Fundamenta Informaticae 59 (2004) 203–219
IOS Press

A Comparison of Rough Set Methods and Representative Inductive Learning Algorithms

Duoqian Miao*
Department of Computer Science and Engineering
Tongji University
Shanghai 200092, P. R. China
miaoduoqian@163.com

Lishan Hou
Department of Mathematics
Shanxi University
Taiyuan 030006, P. R. China
hlslisa@163.com

Abstract. Rough set theory is a kind of new tool to deal with knowledge, particularly when knowledge is imprecise, inconsistent and incomplete. In this paper, we discuss some inductive machine learning techniques in the framework of the knowledge reduction approach based on rough set theory. The Monk's problems introduced in the early of nineties are resolved again employing rough set methods and their results are compared and analyzed with those obtained at that time. As far as accuracy and conciseness are concerned, the learning algorithms based on rough sets have remarkable superiority.

Keywords: rough sets, AQ algorithms, ID algorithms, attribute reduction, dynamic reduct

1. Introduction

During the 2nd Europe Summer School on Machine Learning, held in Coresendonk Priory in Belgium in the summer of 1991, many popular machine learning algorithms at that time (e.g., AQ17-HCI, AQ17-

*Address for correspondence: Department of Computer Science and Engineering, Tongji University, Shanghai 200092, P. R. China

FCI, AQ15-GA, AQR, ID3, IDL and ID5R) were discussed extensively. Some questions have been formulated, like: "Which algorithm would be optimal?" or "Does there exist a kind of method being the foundation for all other algorithms?"

As a consequence of the discussion three problems (called Monk's problems) based on data sets [2] have been created. Each training example from data sets is represented by six discrete-valued attributes. The tasks of learning is to extract a binary function from the given training set, to learn the description of concepts, and to examine the accuracy of the function with respect to the classification quality measured on a given test set, then to compare the capacities of all these algorithms. The data sets differ in size, target concept, and existence of noise in the training examples. In Monk-1, there are 124 training examples, which occupy 30% of the total event space (62 positive and 62 negative); and the testing examples are all possible examples (216 positive and 216 negative). In Monk-2, there are 169 training examples, which occupy 40% of the total event space (64 positive and 105 negative); and the testing examples are all possible examples (190 positive and 242 negative). In Monk-3, there are 122 training examples, which occupy 30% of the total event space (62 positive and 60 negative); and the testing examples are all possible examples (204 positive and 228 negative). We are informed that 5% of the examples are misclassified in Monk-3.

Monk's problems rely on the artificial robot domain, in which robots are described by six different attributes:

- a1 head-shape with values in: {round, square, octagon};
- a2 body-shape with values in: {round, square, octagon};
- a3 is-smiling with values in: {yes, no};
- a4 holding with values in: {sword, balloon, flag};
- a5 jacket-color with values in: {red, yellow, green,blue};
- a6 has-tie with values in: {yes, no}.

The learning task is a binary classification task. Each problem is given by a logical description of a class. Robots belong either to this class or not, but instead of providing a complete class description to the learning problem, only a subset of all 432 possible robots with its classification is given. The learning task is then to generalize over these examples and, if the particular learning technique at hand allows this, to derive a simple class description.

Target concepts are:

- Monk-1 "(head-shape=body-shape) or (jacket-color=red)";
- Monk-2 " exactly two of the six attributes have their first value";
- Monk-3 "(jacket-color is green and holding a sword) or (jacket-color is not blue and body-shape is not octagon)".

Monk-1 is described by a simple formula in disjunctive normal form (DNF) and it is supposed to be easy learnable by all symbolic learning algorithms as AQ and decision trees. Conversely, Monk-2 is similar to parity problems. It combines different attributes in a way which makes it complicated to describe in DNF or CNF using the given attributes only. Monk-3 is again in DNF and serves to evaluate the algorithms under the presence of noise.

However, the comparison to various inductive learning techniques was limited to learning results, learning efficiency, and so on. The inherent connections among the methods have not neen explained from the theoretical point of view. In this paper, several representative machine learning algorithms are compared with the system based on rough sets. Monk's problems are analyzed by the knowledge reduction approach based on rough sets. Comparisons and further conclusions concerning accuracy and conciseness of rules are showing some potential of rough set methods.

2. Theoretical analysis

In 1982, Z. Pawlak put forward the idea of a rough set as a tool to study the representation, learning and induction of the uncertain, incomplete and imprecise information [1, 4]. Since it was proposed, the theory has been applied extensively and approved generally in many fields depending on its effective learning ability. The knowledge reduction approach based on rough sets plays more and more important role.

According to rough sets, possessing knowledge means having the ability of correctly classify objects [3]. Learning knowledge is drawing rules from the given data sets, that is based on knowledge reduction using the rough set approach. Knowledge reduction, which has been one of the central topics investigated in rough sets up to now, consists in removing of superfluous attributes and attribute values. The set of all examples in the training set is called the universe. Variables describing examples are called attributes. Any attribute defines an equivalence relation and can be used to partition the universe. The universe and a family of corresponding equivalence relations create an information system. Moreover, if the attributes are divided into condition attributes and decision attributes, the system turns into a decision table.

