1

Feature Selection Using Rough Sets

Mohua Banerjee[1], Sushmita Mitra[2], and Ashish Anand[1]

[1] Indian Institute of Technology, Kanpur, India
{mohua,aanand}@iitk.ac.in
[2] Indian Statistical Institute, Kolkata, India
sushmita@isical.ac.in

Summary. Feature selection refers to the selection of input attributes that are most predictive of a given outcome. This is a problem encountered in many areas such as machine learning, signal processing, and recently bioinformatics/computational biology. Feature selection is one of the most important and challenging tasks, when it comes to dealing with large datasets with tens or hundreds of thousands of variables. Areas of web-mining and gene expression array analysis provide examples, where selection of interesting and useful features determines the performance of subsequent analysis. The intrinsic nature of noise, uncertainty, incompleteness of data makes extraction of hidden and useful information very difficult. Capability of handling imprecision, inexactness and noise, has attracted researchers to use rough sets for feature selection. This article provides an overview on recent literature in this direction.

1.1 Introduction

Feature selection techniques aim at reducing the number of irrelevant and redundant variables in the dataset. Unlike other dimensionality reduction methods, feature selection preserves the original features after reduction and selection. Benefit of feature selection is many fold: it improves subsequent analysis by removing the noisy data and outliers, makes faster and more cost-effective post-analysis, makes data visualization easier and provides a better understanding of the underlying process that generated the data.

Here, we will consider an example which will serve us as an illustration throughout the chapter. Consider gene selection from microarray data. In this problem, the features are expression levels of genes corresponding to the abundance of mRNA in a sample (e.g. particular time point of development or treatment), for a number of patients and replicates. A typical analysis task is to find genes which are differentially expressed in different cases or can classify different classes with high accuracy. Usually very few data samples are available altogether for testing and training. But, the number of features (genes) ranges from 10,000 to 15,000.

Rough set theory (RST) [13, 14] was developed by Pawlak as a tool to deal with inexact and incomplete data. Over the years, RST has become a topic of great interest to researchers and has been applied to many domains, in particular to knowledge databases. This success is due in part to the following aspects of the theory:

- only the facts hidden in data are analyzed;
- no additional information about the data is required;
- minimal knowledge representation is obtained.

Consider an information system consisting of a domain U of objects / observations and a set A of attributes/features. A induces a partition (classification) of U by A, by grouping together objects having identical attribute values. But the whole set A may not always be necessary to define the classification/partition of U. Many of the attributes may be redundant, and we may find *minimal* subsets of attributes which give the same classification as the whole set A. These subsets are called *reducts* in RST, and correspond to the *minimal feature sets* that are *necessary* and *sufficient* to represent a *correct* decision about classification. Thus RST provides a methodology for addressing the problem of relevant feature selection that could be applied, e.g. to the case of microarray data described earlier.

The task of finding reducts is reported to be NP-hard [15]. The high complexity of this problem has motivated investigators to apply various approximation techniques to find near-optimal solutions. There are some studies reported in literature, e.g., [17, 3], where genetic algorithms [9] have been applied to find reducts.

Genetic algorithms (GAs) provide an efficient search technique in a large solution space, based on the theory of evolution. A population of chromosomes is made to evolve over generations by optimizing a fitness function, which provides a quantitative measure of the fitness of individuals in the pool. When there are two or more conflicting characteristics to be optimized, often the single-objective GA requires an appropriate formulation of the single fitness function in terms of an additive combination of the different criteria involved. In such cases *multi-objective* GAs (MOGAs) [7] provide an alternative, more efficient, approach to search for optimal solutions.

In this article, we present various attempts of using GA's (both single- and multi-objective) in order to obtain reducts, and hence provide some solution to the challenging task of feature selection. The rest of the chapter is organized as follows. Section 1.2 introduces the preliminaries of rough sets and genetic algorithms. Section 1.3 deals with feature selection and the role of rough sets. Section 1.4 and Section 1.5 describe recent literature on single- and multi-objective feature selection using rough sets. Section 1.7 concludes the chapter.

1.2 Preliminaries

In this section we discuss the preliminaries of rough sets and genetic algorithms, with emphasis on multi-objective GAs. The issues relevant to this chapter are explained briefly. For detailed discussion, pointers to references are given at the appropriate places.

