Data reduction based on hyper relations

Hui Wang*, Ivo Düntsch*, David Bell*

School of Information and Software Engineering,

University of Ulster,

Newtownabbey, BT 37 0QB, N.Ireland, UK

{H.Wang,I.Duentsch,DA.Bell}@ulst.ac.uk

Abstract

Data reduction makes datasets smaller but preserves classification structures of interest. It is an important area of research in data mining and databases.

In this paper we present a novel approach to data reduction based on hyper relations. Our method is a generalization of the data filtering method introduced by [4] for one attribute to many attributes. Hyper relations are a generalization of conventional database relations in the sense that we allow sets of values as tuple entries. The advantage of this is that raw data and reduced data can both be represented by hyper relations. The collection of hyper relations can be naturally made into a complete Boolean algebra, and so for any collection of hyper tuples we can find its unique least upper bound (lub) as a reduction of it. However the lub may not qualify as a reduced version of the given set of tuples. Then we turn to find the interior cover – the subset of internal elements covered by the lub. We establish the theoretical result that such an interior cover exists, and we present a method by which we can find it. This interior serves as a reduced version of the data.

The proposed method was evaluated using 7 real world datasets with respect to its test accuracy. The results were quite remarkable in that the cross validated test accuracies were substantially higher than those obtained by C4.5 in 6 out of 7 datasets. The datasets were reduced with reduction ratios up to 99%.

Keywords: data mining, data reduction, hyper relation, interior cover, lattice.

*Equal authorship is implied

1 Introduction

Data reduction is a process which is used to transform raw data into a more condensed form without losing significant semantic information [7]. In data mining, data reduction in a stricter sense refers to feature selection and data sampling [11]. But in a broader sense, data reduction is regarded as a main task of data mining [5] hence any data mining technique can be regarded as a method for data reduction.

In [5], a fundamental question was raised: "Are there meaningful general data reduction techniques that can help people effectively visualize and understand massive datasets? "

In this paper we attempt to answer the above question from an algebraic perspective, with an aim to provide a solution which can potentially be embedded in databases as a "data mining operation". We show that data and models can be represented uniformly by *hyper relations* – a generalization of database relations in the traditional sense. The collection of hyper relations can be turned into a complete Boolean algebra in a natural way; in this algebra a given dataset with known classes can be represented as a partial labeling of the lattice.

Data reduction is taken to be a process to find a set of elements – *interior cover* – in the lattice, which is, in a sense, closest to the least upper bound of the labeled elements, and which is such that all elements in the interior cover are *equilabeled*. This "equilabeledness" guarantees that the original labeling of the lattice is fully preserved, and is also generalized to some other originally unlabeled elements. We prove a theorem showing that the interior cover exists. We also present a method to find the interior cover.

Therefore data reduction in this paper is interpreted as a process to reduce the size of datasets while preserving classification structure. Now we look at an example.

Example 1.1. *To illustrate this sort of data reduction that we are pursuing, let's consider the dataset in Table 1. Take A_3 as the classification attribute, and A_1 and A_2 as predicting attributes. Merging the tuples in sets $\{t_0, t_3\}$, $\{t_1, t_2, t_4, t_5\}$, and $\{t_6, t_7\}$ will preserve the classification labels of the original tuples. This leads to a reduced version shown in Table 2, which clearly agrees with the examples in the original dataset.*

The decision trees generated from the original dataset and the reduced dataset are shown in Figures 1 and 2. We hold that the decision tree in Figure 2 is simpler (in the obvious connectivity sense) and hence easier for humans to understand than the one in Figure 1.

This type of data reduction is useful in the worlds of very large databases and data mining for the following reasons. It reduces the storage requirements of data used mainly for classification; it offers better understandability for the knowledge discovered; it allows feature selection and continuous attribute discretization to be achieved as by-products of data reduction; and it allows computationally demanding algorithms to become serious contenders in the search of knowledge discovery methods (e.g., Bayesian networks).

