A rough set-based effective rule generation method for classification with an application in intrusion detection

Prasanta Gogoi and Dhruba K. Bhattacharyya*

Department of Computer Science and Engineering,
Tezpur University,
Napaam, Tezpur 784028, Assam, India
Email: prasantagogoi24@gmail.com
Email: dkb@tezu.ernet.in
*Corresponding author

Jugal K. Kalita

Department of Computer Science,
College of Engineering and Applied Science,
University of Colorado,
Colorado Springs, CO 80918, USA
Email: kalita@eas.uccs.edu

Abstract: In this paper, we use Rough Set Theory (RST) to address the important problem of generating decision rules for data mining. In particular, we propose a rough set-based approach to mine rules from inconsistent data. It computes the lower and upper approximations for each concept, and then builds concise classification rules for each concept satisfying required classification accuracy. Estimating lower and upper approximations substantially reduces the computational complexity of the algorithm. We use UCI ML Repository data sets to test and validate the approach. We also use our approach on network intrusion data sets captured using our local network from network flows. The results show that our approach produces effective and minimal rules and provides satisfactory accuracy.

Keywords: rough set; LEM2; inconsistency; minimal; redundant; PCS; intrusion detection; network flow data.

Reference to this paper should be made as follows: Gogoi, P., Bhattacharyya, D.K. and Kalita, J.K. (2013) 'A rough set-based effective rule generation method for classification with an application in intrusion detection', *Int. J. Security and Networks*, Vol. 8, No. 2, pp.61–71.

Biographical notes: Prasanta Gogoi received his Bachelor's degree from Jorhat Engineering College, Assam, and Master's degree in Information Technology in 2000 from Tezpur University, Assam, India. He is currently working toward a PhD degree at Tezpur University in the Department of Computer Science and Engineering. His current research interests include network intrusion detection system.

Dhruba K. Bhattacharyya is currently working as a Professor in the Computer Science & Engineering Department at Tezpur University. He received his PhD in Computer Science from Tezpur University in 1999. His research areas include data mining, network security and content-based image retrieval. He has published more than 150 research papers in the leading international journals and conference proceedings. He has also written/edited four books. He is a Programme Committee/Advisory Body member of several International conferences/workshops.

Jugal K. Kalita is a Professor of Computer Science at the University of Colorado, Colorado Springs. He received his PhD degree from the University of Pennsylvania. His research interests are in natural language processing, machine learning, artificial intelligence and bioinformatics. He has published more than 120 papers in international journals and referred conference proceedings and has written two books.

1 Introduction

Rules can be thought of as prescribed standards on the basis of which decisions are made for specific purposes. A rule is a statement that establishes a principle or a standard, and serves as a norm for guiding or mandating action or conduct. A rule can be a conditional statement that tells the system how to react in a particular situation. In data mining, rule generation, in the form of association rules, was first introduced as market-basket analysis (Agrawal et al., 1993). In the case of association rules, rule generation is based on the concept of frequent pattern mining for the discovery of interesting associations and correlations among itemsets. Later, methods were developed for classification rule mining (Han and Kamber, 2001). It usually, rules are represented in terms of a set of *IF-THEN* patterns. The *IF* part (or the left hand side) of a pattern is known as rule antecedent or precondition. The *THEN* part is the consequent. In the rule antecedent, the condition consists of one or more attribute tests which are logically connected by *AND* operations. The consequent contains the class prediction. A rough set-based approach to mining concise rules from inconsistent data is presented in the work of (Sai et al., 2006). In this approach, a heuristic algorithm is used to build concise classification rules. Rule-based methods have been applied successfully in many applications in decision making and prediction such as in medical research (Paetz and Brause, 2002) in economics and finance (Grosan and Abraham, 2006) and network security (Vollmer et al., 2011)).

Three types of rule generation techniques are prevalent: frequent association rule mining, rare association rule mining, and multi-objective rule mining. In Frequent Association Rule Mining (FARM) (Agrawal et al., 1993), an association rule is called frequent if its support is not less than a given minimum support. A frequent association rule is valid if it has a minimum confidence as well, above a user defined threshold.

In Rare Association Rule Mining (RARM) (Kiran and Reddy, 2010) an association rule r is called rare or infrequent if its support is not more than a given maximum support. It means that a rule r is rare if its support is less than a given minimum support. A rare association rule is valid if it is confident.

In Multi-Objective Rule Mining (MORM) (Ghosh and Nath, 2004), the rule mining problem is considered a multi-objective problem rather than a single objective one. In evaluating a rule, measures such as support count, comprehensibility and interestingness are considered to be different objectives; the rule mining problem has to satisfy. *Support count* is the number of records, which satisfy all the conditions present in the rule. *Comprehensibility*, which tries to quantify the understandability of the rule that can be measured by the number of attributes involved in the rule. *Interestingness* can be measured by how surprising a rule is.

The rules generated by the above three approaches often are incapable of

- handling inconsistency in the database,
- generating minimal rule sets, and
- generating non-redundant rule sets as discussed in the work of Slowinski (1992).

In many real life or synthetic data sets, inconsistency is a common problem. Inconsistency is caused by the existence of an indiscernbility relation in the decision table. A data set is represented in Table 1, where each row represents an object or record. Every column represents an attribute that can be measured for each object. This table is a decision table. Attributes are of two categories: condition and decision. The indiscernbility relation occurs in a decision table if in objects of equivalent condition attributes, decision attributes are different. Consider the decision table with objects p_1, p_2, p_3 in Table 1. Condition attributes are A and C, and the decision attribute is D. The attributes of objects p_1 and p_3 are equivalent whereas their decision attributes are different. Here, objects p_1 and p_3 are indiscernible and the decision table has inconsistency.

Table 1 Inconsistent data set

Objects	*Condition Attributes*		*Decision Attribute*
	A	*C*	*D*
p_1	low	high	yes
p_2	low	low	no
p_3	low	high	no

A decision table which contains an indiscernible relation is called inconsistent. In the conventional method, one preprocesses the data set to eliminate the inconsistency, before it is used for classification rule generation (Han and Kamber, 2001). The preprocessing sometimes may alter, or eliminate some data and consequently may lead to the loss of information.

