An Overview of Rough Set Theory from the Point of View of Relational Databases

Written by Rajat Sar (ID-20)

December 14, 2007

Glossary
I. Definition
II. Abstract
III. Introduction
IV. Information Table
V. Knowledge dependencies
VI. Decision Table and Decision Rules
VII. Partial dependencies
VIII. Conclusion
Glossary

Information Table: An information table is a table consists of set of attributes and its values. Each attribute's value must be define in every tuples.

Knowledge dependencies: Equivalence class in a knowledge base is defined by their dependency. Suppose P, Q are two equivalence class in a knowledge base K. If Q is the union of all P-equivalence class then Q is dependend on P or P is finer than Q and Q is coarser than P.

Indiscernibility Relation: If intersection of all equivalence relation in any subset P of equivalence relation R from the knowledge base K is an equivalence relation then it is called indiscernibility relation. It is represented by \( IND(P) \).

Decision Table: A decision table is an information table with an extra attribute called decision attribute. It gives a decision on the basis of attribute values in a tuple.

Reduct: Reduct in Rough set is like the candidate key in Relational database. But it consists of minimum number of columns which have unique attribute values in each and every tuples in a decision table.

Dispensable Attribute: If removal of any attribute from a decision table gives the same decision it has before is called dispensable attribute and all other attributes in the same decision table is called indispensable attributes.

Core: If removal of attributes' values from a tuple in a decision table leads to an contradictory decision (Same values can not give different decision) on the table is called core value on that tuple. It means the attribute has to be there in the decision table.

Value Reduct: Values of the attributes involved on a decision rule in a decision table defining the core as well as give decisions are called value reducts.

Decision Rules: Decision rules are the set of rules on a tuple which are uniquely define the decision from the combination of attributes on a decision table.

Partial Dependencies: If two equivalence classes C and D are partial dependent with each other if and only if some of the equivalence classes
of C or D are inside the positive region of anyone and some of them are outside the region.

T. Y. Lin*1  tylin@cs.sjsu.edu

*1Berkeley Initiative in Soft Computing,
Department of Electrical Engineering and Computer Science,
University of California,
Berkeley, California 94720

Abstract: Both relational database theory (RDB) and rough set theory (RS) are formal theories derived from the study of tables. However, they were based on different philosophies and went on to different directions. RDB assumes semantics of data is known and focuses on organizing data through its semantics. RS, in this table format, assumes data semantics is defined by the given data and focuses on discovering patterns, rules and data semantics through those available data - a data mining theory. In this paper, fundamental notions of two theories are compared and summarized. Further, RS can also take abstract format. In this format it can be used to analyze imprecise, uncertain or incomplete information in data. It is a new set theory complimentary to fuzzy set theory. In this paper, this aspect is only lightly touched.

Keywords: Rough sets, Relational Databases.

1 Introduction

Rough set theory (RS) is a formal theory derived from fundamental research on logical properties of information tables, also known as (Pawlak) information systems or knowledge representation systems, in Polish Academy of Sciences and the University of Warsaw, Poland around 1970’s. Developed independently and differently from relational databases, RS in the table format is another theory on extensional relational databases (ERDB) - snap shots of relational databases. However unlike usual ERDB focusing on storing and retrieving data, RS focuses on discovering patterns, rules and knowledge in data - a modern data mining theory. Fundamentally RS and ERDB are very different theory, even
though the entities, namely tables, of their respective studies are the same. In this paper we compare and summarize their fundamental notions. Further RS in the abstract format can be used to analyze imprecise, uncertain or incomplete information in data - a new set theory complimentary to fuzzy set theory. In this paper, this aspect is only lightly touched.

2 Information Table

The syntax of information tables in RS is very similar to relations in RDB. Entities in RS are also represented by tuples of attribute values. However, the representation may not be faithful, namely, entities and tuples may not be one to one correspondence.

A relation R consists of

1) $U = \{x, y, \ldots\}$ is a set of entities.
2) $T$ is a set of attributes $\{A_1, A_2, \ldots A_n\}$.
3) $\text{Dom}(A_i)$ is the set of values of attribute $A_i$.
   $\text{Dom} = \text{dom}(A_1) \cup \text{dom}(A_2) \cup \ldots \cup \text{dom}(A_n)$
4) Each entity in $U$ is represented uniquely by a map
   
   $t : T \rightarrow \text{Dom},$

   where $t(A_i) \in \text{dom}(A_i)$ for each $A_i \in T$.