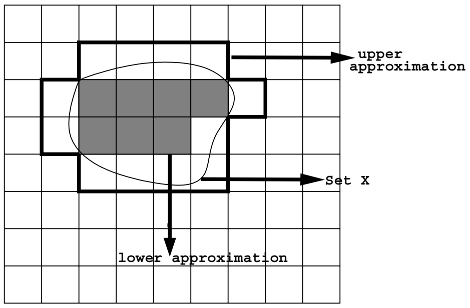

Fig. 1.1. Lower and upper approximations of a rough set

1.2.1 Rough sets

The theory of rough sets deals with uncertainty that arises from granularity in the domain of discourse, the latter represented formally by an *indiscernibility* relation (typically an equivalence) on the domain. The intention is to approximate a *rough* (imprecise) concept in the domain of discourse by a pair of *exact* concepts, that are determined by the indiscernibility relation. These exact concepts are called the lower and upper approximations of the rough concept. The lower approximation is the set of objects *definitely* belonging to the rough concept, whereas the upper approximation is the set of objects *possibly* belonging to the same. Fig. 1.1 illustrates a rough set with its approximations. The small squares represent equivalence classes induced by the indiscernibility relation on the domain. Lower approximation of the set X is shown as the shaded region and upper approximation consists of all the elements inside the thick line. The formal definitions of the above notions and others required for the present work are given below.

Definition 1. *An* **Information System** $\mathcal{A} = (U, A)$ *consists of a non-empty, finite set* U *of* objects *(cases, observations, etc.) and a non-empty, finite set* A *of* attributes a *(features, variables), such that* $a : U \to V_a$*, where* V_a *is a* value set*. Often, the attribute set* A *consists of two parts* C *and* D*, called* condition *and* decision *attributes respectively. In that case the information system* $\mathcal{A}$ *is called a* **decision table**. *Decision tables are termed* consistent*, whenever objects* x, y *are such that for each condition attribute* a*,* $a(x) = a(y)$*, then* $d(x) = d(y)$*, for any* $d \in D$*.*

Definition 2. *Let $B \subseteq A$. A B-***indiscernibility relation** *$IND(B)$ is defined as*

$$IND(B) = \{(x, y) \in U : a(x) = a(y),\ \forall a \in B\}. \tag{1.1}$$

It is clear that $IND(B)$ partitions the universe U into equivalence classes

$$[x_i]_B = \{x_j \in U : (x_i, x_j) \in IND(B)\},\ x_i \in U. \tag{1.2}$$

Definition 3. *The B-***lower** *and B-***upper approximations** *of a given set $X(\subseteq U)$ are defined, respectively, as follows:*
$\underline{B}X\ = \{x \in U : [x]_B \subseteq X\}$,
$\overline{B}X\ = \{x \in U : [x]_B \cap X \neq \phi\}$.

Reducts and Core

Reducts are the basic attributes which induce same partition on universe U as the whole set of attributes. These are formally defined below for both a general information system (U, A), and a decision table $(U, C \cup D)$. In an information system, there may exist many reducts.

For a given information system, $\mathcal{A} = (U, A)$, an attribute $b \in B \subseteq A$ is *dispensable* if $IND(B) = IND(B - \{b\})$, otherwise b is said to be *indispensable* in B. If all attributes in B are indispensable, then B is called *independent* in A. Attribute set $B \subseteq A$ is called *reduct*, if B is independent in A and $IND(B) = IND(A)$.

In a decision table $\mathcal{A} = (U, C \cup D)$, one is interested in eliminating redundant *condition* attributes, and *relative* (D)-reducts are computed.

Let $B \subseteq C$, and consider the B-*positive region* of D, *viz.*, $POS_B(D) = \bigcup_{[x]_D} \underline{B}[x]_D$. An attribute $b \in B(\subseteq C)$ is D-**dispensable** in B if $POS_B(D) = POS_{B\setminus\{b\}}(D)$, otherwise b is D-**indispensable** in B. B is said to be D-**independent** in $\mathcal{A}$, if every attribute from B is D-indispensable in B.

Definition 4. *$B(\subseteq C)$ is called a D-***reduct** *in $\mathcal{A}$, if B is D-independent in $\mathcal{A}$ and $POS_C(D) = POS_B(D)$.*

If a consistent decision table has a single decision attribute d, then $U = POS_C(d) = POS_B(D)$, for any d-reduct B.

The *core* is the set of essential attributes of any information system. Mathematically, $core(A) = \bigcap reduct(A)$, *i.e.*, the set consists of those attributes, which are members of all reducts.

Discernibility Matrix

D-reducts can be computed with the help of D-discernibility matrices [15]. Let $U = \{x_1, \cdots, x_m\}$. A D-**discernibility matrix** $M_D(\mathcal{A})$ is defined as an $m \times m$ matrix of the information system $\mathcal{A}$ with the (i, j)th entry c_{ij} given by:

$$c_{ij} = \{a \in C : a(x_i) \neq a(x_j), \text{and } (x_i, x_j) \notin IND(D)\}, \quad i, j \in \{1, \cdots, m\}. \tag{1.3}$$

A variant of the discernibility matrix, *viz.*, *distinction table* [17] is generally used in many applications to enable faster computation.