We are mainly interested in decision tables in this paper. A decision table can be defined formally as follows: $DT = \langle U, C \cup D, V, f \rangle$, in which U is the universe, C and D are the condition attribute set and the decision attribute set respectively, $C \cap D = \emptyset$, $V = \cup V_a \, (a \in C \cup D)$, V_a is the value set of a, f is the information function, f: $U \times (C \cup D) \rightarrow V$ and $f_x(a) = a(x)$ for every $x \in U$ and $a \in C \cup D$.

Knowledge reduction of decision table consists in eliminating the irrelevant condition attributes and irrelevant condition attribute values with respect to decision attributes. It is necessary when deposing a large scale data, otherwise the cost due to superfluous knowledge is very huge in both time and space. Reduction is one of the most important features of rough sets in comparison to other methods [5, 6, 19].

2.1. Several known knowledge reduction algorithms in rough set framework [14, 15]

A decision table may have more than one reduct, hence there must exist a minimal reduct which consists of all the useful information of the decision table without any superfluity (i.e., in a minimal length form). Unfortunately, finding the minimal reduct is NP-complete problem [10]. Existing reduction algorithms cannot always find out the minimal solution, in many cases one only gets sub-optimal attribute subsets that are approximations to some reducts [19].

Different knowledge representations could lead to different knowledge reduction algorithms. At present there exist several popular reduction algorithms such as the algorithm based on Pawlak's attribute importance, algorithms based on discernibility matrix and its improvements, algorithms based on information entropy and so on. It can be shown that all these approaches can be based on Boolean reasoning (see, e.g., [19], [20]). In the following we will discuss the relations among some of the algorithms mentioned above and inductive machine learning ones.

2.1.1. Pawlak's Algorithm

$$Input\ \ DT = \langle U, C \cup D, V, f \rangle$$
$$Output\ \ \{B \subseteq C | POS_B(D) = POS_C(D)\}$$

Step 1 Compute relative core of DT, i.e. C_O.

Step 2 $B \Leftarrow C_O$.

Step 3 For any $c \in C - B$, compute $Sig(c, B, D)$.

If $Sig(c', B, D) = \max\limits_{c \in C-B} \{Sig(c, B, D)\}$, then $B \Leftarrow B \cup \{c'\}$.

Step 4 If $POS_B(D) = POS_C(D)$, output B and end;

else go to step 3.

in which:

$$Sig(c, B, D) = \frac{card(POS_{B \cup \{c\}}(D)) - card(POS_B(D))}{card U}.$$

X.H.Hu used an algorithm equivalent to Pawlak's algorithm in which the definition of the function $Sig(c, B, D)$ was as follows:

$$Sig(c, B, D) = \frac{card(POS_{B \cup \{c\}}(D)) - card(POS_B(D))}{card(POS_C(D))}.$$

2.1.2. Relations to ID-family

ID3 is one of the most important models in inductive machine learning. It is based on the idea of dynamic partitioning of the space of training examples leading to a decision tree structure.

According to the strategy of ID3, one proceeds using the following procedure: choose an attribute a and partition U into $\{T_a^1, \cdots, T_a^n\}$, in which the superscript k of T will be referred to as a label of a in V_a. Successively, take T_a^k as new universe (example sets) and redo the process mentioned above until all the examples belong to the same decision classification. Thus, we get a decision tree.

According to rough sets, U can be partitioned by every attribute $a \in C$ into U/a.

Pawlak's as well as Hu's algorithm is just based on this partition. For the sake of illustration, let us take attribute a. ${}^{U}/_{a}$ corresponds to $\{T_a^1, \cdots, T_a^n\}$ of the decision tree and $T_a^k \in {}^{U}/_{a},\ k = 1, \cdots, n$, in which $n = card({}^{U}/_{a})$. Therefore, creating a decision description is equivalent to partitioning of the universe by attributes consecutively [13].

Let B be a non-empty attribute set such that $POS_B(D) \neq POS_{B-\{b\}}(D)$, B for any $b \in B$. Then B is independent. If B is independent and $POS_B(D) = POS_C(D)$, then B is just called relative reduct of C (relatively to D) or Pawlak's relative reduct. Actually, relative reducts are kinds of reducts often required. Similarly, there are considered in rough sets local reducts, i.e., reducts relative to objects that preserve discernibility of objects with respect to a given object. In ID3, partitions are created under the restriction of keeping the ttee structure, hence it can be seen as a specialization of a general rough set idea of finding partitions of the universe into indiscernibility classes, corresponding to approximation of local reducts.

On the basis of ID3, new algorithms relevant to different requests have been developed using some strategies. For instance, ID3 with windowing, ID5R that is an incremental decision tree learning algorithm, IDL that uses some heuristics to stimulate a bi-directional search for a tree [11].