One important aspect in the analysis of rule generation is the presence of overlaps among the classification results performed by the rules. If a rule overlaps another rule, the rule is called a *redundant rule*. The presence of a redundant rule can increase the number of rules and consequently cause misclassification in a data set if more than one rule is used to classify a single data object. Usually, when one uses an existing method for rule generation, the creation of non-redundant rules is rare.

None of the previously mentioned techniques can generate classification rules that are minimal. To address this limitation, Rough Set Theory (RST) was introduced for classification rule generation on inconsistent data sets. RST was first introduced by Pawlak (1982) in the year 1982, and is especially well suited to deal with inconsistencies (Slowinski, 1992). One of the major advantages of RST is that it does not require any additional information about the data such as probability distributions or grade memberships. It is also capable of handling inconsistency.

1.1 Motivation

The cost of developing and maintaining rule sets is an important issue for rule-based systems. Rule generation methods have been used for numerous applications in various domains. Many of these methods have been developed to solve focused problems in particular application domains, while others have been developed in a more generic fashion. Rule generation approaches found in the literature (e.g., Agrawal et al., 1993; Kiran and Reddy, 2010; Ghosh and Nath, 2004) have varying scopes and abilities. A new heuristic approach based on RST is found in the work of Liu and Li (2008) for generation of shorter rules and comparison of rules' abilities. The selection of an approach for rule generation depends on the domain of application, data type and availability of labelled data. So, an adequate knowledge of existing approaches is highly essential to select an appropriate method for a specific domain. Based on our study, we observe that most existing methods for rule generation attempt to address the issue of inconsistency handling by eliminating the inconsistent data prior to rule generation. However, such elimination of inconsistent data may lead to loss of information, which is evident from the work of Sai et al. (2006). In this paper, we aim to provide a method of rule generation based on the work of Grzymala-Busse (1997) and Liu and Li (2008), for handling inconsistency in the database as well as generation of minimal and non-redundant rule sets for classification. We also want to establish the effectiveness of the proposed method in the context of classification problems on high-dimensional data, e.g. network intrusion detection used in the work of Gogoi et al. (2010, 2011, 2012) and Bhuyan et al. (2011)

1.2 Our contributions

Our contributions in this paper are following:

- We develop a method to find indiscernibility relations in a data set to find inconsistencies.
- We present a method to determine lower and upper approximations for inconsistent data.
- We generate minimised and non-redundant rule sets by using lower and upper approximations for classification.
- We establish the effectiveness of the proposed rule generation method using several real-life high-dimensional data sets, including network flow data we generate ourselves.

1.3 Organisation of the paper

The remainder of this paper is organised as follows. In the next section, we present related work on rule generation. In Section 3, we describe the proposed method of rule generation. Experimental results are presented in Section 4. In Section 5, we outline conclusions and future work.

2 Related work

Rough sets were introduced to classify in the presence of imprecise and incomplete information. *Reduct* and *core* are two important concepts in RST. A *reduct* is a subset of attributes that is sufficient to describe the decision attributes. Finding all the reduct sets for a data set is an NP-hard problem (Agrawal et al., 1993). Approximation algorithms are used to obtain a reduct set (Kiran and Reddy, 2010). All reducts contain the core. The *core* represents the most important information in the original data set. The intersection of all possible reducts is the core.

There have been attempts at applying RST to rule discovery. Rules and decisions generated from the reducts are representative of the knowledge that can be acquired from a data set. In the work of Li and Cercone (2005), two modules were used in the association rule mining procedure for supporting organisational knowledge management and decision making. Self-organising maps were used to cluster sale actions based on the similarities in the characteristics of a given set of customer records. RST was used on each cluster to determine rules to explain associations. Adetunmbi et al. (2008) used rough sets to all reducts of the data that contain a minimal subset of attributes associated with a class label for classification. RST that can help determine whether there is redundant information in the data so that one can gather the essential data needed for applications. The RST-based rule generation approach is able to generate minimal and non-redundant rule sets in inconsistent data.

2.1 Rough set

- RST is an approach to computing in the presence of vagueness. It is an extension of classical set theory, for use when representing vagueness or imprecision. A rough set is concerned with working on the boundary regions of a set (Pawlak et al., 1995). The basic concept of RST is the notion of *approximation space*, which is an ordered pair $I = (U, R)$, where
- I is an information system,
- U is a nonempty set of objects, called the *universe*, and
- R is an equivalence relation on U, called the *indiscernibility relation*. If $x, y \in U$ and xRy, x and y are indistinguishable in I.

Each equivalence class induced by R, is called an *elementary set* in A, a set of finite attributes is represented as U/R. A *definable set* in I is any finite union of elementary sets in I. For $x \in U$, let $[X]_R$ denote the equivalence class of R containing x. For each $X \subseteq U$, X is characterised in I by a pair of sets, its *lower* and *upper approximations* in I is defined respectively as:

$$\begin{aligned} \underline{R}X &= \{x \in U \mid [X]_R \subseteq X\}, \\ \overline{R}X &= \{x \in U \mid [X]_R \cap X \neq \phi\}. \end{aligned} \tag{1}$$

A rough set in I consists of all subsets of U with the same lower and upper approximations.

Definition 1: *A decision table (Pawlak et al., 1995) is a quadruple*

$$D_t = \langle U, A, \{V_a \mid a \in A\}, \{F_a \mid a \in A\} \rangle \tag{2}$$

where U is a universe of finite non-empty objects, A is a set of finite attributes, V_a is the domain of the attribute $a \in A$, and $F_a : U \to V_a$ is a function, which assigns a value from the domain of attribute a to each object in U. In a decision table, there is an outcome of classification. This posteriori knowledge is expressed by one distinguished attribute called the *decision attribute* $\{d\}$, where $d \notin A$. The elements of A are called *conditional attributes* or *conditions*.

Definition 2: *An indiscernibilty relation (Pawlak et al., 1995) in a decision table with any $R \subseteq A$ is associated with an equivalence relation E_R:*

$$E_R = \{(x, y) \in U \times U : \forall a \in R, F_a(x) = F_a(y)\}. \quad (3)$$

The equivalence relation E_R partitions the set of objects U into disjoint subsets, i.e., equivalence classes. Such a partition of the set U is denoted by U/R. If two elements x and y in U belong to the same equivalence class, x and y are indiscernible from each other by attributes from R.

