To appear in the Third International Workshop on
Rough Sets and Soft Computing (RSSC 94)

A Third Dimension to Rough Sets

Ron Kohavi
Computer Science Dept.
Stanford University
Stanford, CA 94305
ronnyk@CS.Stanford.EDU

Abstract

Rough-sets relative reducts allow one to reduce the number of attributes in a supervised classification problem without sacrificing the consistency of the decision table. In many problems, the cardinality of the relative reducts is large, making the decision table unmanageably big. We propose reducing the size of reducts, and hence the decision table, by increasing the number of label values (values of the decision attribute). The reduction process is accomplished through a two-step process whereby an instance is mapped to an intermediate label using one subset of attributes, and then mapped to the final label using a disjoint subset of attributes. In some cases, the table representation size of the functions generated using this method will be much smaller than the representation of a single unified function. It is possible to repeat the splitting process, creating multiple rough-set approximations, each one containing one attribute less than the previous one, but at a possible increase in the number of labels. The structures formed are isomorphic to oblivious read-once decision graphs (OODGs) and to ordered binary decision diagrams (OBDDs), thus providing an alternative view of how algorithms that construct such graphs operate.

1 Introduction

Rough-sets theory [Pawlak, 1991, Slowinski, 1992] defines a relative reduct as a subset of attributes such that the indiscernibility relation formed using this subset has the same positive region as the original set of attributes with respect to the label, or decision attribute. In many problems, the subset is prohibitively large, and representing the concept of interest as a decision table is impossible if a concise representation is required.

Since one of the main goals of supervised classification learning is to introduce structure into data, thus prompting greater understanding of the underlying concepts, problems with many indispensable attributes pose a great problem to experts who attempt to understand the underlying structure of the data.

We begin by showing that a two-step decomposition process may be a better way to capture structure in the data. In the first step, a decision table employing a subset of the attributes is used to classify a given instance into an intermediate label. The intermediate label is then mapped into the final label using a disjoint subset of attributes.

By generalizing this two-step decomposition to m steps, where m is the number of attributes, one essentially builds an oblivious read-once decision graph (OODG) [Kohavi, 1994a, Kohavi, 1994b], or an ordered binary decision diagram (OBDD) [Bryant, 1992]. The rough-sets view provides an interesting alternative view of the process being carried out by the HOODG algorithm [Kohavi, 1994b].

2 The Two-Step Decomposition

The two-step decomposition divides the attributes into two mutually exclusive and exhaustive sets, such that the first set is used to map to an intermediate label, and the second set is used to map to the goal label. Intuitively, we would like both mapping functions to be "simple," with the sum of the complexities less than the single mapping from the attributes to the label.

Let $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$ and $\mathcal{R} = \{R_1, \ldots, R_r\}$ be two disjoint sets of attributes, such that $\mathcal{Q} \cup \mathcal{R} = \mathcal{P}$, where $\mathcal{P} = \{P_1, \ldots, P_p\}$ is the complete set of attributes. Let L be the label attribute, and let A be a new attribute (the intermediate label). If

$$f : P_1 \times \cdots \times P_p \mapsto L$$

is the original (target) mapping function, then we would like to replace it with two mapping functions g and h such that

$$\begin{array}{l} g : Q_1 \times \cdots \times Q_q \mapsto A \\ h : A \times R_1 \cdots \times R_r \mapsto L \\ \text{Such that } f(p_1, \ldots, p_p) = h(g(q_1, \ldots, q_q), r_1, \ldots, r_r) \end{array}$$

One example of such a decomposition is to divide the attributes such that $\mathcal{Q}$ is the set of attributes in a relative reduct and $\mathcal{R}$ is $\mathcal{P} \setminus \mathcal{Q}$, *i.e.*, the rest of the attributes. By the definition of a relative reduct, the attributes in it are sufficient to determine the label value and we can thus define $A = L$, and be guaranteed that there exists a map g that will map the attributes in the relative reduct to the correct values of the label. h can then be defined as $h(a, r_1, \ldots, r_r) = a$, *i.e.*, ignore all the other attributes.