The ID5R algorithm has been developed as a kind of TDIDT-algorithm that is able to work incrementally, but results finally (i.e., after all training examples) in the same decision as ID3. "Incremental" means that the examples can be given one after another. A very easy solution for the problem of successively given examples would be to generate an ID3 decision tree from scratch with all examples given so far. In contrast to that approach, ID5R always uses the decision tree developed so far for integrating the new example. For that reason the data structure of a node in an ID5R tree has been enlarged by the information necessary to calculate the information gain function of the attributes.

IDL is an algorithm for the incremental induction of decision trees. It is specially designed to find small decision trees. There are various reasons to prefer smaller trees. One reason is efficiency: the fewer decision nodes in a tree, the more efficiently an instance can be classified with it. Another reason is comprehensibility: small trees tend to be easier to understand.

2.1.3. Improvement of discernibility matrix algorithm [17]

Discernibility matrix is an important tool for rough sets. It makes it possible to transform the reduction problem of a database into the simplification problem of a matrix by Boolean reasoning. Skowron (see, e.g., [10, 19, 20]) has proved that Boolean reasoning applied to discernibility matrices is a powerful tool in knowledge reduction and searching for relevant patterns.

Let $DT = \langle U, C \cup D, V, f \rangle$ be a decision table, whose corresponding discernibility matrix is defined as follows:

$$M(DT) = (c_{ij})_{n \times n}, \quad n = card(U),$$

in which,

$$c_{ij} = \begin{cases} \emptyset & [x_i]_{IND(D)} = [x_j]_{IND(D)} \\ \{a \in C | a(x_i) \neq a(x_j)\} & otherwise. \end{cases}$$

In such a way, the decision table analysis can be shifted to analysis of the discernibility matrix (and the Boolean functions constructed on the basis of such matrices). In particular, the higher the frequency of the attribute is in the discernibility matrix, the more examples are distinguished by it, and the more important the attribute is [7, 10].

Below we present an example of algorithm based on the discernibility matrix.

$$Input \;\; DT = \langle U, C \cup D, V, f \rangle$$
$$Output \;\; \{B \subseteq C | POS_B(D) = POS_C(D)\}$$

Step 1 Create the corresponding discernibility matrix $M(DT)$.

Step 2 Calculate core C_O. Classify the elements of $M(DT)$ according to the cardinality of them; $M_1, M_2, \cdots, M_m$ are sets of all entries of $M(DT)$ with $1, \ldots, m$ elements, respectively. It is obvious that M_1 is just the core C_O.

Step 3 $B \Leftarrow C_O$.

Step 4 $M(DT) = M(DT) - \{c_{ij} |\ c_{ij} \cap B \neq \emptyset\}$.

Step 5 If $M(DT) = \emptyset$, then output B and stop.

Step 6 Compare the importance degree of every $c \in C - B$.

$Sig\,(c, B, D)$ is the frequency of c in M_i, in which M_i has the least subscript among the current non-empty M_j $(j = 1, 2, \cdots, m)$.

Denote $Sig(c', B, D) = \max\limits_{c \in C-B} \{Sig(c, B, D)\}$. $B \Leftarrow B \cup \{c'\}$, goto step 4.

Remark We would like to end this section with the following remark. If there are more than one attribute having the highest frequency in M_i, then consider their frequencies in M_{i+1} till the only one attribute is find out, otherwise choose c' randomly from possible candidates. Let us observe that discernibility matrix algorithm is also not complete to independence reduction.

Comparisons with AQ algorithms are included in the following section.

2.1.4. Relations to AQ-family

The basic idea of AQ algorithms of covering by rules of an example can be described as follows:

Step 1 Select a seed example from the set of training examples for a given decision class.

Step 2 Generate a set of alternative most general rules (a star) that cover the seed example, but do not cover any negative example of the class.

Step 3 Select the best rule from the star according to an evaluation function, and remove the examples covered by this rule from the set of positive examples yet to be covered.

Step 4 If this set is not empty, select a new seed from it and go to step 2.

Otherwise, if another decision class still requires rules to be learned, return to step 1, and perform it for the other decision class.

In fact, selecting seed example is equivalent to selecting a column from the discernibility matrix. According to the construction of discernibility matrix, if some element of selected column is empty, its corresponding example belongs to positive examples for a consistent table.

Removing positive examples is equivalent to removing corresponding rows of discernibility matrix. The evaluation function is based on the frequency of the attribute in the column. One can prove that such a strategy is equivalent to that used by AQ algorithms reducing directly attribute value without dealing

with attributes. Hence, AQ-algorithms can be placed in the rough set system. For a more detailed analysis see [21].

In the same way, as a basic method, AQ-algorithms lead to many effective algorithms relevant to specific types of problems by using hypothesis-driven strategy or combining with genetic algorithms (see, e.g., AQ17-HCI (Hypothesis-Driven Constructive Induction), AQ17-FCLS (Flexible Concept Learning), and AQ15-GA (Genetic Algorithm) [12]).

