Finding the Reduct Subject to Preference Order of Attributes

Xiaofeng Zhang, Yongsheng Zhao, and Hailin Zou

School of Computer Science and Technology
Ludong University, Yantai 264025, P.R. China
iamzxf@126.com, jsjzhao@sohu.com, zhl_8655@sina.com

Abstract. In machine learning and knowledge discovery, rough set theory is a useful tool to be employed as a preprocessing step for dimension reduction. However, for a given system, there may be more than one reduct to be selected. Different reducts will lead to discovered knowledge, which may be concise, precise, general, understandable and practically useful in different levels. It is a crucial issue to select the most suitable features or properties of the objects in a dataset in the machine learning process. In this paper, some external information is added to information system and may be simply regarded as user preference on attributes. Consequently, it will guide the procedure of retrieving reducts, which will give birth to the reduct subject to preference order of attributes.

Keywords: Reduct, rough set theory, preference order of attributes.

1 Description of Problem

Information system is the main object in data mining and knowledge discovery. It consists of two major parameters of complexity leading to intractable behavior: the number of attributes in an application domain, namely dimensionality, and the number of examples in a dataset. The latter is typically applied only to the training stage of the system and, depending on intended use, may be acceptable. However, data dimensionality is an obstacle for both the training and runtime phases of a learning system. Many systems exhibit non-polynomial complexity with respect to dimensionality, which imposes a ceiling on the applicability. The curse of dimensionality limits the applicability of learning systems to a great degree.

Rough set theory, proposed by Pawlak Z. [3], is a formal methodology that can be employed in data reduction as a preprocessing step. A fundamental notion supporting this is the concept of reduct, which has been studied extensively by many researchers. A reduct is a subset of attributes which are jointly sufficient and individually necessary for preserving the same information under consideration as provided by the entire set of attributes. However, for a given information system, there may be more than one reduct. The use of different reducts will lead to different discovered knowledge. Typically, discovered knowledge should be concise, precise, general, easy to understand and practically useful, which can

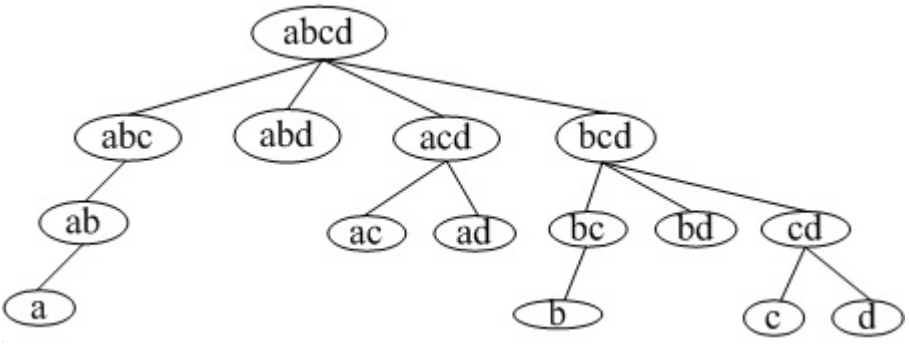
M. Kryszkiewicz et al. (Eds.): RSEISP 2007, LNAI 4585, pp. 202–210, 2007.

be measured according to external information. In this paper, we will consider such external information simply as user preference, which may be weights of attributes, ranking of attributes, and etc. Especially, if the preference is formally a chain, then the reduct subject to preference order of attributes will be unique.

2 Related Research

This section will introduce several proposals associated with "optimal reduct". One is "optimal reduct" in [2], which is in fact the reduct containing the least number of attributes, also is the shortest one. This algorithm makes use of heuristic information in discernibility matrix–attribute frequency to retrieve the shortest reduct, yet they cannot make sure that the reduct they get is affirmatively the shortest one in any case, but in most cases.

In [6], we propose the concept "optimal reduct under dictionary order". We assume that all attributes are ordered lexically, and therefore, all reducts are ordered lexically accordingly. Since dictionary order is formally a chain, the optimal reduct is unique. In this paper, by constructing a special data structure, called dictionary tree, we design a new algorithm to retrieve this reduct under dictionary order. As an example, a typical dictionary tree containing 4 attributes is shown in Figure 1.



Notes: The attribute set serial when accessing the dictionary tree by mid-root mode is $a, ab, abc, abcd, abd, \ldots, d$, which is ordered under dictionary order.