It has been well established in the rough-sets community that finding reducts is important to simplify decision tables, and hence understanding of the problem; however, when all relative reducts are big, a different decomposition is called for.

Example 1 (Multiplexer) Figure 1 depicts a General Logic Diagram (GLD) [Michalski, 1978, Thrun *et al.*, 1991, Wnek and Michalski, 1994] for a concept. Each instance in the space has exactly one box that is marked with an X if it belongs to the concept and with an O if it does not. Readers familiar with Karnaugh maps may note a resemblance, except that the ordering of attribute values does not conform to the hamming distance restriction in Karnaugh maps because GLDs are not restricted to Boolean attributes.

Because all attributes are indispensable for the target concept, the relative core consists of all attributes and is therefore the only relative reduct.

If we select $\mathcal{Q}$ to be the set A_0, A_1, and allow four values for the intermediate label value, the GLD that we get is the one shown in Figure 2. Each of the four symbols stands for a different intermediate label value. The GLD defines the mapping g.

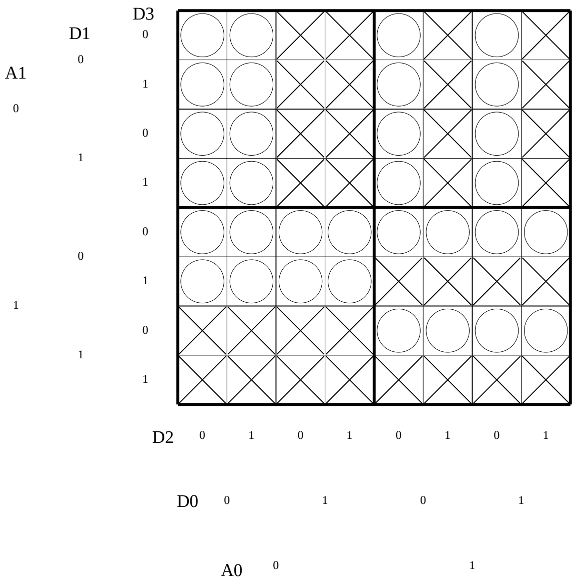

Figure 1: A GLD for an “unknown” concept

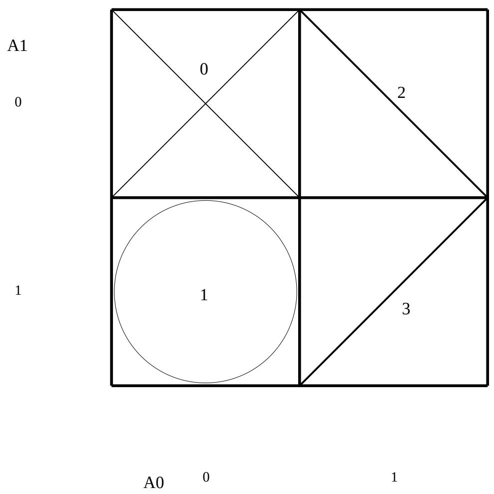

Figure 2: A GLD for the attributes A_0 and A_1

The mapping h can then be defined as:

$$h(a, d_0, d_1, d_2, d_3) = \begin{cases} d_0 \text{ if } a = 0 \\ d_1 \text{ if } a = 1 \\ d_2 \text{ if } a = 2 \\ d_3 \text{ if } a = 3 \end{cases}$$

The function can be represented as four tables, each of size 2, with the g function choosing the appropriate table.

Readers may recognize this as the $2 \rightarrow 4$ multiplexer concept. In general, there are n address bits and 2^n data bits. An instance is labelled positive if and only if the data bit indicated by the address bits is on. For a $2 \rightarrow 4$ multiplexer, the saving is marginal (12 decision rules as opposed to 16), but the saving grows for larger multiplexers.