AQ17-HCI is a module employed in the AQ17 attribute-based multi-strategy constructive learning system. This module implements a new iterative constructive induction capability in which new attributes are generated on the basis of analysis of the hypotheses produced in the previous iteration. Input to the HCI module consists of the example set and a set of rules. The rules are then evaluated according to a rule quality criterion, and the rules that score best for each decision class are combined into new attributes. These attributes are incorporated into the set of training examples, and the learning process is repeated. The process continues until a termination criterion is satisfied. The method is a special implementation of the idea of the "survival of the fittest", and therefore can be viewed as a combination of symbolic learning with a form of genetic algorithm-based learning.

A brief description of the HCI algorithm is as follows:

Step 1 Induce rules for each decision class using a standard AQ algorithm from a subset of the available training examples.

Step 2 Identify variables from the original set that are not present in the rules, and classify them.

Step 3 For each decision class, generate a new attribute that represents the disjunction of the highest quality.

Step 4 Modify the training examples by adding the newly constructed attributes and removing the ones found to be irrelevant.

Step 5 Induce rules from this modified training set.

Step 6 Test these rules against the remainder of the training set. If the performance is not satisfactory, return to step 1. Otherwise, extend the initial complete set of training examples with the attributes from the obtained rules. Induce the final set of rules from this set of examples.

AQ17-FCLS combines both symbolic and numeric representations in generating a concept description. The program is oriented toward learning flexible concepts, i.e, imprecise and context dependent. To characterize such concepts the program creates two-tiered descriptions, which consist of a Basic Concept Representation (BCR) and an Interpretation (ICI) to handle exceptions. In the program, the BCR is in the form of rules, and the ICI is in the form of a weighted evaluation function which sums up the contributions of individual conditions in a rule, and compares it with a THRESHOLD. The learning program induces both the rules and an appropriate value for the THRESHOLD.

AQ15-GA is an approach with attribute selection by a genetic algorithm. Genetic algorithms are used to explore the space of all subsets of attribute set. Each of the selected attribute subsets is evaluated (its fitness measured) by invoking AQ15 and measuring the recognition rate of the rules produced.

The evaluation procedure as shown is divided into three main steps. After an attribute subset is selected, the initial training data, consisting of the entire set of attribute vectors and class assignments

corresponding to examples from each of the given classes, is reduced. This is done by removing the values for attributes that were eliminated from the original attribute vector. The second step is to apply a classification process (AQ15) to the new reduced training data. The decision rules that AQ15 generates for each of the given classes in the training data are then used for classification. The last step is to use the rules produced by the AQ algorithm in order to evaluate the classification and hence, recognition, with respect to the test data.

2.1.5. Attribute reduction algorithm based on information entropy [6, 15]

Entropy measures the average information provided by the information source. The mutual-entropy measures the information of one information source derived from another source.

Let U be a universe, P and Q - equivalence relations on U. We introduce the following definitions:

$$U/IND(P) = \{X_1, X_2 \dots X_n\}, \;\; U/IND(Q) = \{Y_1, Y_2 \dots Y_m\};$$

$$p(X_i) = cardX_i/cardU, \;\; p(Y_j) = cardY_j/cardU;$$

$$p(X_iY_j) = card(X_i \cap Y_j)/cardU, \;\; p(Y_j|X_i) = p(X_i \cap Y_j)/p(X_i);$$

where $1 \leq i \leq n$, $1 \leq j \leq m$.

Information entropy of P is given by

$$H(P) = -\sum_{i=1}^{n} p(X_i) \log p(X_i).$$

Conditional entropy of Q conditioned on P is given by

$$H(Q|P) = -\sum_{i=1}^{n} p(X_i) \sum_{j=1}^{m} p(Y_j|X_i) \log p(Y_j|X_i).$$

The mutual information entropy of P and Q is

$$I(P;\, Q) = H(Q) - H(Q|P).$$

A simple knowledge reduction algorithm based on information entropy can be similar to Pawlak's algorithm, and the difference lies in the definition of the significance function and stop conditions. Using this approach, information entropy can be seen as a kind of measurement of attribute importance, and the attribute that makes mutual-entropy greatest after being selected is the most important, that is,

$$Sig(c, B, D) = I(B \cup \{a\}; D) - I(B; D);$$

$$Sig(c', B, D) = \max_{c \in C-B} \{Sig(c, B, D)\},$$

c' is the most important attribute conditioned on B with respect to D. And the stop condition is:

If $I(B; D) = I(C; D)$, then B is a solution.

It is easy to see this approach is also not complete to independence reduction.

For more details on entropy based reducts see [22].

Remark The speed of computation of the entropy algorithm is high: attributes are eliminated one by one on the basis of comparison of information entropy values.