R	A_1	A_2	A_3
t_0	a	1	0
t_1	a	2	1
t_2	a	3	1
t_3	b	1	0
t_4	b	2	1
t_5	b	3	1
t_6	c	2	0
t_7	c	3	0

Table 1: *A relation on the scheme $\{A_1, A_2, A_3\}$ where A_3 is the classification attribute.*

R'	A_1	A_2	A_3
t'_0	$\{a, b\}$	$\{1\}$	0
t'_1	$\{a, b\}$	$\{2, 3\}$	1
t'_2	$\{c\}$	$\{2, 3\}$	0

Table 2: *A reduced version of the relation in Table 1.*

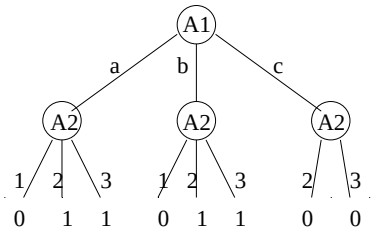

Figure 1: *The decision tree generated from the relation in Table 1 by C4.5 without pruning.*

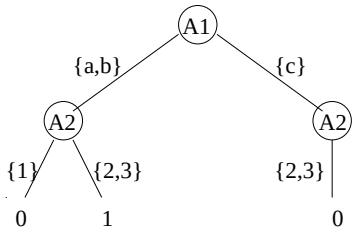

Figure 2: *The decision tree generated from the relation in Table 2 by C4.5 without pruning.*

2 Definitions and notation

Lattice

Suppose that $\mathcal{P} = \langle P, \leq \rangle$ is a partially ordered set, and $T \subseteq P$. We let $\uparrow T = \{y \in P : (\exists x \in T)\ x \leq y\}$. If $T = \{a\}$, we will write $\uparrow a$ instead of $\uparrow \{a\}$; more generally, if no confusion can arise, we shall usually identify singletons with the element they contain. Similarly, we define $\downarrow T$.

A *lattice* L is a partially ordered set such that for each $x, y \in L$ the least upper bound $x \vee y$ and the greatest lower bound $x \wedge y$ exist, and L has a smallest element 0 and a largest element 1. An *atom of* L is an element $a > 0$ with $\downarrow a = \{a, 0\}$. For $M \subseteq L$ we let $At(M) \stackrel{\text{def}}{=} \{a \in \downarrow M : a \text{ is an atom of } L\}$.

For $a, b \in L$, if $a \leq b$ we usually say that a *is below* b. Given $X, Y \subseteq L$, we say X *is covered by* Y *(or* Y *covers* X*)*, written $X \preccurlyeq Y$, if for any $x \in X$ there is $y \in Y$ such that $x \leq y$; in particular, if $X \subseteq Y$, then $X \preccurlyeq Y$.

Conversely, we say X *is dense for* Y, written $X \trianglelefteq Y$, if for any $y \in Y$ there is $x \in X$ such that $x \leq y$.

The reader is invited to consult [6] for unexplained notation and concepts in lattice theory.

Hyper relations

Suppose that U is a finite set of attributes; with each $A \in U$ we associate an attribute domain denoted by $DOM(A)$. Denote the power set (family of all subsets) of $DOM(A)$ by V_A. We denote by $\mathcal{T}$ the Cartesian product of all the power sets, i.e., $\prod_{A \in U} V_A$. We call $\mathcal{T}$ a *universal hyper relation* over U. A *hyper tuple* t over U is an element of $\mathcal{T}$ and its $A \in U$ entry is denoted by $t(A)$. A *hyper relation* is a subset of $\mathcal{T}$, and is usually denoted by R^U.

A hyper tuple is called *simple*, if all its entries have cardinality of 1; a hyper relation is called *simple*, if all its tuples are simple. Simple hyper relations correspond to conventional database relations [1, 10]. Table 2 is an example of a hyper relation, while Table 1 is an example of simple hyper relation.