Definition 3: *A reduct (Pawlak et al., 1995) is a set of attributes that preserve partition. In other words, a reduct is the minimal subset of attributes that enable the same classification of elements of the universe as the whole set of attributes. In order to express the idea of a reduct, let $B \subseteq A$ and $a \in B$ in an information system $I = (U, A)$ where U is the universe of objects, A is the set of attributes, and R(B) is a binary relation.*

Attributes other than the reduct are redundant attributes. The removal of redundant attributes cannot deteriorate the classification. Usually, there are several reducts in a data set.

Definition 4: *The core (Pawlak et al., 1995) is the set of all indispensable attributes, i.e. it is the intersection of all reducts. The core is included in every reduct, i.e, each element of the core belongs to some reduct.*

Thus, the core is the most important subset of attributes, for none of its elements can be removed without affecting the classification. If $Red(B)$ is the set of all reducts of B in an information system $I = (U, A)$ where $B \subseteq A$, the *core* of B is defined as:

$$core(B) = \bigcap Red(B). \quad (4)$$

2.2 HCRI algorithm

It is a heuristic algorithm for mining concise rules from inconsistent data (Sai et al., 2006). This method is based on the variable precision rough set model. It deals with inconsistent data to mine concise rules. It first computes the reduct for each concept, and then computes the reduct for each object. It adopts a heuristic method to build concise classification rules for each concept. To compute the equivalence classes, it uses two hash functions, which substantially reduce the complexity to $O(n)$, $n = |U|$. The hash functions compute the cardinality of the lower approximation. The input to the method is a set of inconsistent objects U and the output is a set of concise rules satisfying a given classification accuracy.

2.3 LEM2

LEM2, Learning by Example Module, Version 2 is a machine learning algorithm based on RST (Grzymala-Busse, 1988). LEM2 learns a discriminant rule set, i.e. learns the smallest set of minimal rules describing a concept. This algorithm can generate both certain and possible rules from a decision table with attributes being numerical as well categorical. LEM2 needs discretisation for numerical attributes.

For inconsistent data, LEM2 induces two sets of rules: the *certain rule set* and the *possible rule set*. The first set is computed from lower approximations of concepts and the second one from upper approximations. It is assumed that the rule set is used automatically by a classification component. Nevertheless, induced rules are available and comprehensible by the user. Thus, it is possible to use rules manually, like in other systems.

Next, the LEM2 algorithm is discussed with an example.

Example: *Consider a data set with records $p_1, p_2, \ldots, p_8$. The conditional attributes are A and C. The decision attribute is D. The values of A and D are y and n while the values of C are n, h and v as shown in Table 2.*

Table 2 Example data set

Records	*Attributes*		*Decision*
	A	*C*	*D*
p_1	y	n	n
p_2	y	h	y
p_3	y	v	y
p_4	n	n	n
P_5	n	h	n
p_6	n	v	y
p_7	n	h	Y
p_8	n	v	n

The two concepts are $\{p_1, p_4, p_5, p_8\}$ and $\{p_2, p_3, p_6, p_7\}$. The elementary sets of the indiscernibility relation defined by the set of attributes $\{A, C\}$ are $\{p_1\}$, $\{p_2\}$, $\{p_3\}$, $\{p_4\}$, $\{p_5, p_7\}$ and $\{p_6, p_8\}$. Clearly, it is a case of inconsistency because, neither $\{p_5, p_7\}$ nor $\{p_6, p_8\}$ are subsets of any of the above concepts.

Let X be a concept. The greatest definable set contained in X and the least definable set containing X are computed. The former is called a lower approximation and the latter is called an upper approximation of X.

The concept $\{p_2, p_3, p_6, p_7\}$ has the lower approximation $\{p_2, p_3\}$ and the upper approximation $\{p_2, p_3, p_5, p_6, p_7, p_8\}$, because the elementary sets $\{p_2\}$ and $\{p_3\}$ are completely present in $\{p_2, p_3, p_6, p_7\}$. Similarly, the concept $\{p_1, p_4, p_5, p_8\}$ has the lower approximation $\{p_1, p_4\}$ and the upper approximation $\{p_1, p_4, p_5, p_6, p_7, p_8\}$.

The elements of the upper approximation, which do not belong to the lower approximation of a concept, form the boundary region. Elements present in the boundary region are not considered members of the concept. Here, the set $\{p_5, p_6, p_7, p_8\}$ is the boundary region for both concepts.

For any concept, the rules induced from a lower approximation are certainly valid. These are known as *certain rules*. The rules induced from an upper approximation are possibly valid, and are known as *possible rules*.

Certain rule from $\{p_1, p_4\}$: $(C,n) \to (D,n)$.

Certain rules from $\{p_2, p_3\}$:

$(A,y) \wedge (C,h) \to (D,y)$, and $(A,y) \wedge (C,v) \to (D,y)$.

Possible rules from $\{p_1, p_4, p_5, p_6, p_7, p_8\}$:

$(A,n) \to (D,n)$, and $(B,n) \to (D,n)$.

Possible rules from $(p_2, p_3, p_5, p_6, p_7, p_8)$:

$(C,h) \to (D,y)$, and $(C,v) \to (D,y)$.

The LEM2 algorithm is a single local covering approach. It yields a single minimal discriminant description (Grzymala-Busse, 1997), which means, it learns the smallest set of minimal rules for every concept. The local coverings are constructed from a minimal complex. Formal definitions of minimal complex and local covering are given in the work of Pawlak et al. (1995).

Definition 5: *Minimal complex and local covering – Let B be a non-empty lower or upper approximation of a concept represented by a decision-value pair (d, w). The set T is a minimal complex of B if and only if B depends on T and no proper subset T' of T exists such that B depends on T'. Let $\Im$ be a non-empty set of attribute-value pairs for equivalence class [T] of T. Then $\Im$ is the local covering of B iff* [(*i*)]

1 Each member T of $\Im$ is a minimal complex of B,

2 $\bigcup_{T \in \Im} [T] = B$, and

3 $\Im$ is minimal, i.e. $\Im$ has the smallest possible number of members.

LEM2 is suitable for rule generation for inconsistent data. In the next section, we introduce an enhanced version of the LEM2 algorithm to handle, especially for the high-dimensional classification problems. The proposed method is superior to LEM2, because of the following features.

- Fast identification of the presence of inconsistency and accordingly computation of lower and upper approximations;
- Incorporation of attribute priority assignment of lower approximation.