2.1.6. Dynamic Attribute Reduction Algorithms [18]

The algorithms based on rough sets mentioned above are sometimes not sufficient for extracting laws from knowledge bases. The reason is that they are not taking into account the fact that different reducts may have different probabilities of occurence in sets of sub-tables created by random samples of a given knowledge base. It is obvious that the larger such probability is, the more objective the reduct is. J.G. Bazan in [18] proposed an idea of dynamic reducts, according to which, dynamic reducts are in some sense the most stable attribute reducts of a given database, i.e., they are the most frequently appearing reducts in sub-tables. At the same time, he proposed a method of the decision rule generation on the basis of dynamic reducts as those rules are better predisposed to classify unseen cases. In what follows, we call the non-dynamic algorithms "standard rough sets methods."

2.2. Summary

In the previous sections, taking the four algorithms as a cue, we discussed the equivalence of knowledge reduction algorithms on the basis of rough sets, ID-family and AQ-family theoretically. This work laid rough sets, a new learning tool, as a foundation of machine learning. That is, the main techniques of inductive machine learning are united in the knowledge reduction approach based on rough sets from the theoretical point of view. In what follows, we are going to examine the superiority and specificity of rough sets in terms of other machine learning techniques in practice.

3. Comparisons and Analysis

As we have already mentioned in the previous sections, the reduction approach based on rough sets covers the traditional representative algorithms of machine learning. What's more, it equals and even superiors of many other methods. The results are promising not only with respect to the accuracy but also in the number of rules. Monk's problems were created specially to test the capability of learning algorithms. We made experiments on Monk's problems with the algorithms mentioned above in order to compare and analyze them.

3.1. Monk-1 problem

In Monk-1, there are 124 training examples, 62 positive and 62 negative and without any noise. Its target concept is "(a1=a2) or (a5=1)".

3.1.1. Employment of rough sets

After reducing the attributes, the core is {a1, a2, a5} and the reduct is just the core. Then reduce the superfluous attribute values. The rules are showed in Table 1.

There are 21 rules, of which 4 belong to positive rules. They are showed in Table 2.

Table 1. All rules extracted from Monk-1 by rough sets

U	a1	a2	a5	d	rule
1	1	1	*	1	if a1=1 and a2=1 then d=1
9	*	*	1	1	if a5=1 then d=1
11	1	2	2	0	if a1=1 and a2=2 and a5=2 then d=0
12	1	2	3	0	if a1=1 and a2=2 and a5=3 then d=0
13	1	2	4	0	if a1=1 and a2=2 and a5=4 then d=0
27	1	3	2	0	if a1=1 and a2=3 and a5=2 then d=0
28	1	3	4	0	if a1=1 and a2=3 and a5=4 then d=0
33	1	3	3	0	if a1=1 and a2=3 and a5=3 then d=0
46	2	1	3	0	if a1=2 and a2=1 and a5=3 then d=0
50	2	1	2	0	if a1=2 and a2=1 and a5=2 then d=0
52	2	1	4	0	if a1=2 and a2=1 and a5=4 then d=0
62	2	2	*	1	if a1=2 and a2=2 then d=1
79	2	3	3	0	if a1=2 and a2=3 and a5=3 then d=0
82	2	3	4	0	if a1=2 and a2=3 and a5=4 then d=0
86	2	3	2	0	if a1=2 and a2=3 and a5=2 then d=0
91	3	1	2	0	if a1=3 and a2=1 and a5=2 then d=0
96	3	1	3	0	if a1=3 and a2=1 and a5=3 then d=0
99	3	2	4	0	if a1=3 and a2=2 and a5=4 then d=0
104	3	2	3	0	if a1=3 and a2=2 and a5=3 then d=0
106	3	2	2	0	if a1=3 and a2=2 and a5=2 then d=0
109	3	3	*	1	if a1=3 and a2=3 then d=1

Table 2. Positive rules extracted from Monk-1 by rough sets

U	a1	a2	a5	d	rule
1	1	1	*	1	if a1=1 and a2=1 then d=1
2	*	*	1	1	if a5=1 then d=1
3	2	2	*	1	if a1=2 and a2=2 then d=1
4	3	3	*	1	if a1=3 and a2=3 then d=1

The value sets of a1 and a2 are {1, 2, 3}, so we can merge rule1, 3, 4 together. In other words, the learning result to Monk-1 by rough sets can be described as (a1=a2) or (a5=1), which is coincident entirely with the target concept. The accuracy accounts for 100% at the test sets (216 positive and 216 negative). This is a scarce result.

Dynamic methods also get a good result: there are 10 rules and the accuracy is 100%.

3.1.2. Employment of AQ-family

We are going to handle Monk-1 employing AQ17-HCI, AQ15-GA and AQR[2].

