Vol. 39 No. 9 Transactions of Information Processing Society of Japan Sep. 1998

Regular Paper

Soft Techniques for Rule Discovery in Data

Ning Zhong,[†] Juzhen Dong,[†] Shinichi Fujitsu[†]
and Setsuo Ohsuga[††]

This paper introduces a new approach to rule discovery in data with uncertainty and incompleteness, using soft techniques. This approach is based on a Generalization Distribution Table (GDT) and its network representation. Inductive learning methods are discussed from the viewpoint of the value of information. The key features of the approach are as follows: (1) both the formal value and the semantic value of information are considered, (2) the uncertainty of a rule, including its ability to predict possible instances, can be explicitly represented in the strength of the rule, and (3) biases can be flexibly selected and background knowledge can be used in the discovery process for constraint and search control.

1. Introduction

Over the last two decades, many researchers have investigated inductive methods for learning *if-then* rules and concepts from instances. According to the value of information, these methods can be divided into two types. The first type is based on the *formal* value of information; that is, the real meaning of data is not considered in the learning process. ID3 and Prism are typical methods of this type[1),11)]. Although *if-then* rules can be discovered by using the methods, it is difficult to use background knowledge in the learning process. The other type of inductive methods is based on the *semantic* value of information; that is, the real meaning of data must be considered by using some background knowledge in the learning process. Dblearn is a typical method of this type[4)]. It can discover rules by means of background knowledge represented by concept hierarchies, but if there is no background knowledge, it can do nothing. The question is, *"how can both the formal value and the semantic value be considered in a discovery system?"*. Unfortunately, so far we have not seen any inductive method that can consider both the formal value and the semantic value of information.

To solve this problem, we would like to discuss inductive methods from another point of view, namely, the style of information processing. From this viewpoint, inductive methods can be divided into two styles: *top-down* and *bottom-up*. Usually, top-down methods such as ID3 and Prism can learn rules very fast, but it is difficult to process data change, to use background knowledge in the learning process, and to perform in a parallel-distributed cooperative mode. On the other hand, bottom-up methods such as *version-space*[9),10)] and *back-propagation*[13)] are incremental (or semi-incremental); that is, learning of a concept is possible not only when instances are input simultaneously but also when they are given one by one.

In *version-space*, proposed by Mitchell as a symbolic approach, the learning task is to search a hypothesis space, subject to constraints imposed by the training instances, to determine plausible generalizations. Although version-space was a landmark project that led to some success and provides a good background for our research, no satisfactory solutions have yet been found to certain issues in real-world applications such as the following:

- How can rules be learned in an environment with uncertainty and incompleteness?
- How can unseen instances be predicted, and how can the uncertainty of a rule, including its ability to predict, be represented explicitly?
- How can biases be selected and altered dynamically for constraint and search control?
- How can the use of background knowledge be selected according to whether background knowledge exists or not?

Back-propagation, proposed by Rumelhart et al.[13)] as a connectionist approach, has proved to

† Department of Computer Science and Systems Engineering, Faculty of Engineering, Yamaguchi University
†† Department of Information and Computer Science, School of Science and Engineering, Waseda University

be a powerful and general technique for machine learning. However, it also has disadvantages, such as the following:

- Without the ability to explain its decisions, it is hard to be confident in the reliability of a network that addresses a real-world problem. Although it learns, it does not provide us with a theory about what it has learned. It is a simple black box that gives an answer but provides no clear idea as to how it arrived at these answers[15];
- It is rather difficult to incorporate background knowledge into the back-propagation networks so that they can learn better;
- The training time needed for back-propagation networks to obtain a satisfactory result is usually long. This is because learning in such networks is essentially based on multiple passes over the training data set.

In view of these disadvantages, back-propagation has not been considered well suited for discovering *if-then* rules in databases.

In this paper, we propose a new rule discovery approach based on a Generalization Distribution Table (GDT) and its network representation, using soft techniques. The key features of this approach are that both the formal value and the semantic value of information are considered, the uncertainty of a rule, including its ability to predict possible instances, can be explicitly represented in the strength of the rule, and biases can be selected for constraint and search control. We first introduce the GDT as a hypothesis search space for generalization, discuss how to select biases based on the three components of the GDT, and describe how the GDT can be represented by networks. We then describe a discovery process based on the network representation, and discuss with the help of an example how to use background knowledge in the discovery process for constraint and search control.

2. GDT and Its Network Representation

This section describes the GDT and its network representation as the basis of our methodology.