3 Proposed work

We have developed an effective rule generation technique using RST based on the LEM2 algorithm. The method works well especially for data sets with inconsistencies. Notations used in describing the proposed method are given in Table 3.

Our method starts with inconsistency checking for each concept in the data set. If it finds inconsistency, it computes an upper approximation and a lower approximation. To compute inconsistency and to find the upper and lower approximations, it uses the following method to support the LEM2-based rule generation technique.

Table 3 Notation of approximation method

Symbol	*Meaning of symbol*
Γ	Set of concepts
C_i, C_j C_i, C_j	Concepts
$\{C_{ij}\}$	Indiscernible objects of concept C_i
c, c'	Objects of Concepts
U_{apprx}	Upper approximation
L_{apprx}	Lower approximation
β	Set of concepts with L_{apprx} and U_{apprx}
$\Im$	A single local covering for the set β, i.e., it yields the smallest set of minimum rules for the entire set β
C_p	The members of β; it is either upper or lower approximation of a concept or a concept itself, depending upon the existence of inconsistency
M	The members of C_p
B	A selected concept
G	Temporary storage of B
T	Set of attribute value pairs
t	Member of T, i.e. $t \in T$
$[t]$	Equivalence class of t, i.e. the set of all objects which have the attribute-value pair t
$T(G)$	Set of attribute-value pairs which are present in objects of G, i.e. $T(G) := \{t \mid [t] \cap G \neq NULL\}$
$[T-\{t\}]$	Set of objects which have attribute-value pairs other than t
S	Member of $\Im$ other than T, i.e. $S \in \Im - \{T\}$

3.1 Generation of upper and lower approximations

The computation of upper and lower approximations for each concept in the data set is given in Algorithm 1. Let Γ be the set of concepts in a data set. The upper and lower approximations of each concept in Γ are created as follows. The attribute-value pairs for each object of a concept are compared with attribute-value pairs for an object of another concept. If attribute-value pairs of both objects match, both objects are indiscernible and the set containing these two objects becomes an elementary set. Let object c of concept C_i and object c' of concept C_j be indiscernible and objects c and c' make an elementary set. Finding indiscernibility and elementary sets takes place in lines 9–12 in Algorithm 1. The upper approximation of concept C_i is the union of all the objects of the elementary sets with respect to concept C_i

and object list of the concept itself. The lower approximation of concept C_i is the subtraction of all the objects of the elementary sets with respect to concept C_i from the object list of concept C_i. Determination of upper and lower approximations of concepts is given in lines 19–21 in Algorithm 1.

Algorithm 1 CLU

```
Input: set of concepts $\Gamma$;
Output: set of concepts $\beta$;
1: while $\Gamma \neq NULL$ do
2:   for each concept $C_i \in \Gamma$
3:   while $\Gamma \neq NULL$ do
4      for each concept $C_j \in \Gamma$
5      if $C_i \neq C_j$ then
6:       while $C_i \neq NULL$ do
7:         for each object $c \in C_i$;
8:         while $C_j \neq NULL$ do
9:           for each object $c' \in C_j$
10:          if all attribute–value pairs of
               $c \equiv$ all attribute–value pairs of $c'$
             then
11             $C_i$ is inconsistent
               and $(c, c')$ inconsistent pair;
12:            indiscernible objects $\{C_{ij}\}$ of
               concept $C_i$ is updated with $(c, c')$;
13:          end if
14:        end while
15:      end while
16:    end if
17:  end if
18:  if $C_i$ is inconsistent then
19:    $U_{apprx}$ is union of $\{C_i\}$ and $\{C_{ij}\}$,
       i.e., $U_{apprx} = \{C_i\} \cup \{C_{ij}\}$;
20:    $L_{apprx}$ is subtraction of $\{C_{ij}\}$ from $\{C_i\}$,
       i.e., $L_{apprx} = \{C_i\} - \{C_{ij}\}$;
21:    update $\beta$ with $U_{approx}$ and $L_{apprx}$;
22:  else
23:    update $\beta$ with $C_i$;
24:  end if
25: end while
```

3.2 Rule generation

The rule generation method is based on the LEM2 algorithm. LEM2 is a single local covering approach and it yields a single minimal discriminant description. In LEM2, the user may or may not consider any attribute priority. In contrast to LEM2, the proposed rule generation method extracts the output of Algorithm 1 to compute upper and lower approximations and it considers the attribute priority for the lower approximation. The method is given in Algorithm 2.

Attribute priority assignment: The lower approximation, L_{approx}, of a concept is given as a higher priority input to the rule generation algorithm than the higher approximation, U_{approx}, of a concept. This is shown in lines 10 and 44 in Algorithm 2.

Algorithm 2 Rule-gen

```
Input: a set of concepts $\beta$;
Output: a set rules $\Im$;
1: while $\beta \neq NULL$ do
2:   for each concept in $\beta$
3:   if found inconsistency then
4:     $L_{approx}$ and $U_{approx}$ will be the member of $C_p$
5:   else
6:     $L_{approx}$ = concept will be the member of $C_p$;
7:   end if
8:   while $C_p \neq NULL$ do
9:     for each member $M \in C_p$
10:    $B = L_{approx}$;
11:    $G := B$;
12:    $\Im := NULL$;
13:    while $G \neq NULL$ do
14:      $T := NULL$;
15:      $T(G) := \{t \mid [t] \cap G \neq NULL\}$;
16:      while $T = NULL$ or $[T] \not\subseteq B$ do
17:        Select an attribute–value pair $t \in T(G)$
             with the highest attribute priority;
18:        if a tie occurs then
19:          Select a $t \in T(G)$
             such that $|t \cap G|$ is maximum;
20           if another tie occurs then
21:            select a $t \in T(G)$
       with the smallest cardinality of $[t]$;
22:            if further tie occurs then
23:              select the first pair;
24:            end if
25:          end if
26:        end if
27:        $T := T \cup \{t\}$;
28:        $G := [t] \cap G$;
29:        $T(G) := \{t \mid [t] \cap G \neq NULL\}$;
30         $T(G) := T(G) - T$;
31:      end while
32:      for each t in T
33:      if $[T - \{t\}] \subseteq B$ then
34:        $T := T - \{t\}$;
35:      end if
36:      $\Im := \Im \bigcup \{T\}$;
```

```
37:             $G := B - \bigcup_{T\in\Im}[T]$;
38:        end while
39:        for each T in $\Im$;
40: if $\bigcup_{S\in\Im-\{T\}}[S] = B$ then
41:   $\Im := \Im - \{T\}$;
42:  end if
43:  if M is inconsistent then
44:        $B = U_{approx}$;
45:     end if
46:  end while
47: end while.
```