AQ17-HCI is a module employed in the AQ17 attribute based multi-strategy constructive learning system. This model implements a new iterative constructive induction capability in which attributes are generated based on the analysis of the hypotheses produced in the previous iteration. Rule is (pos=false), in which pos is the attribute constructed from the original ones, or intermediate ones, as defined below:

c01<:: (a1=1) & (a2=2, 3) & (a5>1)

c05<:: (a1=2) & (a2=1, 3) & (a5>1)

c08<:: (a1=3) & (a2=1, 2) & (a5>1)

c10<:: (a1=1) & (a2=1)

c12<:: (a5=1)

c13<:: (a1=2) & (a2=2)

c15<:: (a1=3) & (a2=3)

pos<:: (c10=false) & (c12=false) & (c13=false) & (c15=false)

neg<:: (c01=false) & (c05=false) & (c08=false)

Actually, (pos=false) means anyone of (c10=false), (c12=false), (c13=false) and (c15=false) is valid. It is coincident with the target concept and the accuracy is also 100%. But it is easy to see that the rule of AQ17-HCI is quite complex and cannot be used to make decisions directly.

AQ15-GA is the fusion of all subsets of a given attribute set. Each of the selected attribute subsets is evaluated by invoking AQ15 and measuring the recognition rate of the rules produced. The approach traverses the whole space of subsets. Huge cost of computing brings about excellent results and its accuracy is 100% too. The AQR algorithm is an implementation of the AQ-family. It produces a rule for each decision class. Monk-1 is a two-class problem, so learning rule is below:

(a2=1) & (a1=1) | (a5=1) | (a3=1) & (a2=2) & (a1=2) | (a2=2) & (a1=2) | (a6=1) & (a2=3) & (a1=3) | (a6=2) & (a1=3) & (a2=3) → class ‘1’.

This rule includes 5 attributes, a1, a2, a3, a5 and a6. But the reduct only includes 3 attributes a1, a2 and a5 according to rough sets, that is, 3 attributes are necessary to keep the capacity of classifying the data. In the rule of the AQR algorithm, there are two irrelevant attributes, so the rule either contains superfluous information or describes the concept too strictly. All these degrade the ability of generation. In this approach, the accuracy is 95.9% and lower than that of the two noted above.

3.1.3. Employment of ID-family

ID-family is a series of algorithms derived from decision trees via introducing some strategies. Its rules are limited to the structure of the corresponding decision tree. As to a new example, search the tree from its root to some leaf and the index of this leaf means the classification information of the example [2]. It seems that ID algorithms describe rules intuitively, but their accuracy is not satisfactory (Table 3).

Table 3. Learning results of Monk-1 by ID-family

	nodes	leaves	accuracy
ID3	13	28	98.6%
ID3 no windowing	32	62	83.2%
ID5R	34	52	79.7%
IDL	36	26	97.2%

Size of a decision tree embodies the conciseness of rules. Nonempty leaf includes the classification information, thus the number of nonempty leaves equals to that of rules. It is quite obvious that the rule numbers of ID-family algorithms are much greater than that rough sets. Its accuracy is relatively low.

Remark We learned rules on Monk-1 using three groups of algorithms. The knowledge reduction of rough sets is very satisfying. Its rule numbers is fewer and accuracy is high. We can say we gained all the knowledge comprised in Monk-1.

3.2. Monk-2 problem

In Monk-2, there are 169 training examples, 64 positive and 105 negative. The number of negative examples is far greater than that of positive examples. Again, there is no noise. Its target concept is: "exactly two of (a1=1), (a2=1), (a3=1), (a4=1), (a5=1) and (a6=1) are valid", which is very complex.

After the attribute reduction, the core is $\{$a1, a2, a3, a4, a5, a6$\}$, and the reduct is just the core. Then reduce the superfluous attribute values. The disposal process is similar to that of Monk-1 and 5 rules are produced. The knowledge we obtained is complicated and different from the target concept to a certain extent. The reason maybe be that too many negative examples exist in Monk-2. The accuracy is only 75%. Thus it can be seen that the standard reduction algorithms on the basis of rough sets need not special but general examples. But the dynamic methods have some advantages in dealing with such problems. They get a satisfying result: 43 rules and 97% accuracy.

Although the new attributes in AQ17-HCI are intricate and AQ17-FCLS summarizes 18 complicated rules, the accuracies of AQ-family on Monk-2 are high, as Table 4 shows:

Table 4. Accuracies of AQ-family to Monk-2

	AQ17-HCI	AQ17-FCLS	AQ15-GA	AQR
Accuracy	93.1%	92.6%	86.8%	79.7%

The results obtained using ID-family are shown in Table 5.

Table 5. Learning results of Monk-2 by ID-family

	nodes	leaves	accuracy
ID3	66	110	67.9%
ID3 no windowing	64	110	69.1%
ID5R	64	99	69.2%
IDL	170	107	66.2%

Obviously, the number of rules is relatively large and the accuracy is quite low. The algorithm is inferior to others on this problem.

3.3. Monk- 3 problem

In Monk-3, there are 122 training examples, 60 negative and 62 positive. The number of negative examples is less than that of positive examples. There are 5% misclassifications, i.e., noise in the training set. Its target concept is "(a5=3 and a4=1) or (a5<>4 and a2<>3)", which can be decomposed into 7 rules.

According to rough sets, the reduct is $\{$a1, a2, a4, a5$\}$, which has more attributes than target concepts by one due to noise. The rules are listed in Table 6.

