A Novel Clustering Method Based on Spatial Operations

Hui Wang

School of Computing and Mathematics, University of Ulster at Jordanstown
Newtownabbey, BT37 0QB, Northern Ireland, UK
H.Wang@ulster.ac.uk

Abstract. In this paper we present a novel clustering method that can deal with both numerical and categorical data with a novel clustering objective and without the need of a user specified parameter. Our approach is based on an extension of database relation – hyperrelations. A hyperrelation is a set of hypertuples, which are vectors of sets.

In this paper we show that hyperrelations can be exploited to develop a new method for clustering both numerical and categorical data. This method merges hypertuples pairwise in the direction of increasing the density of hypertuples. This process is fully automatic in the sense that no parameter is needed from users. Initial experiments with artificial and real-world data showed this novel approach is promising.

1 Introduction

The clustering of data is to organise data by abstracting the underlying structure of the data, either as a grouping of objects or as a hierarchy of groups. The representation can then be investigated to see if these data groups accord to preconceived ideas or to suggest new experiments (1). The objective of clustering is simply to find a convenient and valid organisation of the data. Clustering algorithms are geared toward finding structure in the data, organised around *clusters*. A cluster is comprised of a number of similar objects collected or grouped together.

In the context of knowledge discovery from databases, clustering is the process of discovering a set of categories to which objects should be assigned. Clustering algorithms are required to discover distinct categories using an unlabeled set of data. Objects in the dataset are then assigned (often as a by-product of the clustering process) to these categories.

Most of the existing clustering algorithms are either for numerical data only or for categorical data only. In the case of mixed data (that is, some attributes are numerical while others are categorical) the *numerical-only* clustering algorithms have to treat categorical attributes as numerical in some ways; while the *categorical-only* algorithms have to treat numerical attributes as categorical in some ways (2). Many existing clustering algorithms also needs some parameters from users. For example, the number of clusters (one of the most common parameters demanded from users), the neighbourhood radius and minimum number of points (3), and the number of sub-cells into which to partition a cell (4).

D. Bell and J. Hong (Eds.): BNCOD 2006, LNCS 4042, pp. 140–151, 2006.

It would then be desirable to have a clustering algorithm which can treat numerical and categorical data uniformly and which needs little input from users. In this paper we present a method for clustering having this trait. Our method is based on an extension of database relation.

The extension of a relation is underpinned by a mathematical structure called a *domain lattice* (5), which underlies any relational data scheme. A domain lattice is a set of all *hypertuples* in a problem domain, equipped with a partial ordering. A hypertuple is a vector of sets; in contrast, a database tuple is a vector of single values in its basic form. A *hypertuple* is then a generalisation of a database tuple. A *hyperrelation* is a set of hypertuples. The concept of relations in database theory and applications can be generalised to hyperrelations. Domain lattice has previously been exploited to address the data reduction problem in data mining (5).

In this paper we show that the hyperrelations and the domain lattice can be further exploited to develop a new method for clustering both numerical and categorical data uniformly. In Section 2 we introduce some notation and concepts for use in this paper, including hyperrelations and domain lattice. The central notion of our approach – density of hypertuples – is introduced in Section 3. Section 4 examines three fundamental issues about clustering from our density perspective. An efficient algorithm for clustering is presented and analysed in Section 5. Experimental results are reported in Section 6. Section 7 concludes the paper.

2 Definitions and Notation

Figure 1 illustrates the concepts of simple relations, hyperrelations, and domain lattice, which we define formally below.

2.1 Order and Lattices

A *partial order* on a set $\mathcal{L}$ is a binary relation $\leq$ which is reflexive, antisymmetric and transitive. A *semilattice* $\mathcal{L}$ is a nonempty partially ordered set such that for $x, y \in \mathcal{L}$ the least upper bound $x + y$ exists. Then for $A \subseteq \mathcal{L}$, its least upper bound exists and is denoted by $\text{lub}(A)$. The greatest element of $\mathcal{L}$, if it exists, is denoted by 1; if $\mathcal{L}$ is finite then 1 exists, and it is equal to $\sum_{a \in \mathcal{L}} a$.

The sub-lattice of $\mathcal{L}$ generated from $M \subseteq \mathcal{L}$, written by $[M]$, is $[M] = \{t \in \mathcal{L} : \exists X \subseteq M \text{ such that } t = \text{lub}(X)\}$. The greatest element in $[M]$ is $\text{lub}(M)$.