Definition 5. *A* **distinction table** *is a binary matrix with dimensions* $\frac{(m^2-m)}{2} \times N$*, where N is the number of attributes in A. An entry $b((k,j),i)$ of the matrix corresponds to the attribute a_i and pair of objects (x_k, x_j), and is given by*

$$b((k,j),i) = \begin{cases} 1 \text{ if } a_i(x_k) \neq a_i(x_j), \\ 0 \text{ if } a_i(x_k) = a_i(x_j). \end{cases} \tag{1.4}$$

The presence of a '1' signifies the ability of the attribute a_i to discern between the pair of objects (x_k, x_j).

1.2.2 Genetic Algorithms

Genetic algorithms [9] are heuristic techniques applied to solve complex search and optimization problems. They are motivated by the principles of natural genetics and natural selection. Unlike classical optimization methods, GAs deal with a population of solutions/individuals. With the basic genetic/evolutionary operators, like selection, crossover and mutation, new solutions are generated. A population of chromosomes, representing solutions, is made to evolve over generations by optimizing a fitness function, which provides a quantitative measure of the fitness of individuals in the pool. Selection operator selects better solutions to participate into crossover. Crossover operator is responsible for creating new solutions from the old ones. Mutation also creates new solutions, but only in the vicinity of old solutions. Mutation operator plays a great role in case of multi-modal problems.

When there are two or more conflicting objectives to be optimized, often weighted sum of objectives are taken to convert them as single-objective problem. In such cases multi-objective GAs provide an alternative, more efficient approach to searching for optimal solutions.

Multi-Objective GAs

As the name suggests, a multi-objective optimization problem deals with more than one objective function. In contrast to single-objective problems, multiple objective problems give rise to a set of optimal solutions, known as Pareto-optimal solution [5]. Over the past decade, a number of multi-objective genetic algorithms have been suggested. The basic advantage of multi-objective GAs over classical optimization methods is their ability to find multiple Pareto-optimal solutions in one single simulation run. Detailed discussion about multi-objective genetic algorithms can be found in [7]. Here, we will discuss the main features of Non-dominated Sorting Genetic Algorithm, viz. NSGA-II [8], which is one of the frequently used multi-objective genetic algorithms. This has been used in the studies of multi-objective feature selection using rough sets, as discussed in Section 1.5.

Among the different multi-objective algorithms, it has been observed that NSGA-II has the features required for a good multi-objective GA. It uses the concept of non-domination to select the better individuals, when they are compared with respect to all objectives. NSGA-II uses crowding distance to find the population density near each individual. To get an estimate of the density around the solution i, average distance of two solutions on either side of solution i along each of the objectives is taken. Crowding distance metric is defined in [8] such that the solution which resides in less crowded region will get higher value of crowding distance. Thereby NSGA-II tries to maintain the diversity among the non-dominated solutions. The algorithm assumes each solution in the population has two characteristics:

- a non-domination rank r_i;
- a local crowding distance d_i.

By using crowded tournament selection operator, NSGA-II not only tries to converge to Pareto-front but also tries to have diverse solution on the front. Crowded tournament selection operation is described below.

Definition 6. *It is said that solution i wins tournament with another solution j if any one of the following is true:*

(i) solution i has better rank i.e. $r_i < r_j$;

(ii) both the solutions are in the same front, i.e. $r_i = r_j$ but solution i is less densely located in the search space, i.e. $d_i > d_j$.

The NSGA-II algorithm can be summarized as follows.

1. Initialize the population;
2. Calculate the fitness;
3. Rank the population using the dominance criteria;
4. Calculate the crowding distance;
5. Do selection using crowding selection operator;
6. Do crossover and mutation to generate children population;
7. Combine parent and children population and do non-dominated sorting;
8. Replace the parent population by the best members of the combined population. Initially, members of lower fronts replace the parent population. When it is not possible to accommodate all the members of a particular front, that front is sorted according to the crowding distance. The number of individuals selected on the basis of higher crowding distance, is that which makes size of the new parent population same as size of the old one.

1.3 Feature Selection and Rough Sets

Feature selection plays an important role in data selection and preparation for subsequent analysis. It reduces the dimensionality of a feature space, and removes redundant, irrelevant, or noisy data. It enhances the immediate effects for any application by speeding up subsequent mining algorithms, improving data quality and thereby performance of such algorithms, and increasing the comprehensibility of their output. In this section we highlight the basics of feature selection followed by the role of rough sets in this direction.