Fig. 1. A Dictionary Tree Containing 4 attributes

When the tree is accessed in mid-root traversing, the attribute set serial is $a, ab, abc, abcd, abd, \ldots, d$, which is in the lexical order. Moreover, we prove the correctness of the algorithm to construct the dictionary tree and algorithm to retrieve optimal reduct. However, when the algorithm is applied in real environment, it is hard to be carried out.

Yao presented a formal model of machine learning by considering user preference in [5]. This model combined internal information and external information seamlessly and could be extended to user preference of attribute sets. In that paper, Yao discussed many useful properties of user preference and presented two linear preference order, called *left-to-right* lexical order and *right-to left* one. In addition, two general algorithms are designed to retrieve corresponding optimal reduct under the two linear order from the deletion and addition strategy. However, the two algorithms were implemented based on two concepts–super reduct and partial reduct. How to judge whether a feature set is a super reduct or partial one is still a crucial issue while no reduct is available.

3 Basic Concepts and Theories

3.1 Information System

Definition 1. *An information table is a quadruple:*

$$\mathrm{IT} = (U, A, \{V_a | a \in A\}, \{I_a | a \in A\}) \tag{1}$$

Where

U *is a finite nonempty set of objects,*
A *is a finite nonempty set of attributes,*
V_a *is a nonempty set of values for* $a \in A$,
$I_a : A \to V_a$ *is an information function.*

For simplicity, we only consider information tables characterized by two finite sets: U and A, of which are objects and attributes, formally as (U, A). In general, an information table contains all available information and knowledge about objects under consideration, which are only perceived or measured by using attributes in A.

Given an information system (U, A), for two objects $x, y \in U$ and one attribute $a \in A$, if the value of x on a is equal to that of y on the same attribute, denoted as $x =_a y$, then we say that the two objects are indiscernible on a. For $B \subseteq A$, if for all attributes $b \in B$, $x =_b y$ holds, denoted as $x =_B y$, we call that the two objects are indiscernible on B.

In (U, A), the family of equivalence class with respect to A, denoted as $\mathrm{IND}(A)$, is defined as following.

$$\mathrm{IND}(A) = \{[x]_A | x \in U\} \tag{2}$$

where $[x]_A = \{y | y \in U, \text{ and }, x =_A y\}$ is the equivalence class containing x constructed by A.

$\mathrm{IND}(A)$ is the set of equivalence class, and can be seen as the classification of the given universe. For example, given $U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}$, if $\mathrm{IND}(A) = \{\{u_1, u_2, u_6, u_8\}, \{u_3, u_4, u_5, u_7\}\}$, then we can say that u_1, u_2, u_6 and

u_8 can be seen as the same class by available knowledge in (U, A) and u_3, u_4, u_5 and u_7 can be seen as another different class.

Provided an information system (U, A), $P, Q \subseteq A$, an functional dependency $P \to Q$ holds if equation (3) holds.

$$\forall u, v \in U, u =_P v \Rightarrow u =_Q v \tag{3}$$

Generally, an functional dependency $P \to Q$ has the following properties 1-3 [1]:

Property 1. $P \to Q \Rightarrow P \to q$, for all $q \in Q$; $(R_{inclusion})$

Property 2. $P \to Q \Rightarrow (P \cup V) \to Q$; $(R_{augment})$

Property 3. $P \to V \wedge V \to Q \Rightarrow P \to Q$; (R_{trans})

Definition 2. *In the given information system* (U, A)*, an attribute* $a \in A$ *is* dispensable*, if the following equation holds.*

$$\mathrm{IND}(A) = \mathrm{IND}(A - \{a\}) \tag{4}$$

Lemma 1. $a \in A$ *is dispensable in* (U, A) *if and only if* $A - \{a\} \to a$ *holds.*

Definition 3. *Given an information system* (U, A)*,* $P \subseteq A$ *is* dependent*, if any attribute* $p \in P$ *is not dispensable. Formally,* P *is dependent if and only if*

$$\forall p \in P, \mathrm{IND}(P) \neq \mathrm{IND}(P - \{p\}) \tag{5}$$

3.2 User Preference

In machine learning algorithms, it is implicitly assumed that all attributes are of the same importance from a user's point of view. Consequently, attributes are based solely on their characteristics revealed in an information system. This results in a simple model, which is easy to analyze. At the same time, without considering the semantic information of attributes, the model is perhaps unrealistic. A more applicable model can be built by considering attributes as non-equally important. This type of external information is normally provided by users in addition to the information system, and is referred to as user judgement or user preference [5].

Given an information system (U, A), for any $p, q \in A$, if p is preferred to q by user, we will simply denote it as $p \succ q$.

