Finding Rough Set Reducts with SAT

Richard Jensen[1], **Qiang Shen**[1] and **Andrew Tuson**[2]
{rkj,qqs}@aber.ac.uk

[1] Department of Computer Science, The University of Wales, Aberystwyth
[2] Department of Computing, School of Informatics, City University, London.

Abstract. Feature selection refers to the problem of selecting those input features that are most predictive of a given outcome; a problem encountered in many areas such as machine learning, pattern recognition and signal processing. In particular, solution to this has found successful application in tasks that involve datasets containing huge numbers of features (in the order of tens of thousands), which would be impossible to process further. Recent examples include text processing and web content classification. Rough set theory has been used as such a dataset pre-processor with much success, but current methods are inadequate at finding *minimal* reductions, the smallest sets of features possible. This paper proposes a technique that considers this problem from a propositional satisfiability perspective. In this framework, minimal subsets can be located and verified. An initial experimental investigation is conducted, comparing the new method with a standard rough set-based feature selector.

1 Introduction

Many problems in machine learning involve high dimensional descriptions of input features. It is therefore not surprising that much research has been carried out on dimensionality reduction [4]. However, existing work tends to destroy the underlying semantics of the features after reduction or require additional information about the given data set for thresholding. A technique that can reduce dimensionality using information contained within the dataset and that preserves the meaning of the features (i.e. semantics-preserving) is clearly desirable. Rough set theory (RST) can be used as such a tool to discover data dependencies and to reduce the number of attributes contained in a dataset using the data alone, requiring no additional information [10, 11].

Over the past ten years, RST has indeed become a topic of great interest to researchers and has been applied to many domains. Given a dataset with discretized attribute values, it is possible to find a subset (termed a *reduct*) of the original attributes using RST that are the most informative; all other attributes can be removed from the dataset with very little information loss. However, current methods such as heuristic and stochastic-based search are inadequate at finding minimal reductions. By reformulating the rough set reduction task in a propositional satisfiability (SAT) framework, solution techniques from SAT may be applied that should be able to discover such subsets, guaranteeing their minimality.

The rest of this paper is structured as follows. Section 2 details the main concepts involved in rough set feature selection, with an illustrative example. The third section introduces propositional satisfiability and how the problem of finding rough set reducts can be formulated in this way. The initial experimental results of the application of the new method is presented in section 4. Section 5 concludes the paper, with a discussion of some of the future work in this area.

2 Rough Set-based Feature Selection

Rough set theory [10] is an extension of conventional set theory that supports approximations in decision making. The rough set itself is the approximation of a vague concept (set) by a pair of precise concepts, called lower and upper approximations, which are a classification of the domain of interest into disjoint categories. The lower approximation is a description of the domain objects which are known with certainty to belong to the subset of interest, whereas the upper approximation is a description of the objects which possibly belong to the subset.

There are two main approaches to finding rough set reducts: those that consider the degree of dependency and those that are concerned with the discernibility matrix. This section describes the fundamental ideas behind both approaches. To illustrate the operation of these, an example dataset (table 1) will be used.

$x \in \mathbb{U}$	a	b	c	d	$\Rightarrow e$
0	1	0	2	2	0
1	0	1	1	1	2
2	2	0	0	1	1
3	1	1	0	2	2
4	1	0	2	0	1
5	2	2	0	1	1
6	2	1	1	1	2
7	0	1	1	0	1

Table 1. An example dataset

2.1 Rough Set Attribute Reduction

Central to Rough Set Attribute Reduction (RSAR) [3, 7] is the concept of indiscernibility. Let $I = (\mathbb{U}, \mathbb{A})$ be an information system, where $\mathbb{U}$ is a non-empty set of finite objects (the universe) and $\mathbb{A}$ is a non-empty finite set of attributes such that $a : \mathbb{U} \rightarrow V_a$ for every $a \in \mathbb{A}$. V_a is the set of values that attribute a may take. With any $P \subseteq \mathbb{A}$ there is an associated equivalence relation $IND(P)$:

$$IND(P) = \{(x, y) \in \mathbb{U}^2 \mid \forall\, a \in P,\ a(x) = a(y)\} \tag{1}$$