For $A, B \subseteq \mathcal{L}$, we say A *is covered by* B (or B *covers* A), written $A \preccurlyeq B$, if for each $a \in A$ there is some $b \in B$ such that $a \leq b$. We write $A \prec B$ if $A \preccurlyeq B$ and $B \not\preccurlyeq A$.

2.2 Domain Lattice

Let $\mathbf{D}$ be a relation with a schema $\Omega = \{x_1, \cdots, x_T\}$ and domains V_x of attributes $x \in \Omega$.

Let $\mathcal{L} \stackrel{\text{def}}{=} \prod_{x \in \Omega} 2^{V_x}$. Then $\mathcal{L}$ is a semilattice (in fact, it is a Boolean algebra, but we will not need this here) under the ordering

$$\forall t, s \in \mathcal{L}, t \leq s \Longleftrightarrow t(x) \subseteq s(x) \text{ for all } x \in \Omega. \tag{1}$$

with the least upper bound or the *sum* of $t, s \in \mathcal{L}$ given below

$$t + s \stackrel{\text{def}}{=} \langle t(x) \cup s(x) \rangle_{x \in \Omega} \tag{2}$$

If $t \leq s$ we say t is *below* s. $\mathcal{L}$ is called *domain lattice* for $\mathbf{D}$. The elements of $\mathcal{L}$ are called *hypertuples*; in particular, the elements t of $\mathcal{L}$ with $|t(x)| = 1$ for all $x \in \Omega$ are special hypertuples called *(simple) tuples*. A set of hypertuples is called a *hyperrelation*, and a set of simple tuples is called a *(simple) relation*. Simple relations are database relations in the traditional sense.

Note that $t(x)$ is the projection of tuple t onto attribute x. In practical terms, $t(x)$ can be treated as a set if x is a categorical attribute, and it can be treated as an interval if x is numerical.

In domain lattice $\mathcal{L}$, $\mathbf{D}$ is the set of simple tuples given in the dataset. There is a natural embedding of $\mathbf{D}$ into $\mathcal{L}$ by assigning

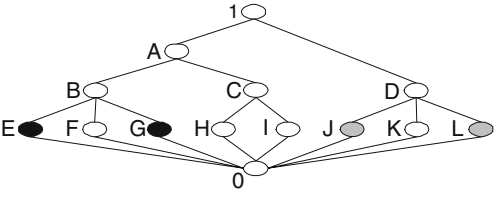
$$\Omega(a) \mapsto \langle \{x_1(a)\}, \{x_2(a)\}, \ldots, \{x_T(a)\} \rangle.$$

and we shall identify $\mathbf{D}$ with result of this embedding. Thus we have $\mathbf{D} \subseteq \mathcal{L}$.

In the sections below our discussion focuses mainly on a subset of the domain lattice (sublattice) $[\mathbf{D}]$ generated from the dataset $\mathbf{D}$.

Node	Size	Colour	Shape	Label
E	Large	Red	Triangle	+
G	Large	Blue	Circle	+
J	Small	Blue	Triangle	-
L	Small	Green	Rectangle	-

(a)



(b)

Node	Size	Colour	Shape	Label
B	Large	{Blue,Red}	{Circle,Triangle}	+
D	Small	{Blue,Green}	{Rectangle,Triangle}	-

(c)

Fig. 1. (a) A relation extension. (b) An abstract *domain lattice* implied in the relation. (c) A hyper relation.

3 Density of Hypertuples

In the sequel we shall use $\mathbf{D}$ as described above as a generic dataset, $\mathcal{L}$ as the domain lattice implied in $\mathbf{D}$, and $[\mathbf{D}]$ as the sublattice generated from $\mathbf{D} \subseteq \mathcal{L}$. The sum operation and partial ordering are as defined in the previous section.