Informally, one can view relation as a table consists of rows of elements. Each row represents an entity uniquely.

An information table (also known as information system, knowledge representation system) consists of:

1) $U = \{u, v, \ldots\}$ is a set of entities.
2) $T$ is a set of attributes $\{A_1, A_2, \ldots A_n\}$. 

4
(3) Dom($A_i$) is the set of values of attribute $A_i$.
Dom = dom($A_1$) $\cup$ dom($A_2$) $\cup$ ... $\cup$ dom($A_n$),

(4)$\rho : U \times T \rightarrow$ Dom, called description function, is a map such that $\rho(u, A_i)$ is in dom($A_i$) for all $u$ in $U$ and $A_i$ in $T$.

Note that $\rho$ induces a set of maps

t = $\rho(u, \bullet) : T \rightarrow$ Dom.

Each map is a tuple:

t = ($\rho(u, A_1), \rho(u, A_2), \ldots, \rho(u, A_i), \ldots, \rho(u, A_n)$)

Note that the tuple $t$ is not necessarily associated with entity uniquely. In an information table, two distinct entities could have the same tuple representation, which is not permissible in relational databases.

A decision table ($DT$) is an information table $(U, T, V, \rho)$ in which the attribute set $T = C \cup D$ is a union of two non-empty sets, $C$ and $D$, of attributes. The elements in $C$ are called conditional attributes. The elements in $D$ are called decision attributes.

3 Knowledge Dependencies

Mathematically, a classifications or partition is a decomposition of the domain $U$ of interest into mutually disjoint subsets, called equivalence classes. Such a partition defines and is defined by a binary relation, called an equivalence relation that is a reflexive, symmetric and transitive binary relation. In this section, we will compare the functional dependencies in RDB with Pawlak theory of equivalence relations derived from the structure of information tables.

3.1 Attributes and Equivalence Relations

In an information table, any subset of attributes induces an equivalence relation as follows: Let $B$ be a non empty subset of $T$. Two entities $u, v$
are equivalent [or indiscernible by $B$] in $U$, denoted by

$$u \equiv v \pmod{B} \text{ if } \rho(u, A_i) = \rho(v, A_i) \text{ for every attribute } A_i \text{ in } B.$$ 

It is not difficult to verify that $\equiv$ is indeed an equivalence relation; $\equiv$ is called indiscernibility relation and denoted by $\text{IND}(B)$. We will denote the equivalence class containing $u$ by $[u]_{\text{IND}(B)}$ or simply by $[u]_B$. Note that $B$ can be a singleton $\{A_i\}$, in this case, we simply denoted by $\text{IND}(A_i)$. It is easy to see that the following is valid:

$$\text{IND}(B) = \cap \{\text{IND}(A_i): A_i \text{ in } B\}.$$ 

As observed earlier the equivalence classes of $\text{IND}(B)$ consists of all possible intersections of equivalence classes of $\text{IND}(A_i)$’s.

A set $B$ of attributes names gives us a finite collection of equivalence relations,

$$\text{RCol}(B) = \{\text{IND}(A_i): A_i \text{ in } B\}.$$ 

In particular, the set of all attributes give us the following set of equivalence relations,

$$\text{RCol}(T) = \{\text{IND}(A_1), \text{IND}(A_2), ..., \text{IND}(A_i), ..., \text{IND}(A_n)\}.$$ 

3.2 Pawlak Knowledge Bases

Let $\text{RCol}$ be a collection of equivalence relations, often a finite collection, over $U$. An ordered pair

$$K = (U, \text{RCol})$$
is called Pawlak knowledge base [Pawlak91]. Pawlak calls a collection of
equivalence relations a knowledge, because our knowledge about a domain
is often represented by its classifications.

Let P and Q be two equivalence relations or classifications in K. If every
Q-equivalence class is a union of P-equivalence classes, then we say that Q
is depended on P, Q is coarse than P, or P is finer than Q. The dependency
is called knowledge dependency (KD) [Pawlak91]. It is easy to verify that
the intersection P \cap Q of two equivalence relations is another equivalence
relation whose partition consists of all possible intersections of P- and Q-
equivalence classes. More generally, the intersection of all the equivalence
relations in RCol, denoted by IND(RCol), is another equivalence relation.

IND(RCol) is referred to as the indiscernibility relation over RCol. Let
PCol and QCol be two sub-collections of RCol. Following Pawlak, we
define the following [Pawlak91]:

(1) QCol depends on PCol iff IND(QCol) is coarse than IND(PCol).