2.1 GDT

The central idea of our methodology is to use a variant of a transition matrix, called the *Generalization Distribution Table* (*GDT*), as a hypothesis search space for generalization, in which probabilistic relationships between concepts and instances over discrete domains are represented[19),21)].

We define a GDT for knowledge discovery as consisting of three components: *possible instances*, *possible generalizations* for instances, and *probabilistic relationships* between possible instances and possible generalizations.

The *possible instances*, which are represented in the top row of a GDT, are all possible combinations of attribute values in a database. The number of the possible instances is $\prod_{i=1}^{m} n_i$, where m is the number of attributes and n is the number of different attribute values in each attribute.

The *possible generalizations* for instances, which are represented in the left column of a GDT, are all possible cases of generalization for all possible instances. The number of possible generalizations is $\prod_{i=1}^{m}(n_i+1) - \prod_{i=1}^{m} n_i - 1$.

The *probabilistic relationships* between the possible instances and the possible generalizations, which are represented in the elements G_{ij} of a GDT, are the probabilistic distribution for describing the strength of the relationship between every possible instance and every possible generalization. The prior distribution (time $t = 0$) is equiprobable, if we do not use any prior background knowledge for creating the GDT. Thus, it is generated by the following equation:

$$G_{ij} = p(PI_j|PG_i) = \begin{cases} \frac{1}{N_{PG_i}} & \text{if } PG_i \supset PI_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where PI_j is the jth possible instance, PG_i is the ith possible generalization, and N_{PG_i} is the number of the possible instances satisfying the ith possible generalization, that is,

$$N_{PG_i} = \prod_{j}^{m} n_j, \quad (2)$$

where $j = 1, \ldots, m$, and $j \neq$ the attribute contained by the ith possible generalization (i.e., j just contains the attributes expressed by the wild card, as shown in **Table 1**).

Thus, in our approach, the basic process of hypothesis generation is to generalize the instances observed in a database by searching and revising the GDT. Here, we need to distinguish two kinds of attributes: *condition* attributes and *decision* attributes (sometimes called class attributes) in a database. Condition attributes as possible instances are used to create the GDT, but decision attributes are not. The deci-

Table 1 GDT for a sample database.

	a0b0c0	a0b0c1	a0b1c0	a0b1c1	...	a1b2c1
*b0c0	1/2				...	
*b0c1		1/2			...	
*b1c0			1/2		...	
*b1c1				1/2	...	
*b2c0					...	
*b2c1					...	1/2
a0*c0	1/3		1/3		...	
a0*c1		1/3		1/3	...	
a1*c0					...	
a1*c1					...	1/3
a0b0*	1/2	1/2			...	
a0b1*			1/2	1/2	...	
a0b2*					...	
a1b0*					...	
a1b1*					...	
a1b2*					...	1/2
**c0	1/6		1/6		...	
**c1		1/6		1/6	...	1/6
b0	1/4	1/4			...	
b1			1/4	1/4	...	
b2					...	1/4
a0**	1/6	1/6	1/6	1/6	...	
a1**					...	1/6

Table 2 Sample database.

No	A	B	C	D
1	a0	b0	c1	y
2	a0	b1	c1	y
3	a0	b0	c1	y
4	a1	b1	c0	n
5	a0	b0	c1	n
6	a0	b2	c1	y

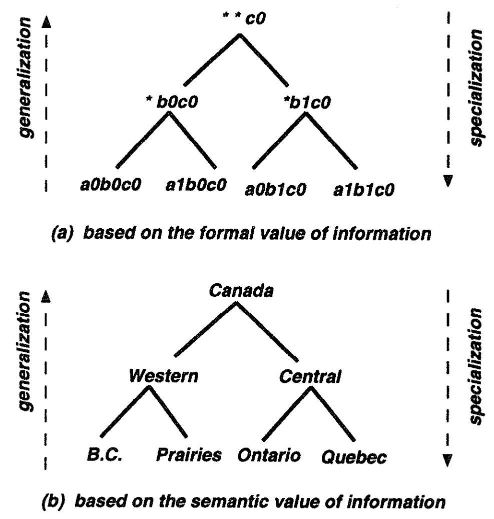


Fig. 1 Two types of generalization.

sion attributes are normally used to distinguish which concept (class) should be described in a rule. Usually a single decision attribute is all that is required.

Table 1 shows an example of a GDT, generated by using three condition attributes, A, B, and C, in the sample database shown in **Table 2**, where $A \in \{a_0, a_1\}$, $B \in \{b_0, b_1, b_2\}$, and $C \in \{c_0, c_1\}$. For example, the real meaning of these condition attributes can be respectively assigned as *Weather*, *Temperature*, *Humidity* in a weather forecast database, or *Temperature*, *Cough*, *Headache* in a medical diagnosis database. Attribute D in Table 2 is used as a *decision* attribute. For example, the real meaning of the decision attribute can be assigned as *Wind* or *Flu* according to the assigned condition attributes.