It can be shown that $\mathcal{T}$ is a lattice under the following ordering:

$$t_1 \leq t_2 \stackrel{\text{def}}{\iff} t_1(A) \subseteq t_2(A),$$

for all $A \in U$. As a product of Boolean algebras, $\mathcal{T}$ is a Boolean algebra.

Atoms have entries $\emptyset$ in all but one place; the nonempty entry has exactly one element.

Given $t_1, t_2 \in \mathcal{T}$, the *hyper similarity* of t_1 to t_2, written $S(t_1, t_2)$, is the number of $A \in U$ such that $t_1(A) \leq t_2(A)$. Clearly, in general, $0 \leq S(t_1, t_2) \leq |U|$; if $t_1 \leq t_2$, $S(t_1, t_2) = |U|$.

3 Data reduction via interior covers

Data reduction can be achieved with universal hyper relations in the following way. Suppose we have a dataset represented as a simple hyper relation R: some tuples are classified as 0 and others as 1 and no tuple is classified as both 0 and 1. We want to reduce the size of it while preserving its classification structure.

Let R_i be the set of all hyper tuples classified as being in class $i \in \{0, 1\}$, and r_i its lub in $\mathcal{T}$, i.e. for each $A \in U$,

$$r_i(A) = \bigcup_{t \in R_i} t(A).$$

We can try to find a set of hyper tuples which, in a sense, is closest to the respective lub but preserves the classification structure. This closest set of hyper tuples will later be called the "interior" contained in the lub. To present our results we need to introduce the following concepts in the context of lattice, not just universal hyper relations.

The classification structure of a dataset in the universal hyper relation can be formally interpreted in general terms as a partial labeling of a lattice.

Definition 3.1 (Lattice labeling). Let $G \subseteq L$. A *labeling of* L is a partial mapping $l :\downarrow G \to B$, where B is a finite set, such that $l(a) = l(b)$ for any $a, b \in \downarrow g$ and $g \in G$.

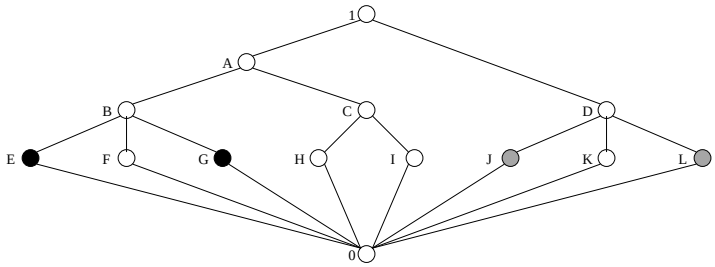

Figure 3: *A labeled lattice.*

G above can be interpreted as a dataset, and B as the set of classes. The functional nature of l guarantees that no element in L is labeled differently. This amounts to assuming that datasets are consistent.

The preservation of a lattice labeling is characterized as follows. Given a labeled lattice, we are interested in sublattices such that in each sublattice, the elements either have the same label or are unlabeled. The unlabeled elements can assume the same labeling as the labeled elements in the sublattice as a generalization. Then the largest elements of these sublattices can be used to represent the lattice labeling. In the context of universal hyper relation, this amounts to reducing the dataset while preserving the classification structure.

Such sublattices and, in particular, the largest elements in them are what we are interested in and they are characterized through the following concepts.

Definition 3.2. $\mathcal{P}(l)$ is the natural partition of G by the function l, i.e. two elements are in the same class iff they have the same value under l.

Definition 3.3. $m \in L$ is *equilabeled* if $\downarrow m$ intersects exactly one class of $\mathcal{P}(l)$.

Note that every element of G is equilabeled.

In sum, the lattice labeling can be represented by a set of equilabeled elements, accompanied by the following simple rule:

> Given an equilabeled element, any elements below it will have the same label as the equilabeled element (if any).