We urge the reader to attempt to define g and h for different $\mathcal{Q}$'s. For example, if $\mathcal{Q} = \{A_0, A_1, D_0, D_1\}$, four label values are still needed; if D_2 is added, only three label values are needed, and if D_1 is deleted, five values are needed. There is no monotonic relation between the number of label values and the size of the set $\mathcal{Q}$.

3 Multi-Step Decomposition

A generalization of the above two-step decomposition can be made by looking at the function h as if it were the original function, and attempting to decompose it. Each decomposition will reduce the number of attributes in the set remaining to be composed, until the process ends.

If at each decomposition step exactly one attribute is chosen, then we can draw the functions as nodes of a levelled graph. Each node represents all the instances which have attribute values leading to it and maps them to the appropriate node with one more attribute value. A levelled graph for the multiplexer function is shown in Figure 3.

Since many nodes ignore their inputs at given levels, that is, they are constant functions, we may remove such nodes from the graph, and end up with the graph (OODG) shown in Figure 4.

Example 2 (Parity) A decision table for k bit parity, i.e., $X_1 \oplus X_2 \oplus \cdots \oplus X_k$ requires 2^k entries in the table. All attributes are in the core, and hence all of them form the only reduct. Using multi-step decomposition, we can generate tables such that their overall size is linear in k:

$$g(X_1, \ldots, X_{k-1}, 0) = 0 \qquad g(X_1, \ldots, X_{k-1}, 1) = 1$$

$$\begin{array}{cc}
h_1(0, X_1, \ldots, X_{k-2}, 0) = 0 & h_1(0, X_1, \ldots, X_{k-2}, 1) = 1 \\
h_1(1, X_1, \ldots, X_{k-2}, 0) = 1 & h_1(1, X_1, \ldots, X_{k-2}, 1) = 0 \\
h_2(0, X_1, \ldots, X_{k-3}, 0) = 0 & h_2(0, X_1, \ldots, X_{k-3}, 1) = 1 \\
h_2(1, X_1, \ldots, X_{k-3}, 0) = 1 & h_2(1, X_1, \ldots, X_{k-3}, 1) = 0 \\
\vdots & \vdots \\
h_{k-2}(0, X_1, 0) = 0 & h_{k-2}(0, X_1, 1) = 1 \\
h_{k-2}(1, X_1, 0) = 1 & h_{k-2}(1, X_1, 1) = 0 \\
h_{k-1}(0, 0) = 0 & h_{k-1}(0, 1) = 1 \\
h_{k-1}(1, 0) = 1 & h_{k-1}(1, 1) = 0
\end{array}$$

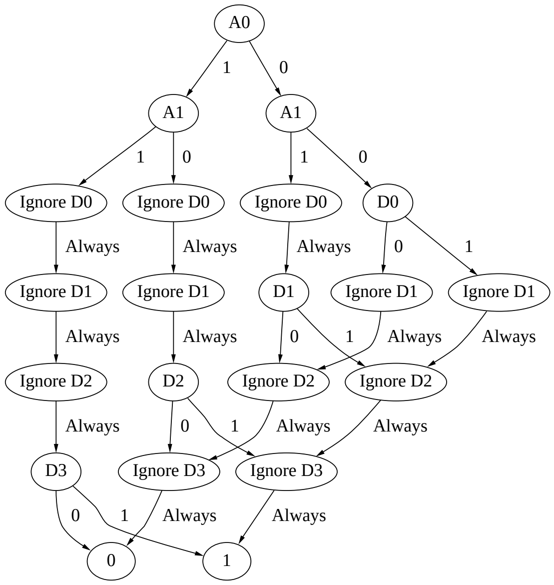

Figure 3: A graphical representation for a $2 \rightarrow 4$ multiplexer as a levelled graph.

