Rough set

From Wikipedia, the free encyclopedia

In computer science, a **rough set**, first described by a Polish computer scientist Zdzisław I. Pawlak, is a formal approximation of a crisp set (i.e., conventional set) in terms of a pair of sets which give the *lower* and the *upper* approximation of the original set. In the standard version of rough set theory (Pawlak 1991), the lower- and upper-approximation sets are crisp sets, but in other variations, the approximating sets may be fuzzy sets.

Contents

Definitions

This section contains an explanation of the basic framework of rough set theory (proposed originally by Zdzisław I. Pawlak), along with some of the key definitions.

Information system framework

Let $I = (\mathbb{U}, \mathbb{A})$ be an information system (attribute-value system), where $\mathbb{U}$ is a non-empty set of finite objects (the universe) and $\mathbb{A}$ is a non-empty, finite set of attributes such that $a : \mathbb{U} \rightarrow V_a$ for every $a \in \mathbb{A}$. V_a is the set of values that attribute a may take. The information table assigns a value $a(x)$ from V_a to each attribute a and object x in the universe $\mathbb{U}$.

With any $P \subseteq \mathbb{A}$ there is an associated equivalence relation $\mathrm{IND}(P)$:

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

The relation $\mathrm{IND}(P)$ is called a P*-indiscernibility relation*. The partition of $\mathbb{U}$ is a family of all equivalence classes of $\mathrm{IND}(P)$ and is denoted by $\mathbb{U}/\mathrm{IND}(P)$ (or $\mathbb{U}/P$).

If $(x,y) \in \mathrm{IND}(P)$, then x and y are *indiscernible* (or indistinguishable) by attributes from P .

Example: equivalence-class structure

For example, consider the following information table:

Sample Information System

Object	P_1	P_2	P_3	P_4	P_5
O_1	1	2	0	1	1
O_2	1	2	0	1	1
O_3	2	0	0	1	0
O_4	0	0	1	2	1
O_5	2	1	0	2	1
O_6	0	0	1	2	2
O_7	2	0	0	1	0
O_8	0	1	2	2	1
O_9	2	1	0	2	2
O_{10}	2	0	0	1	0

When the full set of attributes $P = \{P_1,P_2,P_3,P_4,P_5\}$ is considered, we see that we have the following seven equivalence classes:

$$\begin{cases} \{O_1, O_2\} \\ \{O_3, O_7, O_{10}\} \\ \{O_4\} \\ \{O_5\} \\ \{O_6\} \\ \{O_8\} \\ \{O_9\} \end{cases}$$

Thus, the two objects within the first equivalence class, $\{O_1,O_2\}$, cannot be distinguished from one another based on the available attributes, and the three objects within the second equivalence class, $\{O_3,O_7,O_{10}\}$, cannot be distinguished from one another based on the available attributes. The remaining five objects are each discernible from all other objects. The equivalence classes of the P-indiscernibility relation are denoted $[x]_P$.

It is apparent that different attribute subset selections will in general lead to different indiscernibility classes. For example, if attribute $P = \{P_1\}$ alone is selected, we obtain the following, much coarser, equivalence-class structure:

$$\begin{cases} \{O_1, O_2\} \\ \{O_3, O_5, O_7, O_9, O_{10}\} \\ \{O_4, O_6, O_8\} \end{cases}$$

Definition of a *rough set*

Let $X \subseteq \mathbb{U}$ be a target set that we wish to represent using attribute subset P; that is, we are told that an arbitrary set of objects X comprises a single class, and we wish to express this class (i.e., this subset) using the equivalence classes induced by attribute subset P. In general, X cannot be expressed exactly, because the set may include and exclude objects which are indistinguishable on the basis of attributes P.

For example, consider the target set $X = \{O_1,O_2,O_3,O_4\}$, and let attribute subset $P = \{P_1,P_2,P_3,P_4,P_5\}$, the full available set of features. It will be noted that the set X cannot be expressed exactly, because in $[x]_P$,, objects $\{O_3,O_7,O_{10}\}$ are indiscernible. Thus, there is no way to represent any set X which *includes* O_3 but *excludes* objects O_7 and O_{10}.