However, for a labeled lattice, there are many equilabeled elements. We certainly cannot use all of them. Consider the lattice in Figure 3. Elements A and B are both equilabeled elements. If we don't want to keep both of them, which one should be preferred? Certainly A has greater coverage of unlabeled elements than B, and thus, we can think of B as being of lower complexity in the sense that it is simpler to describe than A, given E and G; in fact B is the lub of E and G. In the spirit of Occam's razor [12], given a set of labeled elements, we prefer a simple generalization. This leads to the following definitions.

Definition 3.4. An *E-cover* is $a \in L$ such that a is the lub of $F \subseteq G$ and a is equilabeled. A pair of E-covers, m and n, is said to be *mergeable* if $m \vee n$ is also an E-cover. An *E-set* is a set of E-covers which are pairwise non-mergeable.

The set of all E-covers is written $\mathcal{E}$. If two E-covers are comparable, they are certainly mergeable. Also, each element of G is an E-cover, and an E-cover is a singleton E-set.

Now our focus is on E-covers instead of individual elements in a lattice. Our next question is: given a collection S of E-covers (e.g., a dataset with known classes), what is the expected (simpler) representation of the lattice labeling? Clearly the lub of S is ideal if it is also an E-cover because, if so, the labeling of these E-covers can be represented by the single lub. Unfortunately this lub may not be an E-cover. But instead we can try to find a set of E-covers which together is, in a sense, closest to the lub of S. Look at Figure 4. Consider $S \stackrel{\text{def}}{=} \{H, I, J, K\}$. There are two sets of E-covers which are below $\operatorname{lub} X$ and cover X: $Y_1 \stackrel{\text{def}}{=} \{A, C\}$ and $Y_2 \stackrel{\text{def}}{=} \{A, B, C\}$. We argue that there is no reason not to include B and hence we prefer Y_2 to Y_1. In general, we expect a maximal set of such E-covers: a collection of all E-covers satisfying the above conditions. We call this expected set of E-covers the *interior cover* of S.

Definition 3.5. The *interior cover* of $A \subseteq \mathcal{E}$, written $E(A)$, is a maximal E-set $B \subseteq \mathcal{E}$ such that $A \preccurlyeq B \preccurlyeq \operatorname{lub} A$ and $A \trianglelefteq B$.

Lemma 3.1. *Let $A \subseteq \mathcal{E}$ and B be an interior cover of A. Then $X \preccurlyeq B$ for any $X \subseteq \mathcal{E}$ such that $A \preccurlyeq X \preccurlyeq \operatorname{lub} A$ and $A \trianglelefteq X$.*

Proof. Consider any $x \in X$. Since $A \trianglelefteq X$, there must be $a \in A$ such that $a \leq x$. Since $A \preccurlyeq B$, there must be $b \in B$ such that $a \leq b$. We need to show that there is $b' \in B$ such that $x \leq b'$. Since $a \leq b$ and $a \leq x$, there are only three possible cases:

- $x \leq b$: obviously $b' = b$.
- $b \leq x$: this means that b is mergeable with b' leading to x, i.e., $b \vee b' = x$. This contradicts the assumption that B is an E-set. Therefore this case is impossible.
- b and x are incomparable: due to the assumption that B is an E-set, they are non-mergeable. Since B is the maximal set of E-covers by definition of interior cover, it follows that $x \in B$. Therefore $b' = x$.

□

Then an important question arises: does the interior cover exist? Now we set out to answer this question.

The following theorem establishes the existence of this interior, and illustrates a way to construct the interior.

Theorem 3.1. *The interior cover of any set of E-covers exists.*

Definition 3.6. Let $A, B \subseteq L$. Then,

$$A + B = \{a \vee b : a \in A,\ b \in B\}$$

is called the *complex sum of A and B*. $\max(A)$ is the set of maximal elements of A, and $Eq(A)$ is the set of all equilabeled elements of A.

Proof. Apply the following procedure to $A \subseteq \mathcal{E}$:

1. $M_0 \stackrel{\text{def}}{=} A$.
2. $C_0 \stackrel{\text{def}}{=} \max(Eq(M_0) \cup A)$.
3. $M_{n+1} \stackrel{\text{def}}{=} C_n + C_n,\ C_{n+1} \stackrel{\text{def}}{=} \max(Eq(M_{n+1}) \cup C_n)$.
4. Continue until $C_n = C_{n+1}$.