Attribute-value pair selection: A list of all the attribute-value pairs $T(G)$ for a concept G is created. For each attribute-value pair t where $t \in T(G)$, a list $\{t\}$ of objects is selected where t exists. Then the algorithm finds a list of intersections between each $\{t\}$ of $t(G)$ and objects list $\{G\}$of concept G. It selects t from the list of intersections with the highest cardinality. If a tie occurs in the highest cardinality, it selects t from among those tied such that the selected one has the smallest cardinality of $\{t\}$. If another tie occurs in the smallest cardinality, it selects t which appears first in the list of tie occurred. The selection of attribute-value pairs is shown in lines 16–22 in Algorithm 2.

Rule generation: Let t be a selected attribute-value pair for concept G. t is the antecedent of a rule, $(t) \to G$, if $\{t\} \subseteq \{G\}$ for the list of objects $\{t\}$ and the list of objects $\{G\}$ of concept G. If $\{t\} \not\subseteq \{G\}$, the list of objects $\{t\}$ is reduced from the list of objects $\{G\}$ of concept G, i.e. $\{G'\} = \{G\} - \{t\}$ and G' is a reduced concept. For the reduced concept, another attribute-value pair t' is selected from set of attribute-value pairs $T(G')$ of the reduced concept G' to build a rule as $(t) \wedge (t') \to \{G\}$. The object list $\{G'\}$ may be further reduced with the object list $\{t'\}$ if intersection of $\{t\}$ and $\{t'\}$ is not a subset of object list $\{G\}$, i.e. $\{t\} \cap \{t'\} \not\subseteq \{G\}$. Additional attribute-value pairs are selected till the intersection of the object lists $\{t\}, \{t'\}, \{t''\}$, etc., of all the selected attribute-value pairs becomes a subset of the object list of concept G, i.e. $\{t\} \cap \{t'\} \cap \ldots \subseteq G$. The rule generation steps are given in lines 26, 29 and 31–35 in Algorithm 2.

Minimal rule set generation: Let $\Im$ be the set of generated rules for all the concepts. To create a minimal rule set from $\Im$, rules which are covered by another rule are removed. Let a rule S' classify a list of objects $\{S'\}$ and a rule S classify a list of objects $\{S\}$. If $\{S'\}$ is a subset of $\{S\}$, rule S' is covered by rule S, and S' is removed from rule set to make rule set $\Im$ minimal. The creation of minimal rule set is shown in lines 39–40 in Algorithm 2.

3.3 Complexity analysis

Let n be the total number of objects in our sample data set. In order to verify that our algorithm handles inconsistency property, we have to compare each individual object with other objects present in our data set. So, the complexity of the computation of upper and lower approximations is $O(n^2)$. LEM2 has the complexity of $O(nm)$ where n is the number of objects and m is the number of attributes. The complexity of our proposed method, comprising of Algorithms 1 and 2, is $O(n^2) + O(nm)$.

Our algorithm expects the sample data set to have inconsistency. The inconsistencies may arise in only some of the concepts but not all. So, our algorithm initially computes U_{approx} and L_{approx} for those concepts only. The concepts which do not have inconsistency are fed to the program without finding upper and lower approximations. The upper and lower approximations are computed before execution of the main algorithm.

4 Experimental results

4.1 Environment used

All necessary experiments were carried out on an Intel workstation with Intel core 2 Quad @2.4 GHz, 2 GB RAM, 160 GB HDD. The programs were developed in *C* in a Linux environment.

4.2 Data sets used

4.2.1 UCI data set

The proposed method was tested on several commonly used data sets from the UCI Machine Learning Repository data set (Blake and Merz, 2001) and a data set mentioned in the work of (Slowinski, 1992). The descriptions of data sets we experimented with are given in Table 4.

Table 4 UCI ML repository (Blake and Merz, 2001) and other (Slowinski, 1992) data set description

	Data sets	*Data Types*	*Instance Sizes*	*Attributes Sizes*
1	Demo (Slowinski, 1992)	Categorical	9	6
2	Breast cancer	Categorical	286	10
3	Congressional Voting	Categorical	435	17
4	Mushroom	Categorical	8124	23
5	Glass	Numeric	214	10
6	Iris	Numeric	150	4
7	New-Thyroid	Numeric	215	6
8	Balloon	Categorical	20	5
9	Hayes-roth	Categorical	160	6
10	Soyabean-Small	Categorical	307	35
11	Balance-scale	Categorical	625	5
12	Liver Disorders	Numerical	345	7

4.2.2 Real life network intrusion data set

The proposed method was also evaluated using our own data set that includes various types of features extracted from network flow data captured using our local network.

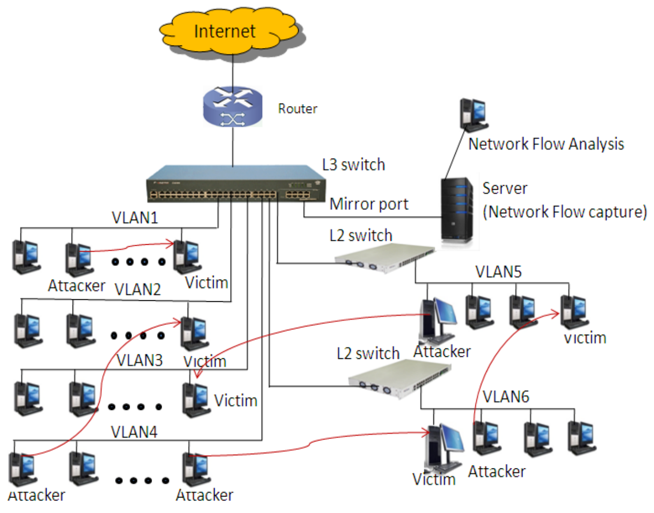
Using existing attack tools, we generated a group of attacks against a local network server and collected the produced traffic as known attack traffic. The existing attacks along with the corresponding tools for their generation are presented in Table 5 (Mixter, 2003). These attacks and tools are also used by Amini et al. (2006).