However, the target set X can be *approximated* using only the information contained within P by constructing the P-lower and P-upper approximations of X:

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

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

Lower approximation and positive region

The P-*lower approximation*, or *positive region*, is the union of all equivalence classes in $[x]_P$ which are contained by (i.e., are subsets of) the target set – in the example, $\underline{P}X = \{O_1, O_2\} \cup \{O_4\}$, the union of the two equivalence classes in $[x]_P$ which are contained in the target set. The lower approximation is the complete set of objects in $\mathbb{U}/P$ that can be *positively* (i.e., unambiguously) classified as belonging to target set X.

Upper approximation and negative region

The P-*upper approximation* is the union of all equivalence classes in $[x]_P$ which have non-empty intersection with the target set – in the example, $\overline{P}X = \{O_1, O_2\} \cup \{O_4\} \cup \{O_3, O_7, O_{10}\}$, the union of the three equivalence classes in $[x]_P$ that have non-empty intersection with the target set. The upper approximation is the complete set of objects that in $\mathbb{U}/P$ that *cannot* be positively (i.e., unambiguously) classified as belonging to the *complement* ($\overline{X}$) of the target set X. In other words, the upper approximation is the complete set of objects that are *possibly* members of the target set X.

The set $\mathbb{U} - \overline{P}X$ therefore represents the *negative region*, containing the set of objects that can be definitely ruled out as members of the target set.

Boundary region

The *boundary region*, given by set difference $\overline{P}X - \underline{P}X$, consists of those objects that can neither be ruled in nor ruled out as members of the target set X.

In summary, the lower approximation of a target set is a *conservative* approximation consisting of only those objects which can positively be identified as members of the set. (These objects have no indiscernible "clones" which are excluded by the target set.) The upper approximation is a *liberal* approximation which includes all objects that might be members of target set. (Some objects in the upper approximation may not be members of the target set.) From the perspective of $\mathbb{U}/P$, the lower approximation contains objects that are members of the target set with certainty (probability = 1), while the upper approximation contains objects that are members of the target set with non-zero probability (probability > 0).

The rough set

The tuple $\langle \underline{P}X, \overline{P}X \rangle$ composed of the lower and upper approximation is called a *rough set*; thus, a rough set is composed of two crisp sets, one representing a *lower boundary* of the target set X, and the other representing an *upper boundary* of the target set X.

The *accuracy* of the rough-set representation of the set X can be given (Pawlak 1991) by the following:

$$\alpha_P(X) = \frac{\left|\underline{P}X\right|}{\left|\overline{P}X\right|}$$

That is, the accuracy of the rough set representation of X, $\alpha_P(X)$, $0 \leq \alpha_P(X) \leq 1$, is the ratio of the number of objects which can *positively* be placed in X to the number of objects that can *possibly* be placed in X – this provides a measure of how closely the rough set is approximating the target set. Clearly, when the upper and lower approximations are equal (i.e., boundary region empty), then $\alpha_P(X) = 1$, and the approximation is perfect; at the other extreme, whenever the lower approximation is empty, the accuracy is zero (regardless of the size of the upper approximation).

Formal properties of rough sets

The important formal properties of rough sets and boundaries are given in Pawlak (1991), and the other sources cited by that book.

Definability

In general, the upper and lower approximations are not equal; in such cases, we say that target set X is *undefinable* or *roughly definable* on attribute set P. When the upper and lower approximations are equal (i.e., the boundary is empty), $\overline{P}X = \underline{P}X$, then the target set X is *definable* on attribute set P. We can distinguish the following special cases of undefinability:

- Set X is *internally undefinable* if $\underline{P}X \neq \emptyset$ and $\overline{P}X = \mathbb{U}$. This means that on attribute set P, there *are* objects which we can be certain belong to target set X, but there are *no* objects which we can definitively exclude from set X.

