

Reduct Generation in Information Systems

Janusz Starzyk
Ohio University, Department of Electrical Engineering and Computer Science
starzyk@bobcat.ent.ohiou.edu

Dale E. Nelson
Air Force Research Laboratory, Sensors Directorate
nelsonde@sensors.wpafb.af.mil

Kirk Sturtz
Veridian Corporation
ksturtz@mbvlab.wpafb.af.mil

***Abstract** – When data sets are analyzed, statistical pattern recognition is often used to find the information hidden in the data. Another approach to information discovery is data mining. Data mining is concerned with finding previously undiscovered relationships in data sets. Rough set theory provides a theoretical basis from which to find these undiscovered relationships. Automatic Target Recognition (ATR) is one area which can benefit from this approach. We present a new theoretical concept, **strong equivalence**, and an efficient algorithm, the **Expansion Algorithm**, for generation of all reducts of an information system. The process of finding reducts has been proven to be NP-hard. Using the elimination method, problems of size 13 could be solved in reasonable times. Using our Expansion Algorithm, the size of problems that can be solved has grown to 40. Further, by using the strong equivalence property in the Expansion Algorithm, additional savings of up to 50% can be achieved. This paper describes the fundamentals of this algorithm and the simulation results obtained from randomly generated information systems. The full paper provides the mathematical foundations of the algorithm.*

1.0 Introduction

In the world today, we are inundated with volumes of data. Businesses have been accumulating vast amounts of data in accounting, inventory and sales records. For decades this has been entered and stored on computers. Business leaders know that there is a wealth of information that could improve business operations if only there was a good way to discover the information contained in the data.

The military is interested in building robust automatic target recognition (ATR) systems. To build these systems, there is a set of measured or synthetic data that can be used for training and test. In the past, statistical pattern recognition has been used to build ATR systems. However, if the training data is viewed as an information system, then the procedures and methods of data mining can be used to find the previously unrecognized relationships in the data that will convert the data to information [3,6,7,10].

An information system can be characterized as a relational database, where the information is stored in a table. Each row in the table represents an individual record. Each column represents some an attribute of the records or a field. The columns could represent weight or height, or even some

measurement of a target such as height or length. Several records can be considered together to represent a logical grouping. For instance, there might be several examples of different kinds of airplanes, or there might be several examples of people who successfully paid off their loan. One operation in data mining is the determination of a minimal set of attributes necessary to distinguish between the different groups in the data. The process of determining which records belong to each of the groups is called **classification**. Each group of attributes that can distinguish between the groups is called a **reduct** [1, 4,7].

In this paper, due to space limitations, we take a naïve approach to explain the concepts involved. Therefore, mathematical rigor will not be enforced. The full paper, on which this summary is based, is mathematically rigorous.

2.0 Elimination Method

The first step in generating reducts is to make the training set non-ambiguous and eliminate duplicates. The training set is said to be **ambiguous** when two signals are identical but belong to two different groups. When this happens both signals should be removed from the training set. This is analogous to a

teacher telling a student that $1+1=2$ and $1+1=3$. One or both of the examples is wrong. Eliminating them eliminates ambiguities. If two or more identical signals represent the same class, all but one of the signals should be eliminated. This reduces computational time.

The procedure from this point is to take all possible combinations of the attribute columns. Each combination forms a set which is checked for ambiguity. If there is no ambiguity, then that set of attributes is a reduct. We call this method the **elimination method** because we are eliminating attributes to check for a reduct.

3.0 Rough Set Theory

Rough set theory was developed by Pawlak [3] for use in reasoning from imprecise data. This theory can also be used to formally develop a method for discovering relationships in data (data mining). Rough set theory, is concerned with three basic components; granularity of knowledge, approximation of sets, and data mining [4, 5]. One aspect of data mining is the finding of all reducts. Skowron [7] has proven this process to be NP-hard. This means that as the size of the problem increases, the time to compute the reducts increases faster than polynomially. Most real world problems involve vast amounts of data. Therefore, it behooves us to find as efficient an algorithm as possible to generate these reducts. The algorithm presented is several orders of magnitude better than the elimination method.

