that are capable of dealing with highly dependent attributes (cf.[22]) to obtain a more general incremental clustering algorithm applicable to mixed data, that is, to data having both nominal and ordinal attributes.

A A proof of inequality (1)

Let π, σ be two partitions of the finite set S, such that $\pi = \{B_1, \ldots, B_m\}$ and $\sigma = \{C_1, \ldots, C_n\}$. It is known (see [3], for example) that $\pi \wedge \sigma$ consists of all sets of the form $B_i \cap C_j$ such that $B_i \cap C_j \neq \emptyset$. On another hand, $\pi \vee \sigma$ has a more complicated description; namely, $x, y \in S$ belong to the same block D of $\pi \vee \sigma$ if there exists a

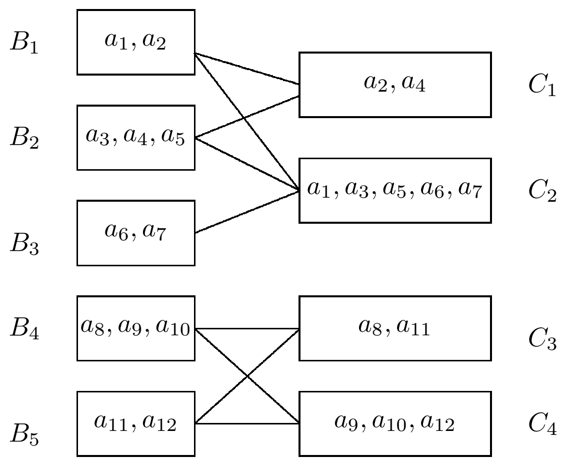

Figure 1. The graph $G_{\pi,\sigma}$

sequence of elements of S, $z_0, \ldots, z_k$ such that $x = z_0$, $z_k = y$ and for each pair (z_p, z_{p+1}) there is a block B_i of π or a block C_j of σ such that both z_p and z_{p+1} belong to B_i or to C_j for $1 \leq p \leq k-1$.

Consider the bipartite graph $G_{\pi,\sigma}$ whose set of vertices consists of the blocks of π and the blocks of σ. An edge (B_i, C_j) exists only if $B_i \cap C_j \neq \emptyset$. If $\mathcal{K}$ is a connected component of this graph it is easy to see that $\bigcup\{B_i \in \pi \mid B_i \in \mathcal{K}\} = \bigcup\{C_j \in \sigma \mid C_j \in \mathcal{K}\}$. Further, each block D of $\pi \vee \sigma$ equals the union of the blocks of π (or the blocks of σ) that belong to a connected component $\mathcal{K}$ of $G_{\pi,\sigma}$.

Example A.1 Let $S = \{a_i \mid 1 \leq i \leq 12\}$ and let $\pi = \{B_i \mid 1 \leq i \leq 5\}$ and $\sigma = \{C_j \mid 1 \leq j \leq 4\}$, where

$$\begin{array}{ll}
B_1 = \{a_1, a_2\}, & C_1 = \{a_2, a_4\}, \\
B_2 = \{a_3, a_4, a_5\}, & C_2 = \{a_1, a_3, a_5, a_6, a_7\}, \\
B_3 = \{a_6, a_7\}, & C_3 = \{a_8, a_{11}\}, \\
B_4 = \{a_8, a_9, a_{10}\}, & C_4 = \{a_9, a_{10}, a_{12}\}, \\
B_5 = \{a_{11}, a_{12}\}. &
\end{array}$$

The graph $G_{\pi,\sigma}$ shown in Figure 1 has two connected components that correspond to the blocks

$$\begin{array}{rcl}
D_1 & = & \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\} \\
& = & B_1 \cup B_2 \cup B_3 \\
& = & C_1 \cup C_2, \\
D_2 & = & \{a_8, a_9, a_{10}, a_{11}, a_{12}\} \\
& = & B_4 \cup B_5 \\
& = & C_3 \cup C_4.
\end{array}$$

of the partition $\pi \vee \sigma$.

The partition $\pi \wedge \sigma$ consists of 9 blocks that correspond to the edges of the graph:

$$\begin{array}{ll}
B_1 \cap C_1 = \{a_2\}, & B_1 \cap C_2 = \{a_1\}, \\
B_2 \cap C_1 = \{a_4\}, & B_2 \cap C_2 = \{a_3, a_5\}, \\
B_3 \cap C_2 = \{a_6, a_7\}, & \\
B_4 \cap C_3 = \{a_8\}, & B_4 \cap C_4 = \{a_9, a_{10}\}, \\
B_5 \cap C_3 = \{a_{11}\}, & B_5 \cap C_4 = \{a_{12}\}.
\end{array}$$