The partition of $\mathbb{U}$, generated by *IND(P)* is denoted $\mathbb{U}$/*IND(P)* (or $\mathbb{U}/P$) and can be calculated as follows:

$$\mathbb{U}/IND(P) = \otimes\{a \in P : \mathbb{U}/IND(\{a\})\}, \tag{2}$$

where

$$A \otimes B = \{X \cap Y : \forall X \in A, \forall Y \in B, X \cap Y \neq \emptyset\} \tag{3}$$

If $(x, y) \in IND(P)$, then x and y are indiscernible by attributes from P. The equivalence classes of the P-indiscernibility relation are denoted $[x]_P$.
Let $X \subseteq \mathbb{U}$. X can be approximated using only the information contained within P by constructing the P-*lower* and P-*upper* approximations of X:

$$\underline{P}X = \{x \mid [x]_P \subseteq X\} \tag{4}$$

$$\overline{P}X = \{x \mid [x]_P \cap X \neq \emptyset\} \tag{5}$$

Let P and Q be equivalence relations over $\mathbb{U}$, then the positive region can be defined as:

$$POS_P(Q) = \bigcup_{X \in \mathbb{U}/Q} \underline{P}X \tag{6}$$

The positive region contains all objects of $\mathbb{U}$ that can be classified to classes of $\mathbb{U}/Q$ using the information in attributes P. For example, let $P = \{b,c\}$ and $Q = \{e\}$, then

$$POS_P(Q) = \bigcup\{\emptyset, \{2, 5\}, \{3\}\} = \{2, 3, 5\}$$

Using this definition of the positive region, the rough set degree of dependency of a set of attributes Q on a set of attributes P is defined in the following way:

For P, $Q \subset \mathbb{A}$, it is said that Q depends on P in a degree k ($0 \leq k \leq 1$), denoted $P \Rightarrow_k Q$, if

$$k = \gamma_P(Q) = \frac{|POS_P(Q)|}{|\mathbb{U}|} \tag{7}$$

In the example, the degree of dependency of attribute $\{e\}$ from the attributes $\{b,c\}$ is:

$$\begin{aligned}\gamma_{\{b,c\}}(\{e\}) &= \tfrac{|POS_{\{b,c\}}(\{e\})|}{|\mathbb{U}|} \\ &= \tfrac{|\{2,3,5\}|}{|\{0,1,2,3,4,5,6,7\}|} = \tfrac{3}{8}\end{aligned}$$

The reduction of attributes is achieved by comparing equivalence relations generated by sets of attributes. Attributes are removed so that the reduced set provides the same predictive capability of the decision feature as the original. A *reduct* is defined as a subset of minimal cardinality R_{min} of the conditional attribute set $\mathbb{C}$ such that $\gamma_R(\mathbb{D}) = \gamma_{\mathbb{C}}(\mathbb{D})$.

The QuickReduct algorithm given in figure 1, attempts to calculate a reduct without exhaustively generating all possible subsets. It starts off with an empty set and adds in turn, one at a time, those attributes that result in the

```
QuickReduct($\mathbb{C}$,$\mathbb{D}$).
$\mathbb{C}$, the set of all conditional features;
$\mathbb{D}$, the set of decision features.

$R \leftarrow \{\}$
do
    $T \leftarrow R$
    $\forall x \in (\mathbb{C} - R)$
        if $\gamma_{R\cup\{x\}}(\mathbb{D}) > \gamma_T(\mathbb{D})$
            $T \leftarrow R \cup \{x\}$
    $R \leftarrow T$
until $\gamma_R(\mathbb{D}) == \gamma_{\mathbb{C}}(\mathbb{D})$
return $R$
```

Fig. 1. The QuickReduct Algorithm

greatest increase in the rough set dependency metric, until this produces its maximum possible value for the dataset.

According to the QuickReduct algorithm, the dependency of each attribute is calculated, and the best candidate chosen. Attribute d generates the highest dependency degree, so that attribute is chosen and the sets $\{a, d\}$, $\{b, d\}$ and $\{c, d\}$ are evaluated. This process continues until the dependency of the reduct equals the consistency of the dataset (1 if the dataset is consistent). In the example, the algorithm terminates after evaluating the subset $\{b, d\}$. The generated reduct shows the way of reducing the dimensionality of the original dataset by eliminating those conditional attributes that do not appear in the set.