The first step in the rough set approach to generating all the reducts is to form a discernibility matrix. The discernibility matrix is an $N \times N$ matrix where N represents the number of records in the training set. For each entry in the table, we are comparing the record represented by the row number with the record represented by the column number. We further assign a label to each of the attributes. We assume each record represents one group. We enter in the table the labels of the attributes which have different values. In other words, these attributes are the ones which allows us to distinguish (discern) that they (the records) are different. We define an operator \vee (called disjunction) which allows us to distinguish between these two records by using attribute 1 **OR** attribute 5 **OR** ..., etc. It is easily seen that the diagonal is empty (there are no attributes that allow us to distinguish a record from itself). Further the matrix is symmetrical about the diagonal (the attributes that allow us to distinguish record 1 from 5 are the same as the attributes to distinguish record 5 from 1).

Using the discernibility matrix, it is now possible to form the discernibility function using another operator, \wedge (called conjunction). For simplicity we use the term “or” to represent our \vee operator and “and” to represent our \wedge operator. We form the discernibility function by or-ing all the values in one entry in the discernibility matrix and then and-ing all these together.

The discernibility function can often be simplified by the process of **absorption**. For example, suppose one of the disjuncts in the discernibility function is $(a \vee b)$, while another disjunct is $(a \vee b \vee c)$. Since attribute a or b is required to satisfy the first disjunct, the second disjunct will be satisfied by either a or b , and attribute c is not required so we can eliminate the $(a \vee b \vee c)$ term. We have determined all reducts when this equation is reduced to a disjunction of conjunctions. This final form yields all possible classifiers for the given information system!

We introduce a new concept, strong equivalence, which can be used to achieve significant speed improvements in our algorithm. We say two attributes are **locally strongly equivalent** if either both attributes are simultaneously present or simultaneously absent in any disjunctive entry of the discernibility function. When two attributes are locally strongly equivalent then they may be represented by a single attribute.

4.0 Expansion Law

There is a simple way to explain the expansion law. First, find the attribute that occurs most frequently (at least twice). **OR** this single attribute with all the other disjunctive terms which **DO NOT** contain the selected attribute. **AND** all the previous terms with all the disjunctive terms in the function removing the selected attribute from each disjunctive term in which it appears. This process is illustrated in the following example in step 3.

5.0 Distribution Algorithm

We now introduce our algorithm for efficiently computing all the reducts. The algorithm is as follows:

Given: $f_A = f_1 \wedge \dots \wedge f_k$ where f_A is the discernibility function.

Step 1. In each component f_i of the discernibility function, apply the absorption law to eliminate all disjunctive expressions which are supersets of

another disjunctive expression; e.g.
 $(a \vee b) \subset (a \vee b \vee c) = (a \vee b)$.

Step 2. Replace each strongly equivalent subset of attributes in each component f_i by a single attribute that represents this class. A strongly equivalent subset is identified in each component f_i if the corresponding set of attributes is simultaneously either present or absent in each subset of its conjuncts.

Step 3. In each component f_i select an attribute which belongs to the largest number of conjunctive sets, numbering at least two, and apply the expansion law. Write the resulting form as a disjunction $f_i = f_{i1} \vee f_{i2}$.

Step 4. Repeat steps 1 through 3 until you cannot apply the expansion law, then f_A is said to be in the **simple form**.

Step 5. For each component f_i of the resulting simple form, substitute all locally strongly equivalent classes for their corresponding attributes.

Step 6. Calculate the reducts by expanding the final discernibility function.

Step 7. Determine the minimal elements, with respect to the inclusion relation, of the set $\bigcup_{i=1}^p Red(f_i)$, where $f_A = f_1 \vee \dots \vee f_p$. These minimal elements are the elements of $Red(A)$

6.0 Example

To illustrate the reduct generation algorithm consider the discernibility function:

$$f_A = \{a \vee b \vee c \vee f\} \wedge \{b \vee d\} \wedge \{a \vee d \vee e \vee f\} \\ \wedge \{a \vee b \vee c \vee d\} \wedge \{b \vee d \vee e \vee f\} \wedge \{d \vee c\}$$

The function should be read as follows: the records in the information system may be discerned from each other by using (attribute a or b or c or f) and (attribute b or d) and (attribute a or d or e or f) and (attribute a or b or c or d) and (attribute b or d or e or f) and (attribute d or c). The reason the \wedge and the \vee are not Boolean operators is that the values of the attributes are not 1 or 0. What we are saying is that the attribute is either used or not used.