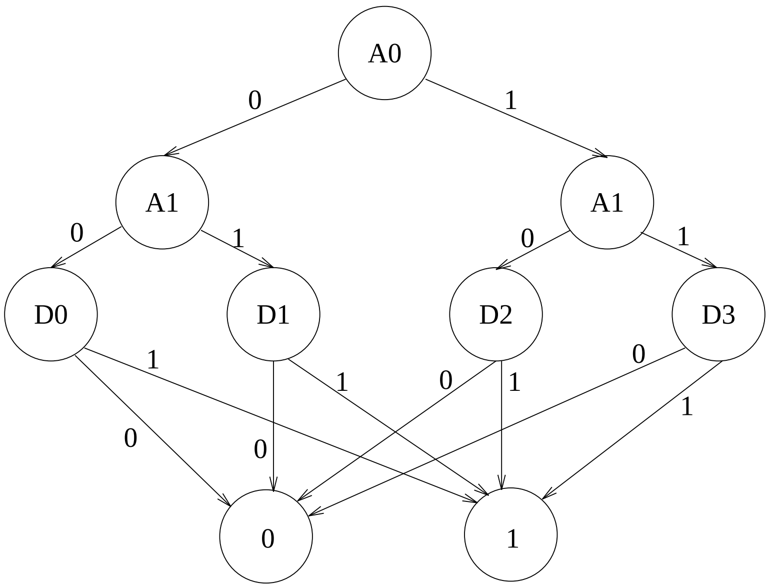

Figure 4: A graphical representation for a $2 \rightarrow 4$ multiplexer with constant functions removed.

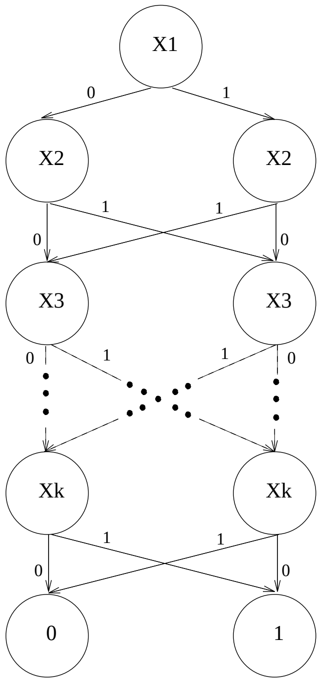

Figure 5: A graphical representation for n bit parity.

Each function in the sequence returns a value which is the first argument to the next function in the multi-step decomposition. The last function, h_{k-1} returns the label for the classification problem. Since all the h functions use only their first and last argument, they can be implemented as decision tables of size four. The overall size of parity under this representation is $4k-2$. Even if we include some overhead for specifying which attributes a table needs to examine, the gap between this representation and a unified table is exponential. Figure 5 shows the OODG representation of this function.

4 Discussion and Related Work

Bryant [Bryant, 1986] introduced Ordered Binary Decision Diagrams (OBDDs), which spawned a plethora of articles and a whole sub-community dealing with ways to build small OBDDs [Bryant, 1992, Brace *et al.*, 1990, Minato *et al.*, 1990, Fujita *et al.*, 1993]. OBDDs have been used for automatically verifying finite state machines, including 64-bit ALUs, with up to 10^{120} states by representing the state space symbolically instead of explicitly [Burch *et al.*, 1990, Burch *et al.*, 1991].

In the machine learning community, Kohavi [1994a, 1994b] investigated the possibility of using oblivious decision graphs as the underlying hypothesis space for supervised classification learning. Modrzejewski's work on feature selection [Modrzejewski, 1993] uses oblivious trees (called "preset tree" by Modrzejewski) as the underlying hypothesis space. The induction algorithm chooses attributes for each level using their significance measure. While oblivious graphs are a more compact representation than trees, it is possible to convert oblivious trees to oblivious graphs, and thus view his algorithm as a multi-step decomposition.

The multi-step decomposition illustrated here provides an interesting view of what algorithms for building OODGs and OBDDs are trying to approximate, namely, multiple precision rough sets using a smaller number of attributes at each level. For example, the HOODG algorithm [Kohavi, 1994b] chooses the attribute which would lead to the minimal number of intermediate label values at the next decomposition step.