1.3.1 Feature Selection

It is a process that selects a minimum subset of M features from an original set of N features ($M \leq N$), so that the feature space is optimally reduced according to an evaluation criterion. Finding the best feature subset is often intractable or NP-hard.

Feature selection typically involves the following steps:

- Subset generation: For N features, the total number of candidate subsets is 2^N. This makes an exhaustive search through the feature space infeasible, even with moderate value of N. Often heuristic and non-deterministic strategies are found to be more practical.
- Subset evaluation: Each generated subset needs to be evaluated by a criterion, and compared with the previous best subset.
- Stopping criterion: The algorithm may stop when either of the following holds.
  - A pre-defined number of features are selected,
  - a pre-defined number of iterations are completed,
  - when addition or deletion of any feature does not produce a better subset, or
  - an optimal subset is obtained according to the evaluation criterion.
- Validation: The selected best feature subset needs to be validated with different tests.

Search is a key issue in feature selection, involving search starting point, search direction, and search strategy. One also needs to measure the goodness of the generated feature subset. Feature selection can be supervised as well as unsupervised, depending on class information availability in data. The algorithms are typically categorized under filter and wrapper models [18], with different emphasis on dimensionality reduction or accuracy enhancement.

1.3.2 Role of Rough Sets

Rough sets provide a useful tool for feature selection. We explain its role with reference to the bioinformatics domain. A basic issue addressed in many practical applications, such as the gene expression analysis example discussed in

Section 1.1, is that the whole set of attributes/features is not always necessary to define an underlying partition/classification. Many of the features may be superfluous, and minimal subsets of attributes may give the same classification as whole set of attributes. For example, only few genes in microarray gene expression study are supposed to define the underlying process and hence working with all genes only reduces the quality and significance of analysis. In rough set terminology, these minimal subsets of features are just the *reducts*, and correspond to the minimal feature sets that are necessary and sufficient to represent underlying classification.

The high complexity of the reduct finding problem has motivated investigators to apply various approximation techniques to find near optimal solutions. There are some studies reported in literature, e.g., [17, 3], where genetic algorithms (GAs) have been applied to find reducts. Each of the studies in [17, 3] employs a single-objective function to obtain reducts.

The essential properties of reducts are:

- to classify among all elements of the universe with the same accuracy as the starting attribute (feature) set, and
- to be of small cardinality.

A close observation reveals that these two characteristics are of a conflicting nature. Hence the determination of reducts is better represented as a bi-objective problem. The idea was first presented in [1], and a preliminary study was conducted. Incorporating some modifications in this proposal, [2] investigates the multi-objective feature selection criteria for classification of cancer microarray data.

We will first discuss single-objective feature selection approach using rough sets in Section 1.4, and then pass on to the multi-objective approach in Section 1.5.

1.4 Single-objective Feature Selection Approach

Over the past few years, there has been a good amount of study in effectively applying GAs to find minimal reducts. First we will discuss algorithms proposed by Wroblewski [17].

Wroblewski's Algorithms

Wroblewski has proposed three heuristic-based approaches for finding minimal reducts. While the first approach is based on classical GAs, the other two are permutation-based greedy approaches.

Method 1

Solutions are represented by binary strings of length N, where N is the number of attributes (features). In the bit representation '1' means that the attribute is present and '0' means that it is not. The following fitness function is considered for each individual:

$$F_1(\boldsymbol{\nu}) = \frac{N - L_{\boldsymbol{\nu}}}{N} + \frac{C_{\boldsymbol{\nu}}}{(m^2 - m)/2}, \tag{1.5}$$

where $\boldsymbol{\nu}$ is a reduct candidate, N is the number of available attributes, $L_{\boldsymbol{\nu}}$ is the number of 1's in $\boldsymbol{\nu}$, $C_{\boldsymbol{\nu}}$ is the number of object combinations that $\boldsymbol{\nu}$ can discern, and m is the number of objects.

First part of the fitness function gives the candidate credit for containing less attributes (few 1's) and the second part of the function determines the extent to which the candidate can discern among objects.

When the fitness function is calculated for each individual, the selection process begins. A particular selection operator 'Roulette Wheel' is used. One-point crossover is used with crossover probability $P_c = 0.7$. Probability P_m of mutation on a single position of individual is taken as 0.05. Mutation of one position means replacement of '1' by '0' or '0' by '1'. Complexity of the algorithm is governed by that of fitness calculation, and it can be shown that the latter complexity is $O(Nm^2)$.