1. Since $\{b \vee d\} \subset \{b \vee d \vee e\}$ and $\{b \vee d\} \subset \{b \vee c \vee d\}$ we use the absorption law to

eliminate conjuncts 4 and 5 and get an equivalent discernibility function:

$$f_A = \{a \vee b \vee c \vee f\} \wedge \{b \vee d\} \wedge \{a \vee d \vee e \vee f\} \\ \wedge \{d \vee c\}$$

2. $\{a, f\}$ is a locally strongly equivalent class so we can represent it by a single attribute g which yields:

$$f_A = \{g \vee b \vee c\} \wedge \{b \vee d\} \wedge \{g \vee d \vee e\} \wedge \{d \vee c\}$$

3. The remaining function attribute d is the most frequent so we apply the expansion law with respect to this attribute to obtain

$$f_A = f_1 \vee f_2 \\ = (\{d\} \wedge \{g \vee b \vee c\}) \vee (\{g \vee b \vee c\} \wedge \{b\} \wedge \{g \vee e\} \wedge \{c\}) \\ = (\{d\} \wedge \{g \vee b \vee c\}) \vee (\{b\} \wedge \{g \vee e\} \wedge \{c\})$$

where the simplification in the last step resulted from the absorption law.

4. All functions f_i are in a simple form.

5. Substituting all strongly equivalent classes for their equivalent attributes we get

$$f_A = f_1 \vee f_2 \\ = (\{d\} \wedge \{a \vee f \vee b \vee c\}) \vee (\{b\} \wedge \{a \vee f \vee e\} \wedge \{c\})$$

6. Reducts which correspond to the f_i are

$$Red(f_1) = \{\{a \vee d\}, \{d \vee f\}, \{b \vee d\}, \{c \vee d\}\}$$

$$Red(f_2) = \{\{b \vee a \vee c\}, \{b \vee f \vee c\}, \{b \vee e \vee c\}\}$$

7. The reducts of A are obtained by determining the minimal elements of the set

$$\bigcup_{i=1}^2 Red(f_i) = \{\{a \vee d\}, \{d \vee f\}, \{b \vee d\}, \{c \vee d\}, \{b \vee a \vee c\}, \\ \{b \vee f \vee c\}, \{b \vee e \vee c\}\}$$

from which we conclude

$$Red(A) = \{\{a \vee d\}, \{d \vee f\}, \{b \vee d\}, \{c \vee d\}, \\ \{b \vee a \vee c\}, \{b \vee f \vee c\}, \{b \vee e \vee c\}\}$$

(The reducts of A are obtained by “throwing away”

supersets in $\bigcup_{i=1}^p Red(f_i)$; in this example there are no supersets.)

7.0 Results

Simulations were run using MATLAB 5.2 on test data generated randomly. A random number generator provided uniformly distributed numbers to represent each attribute of each record. These values were multiplied by 8 and then the fractional part was truncated. This resulted in integer attribute values between 0 and 8. The number of attributes varied from 10 to 40 in steps of 5 and the number of records varied from 10 to 40 in steps of 5. All simulations were accomplished using a dual Pentium Pro 200 MHz computer using 256MB of memory. Figure 1 illustrates how the run times increase with problem

size using the Expansion Algorithm. Note the abscissa is \log_{10} of the run time. The curves shown are for 10, 15, 20, 25, 30, 35, and 40 attributes. Note that the computational time is growing exponentially.

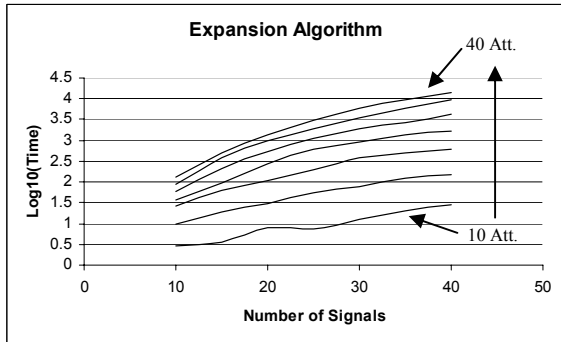


Figure 1. Expansion Algorithm Run Times

Figure 2 shows the difference in time to run the problems using the Elimination Method and the new Distribution Algorithm. The graph only shows the results for 10 and 15 attributes. This is because when the problem size was larger than this, the elimination method required so much time that results could not be obtained without the simulation running for many days! Note that the time expressed is the \log_{10} of the time.