- Set X is *externally undefinable* if $\underline{P}X = \emptyset$ and $\overline{P}X \neq \mathbb{U}$. This means that on attribute set P, there are *no* objects which we can be certain belong to target set X, but there *are* objects which we can definitively exclude from set X.
- Set X is *totally undefinable* if $\underline{P}X = \emptyset$ and $\overline{P}X = \mathbb{U}$. This means that on attribute set P, there are *no* objects which we can be certain belong to target set X, and there are *no* objects which we can definitively exclude from set X. Thus, on attribute set P, we cannot decide whether any object is, or is not, a member of X.

Reduct and core

An interesting question is whether there are attributes in the information system (attribute-value table) which are more important to the knowledge represented in the equivalence class structure than other attributes. Often, we wonder whether there is a subset of attributes which can, by itself, fully characterize the knowledge in the database; such an attribute set is called a *reduct*.

Formally, a reduct is a subset of attributes $\mathrm{RED} \subseteq P$ such that

- $[x]_{\mathrm{RED}} = [x]_P$, that is, the equivalence classes induced by the reduced attribute set RED are the same as the equivalence class structure induced by the full attribute set P.
- the attribute set RED is *minimal*, in the sense that $[x]_{(\mathrm{RED}-\{a\})} \neq [x]_P$ for any attribute $a \in \mathrm{RED}$; in other words, no attribute can be removed from set RED without changing the equivalence classes $[x]_P$.

A reduct can be thought of as a *sufficient* set of features – sufficient, that is, to represent the category structure. In the example table above, attribute set $\{P_3,P_4,P_5\}$ is a reduct – the information system projected on just these attributes possesses the same equivalence class structure as that expressed by the full attribute set:

$$\begin{cases} \{O_1, O_2\} \\ \{O_3, O_7, O_{10}\} \\ \{O_4\} \\ \{O_5\} \\ \{O_6\} \\ \{O_8\} \\ \{O_9\} \end{cases}$$

Attribute set $\{P_3,P_4,P_5\}$ is a legitimate reduct because eliminating any of these attributes causes a collapse of the equivalence-class structure, with the result that $[x]_{\mathrm{RED}} \neq [x]_P$.

The reduct of an information system is *not unique*: there may be many subsets of attributes which preserve the equivalence-class structure (i.e., the knowledge) expressed in the information system. In the example information system above, another reduct is $\{P_1,P_2,P_5\}$, producing the same equivalence-class structure as

$[x]_P$.

The set of attributes which is common to all reducts is called the *core*: the core is the set of attributes which is possessed by *every* legitimate reduct, and therefore consists of attributes which cannot be removed from the information system without causing collapse of the equivalence-class structure. The core may be thought of as the set of *necessary* attributes – necessary, that is, for the category structure to be represented. In the example, the only such attribute is $\{P_5\}$; any one of the other attributes can be removed singly without damaging the equivalence-class structure, and hence these are all *dispensable*. However, removing $\{P_5\}$ by itself *does* change the equivalence-class structure, and thus $\{P_5\}$ is the *indispensable* attribute of this information system, and hence the core.

It is possible for the core to be empty, which means that there is no indispensable attribute: any single attribute in such an information system can be deleted without altering the equivalence-class structure. In such cases, there is no *essential* or necessary attribute which is required for the class structure to be represented.

Attribute dependency

One of the most important aspects of database analysis or data acquisition is the discovery of attribute dependencies; that is, we wish to discover which variables are strongly related to which other variables. Generally, it is these strong relationships that will warrant further investigation, and that will ultimately be of use in predictive modeling.

In rough set theory, the notion of dependency is defined very simply. Let us take two (disjoint) sets of attributes, set P and set Q, and inquire what degree of dependency obtains between them. Each attribute set induces an (indiscernibility) equivalence class structure, the equivalence classes induced by P given by $[x]_P$, and the equivalence classes induced by Q given by $[x]_Q$.