Clustering is a partition $\mathcal{P}$ of $\mathbf{D}$ with classes $\{\mathbf{D}_1, \cdots, \mathbf{D}_K\}$. Each class $\mathbf{D}_i$ is a subset of data objects in $\mathbf{D}$ and is called a *cluster*. In traditional approaches to clustering a cluster is represented by the set of objects in the cluster, or by the center of gravity of the cluster (k-means) or by one of the objects of the cluster located near its center (k-medoid) (6). In our approach we represent a cluster by a hypertuple. For each class $\mathbf{D}_i$ we merge all simple tuples in $\mathbf{D}_i$ by the lattice sum operation in Eq. 2 resulting in a hypertuple $h_i = \text{lub}(\mathbf{D}_i)$. Then we get a hyperrelation $H = \{h_1, \cdots, h_K\}$. Therefore we can take clustering as a process of transforming a simple relation ($\mathbf{D}$) into a hyperrelation (H). Putting this formally, a clustering of $\mathbf{D}$ is a hyperrelation $H \subseteq [\mathbf{D}]$ and therefore $h \in H$ implies there is $A \subseteq \mathbf{D}$ such that $h = \text{lub}(A)$.

Clearly there are many possible partitions of the dataset. To choose one from among them we need a measure of hyperrelations. In our approach we use the measure of density, defined below.

Definition 3.1. *Let $h \in [\mathbf{D}]$ be a hypertuple, and $x \in \Omega$ be an attribute. The* magnitude *of $h(x)$ is defined as*

$$\text{mag}(h(x)) = \begin{cases} \max(h(x)) - \min(h(x)), \text{ if } x \text{ is numerical} \\ |h(x)|, \text{ if } x \text{ is categorical} \end{cases} \tag{3}$$

Note that $h(x)$ is the projection of h onto attribute x, $\min(h(x))$ is the minimal value in $h(x)$ while $\max(h(x))$ is the maximal value.

Definition 3.2. *The* volume *of h is defined as*

$$\text{vol}(h) = \prod_{x \in \Omega} \text{mag}(h(x)) \tag{4}$$

The coverage *of h is* $\text{cov}(h) \stackrel{\text{def}}{=} \{d \in \mathbf{D} : d \leq h\}$.

Definition 3.3. *The* density *of h is defined as*

$$\text{den}(h) = |\text{cov}(h)| / \text{vol}(h) \tag{5}$$

The density *of hyperrelation H,* $\text{den}(H)$*, is then the average density of the hypertuples in H.*

The above definition of density can not be directly applied to compare different hypertuples since different hypertuples may differ at different attributes, and different attributes may have different scales. Therefore we need to re-scale the attributes up to a same *uniform scale*. The re-scaling can be achieved as follows.

Table 1. A relation on the scheme $\{A_1, A_2\}$ where attribute A_1 is categorical and A_2 is numerical

	A_1	A_2
t_0	a	2
t_1	f	10
t_2	c	4
t_3	f	9
t_4	c	3
t_5	e	7
t_6	b	1
t_7	d	6

Let λ be the expected uniform scale. For an attribute $x \in \Omega$, the *re-scaling coefficient* is $s(x) \stackrel{\text{def}}{=} \lambda / \operatorname{mag}(V_x)$. Note that V_x is the domain of attribute x. Then the volume of a hypertuple h after re-scaling is $\operatorname{vol}(h) = \prod_{x \in \Omega} s(x) \times \operatorname{mag}(h(x))$. The density definition can be re-scaled similarly.

This re-scaled notion of density is fine for hypertuples. But there is a problem for simple tuples. Consider Table 1. Suppose the uniform scale is 2. Then $s(A_1) = 2/3$ and $s(A_2) = 1$. Following the above definition, the density for all simple tuples is 0 since the projection of each simple tuple to (numerical) attribute A_2 contains only one value. This is not desirable, the reason for which will be seen in the next section. Therefore we need a method to allocate density values to simple tuples in such a way that the values can be compared with the density values of hypertuples for the purpose of clustering. Our solution is through *quantization of attributes*. For an attribute $x \in \Omega$ the measurement of a unit after re-scaling is $|V_x|/\lambda = 1/s(x)$. For a tuple t, if $t(x)$ is less than the unit value $(1/s(x))$ it should be treated as one unit. If t is a simple tuple then $t(x)$ is treated as a unit for all $x \in \Omega$ and hence $\operatorname{vol}(t) = 1$. Since a simple tuple covers only itself, i.e., $\operatorname{cov}(t) = \{t\}$, we have $\operatorname{den}(t) = 1$. Consequently $\operatorname{den}(H) = 1$ if H is a *simple* relation. If t is a hypertuple, then $\operatorname{den}(t)$ may be greater or less than 1.

In the rest of this paper whenever we talk of density we refer to the re-scaled and quantized density.