The values of a1 are well distributed, so we can draw the rule as (a5=3 & a4=1) | (a5<>4 & a2<>3) | (a5=4 & a4=1) by neglecting a1, which is similar to the target concept. Rough sets is adaptive and rectifiable to some degree. But there is an intersection between the first two rules. We cannot tell the class of the examples belonging to the intersection. For example, we do not know classes of examples satisfying (a2=3 & a4=1 & a5=3). The reason is the data set of Monk-3 is inconsistent and with noise. Inconsistent data may lead to inconsistent rules.

In the test set, there are 12 examples we cannot tell the classes of which and 4 examples we misclassify. The accuracy is 96.4%, which is acceptable and comparative with the accuracy of dynamic algorithms (96.8%).

For comparison we list the results of AQ-family and ID-family in Table 7 and Table 8.

On Monk-3, the gaps among the accuracies of the three groups of algorithms are not very large, but the rules of rough sets are very concise.

It is worthwhile to note that the three algorithms based on rough sets are used to handle Monk's problems. It is interesting that the three reducts happen to be the corresponding cores. Reducts of a subset are not unique, but the core is unique. So the three reduction results on the same problem are identical. All three algorithms are of $O(n^2)$ complexity.

4. Conclusions

In this paper, the inherent connections among the reduction theory of rough sets and several typical inductive machine learning algorithms are discussed in detail, namely that rough sets can be seen as the foundation for others; some experiments and comparisons are made on Monk's problems. Monk's

Table 6. Positive rules extracted from Monk-3 by rough sets

U	a1	a2	a4	a5	d	rule
1	*	1	*	1	1	if a2=1 and a5=1 then d=1
2	*	1	*	2	1	if a2=1 and a5=2 then d=1
6	1	*	2	1	1	if a1=1 and a4=2 and a5=1 then d=1
14	1	1	3	*	1	if a1=1 and a2=1 and a4=3 then d=1
18	*	2	1	3	1	if a2=2 and a4=1 and a5=3 then d=1
19	1	2	*	2	1	if a1=1 and a2=2 and a5=2 then d=1
22	*	2	*	1	1	if a2=2 and a5=1 then d=1
24	1	*	3	2	1	if a1=1 and a4=3 and a5=2 then d=1
26	1	2	3	3	1	if a1=1 and a2=2 and a4=3 and a5=3 then d=1
56	2	1	*	3	1	if a1=2 and a2=1 and a5=3 then d=1
61	2	*	1	3	1	if a1=2 and a4=1 and a5=3 then d=1
66	*	2	3	2	1	if a2=2 and a4=3 and a5=2 then d=1
70	*	2	1	2	1	if a2=2 and a4=1 and a5=2 then d=1
72	2	2	2	3	1	if a1=2 and a2=2 and a4=2 and a5=3 then d=1
91	3	1	*	3	1	if a1=3 and a2=1 and a5=3 then d=1
96	*	1	2	3	1	if a2=1 and a4=2 and a5=3 then d=1
99	3	2	*	2	1	if a1=3 and a2=2 and a5=2 then d=1
101	3	2	2	*	1	if a1=3 and a2=2 and a4=2 then d=1
104	3	*	1	3	1	if a1=3 and a4=1 and a5=3 then d=1
105	3	*	2	1	1	if a1=3 and a4=2 and a5=1 then d=1
107	3	2	*	3	1	if a1=3 and a2=2 and a5=3 then d=1
109	*	3	1	3	1	if a2=3 and a4=1 and a5=3 then d=1
110	3	3	1	4	1	if a1=3 and a2=3 and a4=1 and a5=4 then d=1

Table 7. Accuracies of AQ-family to Monk-3

	AQ17-HCI	AQ17-FCLS	AQ15-GA	AQR
Accuracy	100%	97.2%	100%	87.0%

Table 8. Learning results of Monk-3 by ID-family

	nodes	leaves	accuracy
ID3	13	29	94.4%
ID3 no windowing	14	31	95.6%
ID5R	14	28	95.2%

problems are specially created to test the quality of learning methods. So the conclusions on them are believable and valuable.

The theory of rough set is a new tool for machine learning. It can be seen from the experiments that both standard and dynamic reduction algorithms get good results as to generational data and their rules are accurate, concise and comprehensive. They are superior to AQ-family and ID-family. As for these two series of rough sets methods, the best results of them are comparative at the most cases (e.g., Monk-1 and Monk-3) and the dynamic methods get a very high accuracy in Monk-2. Observe, that in preprocessing, relevant rules, i.e., dynamic rules, are extracted from large number of rules and they rely on time consuming computing of reducts in subtables. However, in the result a small number of high quality rules can be extracted. Reducing knowledge consists in removing of superfluous attributes and attribute values in such a way that the latter can revise the results of the former, for instance, the disposal of a1 in Monk-3. This is another important feature of rough sets. On the other hand, these reduction algorithms are influenced easily by data, that is, they do not have high ability of generalization.