This, however, is not guaranteed to find a *minimal* subset. Using the dependency function to discriminate between candidates may lead the search down a non-minimal path. It is impossible to predict which combinations of attributes will lead to an optimal reduct based on changes in dependency with the addition or deletion of single attributes. It does result in a close-to-minimal subset, though, which is still useful in greatly reducing dataset dimensionality.

2.2 Discernibility Matrix-based Selection

Many applications of rough sets to feature selection make use of discernibility matrices for finding reducts. A discernibility matrix [12] of a decision table $D = (\mathbb{U}, \mathbb{C} \cup \mathbb{D})$ is a symmetric $|\mathbb{U}| \times |\mathbb{U}|$ matrix with entries defined:

$$d_{ij} = \{a \in \mathbb{C} | a(x_i) \neq a(x_j)\} \quad i, j = 1, ..., |\mathbb{U}| \tag{8}$$

Each d_{ij} contains those attributes that differ between objects i and j. For finding reducts, the decision-relative discernibility matrix is of more interest. This only considers those object discernibilities that occur when the corresponding decision attributes differ. Returning to the example dataset, the decision-relative discernibility matrix found in table 2 is produced. For example, it can be seen from the table that objects 0 and 1 differ in each attribute. Although some attributes in objects 1 and 3 differ, their corresponding decisions are the same so no entry appears in the decision-relative matrix. Grouping all entries containing single attributes forms the core of the dataset (those attributes appearing in *every* reduct). Here, the core of the dataset is $\{d\}$.

$x \in \mathbb{U}$	0	1	2	3	4	5	6	7
0								
1	a, b, c, d							
2	a, c, d	a, b, c						
3	b, c		a, b, d					
4	d	a, b, c, d		b, c, d				
5	a, b, c, d	a, b, c		a, b, d				
6	a, b, c, d		b, c		a, b, c, d	b, c		
7	a, b, c, d	d		a, c, d			a, d	

Table 2. The decision-relative discernibility matrix

From this, the discernibility function can be defined. This is a concise notation of how each object within the dataset may be distinguished from the others. A discernibility function f_D is a boolean function of m boolean variables $a_1^*, ..., a_m^*$ (corresponding to the attributes $a_1, ..., a_m$) defined as below:

$$f_D(a_1^*, ..., a_m^*) = \wedge\{\vee c_{ij}^* | 1 \leq j \leq i \leq |\mathbb{U}|, c_{ij} \neq \emptyset\} \tag{9}$$

where $c_{ij}^* = \{a^* | a \in c_{ij}\}$. By finding the set of all prime implicants of the discernibility function, all the minimal reducts of a system may be determined. From table 2, the decision-relative discernibility function is (with duplicates removed):

$$\begin{aligned} f_D(a, b, c, d) = &\{a \vee b \vee c \vee d\} \wedge \{a \vee c \vee d\} \wedge \{b \vee c\} \\ &\wedge \{d\} \wedge \{a \vee b \vee c\} \wedge \{a \vee b \vee d\} \\ &\wedge \{b \vee c \vee d\} \wedge \{a \vee d\} \end{aligned}$$

Further simplification can be performed by removing those sets (clauses) that are supersets of others:

$$f_D(a, b, c, d) = \{b \vee c\} \wedge \{d\}$$

The reducts of the dataset may be obtained by converting the above expression from conjunctive normal form to disjunctive normal form (without negations). Hence, the minimal reducts are $\{b, d\}$ and $\{c, d\}$. Although this is guaranteed to discover all minimal subsets, it is a costly operation rendering the method impractical for even medium-sized datasets.

For most applications, a single minimal subset is required for data reduction. This has led to approaches that consider finding individual shortest prime implicants from the discernibility function. A common method is to incrementally add those attributes that occur with the most frequency in the function, removing any clauses containing the attributes, until all clauses are eliminated [9]. However, even this does not ensure that a minimal subset is found - the search can proceed down non-minimal paths.