The notion of density is also used in some well known clustering methods (7; 3; 4), but their uses of this notion are different from ours: they are defined for numerical attributes only and they are not re-scaled and quantized. Our definition of density applies to both numerical and categorical attributes and, since re-scaled and quantized, can be used to compare among hypertuples and among hyperrelations.

4 Merging Hypertuples to Increase Density

Having a notion of density as defined above we now present our clustering method. Our philosophy for clustering is *merging tuples to increase the density of hyperrelations*: for any set of tuples, if their sum has higher density then

we are inclined to merge them and use their sum to replace this set of tuples. We discuss our method along three fundamental axes regarding any clustering methods: *the criteria for clustering, the determination of the number of clusters* and *the assignment of new tuples to clusters*.

4.1 Criteria for Clustering

An important notion in clustering is *neighbourhood* (or *similarity*). "Similar" objects should be clustered together. In the context of domain lattice, "similar" tuples should end up in same hypertuples. The meaning of neighbourhood varies in different approaches and contexts. For example, the Euclidean distance (or L_p metric [1] in general) and density (7; 8) for numerical data; and the *Jaccard coefficient* [2] (9; 1), the *links* [3] (10), the *co-occurence* in hypergraph (11; 2) [4], the *interestingness* (12) and the *share* (13) for categorical data .

Clustering is then to optimise one or more of these measures one way or another. In hierarchical clustering there is a basic operation — *merge*: two data objects can be merged if they are *neighbouring* or *close enough*. A prerequisite for this approach is the availability of a *proximity matrix* (1). In the case of numerical data this matrix is obtained by some distance measure, e.g., Euclidean distance; in the case of categorical data it is usually not available.

Our approach is hierarchical, and two tuples are deemed neighbours if the density of their sum (see Eq. 2) is higher than the density of the hyperrelation containing only the two tuples (i.e., the average density of the two tuples). More formally, let $\mathbf{D}$ be the dataset and $\mathcal{H} = \{H : H \text{ is a clustering of } \mathbf{D}\}$. Our objective is to find $H_0 \in \mathcal{H}$ such that $\text{den}(H_0) > \text{den}(\mathbf{D})$ and $\text{den}(H_0) = \max\{\text{den}(H) : H \in \mathcal{H}\}$. In other words our expected clustering should have the highest possible density. We call this H_0 the *optimal clustering* of $\mathbf{D}$. From the previous section we know that $\text{den}(\mathbf{D}) = 1$ and hence $\text{den}(H_0)$ should be much greater than or equal to 1.

The optimal clustering of the data in Table 1 is shown in Table 2. Readers can check for themselves that any other hyperrelations obtained by merging simple tuples in the dataset using the lattice sum operation has lower density. For example, merging $\{t_0, \cdots, t_3\}$ and $\{t_4, \cdots, t_7\}$ results in a hyperrelation in Table 3, which has lower density.

With such a criteria we can obtain a proximity matrix for any relational data, no matter it is numerical, categorical or mixed. Table 4 is a proximity matrix for the data in Table 1, where entry (i, j) is 1 if $\text{den}(t_i + t_j) \geq \text{den}(\{t_i, t_j\})$ and 0 otherwise.

[1] $L_p = (\sum_1^d |x_i - y_i|^p)^{1/p}$, $1 \leq p \leq \infty$ and d is the dimensionality of the data points.

[2] The Jaccard coefficient for similarity between two sets S_1 and S_2 is $|S_1 \cap S_2|/|S_1 \cup S_2|$.

[3] The number of links between a pair of data points is the number of common neighbours for the points.

[4] In this approach each tuple in the database is viewed as a set of data objects, and the entire collection of tuples is treated as a hypergraph. The *co-occurence* between two tuples is the number of common elements and is noted as the weight of the edge between the two hyper notes.

Table 2. The optimal clustering of the relation in Table 1 obtained by our method. The uniform scale used is 4, so the re-scaling coefficients are $s(A_1) = 2/3$ and $s(A_2) = 4/9$. The density of this hyperrelation is then 1.313. Note that the density values are re-scaled, and that the density for the original dataset is 1.