This dependency is denoted by PCol \Rightarrow QCol.

(2) PCol and QCol are equivalent iff PCol \Rightarrow QCol, and QCol \Rightarrow PCol.
It is easy to see that PCol and QCol are equivalent iff
IND(PCol)=IND(QCol).

(3) PCol and QCol are independent iff neither
PCol \Rightarrow QCol, nor QCol \Rightarrow PCol

Now we can treat an information table as a Pawlak knowledge base
(U, RCol(T)) and apply the notion of knowledge dependencies to
information tables. Though Pawlak knowledge bases appear to be an
abstract notion, it is in fact a very concrete object.

Proposition. There is an one-to-one correspondence between information
tables and Pawlak knowledge bases.
3.3 Knowledge and Functional Dependencies

In relational databases, a finite collection of attribute names \{A_1, A_2, \ldots, A_n\} is called a relation scheme [Ullmann89]. A relation instance, or simply relation, \( R \) on relation scheme \( R \) is a finite set of tuples, \( t=(t_1, t_2, \ldots, t_n) \) as defined in Section 2. A functional dependency occurs when the values of a tuple on the set of attributes uniquely determine the values of another set of attributes. Formally, let \( X \) and \( Y \) be two subsets of \( T \). A relation \( R \) satisfies an extensional functional dependency EFD: \( X \rightarrow Y \) if for very X-value there is a uniquely determined Y-values in the relation instance \( R \). An intensional function dependency FD: \( X \rightarrow Y \) exists on relation scheme \( R \), if FD is satisfied by all relation instances \( R \) of the scheme \( R \). In database community, FD always refers to intensional FD. One should note that at any given moment, a relation instance may satisfy some family of extensional functional dependencies EFDs, however, the same family may not be satisfied by other relation instances. The family that is satisfied by all the relation instances is the intensional functional dependency FD. In this paper, we will be interested in the extensional functional dependency, so the notation ”\( X \rightarrow Y \)” is an EFD.

As pointed out earlier that attributes induce equivalence relations on information tables. So an EFD can be interpreted as a knowledge dependency (KD). The following proposition is immediate from the definitions.

Proposition. An EFD, \( X \rightarrow Y \), between two sets of attributes \( X \) and \( Y \) is equivalent to a KD, \( RCol(X) \Rightarrow RCol(Y) \), of the equivalence relations induced by the attributes \( X \) and \( Y \).

4 Decision Table and Decision Rules

Rough set theory is an effective methodology to extract rules from information tables, more precisely, from decision tables [Pawlak91]. In this section we will introduce some fundamental concepts and illustrate the procedure by an example.
4.1 Reducts and Candidate Keys

In RDB, an attribute, or a set of attributes \( K \) is called candidate key if all attributes is functionally depended on \( K \), and \( K \) is such a minimal set. We export these notions to extensional world. The "extensional candidate key" is a special form of reduct \([Pawlak91]\).

Let \( S = (U, T = C \cup D, V_\rho) \) be a decision table, where

\[
C = \{ A_1, A_2, ..., A_i, ..., A_n \}, \quad D = \{ B_1, B_2, ..., B_i, ..., B_m \}.
\]

Then there are two Pawlak knowledge bases on \( U \):

\[
RCol(C) = \{ \text{IND}(A_1), \text{IND}(A_2), ..., \text{IND}(A_i), ..., \text{IND}(A_n) \}
\]

\[
RCol(D) = \{ \text{IND}(B_1), \text{IND}(B_2), ..., \text{IND}(B_i), ..., \text{IND}(B_m) \}
\]

\( S \) is a consistent decision table, if \( RCol(C) \Rightarrow RCol(D) \). \( B \) is called a reduct of \( S \), if \( B \) is a minimal subset of \( C \) such that \( RCol(B) \Rightarrow RCol(D) \). It is clear such a \( B \) is not necessarily unique. If we choose \( D=T \), then the reduct is the extensional candidate key.

4.2 Decision Rules and Value Reducts

Rough set theory is an effective methodology to extract rules from information tables (\( IT \)), more precisely, from decision tables (\( DT \)); each \( IT \) can be viewed as many \( DT \)'s. Each entity \( u \) in a \( DT \) can be interpreted as a decision rule. Let \( X \rightarrow Y \) be an EFD (or equivalently a KD), that is, for any \( X \)-value \( c \), there is a unique \( Y \)-value \( d \). We can rephrase it as a decision rule:

If \( t(X)=c \), then \( t(Y)=d \).
Rough set theorists are interested in simplified these decision rules [Pawlak91].