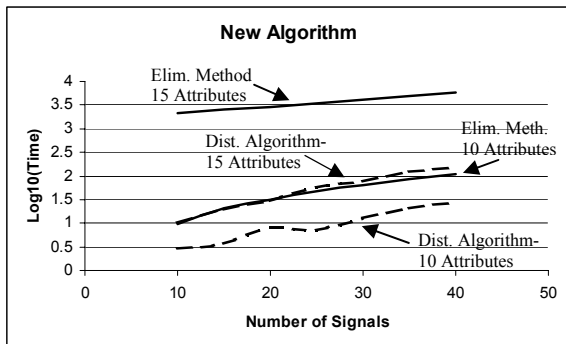


Figure 2. Time Savings of the Distribution Algorithm vs. the Elimination Method

Figure 3 shows the run times for the Distribution Algorithm with and without strong equivalence. Incorporating strong equivalence into the Expansion Algorithm does cost computational time. However, as seen in Figure 3, the time savings can be significant (as much as 50%) when strong equivalence is present.

8.0 Conclusions

We have shown that the use of the Expansion Algorithm allows the generation of all reducts in a much less time than the elimination method. Further, this algorithm is ideal for implementation on

multiprocessor computers. Using this algorithm, larger problems should be able to be addressed.

The addition of strong equivalence to the Expansion Algorithm further reduces computation time when strong equivalence is present. In the simulations we ran, strong equivalence was not always present and thus the run times were increased. It is possible that in real world problems, where structure is present, strong equivalence will manifest itself more frequently. Therefore, the computational time should be reduced in most cases.

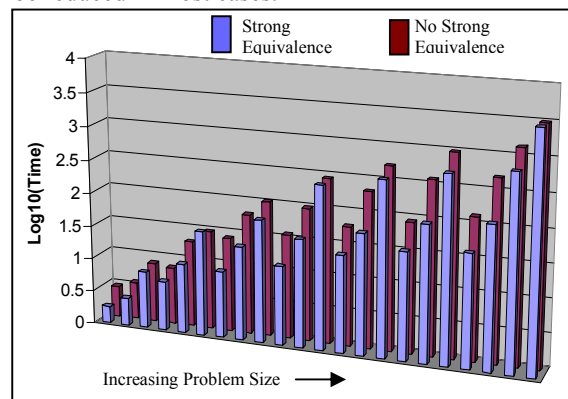


Figure 3. Time Savings When Strong Equivalence is Present vs. When It is Not.

9.0 References

- [1] W. Buszkowski and E. Orłowska, "On the logic of database dependencies", *Bull. Polish Sci. Math.*, Vol 34, pp345-354, 1986.
- [2] A. Nakamura and G. Jian-Miang, "A modal logic for similarity-based data analysis", Hiroshima Univ. Technical Report., 1988.
- [3] Z. Pawlak, "Information systems - theoretical foundations", *Information Systems*, Vol. 6, pp.205-218, 1981.
- [4] Z. Pawlak, "On rough dependency of attributes in information systems", *Bull. Polish Acad. Sci. Tech.* Vol. 33, pp.481-485, 1985.
- [5] Z. Pawlak, *Rough Sets - Theoretical Aspects of Reasoning About Data*, Kluwer Academic Publ., 1991.
- [6] Rasiowa and A. Skowron, "Approximation logic", *Mathematical Methods Specification and Synthesis of Software Systems*, Akademie-Verlag, Berlin, Band 31, pp. 123-139, 1986.
- [7] A. Skowron and J. Stepaniuk, "Towards an approximation theory of discrete problems: Part I", *Fundamenta Informaticae* 15(2), pp.187-208, 1991.
- [8] A. Skowron and C. Rauszer, "The discernibility matrices and functions in information systems", *Fundamenta Informaticae* 15(2), pp.331-362, 1991.
- [9] D. Vakarelov, "Modal logic of knowledge representation systems", *Lecture Notes on Computer Science*, Springer Verlag, 363, pp.257-277, 1989.
- [10] W. Ziarko, "Acquisition of design knowledge from examples", *Math Comput. Modeling* Vol. 10, pp. 551-554.