3 RSAR-SAT

The Propositional Satisfiability (SAT) problem [5] is one of the most studied NP-complete problems because of its significance in both theoretical research and practical applications. Given a boolean formula (typically in conjunctive normal form (CNF)), the SAT problem requires an assignment of variables/features so that the formula evaluates to true, or a determination that no such assignment exists. In recent years search algorithms based on the well-known Davis-Logemann-Loveland algorithm (DPLL) [5] are emerging as some of the most efficient methods for complete SAT solvers. Such solvers can either find a solution or prove that no solution exists.

Stochastic techniques have also been developed in order to reach a solution quickly. These pick random locations in the space of possible assignments and perform limited local searches from them. However, as these techniques do not examine the entire search space, they are unable to prove unsatisfiability.

A CNF formula on n binary variables $x_1, ..., x_n$ is the conjunction of m clauses $C_1, ..., C_m$ each of which is the disjunction of one or more literals. A literal is the occurrence of a variable or its negation. A formula denotes a unique n-variable boolean function $f(x_1, ..., x_n)$. Clearly, a function f can be represented by many equivalent CNF formulas. The satisfiability problem is concerned with finding an assignment to the arguments of $f(x_1, ..., x_n)$ that makes the function equal to 1, signalling that it is satisfiable, or proving that the function is equal to 0 and hence unsatisfiable [14]. By viewing the selection problem as a variant of SAT, with a bound on true assignments, techniques from this field can be applied to reduct search.

3.1 Finding Rough Set Reducts

The problem of finding the smallest feature subsets using rough set theory can be formulated as a SAT problem. Rough sets allows the generation from datasets of clauses of features in conjunctive normal form. If after assigning truth values to all features appearing in the clauses the formula is satisfied, then those features set to true constitute a valid subset for the data. The task is to find the smallest number of such features so that the CNF formula is satisfied. In other words, the problem here concerns finding a minimal assignment to the arguments of $f(x_1, ..., x_n)$ that makes the function equal to 1. There will be at least one solution to the problem (i.e. all x_is set to 1) for consistent datasets. Preliminary work has been carried out in this area [1], though this does not adopt a DPLL-style approach to finding solutions.

The DPLL algorithm for finding minimal subsets can be found in figure 2, where a search is conducted in a depth-first manner. The key operation in this procedure is the unit propagation step, unitPropagate(F), in lines (6) and (7). Clauses in the formula that contain a single literal will only be satisfied if that literal is assigned the value 1 (for positive literals). These are called unit clauses. Unit propagation examines the current formula for unit clauses and automatically assigns the appropriate value to the literal they contain. The elimination of a literal can create new unit clauses, and thus unit propagation eliminates variables by repeated passes until there is no unit clause in the formula. The order of the unit clauses within the formula makes no difference to the results or the efficiency of the process.

Branching occurs at lines (9) to (12) via the function selectLiteral(F). Here, the next literal is chosen heuristically from the current formula, assigned the value 1, and the search continues. If this branch eventually results in unsatisfiability, the procedure will assign the value 0 to this literal instead and continue the search. The importance of choosing good branching literals is well known - different branching heuristics may produce drastically different sized search trees for the same basic algorithm, thus significantly affecting the efficiency of the solver. The heuristic currently used within RSAR-SAT is to select the variable that appears in the most clauses in the current set of clauses. Many other heuristics exist for this purpose [14], but are not considered here.

A degree of pruning can take place in the search by remembering the size of the currently considered subset and the smallest optimal subset encountered so far. If the number of variables currently assigned 1 equals the number of those in the presently optimal subset, and the satisfiability of F is still not known, then any further search down this branch will not result in a smaller optimal subset.