4.3 Illustration

We will illustrate, without losing the general idea from the following table:

<table>
<thead>
<tr>
<th>U</th>
<th>Location</th>
<th>TEST</th>
<th>NEW</th>
<th>CASE</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID-1</td>
<td>Houston</td>
<td>10</td>
<td>92</td>
<td>03</td>
<td>10</td>
</tr>
<tr>
<td>ID-2</td>
<td>San Jose</td>
<td>10</td>
<td>92</td>
<td>03</td>
<td>10</td>
</tr>
<tr>
<td>ID-3</td>
<td>Palo Alto</td>
<td>10</td>
<td>90</td>
<td>02</td>
<td>10</td>
</tr>
<tr>
<td>ID-4</td>
<td>Berkeley</td>
<td>11</td>
<td>91</td>
<td>04</td>
<td>50</td>
</tr>
<tr>
<td>ID-5</td>
<td>NewYork</td>
<td>11</td>
<td>91</td>
<td>04</td>
<td>50</td>
</tr>
<tr>
<td>ID-6</td>
<td>Atlanta</td>
<td>20</td>
<td>93</td>
<td>70</td>
<td>99</td>
</tr>
<tr>
<td>ID-7</td>
<td>Chicago</td>
<td>20</td>
<td>93</td>
<td>70</td>
<td>99</td>
</tr>
<tr>
<td>ID-8</td>
<td>Baltimore</td>
<td>20</td>
<td>93</td>
<td>70</td>
<td>99</td>
</tr>
<tr>
<td>ID-9</td>
<td>Seattle</td>
<td>20</td>
<td>93</td>
<td>70</td>
<td>99</td>
</tr>
<tr>
<td>ID-10</td>
<td>Chicago</td>
<td>51</td>
<td>95</td>
<td>70</td>
<td>94</td>
</tr>
<tr>
<td>ID-11</td>
<td>Chicago</td>
<td>51</td>
<td>95</td>
<td>70</td>
<td>95</td>
</tr>
</tbody>
</table>

Table 1: Original table given on the example

Two rows below the divided line will be cut off.

(4.1) Select a DT: We will consider a decision table from the table given above: U is the universe, RESULT is the decision attribute, and C={TEST, NEW, CASE} is the set of conditional attributes.

(4.2) Split DT: ID-10 and ID-11 form two inconsistent rules. So we split that tables. One consists of entities, ID-1 to ID-9, called consistent table, and another one is ID-10 and ID-11 is called totally inconsistent table. From now on the term in this section ”decision table” is referred to the consistent table.

(4.3) Decision Classes: Using notation of Section 3.1, the equivalence
relation \( \text{IND}(RESULT) \) classifies entities into three equivalence classes, called decision classes

\[
\begin{align*}
\text{DECISION1} = \{\text{ID-1, ID-2, ID-3}\} = [10] \text{RESULT}, \\
\text{DECISION2} = \{\text{ID-4, ID-5}\} = [50] \text{RESULT}, \\
\text{DECISION3} = \{\text{ID-6, ID-7, ID-8, ID-9}\} = [99] \text{RESULT}
\end{align*}
\]

(4.4) Condition Classes: Let \( C = \{\text{TEST, NEW, CASE}\} \) be the conditional attributes. The equivalence relation \( \text{IND}(C) \) classifies entities into four equivalence classes, called condition classes

\[
\begin{align*}
\text{CASE1} = \{\text{ID-1, ID-2}\}, \\
\text{CASE2} = \{\text{ID-3}\}, \\
\text{CASE3} = \{\text{ID-4, ID-5}\}, \\
\text{CASE4} = \{\text{ID-6, ID-7, ID-8, ID-9}\}
\end{align*}
\]

(4.5) Knowledge dependencies: It is not difficult to verify that entities are indiscernible by conditional attributes are also indiscernible by decision attributes, namely we have following inclusions

\[
\begin{align*}
\text{CASE1} \subseteq \text{DECISION1}; \\
\text{CASE2} \subseteq \text{DECISION1}; \\
\text{CASE3} \subseteq \text{DECISION2}; \\
\text{CASE4} \subseteq \text{DECISION3}.
\end{align*}
\]

These inclusions implies that the equivalence relation \( \text{IND}(RESULT) \) depends on \( \text{IND}(C) \). Or equivalently, \( \text{RESULT} \) are KD on \( C \).