Also, how to acquire user preference is a crucial issue. In this paper, for clarity, we simply assume that a user can express preference on the entire attribute set precisely and completely, and this enable us to investigate the real issues without the interference of unnecessary constraints. For simplicity, we assume that any two attributes are preferred in user preference, that is to say, all attributes are assumed to be ordered in a linear order. Formally, $\forall p, q \in C$, either $p \succ q$ holds, or $q \succ p$ holds. Based on user preference on attribute, we can define user preference on the set of attributes as follows.

Definition 4. *Given two feature set* $P=\{p_1, p_2, \ldots, p_m\}$ *and* $Q=\{q_1, q_2, \ldots, q_n\}$ *such that* $p_1 \succ p_2 \succ \ldots \succ p_m$ *and* $q_1 \succ q_2 \succ \ldots \succ q_n$*, where* $\succ$ *is user preference on attributes. Let* $t = min\{m, n\}$*. We say that* P *precedes* Q*, written* $P \succ Q$ *if and only if either of the following two conditions holds:*

(1) *there exist a* $1 \leq i \leq t$ *such that* $p_j = q_j$ *for* $1 \leq j \leq i$ *and* $p_i \succ q_i$

(2) $a_i = b_i$ *for* $0 \leq i \leq t$ *and* $m < n$.

User preference defined above is in fact *left-to-right* lexical order in [5] and there are many applications in practice, such as dictionary order, and etc. If all attributes in (U, A) are linearly ordered in user preference, all reducts of (U, A) must be ordered in user preference and the reduct which is preferred to any other reduct is called *optimal reduct under user preference*. Particularly, if user preference is a linear order, the optimal reduct under assumed preference must be unique.

4 Optimal Reduct Under Preference

In this section, first we present two algorithms associated with optimal reduct, yet both of them have disadvantages. The feature set retrieved by the first algorithm is surely to be one reduct, but not the optimal one. The second algorithm will retain the better attributes, but cannot give birth to one reduct. After the two algorithms, we present the algorithm to retrieve optimal reduct in information system (U, A) and prove its correctness.

4.1 Algorithm to Retrieve Comparatively Optimal Reduct

First we will give an algorithm to retrieve comparatively optimal reduct. The reason why we call a comparatively optimal reduct is that the feature set we retrieve is a reduct but we cannot ensure it is the optimal one under preference. The algorithm is illustrated in *Algorithm 1*.

Data: an information system (U, A), while $A = \{a_1, a_2, \ldots, a_m\}$ satisfying the predefined preference such that $a_1 \succ a_2 \succ \ldots \succ a_m$.
Result: an reduct red of (U, A).
$red = A$;
i=m;
while $i \leq 1$ **do**
  if $red - \{a_i\} \rightarrow a_i$ **then**
    $red = red - \{a_i\}$;
  end
  else
    $i = i - 1$;
  end
end

Algorithm 1. Algorithm $COReductRetrIS(U, A)$

The strategy that Algorithm 1 adopts is deletion strategy, that is to say, the procedure of implementing the algorithm is to delete dispensable attributes one by one. However, the reduct retrieved is not sure to be the optimal one under user preference defined in this paper; this will be illustrated in the following example.

Example 1. Given an information table (U, A) in Table 1 as follows.

Table 1. An Information Table

U	a	b	c
1	1	2	0
2	1	2	0
3	1	1	1
4	0	0	0
5	0	0	0
6	0	3	1

In the given information system, there are two reducts: $\{a, c\}$ and $\{b\}$. According to Algorithm 1, we will get $\{b\}$ as the final output, but it is not the optimal reduct under user preference defined in this paper. However, it is optimal reduct under *right-to-left* lexical order defined in [5], which will be discussed in the expanded version of this paper.

4.2 Algorithm to Retrieve Optimal Feature

Since the reduct retrieved by the algorithm in the former section is not the optimal one, we will extend the algorithm so as to retrieve the optimal feature set, which may not certainly be the reduct. Based on the measure function $\alpha(P)$ on P which is increasing monotonously, the algorithm is illustrated in Algorithm 2 as follows, which can be seen as the revised version of algorithm in [7].

Algorithm 2 attempts to retain the important attributes, however, the final feature set is not sure to be one reduct.

4.3 Algorithm to Retrieve Optimal Reduct

After two attempted algorithms, we will design the algorithm to retrieve the optimal reduct for the given information system (U, A) as follows.

Now let us prove the correctness of the algorithm.