Table 5 Attack list

Attack	*Generation Tool*	*Attack*	*Generation Tool*
bonk	targa2.c	oshare	targa2.c
jolt	targa2.c	saihyousen	targa2.c
nestea	targa2.c	smurf	smuf4.c
newtear	targa2.c	fraggle	fraggle
syndrop	targa2.c	syn	Nmap
teardrop	targa2.c	xmas	Nmap
winnuke	targa2.c	window	targa2.c
1234	targa2.c	land	targa2.

The experimental set-up of the testbed for network flow capture includes one router, one L3 switch, two L2 switches, one server, two workstations and 40 nodes. Six VLANs are created from the L3 switch and L2 switch, and nodes and workstations are connected to separate VLANs. The L3 switch is connected to a router through an internal IP router and the router is connected to the internet through an external IP router. The server is connected to the L3 switch through a port called mirror port to observe traffic activity to the switch. Another LAN of 350 nodes is connected to other VLANs through five L3 and L2 switches and three routers. Attacks are launched within our testbed as well as from another LAN via the internet. In launching attacks within the testbed, nodes of one VLAN are attacked from nodes of another VLAN and as well as the same VLAN. Normal traffic is created within our testbed in restricted conditions after disconnecting another LAN. The traffic activities to our testbed are observed in the computer connected to the mirror port. A diagram of the testbed is shown in Figure 1.

Figure 1 Testbed for generating our data set (see online version for colours)



The network flow is a unidirectional sequence of packets passing through an observation point in the network during a certain time interval between source and destination hosts. All traffic belonging to a particular flow has a set of common properties. The NetFlow protocol, the IPFIX standard (Quittek et al., 2004; Claise, 2004), provides a summarisation of information concerning the router or switch traffic. Network flow is identified by source and destination IP addresses as well as by port numbers. To identify a flow uniquely, NetFlow also uses several fields, viz., types of protocols and types of services (ToS) from the IP header, and the input logical interface of the router or the switch. The flows are stored in the router or the switch cache and exported to a collector under the following constraints.

- The flows that have been idle for a specified time are expired where the default setting is 15 seconds, or the user can configure this time to be between 10 and 600 seconds.
- The flows lived longer than 30 minutes are expired.
- If the cache reaches its maximum size, a number of heuristic expiry functions are applied to export flows.
- A TCP connection has finished with flag FIN or RST.

A flow collector tool, viz., nfdump (Haag, 2010), receives flow records from the flow exporter and stores them in a form suitable for further monitoring or analysis. A flow record is the information stored in the flow exporter cache. A flow exporter protocol defines how expired flows are transferred by the exporter to the collector. The information exported to the collector is referred to as flow record. NetFlow version 5 (Cisco.com., 2010) is a simple protocol that exports flow records of fixed size (48 bytes in total).

We used the NetFlow version 5 protocol to export flow records and used `nfdump` to obtain flow records. All data are stored to disc before analysing. This separates the process of storing and analysing the data. The data are organised in a time-based fashion. `Nfdump` has a flow record capturing daemon process `nfcapd` which reads data from the network and stores the data into files. Automatically, every *n* minutes, typically 5, `nfcapd` rotates and renames each output file with time stamp `nfcapd.YYYYMMddhhmm`. For example, `nfcapd.201012110845` contains data from 11th December 2010, 08:45, onward. Based on a five-minute time interval, this results in 288 files per day. Analysis of the data is done by concatenation of several files for a single run. The output is stored either in ASCII text or as binary data in a file and it is ready to be processed again with the same tool. We used *C* programs to filter the captured data to extract new features. Unnecessary parameters were removed and the retained parameters were flow-start, duration, protocol, source-IP, source-Port, destination-IP, destination-Port, flags, ToS, bytes, packets-per-second (pps), bits-per-second (bps) and bytes-per-packet (Bps). Network traffic corresponding to attack and normal traffic was gathered using our local network within a four-week period. We collected 148,712 flow records of 16 attack types and normal records. The extracted features were of 23 types and were classified into three groups of features: (a) basic, (b) time-window based and (c) connection-based features. The list of features (Gogoi et al., 2012) is given in Table 6.

Table 6 Features of network flow intrusion data set

Features		*Description*
		Basic features
1	Duration	Length of the flow (in seconds)
2	Protocol-type	Type of protocols e.g. TCP, UDP, ICMP
3	Src IP	Source node IP address
4	Dest IP	Destination IP address
5	Src port	Source port
6	Dest port	Destination port
7	ToS	Type of service
8	URG	Urgent flag of TCP header
9	ACK	Acknowledgement flag
10	PSH	Push flag
11	RST	Reset flag
12	SYN	SYN flag
13	FIN	FIN flag
14	Source byte	Number of data bytes transferred from source IP to destination IP
15	Land	Same source IP/source port are equal to Destination IP/Destination port
		Time-window features
16	count-dest	Number of flows to unique destination IP addresses inside the network in the last T seconds from the same source
17	count-src	Number of flows from unique source IP addresses inside the network in the last T seconds to the same destination
18	count-serv-src	Number of flows from the source IP to the same destination port in the last T seconds.
19	count-serv-dest	Number of flows to the destination IP using same source port in the last T seconds.
		Connection based features
20	count-dest-conn	Number of flows to unique destination IP addresses inside the network in the last N flows from the same source
21	count-src-conn	Number of flows from unique source IP addresses inside the network in the last N flows to the same destination
22	count-serv-src-conn	Number of flows from the source IP to the same destination port in the last N flows.
23	count-serv-dest-conn	Number of flows to the destination IP using same source port in the last N flows.

4.3 Validity measure

The accuracy of each experiment was measured based on the percentage of successful classification (PSC) (Adetunmbi et al., 2008) on the evaluated data set, where

$$PSC = \frac{No.\,of\ Correctly\ Classified\ Instances}{No.\,of\ Instances\ in\ Data\ set} \times 100. \quad (5)$$

4.4 Result analysis

4.4.1 UCI data set

The results of the experiments are reported in Table 7.