Method 2

This method uses greedy algorithms to generate the reducts. Here the aim is to find the proper order of attributes. We can describe this method as follows:

Step 1 : Generate an initial set of random permutations of attributes $\tau(a_1, \ldots, a_N)$, each of them representing an ordered list of attributes, i.e., $(b_1,\ldots,b_N) = \tau(a_1,\ldots,a_N)$.
Step 2 : For each ordered list, start with empty reduct $R = \phi$ and set $i \leftarrow 0$.
Step 3 : Check whether R is a reduct. If R is a reduct, Stop.
Step 4 : Else, add one more element from the ordered list of attributes, i.e. define $R := R \cup b_{i+1}$.
Step 5 : Go to step 3.

The result of this algorithm will be either a reduct or a set of attributes containing a reduct as a subset. GAs help to find reducts of different order.
Genetic Operators: Different permutations represent different individuals. The fitness function of an individual $\boldsymbol{\nu}$ is defined as:

$$F(\tau) = \frac{1}{L_{\boldsymbol{\nu}}}, \tag{1.6}$$

where $L_{\boldsymbol{\nu}}$ is the length of the subset R found by the greedy algorithm.

The same selection methods are used as in method 1. But different mutation and crossover operators are used, with an interchange of two randomly chosen attributes being done in mutation with some probability. Although one can choose any order-based crossover method, Wroblewski [17] has suggested the use of PMX (Partially Mapped Crossover [9]).

Method 3

This method again uses greedy algorithms to generate reducts. We can describe this method as follows:

Step 1 : Generate an initial set of random permutations of attributes $\tau(a_1, \ldots, a_N)$, each of which represents an ordered list of attributes, i.e., $(b_1,\ldots,b_N) = \tau(a_1,\ldots,a_N)$.
Step 2 : For each ordered list, define reduct R as the whole set of attributes.
Step 3 : Set $i \leftarrow 1$ and let $R := R - b_i$.
Step 4 : Check whether R is a reduct. If it is not, then undo step 3 and $i \leftarrow i + 1$. Go back to step 3.

All genetic operators are chosen as in method 2. The result of this algorithm will always be a reduct, the proof of which is discussed in [17]. However, a disadvantage of this method is its high complexity.

'Rough Enough' Approach to Calculating Reducts

Bjorvand [3] has proposed another variant of finding reducts using GAs. In his approach, a different fitness function is used. The notations used are the same as those for the previous methods.

$$F_1(\boldsymbol{\nu}) = \begin{cases} (\frac{N-L_{\boldsymbol{\nu}}}{N} + \frac{C_{\boldsymbol{\nu}}}{(m^2-m)/2})^2 & \text{if } C_{\boldsymbol{\nu}} < (m^2-m)/2 \\ (\frac{N-L_{\boldsymbol{\nu}}}{N} + (\frac{C_{\boldsymbol{\nu}}}{(m^2-m)/2} + \frac{1}{2}) \times \frac{1}{2})^2 & \text{if } C_{\boldsymbol{\nu}} = (m^2-m)/2. \end{cases}$$

Bjorvand argues that by squaring the fitness values it becomes easy to separate the different values. In case of total coverings (candidate is possibly a reduct), the second part of fitness values is added and then multiplied by 1/2 to avoid getting low fitness values as compared to the candidates almost covering all objects and also having a low number of attributes. Instead of constant mutation rate, Bjorvand uses adaptive mutation. If there are many individuals with same fitness value, higher mutation rate is chosen to avoid premature convergence or getting stuck at local minima. To give more preference to finding shorter reducts, higher mutation probability is chosen for mutation from 1 to 0 than for the reverse direction.

Other Approaches

In literature, there are some more approximation approaches to calculate reducts. Among these algorithms are Johnson's algorithm [12] and hitting set approach by Vinterbo *et. al* [16]. Johnson's algorithm is a greedy approach to find a single reduct. In the hitting set approach, non-empty elements of the discernibility matrix are chosen as elements of a multiset ℓ. The minimal hitting sets of ℓ are exactly the reducts. Since, finding minimal hitting sets is again an NP-hard problem [16], GAs are used for finding the approximate hitting sets (reducts).

Here the fitness function again has two parts, and a weighted sum of the two parts are taken. The following fitness function is defined for each candidate solution $\boldsymbol{\nu}$:

$$F(\boldsymbol{\nu}) = (1-\alpha) \times \frac{cost(A) - cost(\boldsymbol{\nu})}{cost(A)} + \alpha \times min\{\varepsilon, \frac{|[S \in \ell | S \cap \boldsymbol{\nu} \neq \phi]|}{|\ell|}\}. \quad (1.7)$$

In the above equation, α lies between 0 and 1, A is the set containing elements of the discernibility matrix. Candidate solutions $\boldsymbol{\nu}(\subset A)$ are found through evolutionary search algorithms. The parameter ε signifies a minimal value for the hitting fraction. First term of the above equation rewards the shorter element and the second term tries to ensure that hitting sets get reward. Cost function in above definition specifies the cost of an attribute subset. Means of defining the cost function are discussed in Rosetta [12]. One can trivially define a cost function as "the cardinality of the candidate $\boldsymbol{\nu}$, $|\boldsymbol{\nu}|$". Rosetta describes all the required parameters in brief, and detailed description of the fitness function and all parameters can be found in [16].