Proof. In order to prove the correctness of the designed algorithm, there are three problems to be explained:

(1) $\mathrm{IND}(redr) = \mathrm{IND}(A)$;
(2) $redr$ is dependent;
(3) There is no other reduct $redr' \subseteq A$ such that $redr' \succ redr$.

Now we will prove the above three sub-problems one by one.

Data: An decision information table (U, A),while $A = a_0, a_1, \ldots, a_n$.
Result: A feature set $GloBalFea$.
$GlobalFea = \Phi$;
L1:
for $(i = 1\ to\ n\ do\ )$ **do**
  $TempGlobalFea = \{S_q | \alpha(S_q) = max\{\alpha(S_p), p = 1..n\}\}$
  if $(TempGlobalFea = GlobalFea)$ **then**
    Return $GlobalFea$;
    End Algorithm;
    else
      $GlobalFea = TempGlobalFea$
      for $(j = 1..n)$ **do**
        $S_j = Sj \bigcup GlobalFea$
      end
      Goto L1;
    end
  end
end
return $GlobalFea$;

Algorithm 2. Algorithm for optimal features

Data: an information system (U, A), while $A = \{a_1, a_2, \ldots, a_m\}$ satisfying the predefined preference such that $a_1 \succ a_2 \succ \ldots \succ a_m$.
Result: an reduct $red \subseteq A$.
i=1;
while $i \leq m$ **do**
  $redr = \{a_i\}$;
  $j = i + 1$;
  while $j \leq m$ **do**
    if $redr \not\rightarrow a_j$ **then**
      if $\forall a_p \in redr$, *such that* $(red - \{a_p\}) \cup \{a_j\} \not\rightarrow a_p$ **then**
        $redr = redr \cup \{a_j\}$;
        if IND($redr$) = IND(A) **then**
          return $redr$, algorithm end.
        end
      end
    end
    $j = j + 1$;
  end
  $i = i + 1$;
end

Algorithm 3. Algorithm for optimal reduct

(1) According to the step which can end the algorithm in the algorithm, the first formula is apparent;

(2) In the algorithm, the final reduct $redr$ is produced by adding attributes one by one. Therefore, we can prove that it is dependent in the following two steps.

a.) In the initial step, since $redr = \{a_i\}$, and there is only one attribute, obviously it is dependent;

b.) Suppose that in the q^{th} step, $redr = \{a_{k_1}, a_{k_2}, \ldots, a_{k_q}\}$ satisfies the dependent property, that is to say, any attribute a_p in $redr$ satisfies $\mathrm{IND}(redr) \neq \mathrm{IND}(redr - \{a_p\})$. According to the algorithm, if one attribute a_j is added to $redr$, it must satisfy the following two properties:

• $redr \not\rightarrow a_j$, which ensures that a_j is not dispensable at the $(q+1)^{th}$ step based on the result of ***Lemma 1***;

• $\forall a_u \in redr, (redr - \{a_u\}) \cup a_j \not\rightarrow a_u$, which ensures that all attributes in current attribute set $redr$ are not dispensable in the new attribute set $redr \cup \{a_j\}$ retrieved in $(q+1)^{th}$ step based on the result of ***Lemma 1***.

Therefore, the attribute set $redr \cup a_j$ retrieved in $(q+1)^{th}$ step is dependent.

(3) Suppose the reduct retrieved by applying the designed algorithm is $redr = \{a_{k_1}, a_{k_2}, \ldots, a_{k_u}\}$. If it is not the optimal one, there must be another reduct $redr' = \{a_{l_1}, a_{l_2}, \ldots, a_{l_v}\}$ such that $redr \succ redr'$.

According to Definition 4, $redr \succ redr'$ holds must satisfy either of the following two conditions, from which we will prove the sub-theorem.

a.) $l_v \leq k_u$ and $\forall q \leq l_v, a_{k_q} = a_{l_q}$.

According to the designed algorithm, this case will not occur in the designed algorithm. For if $redr' = \{a_{l_1}, a_{l_2}, \ldots, a_{l_v}\}$ and $redr = \{a_{k_1}, a_{k_2}, \ldots, a_{k_u}\}$ are two reducts, according to description in this case, $\forall q \leq l_v, a_{k_q} = a_{l_q}$ and $l_v < k_u$, which is to say, $redr' \subset redr$, which is contradict to the definition of reduct.

b.) $\exists w,$ such that $\forall q \leq w, a_{k_q} = a_{l_q},$ and $a_{l_{q+1}} \succ a_{k_{q+1}}$.