Let $[x]_Q = \{Q_1, Q_2, Q_3, \dots, Q_N\}$, where Q_i is a given equivalence class from the equivalence-class structure induced by attribute set Q. Then, the *dependency* of attribute set Q on attribute set P, $\gamma_P(Q)$, is given by

$$\gamma_P(Q) = \frac{\sum_{i=1}^{N} |\underline{P}Q_i|}{|\mathbb{U}|} \leq 1$$

That is, for each equivalence class Q_i in $[x]_Q$, we add up the size of its lower approximation by the attributes in P, i.e., $\underline{P}Q_i$. This approximation (as above, for arbitrary set X) is the number of objects which on attribute set P can be positively identified as belonging to target set Q_i. Added across all equivalence classes in $[x]_Q$, the numerator above represents the total number of objects which – based on attribute set P – can be positively categorized according to the classification induced by attributes Q. The dependency ratio therefore expresses the proportion (within the entire universe) of such classifiable objects. The dependency $\gamma_P(Q)$ "can be interpreted as a proportion of such objects in the information system for which it suffices to know the values of attributes in P to determine the values of attributes in Q".

Another, intuitive, way to consider dependency is to take the partition induced by Q as the target class C, and consider P as the attribute set we wish to use in order to "re-construct" the target class C. If P can completely reconstruct C, then Q depends totally upon P; if P results in a poor and perhaps a random reconstruction of C, then Q does not depend upon P at all.

Thus, this measure of dependency expresses the degree of *functional* (i.e., deterministic) dependency of attribute set Q on attribute set P; it is *not* symmetric. The relationship of this notion of attribute dependency to more traditional information-theoretic (i.e., entropic) notions of attribute dependence has been discussed in a number of sources (e.g., Pawlak, Wong, & Ziarko 1988; Yao & Yao 2002; Wong, Ziarko, & Ye 1986).

Rule extraction

The category representations discussed above are all *extensional* in nature; that is, a category or complex class is simply the sum of all its members. To represent a category is, then, just to be able to list or identify all the objects belonging to that category. However, extensional category representations have very limited practical use, because they provide no insight for deciding whether novel (never-before-seen) objects are members of the category.

What is generally desired is an *intentional* description of the category, a representation of the category based on a set of *rules* that describe the scope of the category. The choice of such rules is not unique, and therein lies the issue of inductive bias. See Version space and Model selection for more about this issue.

There is a few rule-extraction methods. We will start from a rule-extraction procedure based on Ziarko & Shan (1995).

Decision matrices

Let us say that we wish to find the minimal set of consistent rules (logical implications) that characterize our sample system. For a set of *condition* attributes $\mathcal{P} = \{P_1, P_2, P_3, \dots, P_n\}$ and a decision attribute $Q, Q \notin \mathcal{P}$, these rules should have the form $P_i^a P_j^b \dots P_k^c \to Q^d$, or, spelled out,

$$(P_i = a) \land (P_j = b) \land \dots \land (P_k = c) \to (Q = d)$$

where $\{a, b, c, \dots\}$ are legitimate values from the domains of their respective attributes. This is a form typical of association rules, and the number of items in $\mathbb{U}$ which match the condition/antecedent is called the *support* for the rule. The method for extracting such rules given in Ziarko & Shan (1995) is to form a *decision matrix* corresponding to each individual value d of decision attribute Q. Informally, the decision matrix for value d of decision attribute Q lists all attribute–value pairs that *differ* between objects having $Q = d$ and $Q \neq d$.

This is best explained by example (which also avoids a lot of notation). Consider the table above, and let P_4 be the decision variable (i.e., the variable on the right side of the implications) and let $\{P_1,P_2,P_3\}$ be the condition variables (on the left side of the implication). We note that the decision variable P_4 takes on two different values, namely $\{1,2\}$. We treat each case separately.