1.5 Multi-objective Feature Selection Approach

All the algorithms discussed above concentrate more on finding the minimal reducts and thus use variations of different single fitness functions. In many applications, such as gene expression analysis, a user may not like to just have a minimal set of genes, but explore a range of different sets of features and the relations among them. Multi-objective criterion has been used successfully in many engineering problems, as well as in feature selection algorithms [11]. Here, the basic idea is to give freedom to the user to choose features from a wide spectrum of trade-off features, which will be useful for them. As discussed in Section 1.3.2, a reduct exhibits a conflicting nature of having small cardinality and ability to discern among all objects. Combining this conflicting nature of reducts with MOGAs may give the desired set of trade-off solutions. This motivated the work in [1], to use multi-objective fitness functions for finding reducts of all lengths.

In this section we first discuss the multi-objective reduct finding algorithm proposed initially in [1]. This is followed by a discussion on a modification and implementation of this proposal, for gene expression classification problem, in [2].

1.5.1 Finding Reducts Using MOGA – I

In [1] the fitness function proposed by Wroblewski (eqn. (1.5)) was split into two parts, to exploit the basic properties of reducts as two conflicting objectives. The two fitness functions F_1 and F_2 are as follows:

$$F_1(\boldsymbol{\nu}) = \frac{N - L_{\boldsymbol{\nu}}}{N}, \tag{1.8}$$

$$F_2(\boldsymbol{\nu}) = \frac{C_{\boldsymbol{\nu}}}{(m^2 - m)/2}. \tag{1.9}$$

Hence, in this case, the first fitness function gives the solution credit for containing less attributes (few 1's) and the second fitness function determines the extent to which the solution can discern among objects.

Non-domination sorting brings out the difference between the proposed algorithm and the earlier algorithms. The algorithm makes sure that

- the true reducts come to the best non-domination front, and
- two different candidates also come into the same non-domination front, if one is not the superset of the other and the two can discern between the same number of objects.

For two solutions i and j, non-domination procedure is outlined as follows:

```
**if** $F_2^i = F_2^j$ and $F_1^i \neq F_1^j$
   **if** one (i, say) is superset of other (j)
      **then** put i at inferior domination level
   **else** put both in same domination level
**else**
   do regular domination checking with respect to two fitness values.
```

Remark 1. If the two solutions discern the same number of pair of objects, then their non-domination level is determined by the first objective. In this way, we make sure that candidates with different cardinality can come to the same non-domination level, if they do not violate the superset criteria, i.e. one solution is not a superset of the other. Explicit checking of superset is intended to ensure that only true reducts come into the best non-domination level.

The representation scheme of solutions discussed in Section 1.4, and the usual crowding binary tournament selection as suggested in Section 1.2.2 are used. The complete algorithm can be summarized in the following steps:

Step 1 : A random population of size n is generated.

Step 2 : The two fitness values for each individual is calculated.
Step 3 : Non-domination sorting is performed, to identify different fronts.
Step 4 : Crowding sort is performed to get a wide spread of the solution.
Step 5 : Offspring solution is created using crowded tournament selection, crossover and mutation operators.
Step 6 : Steps 2 to 5 are repeated for a pre-specified number of generations.

An advantage of the multi-objective approach can be shown by taking an example. Consider two solutions (a, b) and (c, d, e) giving the same classification. Then the earlier approach will give less preference to the second solution, whereas the proposed algorithm puts both solutions in the same non-domination level. Thus the probability of selecting reducts of larger cardinalities is the same as that of smaller cardinalities. The explicit check of superset also increases the probability of getting only true reducts.

This algorithm was implemented on some simple data sets. However, there are complexity problems when faced with large data.

1.5.2 Finding Reducts Using MOGA – II

For a decision table $\mathcal{A}$ with N condition attributes and a single decision attribute d, the problem of finding a d-reduct is equivalent to finding a minimal subset of columns $R(\subseteq \{1, 2, \cdots, N\})$ in the distinction table, satisfying

$$\forall (k, j) \exists i \in R : b((k, j), i) = 1, \text{whenever } d(x_k) \neq d(x_j).$$

So, in effect, the distinction table consists of N columns, and rows corresponding to only those object pairs (x_k, x_j) such that $d(x_k) \neq d(x_j)$. We call this shortened distinction table, a *d-distinction table*. Note that, as $\mathcal{A}$ is taken to be consistent, there is no row with all 0 entries in a d-distinction table.