Supposing that in some step of the implement of the algorithm, we retrieve the attribute set is $redr = \{a_{k_1}, a_{k_2}, \ldots, a_{k_q}\}$, if we can explain that $redr'$ will be the reduct we retrieved in the algorithm, then we can prove the sub-question.

First we can get the conclusion that current attribute set $redr$ is not a reduct, otherwise the algorithm will stop. Then according to the hypothesis $a_{l_{q+1}} \succ a_{k_{q+1}}$, the two attributes must in the same order in given preference. In the procedure of the algorithm, any attribute a_j between a_{l_q} and $a_{l_{q+1}}$ will not be added to $redr$, either $redr \rightarrow a_j$ holds, or there exists better attribute $a_q \in redr$, such that $(redr - \{a_q\}) \cup \{a_j\} \rightarrow a_q$ holds.

Since attributes are added one by one in the order of given preference, $a_{l_{q+1}}$ will be faced ahead of $a_{k_{q+1}}$. Furthermore, for current attribute set $redr$ is not the reduct and there exists a reduct $\{a_{l_1}, a_{l_2}, \ldots, a_{l_v}\}$ containing $a_{l_{q+1}}$ according to the hypothesis, therefore, the following equation $\mathrm{IND}(redr) \neq \mathrm{IND}(redr \cup \{a_{l_{q+1}}\})$ holds. From this *Step* , $a_{l_{q+1}}$ will not be added in $redr$ if and only if $\nexists P \subseteq \{a_{l_{q+2}}, \ldots, a_m\}$, such that $\mathrm{IND}(redr \cup P \cup \{a_{l_{q+1}}\}) \neq \mathrm{IND}(A)$, of course this is impossible, for there exists a reduct $redr'$.

From the above three aspects, we can illustrate that the reduct retrieved in the algorithm is the one subject to preference order of attributes.

4.4 Complexity Analysis

Suppose $|A| = m$ and $|U| = n$. The worst case is that the reduct is the last attribute. In such case, when $i = 1$, we will add all other attributes one by one. We will judge all functional dependencies in 2-attribute set, 3-attributes, ..., m-attribute set. When judging in 2-attribute set, we will test whether two functional dependencies only containing one attribute in the right hold, the complexity of each is $2 * n^2$, so finding all functional dependencies in 2-attribute set is $(2n)^2$. Also, finding all functional dependencies in 3-attribute set, 4-attribute set, ..., m-attribute set are of $(3n)^2, (4n)^2, \ldots, (mn)^2$ complexity. Therefore, the complexity to judge whether the first attribute is included in the reduct is $(2n)^2 + (3n)^2 + \ldots + (mn)^2$.

In the same mode, judging whether the second attribute is included in the reduct is of the following complexity: $(2n)^2+(3n)^2+\ldots+((m-1)n)^2$,..., judging the $(m-1)^{\text{th}}$ attribute is of $(2n)^2$ complexity.

Therefore, the total complexity of this algorithm is
$(2n)^2 + \ldots + (mn)^2 + (2n)^2 + \ldots + ((m-1)n)^2 + \ldots + (2n)^2 = O(n^2m^4)$

5 Conclusions

This paper investigates how to retrieve the reduct subject to preference order of attributes in the information system. However, how good is the reduct subject to preference order of attributes for both knowledge representation and prediction? Algorithms should take into account not only the length and the preference order of attributes, but also the intended application of the obtained reduct.

References

1. Dan, Simovici, A.: Relational Database System. Academic Press, San Diego (2002)
2. Hu,K., Diao,L., Lu,Y., Shi,C.: Sampling for approximate reduct in very large databases. URL: `//citeseer.ist.psu.edu/587308.html`
3. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11(5), 341–356 (1982)
4. Yao, Y., Zhao, Y., Wang, J.: On reduct construction algorithms. In: Wang, G.-Y., Peters, J.F., Skowron, A., Yao, Y. (eds.) RSKT 2006. LNCS (LNAI), vol. 4062, pp. 297–304. Springer, Heidelberg (2006)
5. Yao,Y., Zhao,Y., Wang,J., Han,S.: A model of Machine Learning Baed on User Preference of Attributes (to be published)
6. Zhang, X., Zhang, F., Li, M., Wang, N.: Research of Optimal reduct under preference. Computer Engineering and Design 26, 2103–2106 (2005)
7. Zhao, Y., Zhang, X., Jia, S., Zhang, F.: Applying PSO in finding useful features. In: Wang, G.-Y., Peters, J.F., Skowron, A., Yao, Y. (eds.) RSKT 2006. LNCS (LNAI), vol. 4062, pp. 580–585. Springer, Heidelberg (2006)