Let $C = C_n = C_{n+1}$. Each $c \in C$ is equilabeled, and $A \preccurlyeq C$. Assume that $s, t \in C_n$ such that $s + t$ is equilabeled. Then, $s + t \notin C_n$, because of maximality, and $s + t \in Eq(M_{n+1}) \setminus C_n$, contradicting that $C_n = C_{n+1}$. Thus, C is an E–set.

It is clear that $A \preccurlyeq C \preccurlyeq \operatorname{lub} A$ and $A \trianglelefteq C$, and all E-covers which densely cover A are included in C hence maximal. Therefore the interior cover exists. □

Theorem 3.1 indicates a way to construct the interior of any collection of E-covers. Algorithms based on this theorem can be designed easily.

4 Worked examples

We now illustrate the use of the above theorem and its implied algorithm using some examples. First we present a simple illustration using an abstract lattice.

Example 4.1. *Consider the labeled lattice shown in Figure 4. Let* $X = \{H, I, J, K, M, O\}$. *Then the partial labeling function is* $X \to \{darkblack, lightblack\}$. *The set of E-covers is* $\{H, I, J, K, M, O, A, B, C, \}$. *Following the algorithm in Theorem 3.1 we get the interior* $Y = \{A, B, C, M, O\}$. *Clearly* $X \preccurlyeq Y$. *Any other collection* Y' *of E-covers such that* $X \preccurlyeq Y'$ *is covered by* Y. *For example,* $Y' = \{A, C, E, M\}$ *covers* X *and, clearly,* $Y' \preccurlyeq Y$.

Now we look at an example in the context of universal hyper relation.

Example 4.2. *Consider a relation scheme* $\{X, Y, Z\}$, *where* Z *is the classification attribute,* $DOM(X) = \{a, b, c, d, e\}$ *and* $DOM(Y) = \{1, 2, 3, 4, 5\}$. *Suppose that we have a dataset as in Table 3(a), which is a hyper relation. Following the algorithm in Theorem 3.1 we get the interior cover shown in Table 3(b). The interior serves as reduced versions of the original datasets.*

Relating these examples with the pattern of usage in Example 1.1, the promise of this reduction technique (e.g., to get understandable decision trees, or reduce storage space and execution time) can readily be appreciated.

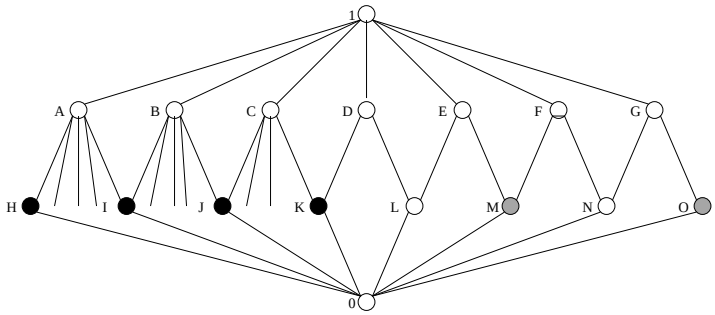

Figure 4: *A lattice and its labeling.*

	X	Y	Z
t_0	$\{a, b\}$	$\{1, 2\}$	0
t_1	$\{b, c\}$	$\{2, 3\}$	0
t_2	$\{c, d\}$	$\{3, 4\}$	0
t_3	$\{d, e\}$	$\{4, 5\}$	0
t_4	$\{d\}$	$\{1\}$	1
t_5	$\{a\}$	$\{5\}$	1

(a)

	A_1	A_2	A_3
t'_0	$\{a, b, c\}$	$\{1, 2, 3\}$	0
t'_2	$\{b, c, d, e\}$	$\{2, 3, 4, 5\}$	0
t'_3	$\{d\}$	$\{1\}$	1
t'_4	$\{a\}$	$\{5\}$	1

(b)

Table 3: *(a) A relation on the scheme $\{X, Y\}$ and Z is the labeling (classification) attribute; (b) an interior cover.*

5 Data reduction as an approach to data mining

As mentioned in [5], in a general sense any data mining algorithms can be regarded as a process of data reduction. What we discussed above is aimed at reducing data while preserving its classification structure. This method can in turn be used as an approach to data mining – building models from data.