Table 7 Experimental results on UCI ML repository (Blake and Merz, 2001) and other data set (Slowinski, 1992)

Data sets	*Training Set Size*	*Test Set Size*	*Rules Generated*	*Accuracy (PSC)*
Demo (Slowinski, 1992)	9	9	7	100%
Breast cancer	500	149	17	97.32%
Congressional Voting	300	135	14	98.52%
Mushroom	450	150	18	92.67%
Glass	144	70	13	85.71%
Iris	150	150	18	93.5%
New-Thyroid	215	215	49	86.5%
Balloon	20	20	7	98%
Hayes-roth	160	160	50	95.5%
Soyabean-Small	625	625	80	91.7%
Balance-scale	625	625	50	94.8%
Liver Disorders	345	345	33	96.8%

We observe from the table that our method performs consistently well for categorical data sets. Since, the method is especially designed for handling inconsistency, it expects the occurrence of at least some inconsistencies in the data set. Another important advantage of the method is its input order independence. We observe in Table 7 that for the UCI Machine Learning Repository data sets such as *mushroom*, *glass identification*, and *breast cancer*, the algorithm is able to generate rules which classify with more than 90% accuracy. Examples of generated rules for different UCI data sets are given in Table 8. An interesting observation is that the number of rules generated is not dependent on the number of instances in the data set. For example, as shown in Table 4, the mushroom data set has the maximum number of instances, 8124. However, the number of rules generated (see Table 7) for this data set is not the maximum. However, with the increase in dimensionality, the number of rules generated also increases, as given in Table 7. This is evident in the case of *soyabean-small* data set given in Table 4. In the *balance-scale* data set, examples of inconsistency present in records are given in Table 9. In the presented records, the values for the four condition attributes, viz., *Class name*, *Left weight*, *Left distance* and *Right weight* are similar whereas the value for the decision attribute *Right distance* is different. The rules are generated by an upper approximation of the concept including the inconsistent records. So, these rules are *possible rules*. The *possible rules* generated by the method for these data are as follows.

1 $(Left\ weight, 1) \rightarrow (Right\ distance, 1)$

2 $(Left\ weight, 1) \rightarrow (Right\ distance, 2)$

3 $(Left\ weight, 1) \rightarrow (Right\ distance, 3)$

4 $(Class\ name, R) \wedge (Left\ weight, 1) \rightarrow (Right\ distance, 5)$

Table 8 Sample rules on UCI ML repository and other data set

Data set		*Rules*
Demo	1	$(Hemoglobin, fair) \wedge (Temperature, low) \rightarrow (Comfort, low)$
	2	$(Blood\_Pressure, high) \rightarrow (Comfort, very\_low)$
Breast cancer	1	$(Fractal\_Dimension, 2) \wedge (Concavity, 2) \rightarrow (Diagonis, 2)$
	2	$(Fractal\_Dimension, 2) \wedge (Summary, 8) \rightarrow (Diagonis, 4)$
Mushroom	1	$(stalkcolor\_above\_ring, c) \wedge (stalk\_root, b) \rightarrow (habitat, m)$
	2	$(geil\_color, f) \wedge (ring\_type, g) \rightarrow (habitat, g)$
Glass	1	$(Aluminium, 1.32) \rightarrow (type\_of\_glass, building\_windows\_float\_processed)$
	2	$(Potassium, 0.00) \wedge (Iron, 0.00) \rightarrow (type\_of\_glass, tableware)$

Table 9 Inconsistent data in balance-scale data set

Records	*Class name*	*Left weight*	*Left distance*	*Right weight*	*Right distance*
6	R	1	1	2	1
7	R	1	1	2	2
8	R	1	1	2	3
10	R	1	1	2	5

4.4.2 Real life network intrusion data set

The results of the network flow intrusion data set are given in Table 10. The detection performance of the method in the network flow intrusion data set is excellent. A total of 29 rules were generated for the all-attacks and normal classes. The percentage of successful classification (PSC) in the network flow intrusion data set, in case of the normal class, is 99.94%; whereas for the all-attacks class it is 96.21%. Examples of generated rules for the network flow intrusion data set are given in Table 11.

A comparison of results of rule generation from similar data sets using different rule generation methods is shown in Table 12. Most existing rule generation methods generate significantly large number of rules in comparison to the proposed method. Our method generates classification rules in the presence of inconsistency in the data sets and with no reduction in the size of data sets. The rule generation for *Breast cancer* data set by Apriori (Agrawal et al., 1993) and Apriori-Rare (Agrawal et al., 1993) algorithms give 125 and 11192 rules, respectively, whereas our method gives 17 rules only. In the *Iris* data set, Apriori-Rare algorithm generates 88 rules and the HCRI algorithm (Sai et al., 2006) generates 13 rules after reduction in the data set, whereas our method generates 18 rules without any reduction in the data set size. The HCRI algorithm and our approach generate seven rules each for the *Balloon* data set. In case of the *Liver disorders* data set, the proposed method performs significantly better than the HCRI algorithm by generating only 33 rules against 221 rules generated by the HCRI algorithm. Another important advantage of the proposed method is that it is capable of operating both with categorical as well as numeric data sets.

Table 10 Results on network flow intrusion data set

Class name	*Records*	*Detection*	*Accuracy (PSC)*
bonk	13,000	12,562	96.63%
jolt	1394	1374	98.57%
nestea	92	92	100%
newtear	137	136	99.27%
syndrop	66	65	98.48%
teardrop	130	130	100%
winnuke	12,000	11,559	96.33%
1234	30,000	28,929	96.43%
oshare	12,000	11,672	92.27%
silhyousen	252	25	99.60%
Smurf	30	30	100%
fraggle	12,000	10,784	89.87%
syn	8000	7598	94.98%
xmas	13,000	12,942	99.55%
window	14,000	13,565	96.89%
land	2	2	100%
All attacks	116,096	111,691	96.21%
normal	32,616	32,598	99.94%

Table 11 Sample rules on network flow intrusion data set

No.	*Rules*
1	$(protocol, ICMP) \wedge (source\ port, 0) \rightarrow (class, 1234)$
2	$(protocol, UDP) \wedge (RST, 0) \rightarrow (class, bonk)$
3	$(sourc\ IP, 172.16.15.12) \wedge (protocol, UDP) \rightarrow (class, fraggle)$
4	$(count\_serv\_dest, 1385) \rightarrow (class, jolt)$
5	$(protocol, UDP) \wedge (URG, 1) \rightarrow (class, nestea)$

Table 12 Comparison of results

Data sets (Blake and Merz, 2001)	*Number of Generated Rules*	
	HCRI algorithm (Sai et al., 2006)	*Our Method*
Iris	13	18
New-Thyroid	27	49
Ballon	7	7
Liver Disorders	221	33

5 Conclusion and future works

The work presented in this paper proposes a classification rule generation method based on the LEM2 algorithm (Grzymala-Busse, 1988). The proposed method is typically employable in data sets that have inconsistencies. The method exhibits satisfactory performance whenever the data set contains inconsistencies, at least for some concepts. We have tested our rule generation method on several real life data sets from the UCI Machine Learning Repository and the results are satisfactory. Thus, experimental results demonstrate the effectiveness of the proposed method. The method works well with our network flow intrusion data set as well.