	A_1	A_2	Coverage cov()	Density den()
t'_0	$\{a, b, c\}$	$\{1, 2, 3, 4\}$	$\{t_0, t_2, t_4, t_6\}$	1.500
t'_1	$\{d, e, f\}$	$\{6, 7, 9, 10\}$	$\{t_1, t_3, t_5, t_7\}$	1.125

Table 3. An arbitrary hyperrelation obtained by merging simple tuples in Table 1. The uniform scale used is the same as in Table 2, so are the re-scaling coefficients. The density of this hyperrelation is 0.5625.

	A_1	A_2	Coverage cov()	Density den()
t''_0	$\{a, c, f\}$	$\{2, 4, 9, 10\}$	$\{t_0, t_1, t_2, t_3\}$	0.5625
t''_1	$\{b, c, d, e\}$	$\{1, 3, 6, 7\}$	$\{t_4, t_5, t_6, t_7\}$	0.5625

Table 4. A proximity matrix for the relation in Table 1

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
t_0	1	0	1	0	1	0	1	0
t_1	0	1	0	1	0	1	0	0
t_2	1	0	1	0	1	1	1	1
t_3	0	1	0	1	0	1	0	1
t_4	1	0	1	0	1	0	1	1
t_5	0	1	1	1	0	1	0	1
t_6	1	0	1	0	1	0	1	0
t_7	0	0	1	1	1	1	0	1

This approach has a major advantage: *numerical and categorical attributes can be treated uniformly.* Simple tuples, either numerical or categorical or a mixture of the two, can thus be *uniformly* measured for their neighbourhood.

4.2 Determination of the Number of Clusters

Some clustering algorithms require the number of clusters be given by users. Our approach can determine the number of clusters automatically; it can also be tuned to find a required number of clusters as stated by a user.

As discussed in the previous section our criteria for clustering is to maximise the density of hyperrelations. Naturally the *optimal* number of clusters should be the number of hypertuples in the optimal clustering. Whether or not we can find the optimal clustering depends on the algorithm used.

For the data in Table 1, its optimal clustering is shown in Table 2. Therefore the optimal number of clusters for this dataset is 3.

In some data mining exercises, however, we have a preconceived (given) number of clusters and we wish the clustering algorithm to find the given number of

clusters for us. Examples are: in business systems, to allocate stock to a given number of warehouses, or to allocate tuples to a given number of disk volumes in physical database design (14; 15). Some clustering algorithms require the number of clusters be given in this way.

Suppose we wish to find N clusters. With the availability of a proximity matrix we can take an agglomerative hierarchical approach. Assume we have a hierarchy of clusterings (hyperrelations) of the dataset, $Q_0, Q_1, \cdots, Q_q$, where $Q_0 = \mathbf{D}$, $Q_q = \text{lub}(\mathbf{D})$ and $|Q_i| = |Q_{i-1}| - 1$. Clearly we can select a hyperrelation Q_k in the hierarchy such that $|Q_k| = N$.

4.3 Assignment of New Data Tuples to Clusters

Suppose we have already clustered a dataset, resulting in a clustering $H = \{h_1, \cdots, h_K\}$ where each h_i is a hypertuple. When new data arrives we may need to assign the new data objects to the existing clusters to get a new clustering[5]; or we may want to assign new data objects to clusters for data mining purposes like categorisation, classification or association. In traditional approaches to clustering this is usually done via calculating distances (or proximities) between a new data object and the clusters using some metric and assigning the new data object to whichever cluster is closest to the new data object (1). In our approach we have available proximity matrices so we can do it in a similar way, but in a more general context (since our method works for both numerical and categorical data). Specifically, we sum the new data object with each cluster and see if, and to what extent, the sum increases density. If none of the sums increases density, then it is likely the new data object belongs to a new cluster; otherwise the new data object is assigned to one cluster the sum of which with the new data object has the greatest increase of density.

More formally, let t be a new data object – a simple tuple. We have two main steps for the assignment procedure:

- If $\text{den}(h_i + t) < \text{den}(\{h_i, t\})$ for all $i = 1, \cdots, K$, then t is taken as a new cluster on its own.
- Otherwise assign t to cluster h_{i_0} such that $(\text{den}(h_{i_0} + t) - \text{den}(\{h_{i_0}, t\}))/\text{den}(\{h_{i_0}, t\})$ is highest.