In this approach, both data and models are represented as hyper relations, though almost all datasets we use in data mining are usually simple relations – a special case of hyper relation. Training process is to find the interior cover of given data in the context of universal hyper relations. Recall that an interior cover is a set of pairwise non-mergeable E-covers, and an E-cover is a hyper tuple which covers a set of (simple or hyper) tuples equally labeled. The procedure has been given in the proof of Theorem 3.1. Since a hyper tuple is simply a vector of sets, classification can be done via set inclusion. Specifically, suppose we have a set, M, of hyper tuples with known classes – the result of data reduction – and a simple tuple, d, the class of which is unknown. The classification of t is done using the ordering $\leq$ in the universal hyper relation as follows.

- If $d \leq m$ for $m \in M$, then the class of d is identified with that of m.
- If there are multiple such m, then select the one which has the greatest coverage of simple tuples resulting from the data reduction process, and identify the class of d with that of this hyper tuple.

- If there is no such m, then select the one which has the greatest hyper similarity value, and identify the class of d with that of this hyper tuple.

We have implemented a system, called **DR**, which can reduce a dataset resulting in a model of it, and classify unlabeled data using the model. The data reduction part is a straightforward implementation of the procedure described in the proof of Theorem 3.1 and the classification part is based on the above procedure. DR was evaluated using real world datasets and is reported in the following section.

6 Experiment and evaluation

Objective of the experiment

The ultimate goal of data reduction is in improving learning performance. So the objective of the experiment is set to evaluate the proposed data reduction method to see how well it performs in prediction with real world datasets. This is measured in terms of test accuracies using cross validation. The results are compared with those by standard data mining methods.

Datasets description

We chose 7 public datasets: Australian Credit, Diabetes, Heart, Iris, German Credit, Tic-Tac-Toe, and Vote from the UCI machine learning repository [9, 8]. The tuples with missing values in Tic-Tac-Toe and Vote are deleted. Some general information about these datasets is given in Table 4.

Experiment procedure

To achieve our objective, we need a standard data mining algorithm for benchmarking. We chose C4.5 as it is one of the most extensively used algorithms in independent research in the literature and it is easily available so that the experiment results can be easily repeated, if needed. We used the C4.5 module in the Clementine package [2] in our experiment.

The evaluation method we used is 5-fold cross validation for both C4.5 and DR. We observed the test accuracies for both methods. We also recorded the reduction ratio by DR. The reduction ratio we used is defined as (the number of tuples in the original datasets - the number of hyper tuples in the model) / (the number of tuples in the original datasets). The results are shown in Table 5.

Discussion of the experiment results

From Table 5 we see that our DR algorithm outperforms C4.5 with respect to the cross validated test accuracy for all datasets but Vote. The reason for this is that the Vote dataset is binary, i.e., all attributes have only binary values, and there is no reduction

Datasets	NoA	NoN	NoO	NoB	NoE	NoC	CD
Australian	14	4	6	4	690	2	383 : 307
Diabetes	8	0	8	0	768	2	268 : 500
Heart	13	3	7	3	270	2	120 : 150
Iris	4	0	4	0	150	3	50 : 50 : 50
German	20	11	7	2	1000	2	700 : 300
Tic-Tac-Toe	9	9	0	0	958	2	332 : 626
Vote	18	0	0	18	232	2	108 : 124

Table 4: *General information about the datasets. The acronyms above are: NoA – Number of attributes, NoN – Number of Nominal attributes, NoO – Number of Ordinal attributes, NoB – Number of Binary attributes, NoE – Number of Examples, NoC – Number of Classes, and CD – Class Distribution*