In [2], NSGA-II is modified to effectively handle large datasets. We focus on two-class problems. An initial redundancy reduction is done to generate a reduced attribute value table $\mathcal{A}_r$. From this we form the d-distinction table consisting of N columns, with rows corresponding to only those object pairs (x_k, x_j) such that $d(x_k) \neq d(x_j)$. As object pairs corresponding to the same class do not constitute a row of the d-distinction table, there is a considerable reduction in its size, thereby leading to a decrease in computational cost.

The modified feature selection algorithm is implemented on microarray data consisting of three different cancer samples, as summarized in Table 1.1 [2]. After the initial redundancy reduction, the feature sets are reduced:

- Colon dataset: 1102 attributes for the normal and cancer classes,
- Lymphoma dataset: 1867 attributes for normal and malignant lymphocyte cells, and
- Leukemia dataset: 3783 attributes for classes ALL and AML.

The algorithm is run on the d-distinction table, with different population sizes, to generate reducts upon convergence. Fitness functions of eqns. (1.8)-(1.9),

Table 1.1. Usage details of the two-class microarray data

Data used	# Attributes	Classes	# Samples
Colon	2000	Colon cancer	40
		Normal	22
Lymphoma	4026	Other type	54
		B-cell lymphoma	42
Leukemia	7129	ALL	47
		AML	25

adapted to the case of two-class problems, are used. Results indicate convergence to 8, 2, 2 attributes respectively, for the *minimal* reduct on the three sets of two-class microarray gene expression data after 15,000 generations.

On the other hand, feature selection (without rough sets) in microarray gene expression analysis has also been reported in literature [10, 4, 6]. Huang [10] uses a probabilistic neural network for feature selection, based on correlation with class distinction. In case of *Leukemia* data, they report a reduction to a ten-genes set. For *Colon* data, a ten-genes set is generated. Chu *et al.* [6] employ a t-test based feature selection with a fuzzy neural network. A five-genes set is generated for *Lymphoma* data.

1.6 Example

In this section we will explain single-objective and multi-objective based feature selection through an example. For illustration, a sample data viz. *Cleveland* data set is taken from Rosetta [12]. The data has 14 attributes and it does not contain any decision variable. We have removed all objects with missing values and hence there are 297 objects. A part of the data with 10 attributes and 3 objects, is shown in Table 1.2.

Table 1.2. Part of Cleveland data, taken from Rosetta

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
63	M	Typical angina	145	233	T	LV hypertrophy	150	N	2.3
67	M	Asymptomatic	160	286	F	LV hypertrophy	108	Y	1.5
67	M	Asymptomatic	120	229	F	LV hypertrophy	129	Y	2.6
37	M	Non-anginal pain	130	250	F	Normal	187	N	3.5

To illustrate the single-objective feature selection approach, we have used Wroblewski's algorithm implemented in Rosetta, to find reducts. For multi-objective feature selection approach, the algorithm proposed in [1] is used on the same data set.

1.6.1 Illustration for the Single-objective Approach

Wroblewski's algorithm searches for reducts using GA until either it has exhausted the search space, or a pre-defined number of reducts has been found. Three different parameters can be chosen to control the thoroughness and speed of the algorithm.

Rosetta gives an option of finding reducts based on a complete set of objects or a subset of objects. In this example, we have chosen discernibility based on all objects. In case of decision table, one can select *modulo decision* to avoid discerning between objects belonging to the same decision class. Thus the resultant distinction table consists of object pairs belonging to different classes only. Rosetta also provides options to users for selecting parameters such as *number of reducts*, *seed* to start with different random populations, and *calculation speed* to choose one of the different versions of Wroblewski's algorithm discussed in Section 1.4. We have chosen *normal calculation speed* and *number of reducts = 50.* It may be remarked that other approaches (e.g. Vinterbo's method [16]) for finding reducts have also been implemented in Rosetta.

Results

On running Wroblewski's algorithm implemented in Rosetta, we obtain 25 reducts. Table 1.3 summarizes the results. All reducts were tested for reduct membership [15] and found to be true reducts.

Table 1.3. Results of Rosetta

Reduct Length	# Reducts
3	10
4	10
5	2
6	3

1.6.2 Illustration for the Multi-objective Approach

Solutions or chromosomes are binary strings of 1 and 0, of length equal to the total number of attributes. 1 indicates that the particular attribute is present. For example, 10001000000001 means that 1st, 5th and 14th attributes are present.