5 Summary

Attempts to construct decision tables or similar structures directly using reducts sometimes results in large incomprehensible structures. Instead of using rough sets on the original space, we introduced a third dimension to the usual diagram representing rough sets—a dimension that allows varying the number of label values. We illustrated how a multi-step decomposition creates decomposed tables that may be exponentially smaller.

The multi-step decomposition process creates a structure that is isomorphic to oblivious decision graphs (OODGs) and ordered binary decision diagrams (OBDDs), and thus provides an interesting an alternative view of what such algorithms are attempting to optimize. It is conceivable that after this connection has been established, results from rough-sets theory will be applicable to algorithms constructing OODGs and OBDDs, and improve them.

Acknowledgments The work in this paper was done using the $\mathcal{MLC}$++ library, partly funded by ONR grant N00014-94-1-0448 and NSF grant IRI-9116399. Yoav Shoham and Nils Nilsson provided support for the $\mathcal{MLC}$++ project. Pat Langley deserves the credit for prompting me to look at rough sets after seeing my work on OODGs.

References

[Brace *et al.*, 1990] Karl S. Brace, Richard L. Rudell, and Randal E. Bryant. Efficient implementation of a BDD package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 40–45, 1990.

[Bryant, 1986] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[Bryant, 1992] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[Burch *et al.*, 1990] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Fifth Annual IEEE Symposium on Logic in Computer Science.*, pages 428–439. IEEE Comput. Soc. Press, 1990.

[Burch *et al.*, 1991] J. R. Burch, E. M. Clarke, and D. E. Long. Representing circuits more efficiently in symbolic model checking. In *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pages 403–407, 1991.

[Fujita *et al.*, 1993] Masahiro Fujita, Hisanori Fujisawa, and Jusuke Matsunaga. Variable ordering algorithms for ordered binary decision diagrams and their evaluation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(1):6–12, 1993.

[Kohavi, 1994a] Ron Kohavi. Bottom-up induction of oblivious, read-once decision graphs. In *Proceedings of the European Conference on Machine Learning*, April 1994. Paper available by anonymous ftp from
`starry.Stanford.EDU:pub/ronnyk/euroML94.ps`.

[Kohavi, 1994b] Ron Kohavi. Bottom-up induction of oblivious, read-once decision graphs : Strengths and limitations. In *Twelfth National Conference on Artificial Intelligence*, 1994. Paper available by anonymous ftp from
`Starry.Stanford.EDU:pub/ronnyk/aaai94.ps`.

[Michalski, 1978] Ryszard S. Michalski. A planar geometric model for representing multidimensional discrete spaces and multiple-valued logic functions. Technical Report UIUCDCS-R-78-897, University of Illinois at Urbaba-Champaign, 1978.

[Minato *et al.*, 1990] Shin-ichi Minato, Nagisa Ishiura, and Shuzo Yajima. Shared binary decision diagram with attributed edges for efficient boolean function manipulation. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 24–28, 1990.

[Modrzejewski, 1993] Maciej Modrzejewski. Feature selection using rough sets theory. In Pavel B. Brazdil, editor, *Proceedings of the European Conference on Machine Learning*, pages 213–226, 1993.

[Pawlak, 1991] Zdzislaw Pawlak. *Rough Sets.* Kluwer Academic Publishers, 1991.

[Slowinski, 1992] Roman Slowinski. *Intelligent decision support : handbook of applications and advances of the rough sets theory.* Kluwer Academic Publishers, 1992.

[Thrun *et al.*, 1991] S.B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S.E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van de Weldel, W. Wenzel, J. Wnek, and J. Zhang. The monk's problems: A performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.

[Wnek and Michalski, 1994] Janusz Wnek and Ryszard S. Michalski. Hypothesis-driven constructive induction in AQ17-HCI : A method and experiments. *Machine Learning*, 14(2):139–168, 1994.