DPLL(F).
F, the formula containing the current set of clauses.

```
(1)   if ($F$ contains an empty clause)
(2)       return unsatisfiable
(3)   if ($F$ is empty)
(4)       output current assignment
(5)       return satisfiable
(6)   if ($F$ contains a unit clause $\{l\}$)
(7)       $F' \leftarrow$ unitPropagate($F$)
(8)       return DPLL($F'$)
(9)   $x \leftarrow$ selectLiteral($F$)
(10)  if ( DPLL($F \cup \{x\}$) is satisfiable)
(11)      return satisfiable
(12)  else return DPLL($F \cup \{-x\}$)
```

Fig. 2. The definition of the DPLL algorithm

Although stochastic methods have been applied to SAT problems [6], these are not applicable here as they provide no guarantee of solution minimality. The DPLL-based algorithm will always find the minimal optimal subset. However, this will come at the expense of time taken to find it.

3.2 Pre-processing Clauses

The discernibility function can be simplified by replacing those variables that are simultaneously either present or absent in all clauses by single representative variables. For instance, in the formula below, variables a and f can be replaced by a single variable.

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

The first and third clauses may be considered to be $\{\{a \vee f\} \vee b \vee c\}$ and $\{\{a \vee f\} \vee d \vee e\}$ respectively. Replacing $\{a \vee f\}$ with g results in

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

If a reduct resulting from this discernibility function contains the new variable g, then this variable may be replaced by either a or f. Here, $\{g, d\}$ is a reduct and so $\{a, d\}$ and $\{f, d\}$ are reducts of the original set of clauses. Hence, fewer attributes are considered in the reduct-determining process with no loss of information [13]. The complexity of this (optional) pre-processing step is $\mathrm{O}(a * c + a^2)$, where a is the number of attributes and c is the number of clauses.

From the generation of the discernibility matrix, the core attributes are immediately determined (as discussed in section 2.2). These may then be removed from the discernibility function as they will appear in every rough set reduct. Hence, if the union of the core attributes for a dataset results in a reduct, no search is required as this will be the minimal subset.

4 Evaluation

Initial experimentation has been carried out using the algorithm outlined previously. The datasets have been obtained from [2]. Table 3 shows the average time taken for the preprocessing of each dataset. For RSAR, this involves constructing partitions for each attribute. For RSAR-SAT, the discernibility matrix is calculated and simplified. It can be seen from the table that RSAR-SAT requires more pre-processing time. Included in this table are the number of clauses appearing in the resultant discernibility function for the RSAR-SAT method.

The average times of the execution of these algorithms are also presented in table 3. The time taken for RSAR-SAT is split into two columns. The first indicates the average length of time taken to find the minimal subset, the second how long it takes to verify that this is indeed minimal. For RSAR, an asterisk next to the time indicates that it found a non-minimal reduct.

Dataset	No. of clauses	No. of Features	RSAR setup (s)	SAT setup (s)	RSAR (s)	SAT: Minimal (s)	SAT: Full (s)
M-of-N	6	13	0.164	2.333	0.171*	0.001	0.007
Exactly	6	13	0.146	2.196	0.304*	0.001	0.008
Exactly2	10	13	0.136	1.898	0.823*	0.001	0.008
Heart	12	13	0.085	0.380	0.207*	0.002	0.009
Vote	12	16	0.076	0.333	0.170*	0.004	0.009
Credit	200	20	0.148	3.873	1.988*	0.077	0.094
LED	167	24	0.125	68.20	0.097*	0.041	0.051
Letters	57	25	0.019	0.074	0.067*	0.024	0.116
Derm	1126	34	0.187	11.31	0.758*	0.094	0.456
Derm2	1184	34	0.133	6.796	0.897*	0.104	0.878
WQ	6534	38	0.168	87.85	9.590*	0.205	116.1
Lung	171	56	0.032	0.125	0.059	0.023	0.786
DNA	3861	58	0.139	30.40	1.644*	0.227	53.81

Table 3. Runtimes for RSAR and RSAR-SAT

The results show that RSAR-SAT is comparable to RSAR in the time taken to find reducts. However, RSAR regularly fails to find the smallest optimal subset, being misled in the search process. For larger datasets, the time taken for RSAR-SAT verification exceeds that of RSAR. Note that the verification stage involves simple chronological backtracking. There are ways in which this can be made more effective and less time-consuming.