First, we look at the case $P_4 = 1$, and we divide up $\mathbb{U}$ into objects that have $P_4 = 1$ and those that have $P_4 \neq 1$. (Note that objects with $P_4 \neq 1$ in this case are simply the objects that have $P_4 = 2$, but in general, $P_4 \neq 1$ would include all objects having any value for P_4 *other than* $P_4 = 1$, and there may be several such classes of objects (for example, those having $P_4 = 2,3,4,etc.$).) In this case, the objects having $P_4 = 1$ are $\{O_1,O_2,O_3,O_7,O_{10}\}$ while the objects which have $P_4 \neq 1$ are $\{O_4,O_5,O_6,O_8,O_9\}$. The decision matrix for $P_4 = 1$ lists all the differences between the objects having $P_4 = 1$ and those having $P_4 \neq 1$; that is, the decision matrix lists all the differences between $\{O_1, O_2, O_3, O_7, O_{10}\}$ and $\{O_4, O_5, O_6, O_8, O_9\}$. We put the "positive" objects ($P_4 = 1$) as the rows, and the "negative" objects $P_4 \neq 1$ as the columns.

Decision matrix for $P_4 = 1$

Object	O_4	O_5	O_6	O_8	O_9
O_1	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2
O_2	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2
O_3	P_1^2, P_3^0	P_2^0	P_1^2, P_3^0	P_1^2, P_2^0, P_3^0	P_2^0
O_7	P_1^2, P_3^0	P_2^0	P_1^2, P_3^0	P_1^2, P_2^0, P_3^0	P_2^0
O_{10}	P_1^2, P_3^0	P_2^0	P_1^2, P_3^0	P_1^2, P_2^0, P_3^0	P_2^0

To read this decision matrix, look, for example, at the intersection of row O_3 and column O_6, showing P_1^2, P_3^0 in the cell. This means that *with regard to* decision value $P_4 = 1$, object O_3 differs from object O_6 on attributes P_1 and P_3, and the particular values on these attributes for the positive object O_3 are $P_1 = 2$ and $P_3 = 0$. This tells us that the correct classification of O_3 as belonging to decision class $P_4 = 1$ rests on attributes P_1 and P_3; although one or the other might be dispensable, we know that *at least one* of these attributes is *in*dispensable.

Next, from each decision matrix we form a set of Boolean expressions, one expression for each row of the matrix. The items within each cell are aggregated disjunctively, and the individuals cells are then aggregated conjunctively. Thus, for the above table we have the following five Boolean expressions:

$$\begin{cases}
(P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \\
(P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \\
(P_1^2 \vee P_3^0) \wedge (P_2^0) \wedge (P_1^2 \vee P_3^0) \wedge (P_1^2 \vee P_2^0 \vee P_3^0) \wedge (P_2^0) \\
(P_1^2 \vee P_3^0) \wedge (P_2^0) \wedge (P_1^2 \vee P_3^0) \wedge (P_1^2 \vee P_2^0 \vee P_3^0) \wedge (P_2^0) \\
(P_1^2 \vee P_3^0) \wedge (P_2^0) \wedge (P_1^2 \vee P_3^0) \wedge (P_1^2 \vee P_2^0 \vee P_3^0) \wedge (P_2^0)
\end{cases}$$

Each statement here is essentially a highly specific (probably *too* specific) rule governing the membership in class $P_4 = 1$ of the corresponding object. For example, the last statement, corresponding to object O_{10}, states that all the following must be satisfied:

1. Either P_1 must have value 2, or P_3 must have value 0, or both.
2. P_2 must have value 0.
3. Either P_1 must have value 2, or P_3 must have value 0, or both.
4. Either P_1 must have value 2, or P_2 must have value 0, or P_3 must have value 0, or any combination thereof.
5. P_2 must have value 0.