For an example, consider Table 2. If we are given a new tuple $t = \langle b, 8\rangle$, then we have $\text{den}(t'_0 + t) = 0.804$ and $\text{den}(t'_1 + t) = 1.055$. However $\text{den}(\{t'_0, t\}) = 1.25$ and $\text{den}(\{t'_1, t\}) = 1.063$. This indicates that the new tuple should stand as a new cluster. If the new tuple is $t = \langle b, 5\rangle$, then $\text{den}(t'_0 + t) = 1.406$ and $\text{den}(t'_1 + t) = 0.844$, and $\text{den}(\{t'_0, t\})$ and $\text{den}(\{t'_1, t\})$ remain the same. This indicates that we should assign this new tuple to t'_0.

5 An Efficient Algorithm for Clustering

Based on the above discussions we designed an efficient clustering algorithm, LM/Clus. The following is an outline of the algorithm.

[5] This is in fact incremental clustering (8).

- Input: $\mathbf{D}$ as defined above.
- Initialisation: $Q_0 = \mathbf{D}$; $flag = 1$; $i = 0$;
- WHILE $(flag = 1)$
  1. $flag = 0$; $H = Q_i$;
  2. WHILE (there are $x, y \in Q_i$ that haven't been examined)
     Let $w = x + y$;
     IF $(\text{den}(w) > \text{den}(\{x, y\})$
     THEN $Q_{i+1} = Q_i \cup \{w\} \setminus \{x, y\}$; $i = i + 1$; $flag = 1$;
     //`Replace` x `and` y `by their sum.`
- Output: H.

Execution starts with $Q_0 = \mathbf{D}$. Then a pair of elements $x, y \in Q_0$ such that $x + y$ increases density is merged, resulting in Q_1. The next loop starts from Q_1 and results in Q_2. This process continues until Q_m where no pair of elements can be merged in this way. Thus we get a sequence $Q_0, Q_1, \cdots, Q_m$, where Q_0 is the original dataset and Q_m is the quasi-optimal clustering. Clearly $Q_0 \preccurlyeq Q_1 \preccurlyeq \cdots \preccurlyeq Q_m$, and $|Q_i| = |Q_{i-1}| - 1$. Therefore this is an agglomerative hierarchical clustering algorithm (1).

This algorithm has a worst case complexity of $O(n \log n)$ where $n = |\mathbf{D}|$. In our implementation of the algorithm we take advantage of the operations in our Lattice Machine (5) so the average computational complexity is close to linear (see below). The algorithm is implemented as part of our Lattice Machine based KDD suite, called DR .

6 Experimental Results

In this section we present experimental results showing the effectiveness and efficiency of our clustering algorithm LM/CLUS. We used two types of data: artificial data and real world data. These datasets are a good mix of numerical and categorical data. Artificial datasets were generated from known clusters with added noise, and they are mainly used to show the effectiveness of the algorithm (i.e., can the known clusters be discovered?) as well as efficiency. The data generator we used is also available in our DR system. Real world datasets are public and are frequently used in KDD literature. They are used in our experiment mainly to show the efficiency of the algorithm since the underlying clusters are not known.

6.1 Artificial Datasets

We used two seeds to generate our artificial datasets. The seeds are described in Table 5. For each seed we generated four datasets of varying sizes with 2% random noise added, and with $100, 1000, 5000, 10000$ tuples respectively. The time used to cluster these data is shown in Table 6. From this table we can see that the algorithm is close to linear in the number of tuples. The underlying cluster structures were fully recovered. Readers are invited to evaluate our system which is available online (for the Web address see the footnote on page 148).

Table 5. Two seeds used to generate artificial data

	*Attribute*1	*Attribute*2	*Attribute*3
Cluster 1	$[0, 4]$	$[100, 130]$	$\{a, b, c\}$
Cluster 2	$[6, 10]$	$[160, 199]$	$\{c, d, e\}$

(a) Seed one: gd1. The first two attributes are numerical and the third is categorical.

	*Attribute*1	*Attribute*2	*Attribute*3	*Attribute*4
Cluster 1	$[0, 4]$	$[100, 130]$	$[1000, 1300]$	$\{a, b, c\}$
Cluster 2	$[6, 10]$	$[160, 199]$	$[1400, 1650]$	$\{d, e, f\}$
Cluster 3	$[3, 7]$	$[140, 150]$	$[1750, 1999]$	$\{c, d\}$

(b) Seed two: gd2. The first three attributes are numerical and the fourth is categorical.