A random population of size n is generated. Each individual is nothing but a binary string, as just explained. Fitness functions are calculated and non-domination ranking is done, as discussed in Section 1.5.1. For illustration, let us take 4 individuals with the following strings and assume that all 4 individuals can discern among all the objects, i.e. second fitness function of all individuals is 1.0.

Individual 1: 10001000100000
Individual 2: 01001000010000
Individual 3: 11001000010000
Individual 4: 01001001000100

Since individual 3 is a superset of individual 2, individual 2 dominates it. But individuals 1 and 2 are non-dominated, and are put in the same front. Though individual 4 has cardinality four, which is more than the cardinality of individual 1 and individual 2, it is still kept in the same non-dominated front as individuals 1 and 2.

Offspring solutions are created using crowding selection, crossover and mutation operators.

Results

The following parameters are used to run the multi-objective algorithm.
Population Size = 50
Number of Generations = 500
Crossover Probability = 0.6
Mutation Probability = 0.08
Table 1.4 summarizes the results. Again, all reducts were tested for reduct membership [15] and found to be true reducts. A comparison with the results obtained using the single-objective algorithm indicates a greater effectiveness of the multi-objective approach.

Table 1.4. Results of multi-objective implementation

Reduct Length	# Reducts
3	10
4	17
5	14
6	7
7	2

1.7 Conclusion

In this article we have provided a study on the use of rough sets for feature selection. Handling of high-dimensional data requires a judicious selection of attributes. Feature selection is hence very important for such data analysis. *Reducts* in rough set theory, prove to be relevant for this task. However, reduct computation is a hard problem. It is found that evolutionary algorithms, particularly multi-objective GA, is useful in computing optimal reducts. Application to microarray data is described. An illustrative example is also provided, to explain the single- and multi-objective approaches.

Identifying the essential features amongst the non-redundant ones, also appears to be important in feature selection. The notion of *core* (the common part of all reducts) of an information system, could be relevant in this direction, and it may be a worthwhile future endeavor to conduct an investigation into its role.

Acknowledgment

Ashish Anand is financially supported by DBT project no. DBT/BSBE/20030360. We thank the referees for their suggestions.

References

[1] A. Anand. Representation and learning of inexact information using rough set theory. Master's thesis, Department of Mathematics, Indian Institute of Technology, Kanpur, India, 2002.

[2] M. Banerjee, S. Mitra, and H. Banka. Evolutionary-rough feature selection in gene expression data. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2005. Accepted.

[3] A. T. Bjorvand. 'Rough Enough' – A system supporting the rough sets approach. In *Proceedings of the Sixth Scandinavian Conference on Artificial Intelligence*, pages 290–291, Helsinki, Finland, 1997.

[4] L. Cao, H. P. Lee, C. K. Seng, and Q. Gu. Saliency analysis of support vector machines for gene selection in tissue classification. *Neural Computing and Applications*, 11:244–249, 2003.

[5] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making Theory and Methodology.* North-Holland, 1983.

[6] F. Chu, W. Xie, and L. Wang. Gene selection and cancer classification using a fuzzy neural network. In *Proceedings of 2004 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS 2004)*, volume 2, pages 555–559, 2004.

[7] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, London, 2001.

[8] K. Deb, S. Agarwal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[9] D.E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, 1989.

[10] C. -J. Huang. Class prediction of cancer using probabilistic neural networks and relative correlation metric. *Applied Artificial Intelligence*, 18:117–128, 2004.

[11] R. Jensen. *Combining rough and fuzzy sets for feature selection.* PhD thesis, School of Informatics, University of Edinburgh, 2004.

[12] J. Komorowski, A. Øhrn, and A. Skowron. The rosetta rough set software system. In W. Klasgen and J. Zytkow, editors, *Handbook of Data Mining and Knowledge Discovery*, chapter D.2.3. Oxford University Press, 2002.

[13] Z. Pawlak. Rough sets. *International J. Comp & Inf. Sc.*, 1982.

[14] Z. Pawlak. *Rough Sets, Theoretical Aspects of Reasoning about Data.* Kluwer Academic Publishers, Dordrecht, 1991.

[15] A. Skowron and C. Rauszer. The discernibility matrices and functions in information systems. In R. Slowinski, editor, *Handbook of Applications and Advances of the Rough Set Theory*, pages 331–362. Kluwer Academic Publishers, Dordrecht, 1992.

[16] S. Vinterbo and A. Øhrn. Minimal approximate hitting sets and rule templates. *International Journal of Approximate Reasoning*, pages 123–143, 2000.

[17] J. Wroblewski. Finding minimal reducts using genetic algorithms. In *Second Annual Joint Conference on Information Sciences*, pages 186–189, 1995.

[18] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, 2004.