It is clear that there is a large amount of redundancy here, and the next step is to simplify using traditional Boolean algebra. The statement $(P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2)$ corresponding to objects $\{O_1, O_2\}$ simplifies to $P_1^1 \vee P_2^2$, which yields the implication

$$(P_1 = 1) \vee (P_2 = 2) \to (P_4 = 1)$$

Likewise, the statement $(P_1^2 \vee P_3^0) \wedge (P_2^0) \wedge (P_1^2 \vee P_3^0) \wedge (P_1^2 \vee P_2^0 \vee P_3^0) \wedge (P_2^0)$ corresponding to objects $\{O_3, O_7, O_{10}\}$ simplifies to $P_1^2 P_2^0 \vee P_3^0 P_2^0$. This gives us the implication

$$(P_1 = 2 \wedge P_2 = 0) \vee (P_3 = 0 \wedge P_2 = 0) \to (P_4 = 1)$$

The above implications can also be written as the following rule set:

$$\begin{cases} (P_1 = 1) \to (P_4 = 1) \\ (P_2 = 2) \to (P_4 = 1) \\ (P_1 = 2) \wedge (P_2 = 0) \to (P_4 = 1) \\ (P_3 = 0) \wedge (P_2 = 0) \to (P_4 = 1) \end{cases}$$

It can be noted that each of the first two rules has a *support* of 2 (i.e., the antecedent matches two objects), while each of the last two rules has a support of 3. To finish writing the rule set for this knowledge system, the same procedure as above (starting with writing a new decision matrix) should be followed for the case of $P_4 = 2$, thus yielding a new set of implications for that decision value (i.e., a set of implications with $P_4 = 2$ as the consequent). In general, the procedure will be repeated for each possible value of the decision variable.

LERS rule induction system

The data system LERS (Learning from Examples based on Rough Sets) Grzymala-Busse (1997) may induce rules from inconsistent data, i.e., data with conflicting objects. Two objects are conflicting when they are characterized by the same values of all attributes, but they belong to different concepts (classes). LERS uses rough set theory to compute lower and upper approximations for concepts involved in conflicts with other concepts.

Rules induced from the lower approximation of the concept *certainly* describe the concept, hence such rules are called *certain*. On the other hand, rules induced from the upper approximation of the concept describe the concept *possibly*, so these rules are called *possible*. For rule induction LERS uses three algorithms: LEM1, LEM2, and IRIM.

The LEM2 algorithm of LERS is frequently used for rule induction and is used not only in LERS but also in other systems, e.g., in RSES (Bazan et al. (2004). LEM2 explores the search space of attribute-value pairs. Its input data set is a lower or upper approximation of a concept, so its input data set is always consistent. In general, LEM2 computes a local covering and then converts it into a rule set. We will quote a few definitions to describe the LEM2 algorithm.

The LEM2 algorithm is based on an idea of an attribute-value pair block. Let X be a nonempty lower or upper approximation of a concept represented by a decision-value pair (d,w). Set X *depends* on a set T of attribute-value pairs $t = (a,v)$ if and only if

$$\emptyset \neq [T] = \bigcap_{t \in T} [t] \subseteq X.$$

Set T is a *minimal complex* of X if and only if X depends on T and no proper subset S of T exists such that X depends on S. Let $\mathbb{T}$ be a nonempty collection of nonempty sets of attribute-value pairs. Then $\mathbb{T}$ is a *local covering* of X if and only if the following three conditions are satisfied:

each member T of $\mathbb{T}$ is a minimal complex of X,

$$\bigcup_{t \in \mathbb{T}} [T] = X,$$

$\mathbb{T}$ is minimal, i.e., $\mathbb{T}$ has the smallest possible number of members.

For our sample information system, LEM2 will induce the following rules:

$$\begin{cases} (P_1, 1) \to (P_4, 1) \\ (P_5, 0) \to (P_4, 1) \\ (P_1, 0) \to (P_4, 2) \\ (P_2, 1) \to (P_4, 2) \end{cases}$$

Other rule-learning methods can be found, e.g., in Pawlak (1991), Stefanowski (1998), Bazan et al. (2004), etc.

Incomplete data

Rough set theory is useful for rule induction from incomplete data sets. Using this approach we can distinguish between three types of missing attribute values: *lost values* (the values that were recorded but currently are unavailable), *attribute-concept values* (these missing attribute values may be replaced by any attribute value limited to the same concept), and *"do not care" conditions* (the original values were irrelevant). A *concept* (*class*) is a set of all objects classified (or diagnosed) the same way.