Table 6. Time in seconds used to cluster the artificial data

	gd1.100x2	gd1.1000x20	gd1.5000x100	gd1.10000x200
Time	0.44	3.84	22.24	93.65

	gd2.100x2	gd2.1000x20	gd2.5000x100	gd2.10000x200
Time	1.15	10.27	61.24	229.98

Table 7. Some general information about the real world data and the time in seconds used to cluster the data

Dataset	#Attributes	#Size	#Numeric Attribute	#Categorical Attribute	Clustering Time
German	20	1000	6	14	820.14
Heart	13	270	9	4	56.68
Iris	4	150	4	0	2.53

6.2 Real World Datasets

We chose three public datasets to show the efficiency of the algorithm for clustering: `German Credit`, `Heart Disease` and `Iris`, all available from UCI Machine Learning Data Repository. Some general information about the datasets and clustering time are shown in Table 7.

7 Conclusion

In this paper we present a novel method of automatically clustering both numerical and categorical data or mixed data in a uniform way. The first major contribution of this paper is the provision of a uniform measure of density for both numerical and categorical data. After re-scaling and quantization this measure can be used to compare among any (simple or hyper) tuples and among

any (simple or hyper) relations to see which is denser. Since the density measure is *local*, its calculation is very efficient. Based on this measure our clustering method is simply to transform simple relations (original data) to hyperrelation guided by the density measure. Data tuples are merged with the aim of increasing the density of hyperrelations. The optimal clustering is the hyperrelation with highest possible density, and the number of hypertuples in this hyperrelation is the *optimal* number of clusters.

Another major contribution of this paper is the provision of an efficient algorithm for clustering, LM/Clus. This algorithm is (agglomerative) hierarchical and it takes advantage of our (local) measure of density. It examines pairs of tuples and merges those which increase the density of the relation. This process continues until the density of the relation cannot be increased. Experiments with both artificial data and real world data showed that this algorithm is very efficient and is, in average, close to linear in the number of data tuples. Experiments with the artificial datasets showed that this algorithm is also effective as it recovers completely the underlying cluster structures used to generate the datasets.

Bibliography

[1] Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs, New Jersey (1988)
[2] Gibson, D., Kleinberg, J., Raghavan, P.: Clustering categorical data: An approach based on dynamical systems. In: Proc. 24th International Conference on Very Large Databases, New York (1998)
[3] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press (1996) 226–231
[4] Wang, W., Yang, J., Muntz, R.: STING: A statistical information grid approach to spatial data mining. In: Proc. 23rd Int. Conf. on Very Large Databases, Morgan Kaufmann (1997) 186–195
[5] Wang, H., Düntsch, I., Bell, D.: Data reduction based on hyper relations. In: Proceedings of KDD98, New York. (1998) 349–353
[6] Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons (1990)
[7] Schikuta, E.: Grid clustering: an efficient hierarchical clustering method for very large data sets. In: Proc. 13th Int. Conf. on Pattern Recognition. Volume 2., IEEE Computer Society Press (1996) 101–105
[8] Ester, M., Kriegel, H.P., Sander, J., Wimmer, M., Xu, X.: Incremental clustering for mining in a data warehousing environment. In: Proc. 24th International Conference on Very Large Databases. (1998)
[9] Duda, R.O., Hart, P.E.: Pattern classification and scene analysis. John Wiley & Sons (1973)
[10] Guha, S., Rastogi, R., Shim, K.: ROCK: A robust clustering algorithm for categorical attributes. Technical Report 208, Bell Laboratories (1998)

[11] Han, E.H., Karypis, G., Kumar, V., Mobasher, B.: Clustering based on association rule hypergraphs. In: 1997 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. (1997)
[12] Gray, B., Orlowska, M.E.: Clustering categorical attributes into interesting association rules. In: Proc. PAKDD98. (1998)
[13] Hilderman, R.J., Carter, C.L., Hamilton, H.J., Cercone, N.: Mining market basket data using share measures and characterized itemsets. In: Proc. PAKDD98. (1998)
[14] Bell, D.A., McErlean, F., Stewart, P., Arbuckle, W.: Clustering related tuples in databases. Computer Journal **31**(3) (1988) 253–257
[15] Stewart, P., Bell, D.A., McErlean, F.: Some aspects of a physical database design and reorganisation tool. Journal of Data and Knowledge Engineering (1989) 303–322