Dataset	Test accuracy with C4.5	Test accuracy with DR	Reduction Ratio by DR
Australian	85.2	87.0	70.6
Diabetes	72.9	78.6	68.6
Heart	77.1	83.3	74.1
Iris	94.0	96.7	94.0
German	70.5	78.0	73.1
Tic-Tac-Toe	86.2	86.9	81.5
Vote	96.1	87.0	99.1
Average	83.1	85.4	80.1

Table 5: *Test accuracies with C4.5 and DR, and the reduction ratios obtained by DR. The evaluation method we used is 5-fold cross validation. C4.5 we used is implemented in the Clementine package.*

possible because the partitions obtained from binary attributes are co–atoms in the partition lattice of the object set , see e.g. [3].

7 Conclusion and future work

In this paper we have presented a novel approach to data reduction based on hyper relations. We have shown that hyper relations are a generalization of database relations in the traditional sense, and they can be used to represent both raw data and reduced data. The reduced data can be regarded as a model of the raw data. We have shown that data reduction can be regarded as a process to find a set of interiors contained in the least upper bounds of individual classes of tuples in the dataset. In the context of lattice, we have proved that the interior exists, and have demonstrated a way in which we can find the expected interiors.

We illustrated that the proposed data reduction method can be regarded as a novel approach to data mining. We also discussed a data mining system, called DR. The training module of DR is a straightforward implementation of our data reduction procedure, and the classification module is simply based on set inclusion.

Results from initial experiments with DR are quite remarkable. DR outperformed C4.5

substantially in 6 out of 7 datasets with respect to the test accuracies by 5-fold cross validation. DR underperformed C4.5, again substantially, in one dataset – Vote, since Vote is binary and hence there is no reduction possible [3]. We hypothesized that data reduction with all binary attributes should be addressed explicitly beyond the current version of DR.

Since our method is new in many respects, most of data mining issues should be addressed with regard to the proposed method, notably, efficiency, windowing, missing values, incremental and parallel data mining, and so on.

Since data and knowledge can both be represented as hyper relations uniformly and the process of discovering knowledge can be modeled as finding the interior cover of the data, another possible further work can be directed towards generalizing the conventional relational data model to *hyper relational data model* so that database systems have a uniform representation for data and models and can be used for both data archiving and data modeling incorporating the interior cover operation.

References

[1] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 3(6):377–387, 1970.

[2] Clementine: A data mining system. Integral Solutions Limited (ISL). `http://www.isl.co.uk/`.

[3] Ivo Düntsch and Günther Gediga. Algebraic aspects of attribute dependencies in information systems. *Fundamenta Informaticae*, 29:119–133, 1997.

[4] Ivo Düntsch and Günther Gediga. Simple data filtering in rough set systems. *International Journal of Approximate Reasoning*, 1998. To appear.

[5] Usama M. Fayyad. Editorial. *Data Mining and Knowledge Discovery – An International Journal*, 1(3), 1997.

[6] George Grätzer. *General Lattice Theory*. Birkhäuser, Basel, 1978.

[7] Medical Informatics Handbook. `http://www.mieur.nl/mihandbook/r_1/glossary/glossary.htm`.

[8] D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine learning, neural and statistical classification*. Ellis Horwood, 1994.

[9] P. M. Murphy and D. W. Aha. *UCI Repository of Machine Learning Databases and Domain Theories*. Irvin, CA, 1994. `ftp://ftp.ics.uci.edu`.

[10] J. D. Ullman. *Principles of Database Systems*. Computer Science Press, 2 edition, 1983.

[11] Sholom M. Weiss and Nitin Indurkhya. *Predictive Data Mining: A Practical Guide.* Morgan Kaufmann Publishers, Inc., 1997.

[12] D. H. Wolpert. The relationship between Occam's Razor and convergent guessing. *Complex Systems*, 4:319–368, 1990.