AQ-family and ID-family are typical algorithms in traditional machine learning. The adaptive AQ-family can be used to different data sets and have good quality, but their rules are complicated and not accessible. ID-family are simple and easily available, but their results are inferior to AQ-family and rough sets.

5. Acknowledgements

Thanks to Professor A. SkowronDan and Professor L. Polkowski especially for their valuable comments and suggestions to this paper. This research was supported by grant No.60175016 from the National Natural Science Foundation of China.

References

[1] Z. Pawlak. Rough Sets-Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordecht, 1991.

[2] S.B. Thrun et al. The Monk's Problems-A Performance Comparison of Different Learning Algorithms. CMU-CS-91-197, Pittsburgh, PA, 1991: 2-80.

[3] I. Duntsch, G. Gediga. *Rough Set Data Analysis*, Encyclopedia of Computer Science and Technology, vol 43, 2000: 281-301.

[4] I. Duntsch, G. Gediga. *Uncertainty measures of Rough Set Prediction*, Artificial Intelligence, vol 106, 1998: 109-137.

[5] H.T. Nguyen. *Some Mathematical Structures for Computational Information*, Information Sciences, vol 128, 2000: 67-89.

[6] J.W. Guan, D.A. Bell. *Rough Computational Methods for Information Systems*, Artificial Intelligence, vol 105, 1998: 77-103.

[7] J.W. Guan, D.A. Bell. *Matrix Computation for Information Systems*, Information Sciences, vol 131, 2001: 129-156.

[8] M. Kryszkiewicz. *Rules in Incomplete Information Systems*, Information Sciences, vol 113, 1999: 271-192.

[9] L. Polkowski, A. Skowron. *Rough Sets: Perspectives, Rough Sets in Knowledge Discovery*, Physica-Verlag, 1998: 1-27.

[10] A. Skowron, C. Rauszer. *The Discernibility Matrix and Functions in Information Systems*, Handbook of Applications and Advances of the Rough Set Theory. Kluwer Academic Publishers, 1992: 331-362.

[11] B. Cestnik, I. Bratko. On Estimating Probabilities in Tree Pruning. Proc. of EWSL 91, Porto, Portugal, March 6-8, 1991, Lecture Notes in Artificial Intelligence Vol 482, Springer-Verlag, Heidelberg, 1991, 138-150.

[12] J.R. Quinlan. *Learning Logical Definitions from Relations*. Machine Learning, 1990, vol 5(3): 239-266.

[13] J. Wang. *Contributions on Rough Set Theory to Inductive Machine Learning*, Computer Science, 2001, vol 28(5): 5-7.

[14] D.Q. Miao, G.R. Hu. *A Heuristic Algorithm of Knowledge Reduction*, Computer Research and Development, 1999, 36(6): 681-684.

[15] D.Q. Miao, J. Wang. *An Information Representation of Concepts and Operations in Rough Sets*, Journal of Software, 1999, 10(2): 113-116.

[16] D.Q. Miao, J. Wang. *Analysis on Attribute Reduction Strategies of Rough Set*, Chinese Journal of Computer Science and Technology, vol 13(2), 1998: 189-192.

[17] D.Q. Miao, L.S. Hou. *A Heuristic Algorithm for Reduction of Knowledge Based on Discernibility Matrix*, International Conference on Intelligent Information Technology (ICIIT-02), September 22-25, 2002, Beijing, China. 276-279.

[18] J.G. Bazan. *A Comparison of Dynamic and Non-dynamic Rough Set Methods for Extracting Laws from Decision Tables*, In: L. Polkowski, A. Skowron, Rough Sets in Knowledge Discovery, Physica-Verlag, Heidelberg. 321-365.

[19] J. Komorowski, Z. Pawlak, L. Polkowski, A. Skowron. *Rough Sets: A Tutorial*, In: S.K. Pal, A. Skowron (eds.), *Rough-Fuzzy Hybridization: A New Trend in Decision Making*, Springer-Verlag, Singapore, 1999, 3–98.

[20] A. Skowron. Rough Sets in KDD (plenary talk), In: Z. Shi, B. Faltings, and M.Musen (eds.), Proceedings of the 16-th World Computer Congress (IFIP'2000), Beijing, 2000. Publishing House of Electronic Industry, 2000, 1-14.

[21] A. Skowron, J. Stepaniuk. *Decision Rules Based on Discernibility Matrices and Decision Tables*, In: T.Y. Lin and A.M. Wildberger (eds.), The Third International Workshop on Rough Sets and Soft Computing Pro-

ceedings (RSSC'94), San Jose State University, San Jose, California, USA, November 10-12, 1994, 602-609; see also: Decision Rules Based on Discernibility Matrices and Decision Matrices", In: T.Y. Lin, and A.M. Wildberger (eds.), Soft Computing, Simulation Councils, Inc., San Diego, 1995, 6-9; also In: ICS Research Report 40/94, Warsaw University of Technology.

[22] D. Slezak. *Approximate Entropy Reducts*, Fundamenta Informaticae, 53(3-4): 365-387, 2002.