Two special data sets with missing attribute values were extensively studied: in the first case, all missing attribute values were lost (Stefanowski and Tsoukias, 2001), in the second case, all missing attribute values were "do not care" conditions (Kryszkiewicz, 1999).

In attribute-concept values interpretation of a missing attribute value, the missing attribute value may be replaced by any value of the attribute domain restricted to the concept to which the object with a missing attribute value belongs (Grzymala-Busse and Grzymala-Busse, 2007). For example, if for a patient the value of an attribute Temperature is missing, this patient is sick with flu, and all remaining patients sick with flu have values high or very-high for Temperature when using the interpretation of the missing attribute value as the attribute-concept value, we will replace the missing attribute value with high and very-high. Additionally, the *characteristic relation*, (see, e.g., Grzymala-Busse and Grzymala-Busse, 2007) enables to process data sets with all three kind of missing attribute values at the same time: lost, "do not care" conditions, and attribute-concept values.

Applications

Rough set methods can be applied as a component of hybrid solutions in machine learning and data mining. They have been found to be particularly useful for rule induction and feature selection (semantics-preserving dimensionality reduction). Rough set-based data analysis methods have been successfully applied in bioinformatics, economics and finance, medicine, multimedia, web and text mining, signal and image processing, software engineering, robotics, and engineering (e.g. power systems and control engineering).

Extensions

Dominance-based rough set approach (DRSA) is an extension of rough set theory for multi-criteria decision analysis (MCDA), introduced by Greco, Matarazzo and Słowiński (2001). The main change in this extension of classical rough sets is the substitution of the indiscernibility relation by a *dominance* relation, which permits the formalism to deal with inconsistencies typical in consideration of criteria and preference-ordered decision classes.

Decision-theoretic rough sets (DTRS) is a probabilistic extension of rough set theory introduced by Yao, Wong, and Lingras (1990). It utilizes a Bayesian decision procedure for minimum risk decision making. Elements are included into the lower and upper approximations based on whether their conditional probability is above thresholds α and β. These upper and lower thresholds determine region inclusion for elements. This model is unique and powerful since the thresholds themselves are calculated from a set of six loss functions representing classification risks.

History

The idea of rough set was proposed by Pawlak (1981) as a new mathematical tool to deal with vague concepts. Comer, Grzymala-Busse, Iwinski, Nieminen, Novotny, Pawlak, Obtulowicz, and Pomykala have studied algebraic properties of rough sets. Different algebraic semantics have been developed by P. Pagliani, I. Duntsch, M. K. Chakraborty, M. Banerjee and A. Mani; these have been extended to more generalized rough sets by D. Cattaneo and A. Mani, in particular. Rough sets can be used to represent ambiguity, vagueness and general uncertainty. Fuzzy-rough sets further extend the rough set concept through the use of fuzzy equivalence classes.

See also

- Algebraic semantics
- Alternative set theory
- Description logic
- Fuzzy logic
- Fuzzy set theory

- Generalized rough set theory
- Granular computing
- Near sets
- Rough fuzzy hybridization
- Semantics of rough set theory
- Soft computing
- Type-2 fuzzy sets and systems
- Decision-theoretic rough sets
- Variable precision rough set
- Version space
- Dominance-based rough set approach