The method covers only the local covering option. For every concept, it generates a minimum, non-redundant set of classification rules. However, the method does not address the generation of a minimum, non-redundant classification rule set collectively considering the whole data set. That is, it does not perform global covering. Thus, it will be useful to expand the method to consider the idea of global covering as well. It might yield better results if we use a mixed approach with local as well as global coverings.

References

Adetunmbi, A.O., Falaki, S.O., Adewale, O.S. and Alese, B.K. (2008) 'Network intrusion detection based on rough set and k-nearest neighbour', *International Journal of Computing and ICT Research*, Vol. 2, pp.60–66.

Agrawal, R., Imielinski, T. and Swami, A. (1993) 'Mining association rules between sets of items in large databases', *Proceedings of 1993 ACM SIGMOD*, ACM, New York, NY, USA, pp.207–216.

Amini, M., Jalili, R. and Shahriari, H.R. (2006) 'RT-UNNID: a practical solution to real-time network-based intrusion detection using unsupervised neural networks', *Computers & Security*, Vol. 25, No. 6, pp.459–468.

Bhuyan, M.H., Bhattacharyya, D.K. and Kalita, J.K. (2011) 'Surveying port scans and their detection methodologies', *The Computer Journal*, Vol. 54, No. 4, pp.1–17.

Blake, C.L. and Merz, C.J. (2001) *UCI Machine Learning Repository*, University of California, Department of Information and Computer Science, Irvine, CA. Available online at: http://www.ics.uci.edu/~mlearn/MLRepository.html

Cisco.com. (2010) *Cisco IOS netflow con guide*, Release 12.4. Available online at: http://www.cisco.com

Claise, B. (2004) *RFC 3954: Cisco systems netflow services export version 9*. Available online at: http://www.ietf.org/rfc/rfc 3954.txt

Ghosh, A. and Nath, B.T. (2004) 'Multi-objective rule mining using genetic algorithms', *Information Sciences: an International Journal*, Vol. 163, pp.123–133.

Gogoi, P., Bhattacharyya, D.K., Borah, B. and Kalita, J.K. (2011) 'A survey of outlier detection methods in network anomaly identification', *The Computer Journal*, Vol. 54, No. 4, pp.570–588.

Gogoi, P., Bhuyan, M.H., Bhattacharyya, D.K. and Kalita, J.K. (2012) 'Packet and flow based network intrusion dataset', *Proceeding of the 5th International Conference on Contemporary Computing (IC3-2012)*, Springer, India, pp.322–334.

Gogoi, P., Borah, B. and Bhattacharyya, D.K. (2010) 'Anomaly detection analysis of intrusion data using supervised & unsupervised approach', *Journal of Convergence Information Technology*, Vol. 5, No. 1, pp.95–110.

Gogoi, P., Borah, B., Bhattacharyya, D.K. and Kalita, J.K. (2012) 'Outlier identification using symmetric neighborhoods', *Proceedings of 2nd International Conference of Communication Computing & Security (ICCCS 2012)*, Vol. 6, Procedia Technology, Elsevier, India, pp.239–246.

Grosan, C. and Abraham, A. (2006) 'Stock market modeling using genetic programming ensembles', *Genetic Systems Programming: Theory and Experiences*, Vol. 13, pp.133–148.

Grzymala-Busse, J.W. (1988) 'Knowledge acquisition under uncertainty a rough set approach', *Journal of Intelligent & Robotic Systems*, Vol. 1, No. 1, pp.3–16.

Grzymala-Busse, J.W. (1997) 'A new version of the rule induction system LERS', *Fundamenta Informaticae*, Vol. 31, No. 1, pp.27–39.

Haag, P. (2010) *NFDUMP & NFSEN*. Available online at: http://nfdump.sourceforge.net/

Han, J. and Kamber, M. (2001) *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, CA.

Kiran, R.U. and Reddy, P.K. (2010) 'Mining rare association rules in the datasets with widely varying items' frequencies', *Lecture Notes in Computer Science*, Vol. 5981, pp.49–62.

Li, J. and Cercone, N. (2005) 'A rough set based model to rank the importance of association rules', *Lecture Notes in Computer Science*, Vol. 3642, pp.109–118.

Liu, Z. and Li, Y. (2008) 'A new heuristic algorithm of rules generation based on rough sets', *Proceeding of ISBM 2008*, IEEE Computer Society, Washington, DC, USA, pp.291–294.

Mixter, D. (2003) *Attacks Tools and Information*. Available online at: http://packetstormsecurity.nl/index.html

Paetz, J. and Brause, R. (2002) 'Rule generation and model selection used for medical diagnosis', *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology – Challenges for future intelligent systems in biomedicine*, Vol. 12, No. 1, pp.69–78.

Pawlak, Z., Grazymala-Busse, J.W., Slowinski, R. and Ziarko, W. (1995) 'Rough sets', *Communications of the ACM*, Vol. 38, pp.88–95.

Quittek, J., Zseby, T., Claise, B. and Zender, S. (2004) *Requirements for IP flow information export: IPFIX*, Hawthorn Victoria. Available online at: http://www.ietf.org/ rfc/rfc3917.txt

Sai, Y., Nie, P., Xu, R. and Huang, J. (2006) 'A rough set approach to mining concise rules from inconsistent data', *Proceedings of IEEE GRC 2006*, IEEE, Atlanta USA, pp.333–336.

Slowinski, R. (1992) *In Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory*, Kluwer Academic Publishers, Norwell, MA, USA.

Vollmer, T., Foss, J.A. and Manic, M. (2011) 'Autonomous rule creation for intrusion detection', *Proceedings of SCCI 2011*, IEEE, Paris, France, pp.1–8.