Furthermore, "$*$" in Table 1, which specifies a wild card, denotes a generalization for instances. For example, the generalization $*b_0c_1$ means that the attribute A is unimportant for describing a concept. In this example, the number of possible instances in the GDT is 12, and the number of possible generalizations is 23.

Here we would like to distinguish between two types of generalization from the viewpoint of the value of information, as described in Section 1. The first type is based on the formal value of information, as shown in **Fig. 1** (a) (i.e., a generalization and specialization hierarchy such as *version-space*); the other type is based on the semantic value of information, as shown in Fig. 1 (b) (i.e., background knowledge represented by a concept hierarchy). Basically, the generalization used in a GDT belongs to the first type, but the second type can also be selected for constraint and preprocessing. In particular, interesting rules occur at a relatively high concept level in many applications, but data in databases often contain detailed information at primitive concept levels[2),18)]. The data at a primitive concept level can be first generalized into a relatively high concept level by using concept hierarchies like the one shown in Fig. 1 (b). This generalized database is then used to create a GDT.

2.2 Biases

Since our approach is based on the GDT, rule discovery can be constrained by three types of bias, corresponding to the three components of the GDT defined in Section 2.1.

The first type of bias is related to the possible generalizations in a GDT. It is used to decide which concept description should be considered first. To obtain the best concept descriptions, all possible generalizations should be considered, but not all of them need to be considered at the same time. We divide possible generalizations (concept descriptions) into several levels of generalization according to the number of wild cards in a generalization: the greater the number of wild cards, the higher the level. For example, all possible generalizations shown in Table 1 are divided into two levels of generalization:

$$Level_1 \in \{*b_0c_0, *b_0c_1, \dots, a_1b_2*\}$$
$$Level_2 \in \{**c_0, **c_1, \dots, a_1**\}.$$

Thus, we can see that any generalization in a lower level is properly contained by one or more generalizations in an upper level. As the default, our approach prefers more general concept descriptions in an upper level to more specific ones in a lower level. However, if necessary, we can use a meta control to alter the bias so that more specific descriptions are preferred to more general ones. This issue will be further discussed in Section 3.2.

The second type of bias is related to the probability values denoted in G_{ij} of a GDT. It is used to adjust the strength of the relationship between an instance and a generalization. If no prior background knowledge as a bias is available, as the default, we create the prior distribution of a GDT by considering that the occurrence of all possible instances is equiprobable, as shown in Table 1. However, a bias such as background knowledge can be used during the creation a GDT. The prior distributions will be dynamically updated according to the real data in a database, and the posterior distributions will converge to the real data. This issue will be further discussed in Section 3.2.

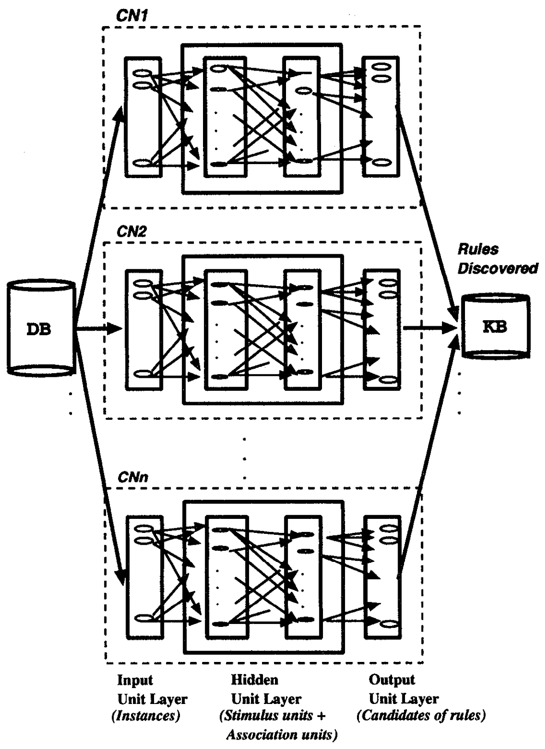

CN1
CN2
CNn
DB
KB
Rules Discovered
Input Unit Layer (Instances)
Hidden Unit Layer (Stimulus units + Association units)
Output Unit Layer (Candidates of rules)

Fig. 2 Network representation of the GDT for knowledge discovery.