References

- Pawlak, Zdzisław (1982). "Rough sets". *International Journal of Parallel Programming* **11** (5): 341–356. doi:10.1007/BF01001956 (http://dx.doi.org/10.1007%2FBF01001956) .
- Bazan, Jan; Szczuka, Marcin and Wojna, Arkadiusz and Wojnarski, Marcin (2004). "On the evolution of rough set exploration system". *Proceedings of the RSCTC 2004*: 592–601.
- Dubois, D.; Prade, H. (1990). "Rough fuzzy sets and fuzzy rough sets". *International Journal of General Systems* **17**: 191–209. doi:10.1080/03081079008935107 (http://dx.doi.org/10.1080%2F03081079008935107) .
- Greco, Salvatore; Matarazzo, Benedetto and Słowiński, Roman (2001). "Rough sets theory for multicriteria decision analysis". *European Journal of Operational Research* **129** (1): 1–47. doi:10.1016/S0377-2217(00)00167-3 (http://dx.doi.org/10.1016%2FS0377-2217%2800%2900167-3) .
- Grzymala-Busse, Jerzy (1997). "A new version of the rule induction system LERS". *Fundamenta Informaticae* **31**: 27–397.
- Grzymala-Busse, Jerzy; Grzymala-Busse, Witold (2007). "An experimental comparison of three rough set approaches to missing attribute values". *Transactions on Rough Sets* **6**: 1–47.
- Kryszkiewicz, Marzena (1999). "Rules in incomplete systems". *Information Sciences* **113**: 271–292. doi:10.1016/S0020-0255(98)10065-8 (http://dx.doi.org/10.1016%2FS0020-0255%2898%2910065-8) .
- Pawlak, Zdzisław; Wong, S. K. M. and Ziarko, Wojciech (1988). "Rough sets: Probabilistic versus deterministic approach". *International Journal of Man-Machine Studies* **29**: 81–95. doi:10.1016/S0020-7373(88)80032-4 (http://dx.doi.org/10.1016%2FS0020-7373%2888%2980032-4) .
- Pawlak, Zdzisław (1991). *Rough Sets: Theoretical Aspects of Reasoning About Data*. Dordrecht: Kluwer Academic Publishing. ISBN 0-7923-1472-7.
- Slezak, Dominik; Wroblewski, Jakub; Eastwood, Victoria and Synak, Piotr (2008). "Brighthouse: an analytic data warehouse for ad-hoc queries (http://www.vldb.org/pvldb/1/1454174.pdf) ". *Pvldb 1(2)*: 1337–1345.
- Stefanowski, Jerzy (1998). "On rough set based approaches to induction of decision rules". In Polkowski, Lech and Skowron, Andrzej. *Rough Sets in Knowledge Discovery 1: Methodology and Applications*. Heidelberg: Physica-Verlag. pp. 500–529.
- Stefanowski, Jerzy; Tsoukias, Alexis (2001). "Incomplete information tables and rough classification". **17**. 545–566.
- Wong, S. K. M.; Ziarko, Wojciech and Ye, R. Li (1986). "Comparison of rough-set and statistical methods in inductive learning". *International Journal of Man-Machine Studies* **24**: 53–72.
- Yao, J. T.; Yao, Y. Y. (2002). "Induction of classification rules by granular computing". *Proceedings of the Third International Conference on Rough Sets and Current Trends in Computing (TSCTC'02)*. London, UK: Springer-Verlag. pp. 331–338.
- Ziarko, Wojciech (1998). "Rough sets as a methodology for data mining". *Rough Sets in Knowledge Discovery 1: Methodology and Applications*. Heidelberg: Physica-Verlag. pp. 554–576.
- Ziarko, Wojciech; Shan, Ning (1995). "Discovering attribute relationships, dependencies and rules by using rough sets". *Proceedings of the 28th Annual Hawaii International Conference on System Sciences (HICSS'95)*. Hawaii. pp. 293–299.

External links

- The International Rough Set Society (http://www.roughsets.org)
- Rough set tutorial (http://www.uit.edu.vn/forum/index.php?act=Attach&type=post&id=19757)
- Example-based simple tutorial (http://www.ghastlyfop.com/blog/2006/01/rough-sets-quick-tutorial.html)
- Rough Set Exploration System (http://logic.mimuw.edu.pl/~rses/)
- Rough Sets in Data Warehousing (http://rsctc2008.cs.uakron.edu/Invited%20Speakers/Presentations/Slezak%20Revised%20RSCTC%20Presentation.ppt)

Retrieved from "http://en.wikipedia.org/wiki/Rough_set"
Categories: Systems of set theory | Theoretical computer science | Machine learning