(4.6) Inference Rules: The relationship induces inference rules:

1. If \( \text{TEST} = 10 \), \( \text{NEW} = 92 \), \( \text{CASE} = 03 \), then \( \text{RESULT} = 10 \),
2. If \( \text{TEST} = 10 \), \( \text{NEW} = 90 \), \( \text{CASE} = 02 \), then \( \text{RESULT} = 10 \),
3. If \( \text{TEST} = 11 \), \( \text{NEW} = 91 \), \( \text{CASE} = 04 \), then \( \text{RESULT} = 50 \),
4. If \( \text{TEST} = 20 \), \( \text{NEW} = 93 \), \( \text{CASE} = 70 \), then \( \text{RESULT} = 99 \).
One can rewrite these four rules into one universal inference rule “
\((TEST, NEW, CASE)\) implies RESULT.”

(4.7) Reducts: It is clear that \{TEST, NEW\} and \{CASE\} are two reducts. So the conditions on CASE can be deleted from the rules given above.

(4.8) Value Reducts: Note that the rule 1 is derived from the inclusion \(CASE1 \subseteq DECISION1\). We can sharpen the description of \(CASE1\) by the condition \(TEST=10\) alone, called value reducts, i.e.,

\[ CASE1=\{ u: u.TEST=10 \}. \]

By similar arguments, we can simplify the four rules to:

1. If \(TEST=10\), then \(RESULT=10\),
2. If \(TEST=11\), then \(RESULT=50\),
3. If \(TEST=20\), then \(RESULT=99\).

We will have another set of rules, if the other reduct is used.

5 Partial dependencies

The table given above is assuming that data have no noise. In practices, noise are unavoidable. So we will examine next the partial dependency. As explained, \(D\) and \(C\) may induce two distinct partitions of the universe. Consider the problem of approximating equivalence classes of \(IND\) using the equivalence classes of \(IND\). For an equivalent class \([X]_D\), the lower approximation is given by:

\[ C_L(X) = u \mid [u]_C \subseteq C \]

\[ = \bigcup \{ [u]_C \mid [u]_C \subseteq C \} \]

The upper approximation of \(X\) is defined as,
\[ C_H(X) = \{ u \mid [u]C \cap C \neq \emptyset \} \]
\[ = \bigcup \{ [u]C \mid [u]C \cap C \neq \emptyset \} \]

With the lower approximations of every equivalence class of D, the C-positive region of D, denoted by \( POS_C(D) \), is defined by:

\[ POS_C(D) = \bigcup \{ C_L(X) : X \text{ are all decision classes} \} \]

The number
\[ k = | POS_C(D) | / | U | \]

is called degree of dependency of D on C, where \( | \bullet | \) denotes the cardinality of a set. If degree =1, then we have the knowledge dependency. By applying k-dependency, one can convert the previous example into an example of approximate rules or soft rules [Lin93], [Ziarko93], [Lin96c]. In RDB, there are no such a concept of partial dependencies.

6 Conclusion

In this paper, we use relational databases to explain some aspects of rough set theory. Rough set theory, in its table form, has been an effective method in rule mining for small to median size data. The theory and methodology has been extended to very large databases [Lin96b]. Some applications to intelligent control design have been examined[Lin96a, b], [Mrozek96], [Munakata96]. RS, in its abstract format, is a new set theory complimentary to fuzzy theory [Zimmermann91]. Its approximation aspect is closely related to modal logic and (pre-)topological spaces (neighborhood systems) [Chellas80], [Lin88]. Rough set theory provides a new point of views to the theory of relation databases. We believe some profound applications in databases through this view may soon be flourished.

References


[Lin93] Lin, T. Y., ”Coping with Imprecision information—“Fuzzy” Logic,” Downsizing Expo, Santa Clara Convention Center, Aug.3-5, 1993


[Ulmann89] Ulman, J., Principles of Database and Knowledge-Base


Tsau Young (T. Y.) Lin received his Ph.D from Yale University, and now is a Professor at San Jose State University and Visiting Scholar at BISC, University of California-Berkeley. He has been chairs and members of program committees in various conferences and workshops, associate editors and members of editorial boards of several international journals. He is the founding president of International Rough Set Society. His interests include approximate retrieval and reasoning in database and knowledge-base systems, data mining, data security, fuzzy sets, intelligent control, neighborhood systems (geometric rough sets), and Petri nets as automata (alphabetical order).