5 Conclusion

This paper has presented a new DPLL-based technique for locating and verifying minimal subsets in the rough set context. The initial experimentation has shown that the method performs well in comparison to RSAR, which often fails to find the smallest subsets. Additional investigations to be carried out here include evaluating the proposed work against further well established heuristic-based approaches to reduct finding other than RSAR. Typical methods can be found in [7–9, 13].

DPLL resorts to chronological backtracking if the current assignment of variables results in the unsatisfiability of F. Much research has been carried out in developing solution techniques for SAT that draws on related work in solvers for constraint satisfaction problems (CSPs). Indeed the SAT problem can be translated to a CSP by retaining the set of boolean variables and their $\{0, 1\}$ domains, and to translate the clauses into constraints. Each clause becomes a constraint over the variables in the constraint. Unit propagation can be seen to be a form of forward checking.

In CSPs, more intelligent ways of backtracking have been proposed such as backjumping, conflict-directed backjumping and dynamic backtracking. Many aspects of these have been adapted to the SAT problem solvers. In these solvers, whenever a conflict (dead-end) is reached, a new clause is recorded to prevent the occurrence of the same conflict again during the subsequent search. Non-chronological backtracking backs up the search tree to one of the identified causes of failure, skipping over irrelevant variable assignments.

With the addition of intelligent backtracking, RSAR-SAT should be able to handle datasets containing large numbers of features. As seen in the preliminary results, the bottleneck in the process is the verification stage - the time taken to confirm that the subset is indeed minimal. This requires an exhaustive search of all subtrees containing fewer variables than the current best solution. Much of this search could be avoided through the use of more intelligent backtracking. This would result in a selection method that can cope with many thousands of features, whilst guaranteeing resultant subset minimality - something that is particularly sought after in feature selection.

References

1. A.A. Bakar, M.N. Sulaiman, M. Othman, M.H. Selamat. IP algorithms in compact rough classification modeling. Intelligent Data Analysis, Vol. 5, No. 5, pp. 419–429. 2001.
2. C.L. Blake and C.J. Merz. UCI Repository of machine learning databases. Irvine, University of California. 1998. `http://www.ics.uci.edu/~mlearn/`.
3. A. Chouchoulas and Q. Shen. Rough set-aided keyword reduction for text categorisation. Applied Artificial Intelligence, Vol. 15, No. 9, pp. 843–873. 2001.
4. M. Dash and H. Liu. Feature Selection for Classification. Intelligent Data Analysis, Vol. 1, No. 3, pp. 131–156. 1997.
5. M. Davis, G. Logemann and D. Loveland. A machine program for theorem proving. Communications of the ACM, vol. 5, pp. 394–397, 1962.
6. H.H. Hoos and T. Stützle. Towards a Characterisation of the Behaviour of Stochastic Local Search Algorithms for SAT. Artificial Intelligence, Vol. 112, pp. 213–232. 1999.
7. R. Jensen and Q. Shen. Semantics-Preserving Dimensionality Reduction: Rough and Fuzzy-Rough Based Approaches. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 12, pp. 1457–1471. 2004.
8. M. Kryszkiewicz. Comparative Study of Alternative Types of Knowledge Reduction in Inconsistent Systems. International Journal of Intelligent Systems, Vol. 16, No. 1, pp. 105–120. 2001.
9. H.S. Nguyen and A. Skowron. Boolean Reasoning for Feature Extraction Problems. In Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems, pp. 117–126. 1997.
10. Z. Pawlak. Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishing, Dordrecht. 1991.
11. L. Polkowski. Rough Sets: Mathematical Foundations. Advances in Soft Computing. Physica Verlag, Heidelberg, Germany. 2002.
12. A. Skowron and C. Rauszer. The discernibility matrices and functions in Information Systems. In: R. Slowinski (Ed.), Intelligent Decision Support, Kluwer Academic Publishers, Dordrecht, pp. 331–362. 1992.
13. J.A. Starzyk, D.E. Nelson, and K. Sturtz. Reduct Generation in Information Systems. Bulletin of the International Rough Set Society, Vol. 3, No. 1–2, pp. 19–22. 1999.
14. L. Zhang and S. Malik. The Quest for Efficient Boolean Satisfiability Solvers. In Proceedings of the 18th International Conference on Automated Deduction, pp. 295–313. 2002.