$\square$

Let $D_1, \ldots, D_r$ be the blocks of the partition $\pi \vee \sigma$. For a block D_k define the sets $I_k \subseteq \{1, \ldots, m\}$ and $J_k \subseteq \{1, \ldots, n\}$ where $I_k = \{i \mid B_i \cap D_k \neq \emptyset\}$ and $J_k = \{j \mid B_i \cap D_k \neq \emptyset\}$. Note that

$$\begin{array}{rcl}
v(\pi \vee \sigma) & = & \sum_{k=1}^{r} |D_k|^2, \\
v(\pi \wedge \sigma) & = & \sum_{k=1}^{r} \sum_{i \in I_k} \sum_{j \in J_k} |B_i \cap C_j|^2, \\
v(\pi) & = & \sum_{k=1}^{r} \sum_{i \in I_k} |B_i|^2 \\
& = & \sum_{k=1}^{r} \sum_{i \in I_k} \left(\sum_{j \in J_k} |B_i \cap C_j| \right)^2, \\
v(\sigma) & = & \sum_{k=1}^{r} \sum_{j \in J_k} |C_j|^2 \\
& = & \sum_{k=1}^{r} \sum_{j \in J_k} \left(\sum_{i \in I_k} |B_i \cap C_j| \right)^2.
\end{array}$$

It is immediate to verify the inequality:

$$\left(\sum_{i \in I_k} \sum_{j \in J_k} |B_i \cap C_j|\right)^2 + \sum_{i \in I_k} \sum_{j \in J_k} |B_i \cap C_j|^2 \\ \geq \sum_{i \in I_k} \left(\sum_{j \in J_k} |B_i \cap C_j|\right)^2 + \\ \sum_{j \in J_k} \left(\sum_{i \in I_k} |B_i \cap C_j|\right)^2$$

This is equivalent to

$$|D_k|^2 + \sum_{i \in I_k} \sum_{j \in J_k} |B_i \cap C_j|^2 \\ \geq \sum_{i \in I_k} \left(\sum_{j \in J_k} |B_i \cap C_j|\right)^2 + \\ \sum_{j \in J_k} \left(\sum_{i \in I_k} |B_i \cap C_j|\right)^2$$

Adding up the similar inequalities for $1 \leq k \leq r$ we have

$$v(\pi \vee \sigma) + v(\pi \wedge \sigma) \geq v(\pi) + v(\sigma),$$

which is the desired inequality.

References

[1] J. Barthélemy. Remarques sur les propriétés metriques des ensembles ordonnés. *Math. Sci. hum.*, 61:39–60, 1978.

[2] J. Barthélemy and B. Leclerc. The median procedure for partitions. In *Partitioning Data Sets*, pages 3–34, Providence, 1995. American Mathematical Society.

[3] G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, 1973.

[4] C. L. Blake and C. J. Merz. *UCI Repository of machine learning databases*. University of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[5] F. Can. Incremental clustering for dynamic information processing. *ACM Transaction for Information Systems*, 11:143–164, 1993.

[6] F. Can, E. A. Fox, C. D. Snavely, and R. K. France. Incremental clustering for very large document databases: Initial MARIAN experience. *Inf. Sci.*, 84:101–114, 1995.

[7] G. Carpenter and S. Grossberg. Art3: Hierachical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, 3:129–152, 1990.

[8] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *STOC*, pages 626–635, 1997.

[9] A. Cornuéjols. Getting order independence in incremental learning. In *European Conference on Machine Learning*, pages 196–212, 1993.

[10] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proc. of the 12th International Conference on Machine Learning*, pages 194–202, 1995.

[11] M. Ester, H. P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *VLDB*, pages 323–333, 1998.

[12] U. M. Fayyad. *On the Induction of Decision Trees for Multiple Concept Learning*. PhD thesis, University of Michigan, 1991.

[13] U. M. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classifi cation learning. In *Proc. of the 12th International Joint Conference on Artifi cial Intelligence*, pages 1022–1027, 1993.

[14] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[15] J. A. Hartigan. *Clustering Algorithms*. John Wiley, New York, 1975.

[16] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31:264–323, 1999.

[17] I. Kononenko. Naive bayes classifi er and continuous attributes. *Informatica*, 16:1–8, 1992.

[18] I. Kononenko. Inductive and Bayesian learning in medical diagnosis. *Applied Artifi cial Intelligence*, 7:317–337, 1993.

[19] T. Langford, C. G. Giraud-Carrier, and J. Magee:. Detection of infectious outbreaks in hospitals through incremental clustering. In *Proceedings of the 8th Conference on AI in Medicine (AIME)*, pages 30–39. Springer, 2001.

[20] J. Lin, M. Vlachos, E. J. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *EDBT*, pages 106–122, 2004.

[21] B. Monjardet. Metrics on parially ordered sets – a survey. *Discrete Mathematics*, 35:173–184, 1981.

[22] M. Robnik and I. Kononenko. Discretization of continuous attributes using relieff. In *Proc. of ERK-95*, pages 149–152, 1995.

[23] J. Roure and L. Talavera. Robust incremental clustering with bad instance orderings: A new strategy. In *IBERAMIA*, pages 136–147, 1998.

[24] D. Simovici and R. Butterworth. A metric approach to supervised discretization. In *Extraction et Gestion des Connaisances (EGC'2004)*, pages 197–202, Toulouse, France, 2004. Cépaduès-Éditions.

[25] Y. Yang and G. I. Webb. Proportional k-interval discretization for naive-Bayes classifi ers. In *Proc. of the 12th European Conference on Machine Learning*, pages 564–575, 2001.

[26] Y. Yang and G. I. Webb. Weighted proportional k-interval discretization for naive-Bayes classifi ers. In *Proc. of the PAKDD*, 2003.