The third type of bias is related to the possible instances in a GDT. In our approach, the strength of the relationship between every possible instance and every possible generalization depends to a certain extent on how the possible instances are created and defined. This issue will be further described in Section 4.

2.3 Representing the GDT by Networks

We know that the connectivity of a network represented by a network drawing (a network representation, for short) can be naturally represented in a matrix (a matrix representation, for short)[14)]. In contrast, a matrix representation can obviously be changed into a network representation. Since the GDT is a variant of a transition matrix, it can be represented by networks[20)].

Figure 2 shows a network representation of the GDT for our purpose. We can see that the networks consist of three layers: the *input unit* layer, the *hidden unit* layer, and the *output unit* layer. A unit that receives instances from a database is called an *input unit*, and the number of input units corresponds to the number of condition attributes. A unit that receives a result of learning in a hidden unit, which is used as one of the rule candidates discovered, is called an *output unit*. A unit that is neither input nor output unit is called a *hidden unit*. Let the hidden unit layer be further divided into *stimulus units* and *association units*. Since the *stimulus units* are used to represent the possible instances, like the top row in a GDT, and the *association units* are used to represent the possible generalizations of instances, like the left column in a GDT, they are also called *instance units* and *generalization units*, respectively. Furthermore, there is a link between a stimulus unit and an association unit if the association unit represents a possible generalization of a possible instance represented by the stimulus unit. Moreover, the probabilistic relationship between a possible instance and a possible generalization is represented in the weight of the link, and the initial weights are equiprobable, like the G_{ij} of an initial GDT[☆].

Here, we need to distinguish two kinds of links: *excitatory links* and *inhibitory links*. A link is *excitatory* if the stimulus unit corresponding to the link was activated by an instance. In contrast, a link is *inhibitory* if the stimulus unit corresponding to the link was not activated by an instance. Let $W^t_{exc}(i,j)$ be the weight of the excitatory link that is related to the ith stimulus unit activated by an instance and the jth association unit (i.e., the jth generalization) over the time period from 0 to t, and let $W^t_{inh}(k,j)$ be the weight of the inhibitory link that is related to the kth stimulus unit (i.e., the kth possible instance) and the jth association unit (i.e., the jth generalization) over the time period from 0 to t. Furthermore, let $O(j)$ be the output of the jth association unit. **Figure 3** shows the relationship between stimulus units and an association unit with different links.

The network representation of the GDT provides many advantages; for example, the computation of weights can be done in a parallel-distributed mode, a large GDT can be conveniently decomposed into smaller ones, and space can be saved, because we only need to

[☆] For simplicity, we assume that prior background knowledge is not currently used.

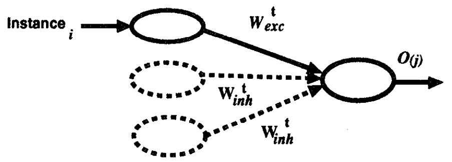


Fig. 3 Relationship between stimulus units and an association unit with different links.

record the values related to excitatory links. Here we would like to stress that

- In the representation, a network is only used to represent the possible generalizations for the possible instances in one of the levels of generalization that were defined in Section 2.2. Hence, we need to create n networks if there are n levels of generalization, as shown in Fig. 2. Thus, the number of networks that should be created, N_{cns}, is
$$N_{cns} = N_{attrs} - 2, \qquad (3)$$
where N_{attrs} is the number of attributes in a database. The merit of this approach is obviously that rule candidates can be generated in parallel. For example, we need to use two networks for the sample database shown in Table 2.
- Since the creation of the networks is based on the GDT, the meaning of every unit in the networks can be explained clearly. Thus, not only the trained results in the networks are explicitly represented in the form of *if-then* rule with *strength*, but background knowledge can also be used for dynamically revising and changing the network representation in the discovery process. Hence, the networks are called *knowledge-oriented* networks.
- The networks do not need to be explicitly created in advance. They can be embodied in the search algorithm, and we only need to record the weights related to stimulus units activated by instances. In other words, although the weights of inhibitory links also need to be calculated and revised in principle, it is not necessary for our rule discovery process currently. In fact, the instances collected, in many real-world databases, are generally a small subset of all possible instances. Hence, the real size of the networks is much smaller than the size of the corresponding GDT.

3. Rule Discovery

Building on the preparatory in Section 2, this section describes the fundamental methodology of rule discovery based on the network representation of a GDT.

3.1 Rule Representation

In our approach, the discovered rules are typically expressed in the form

$X \to Y$ with S.

That is, "if X then Y with strength S", where X denotes the conjunction of conditions that a concept must satisfy, Y denotes the concept that the rule describes, and S is a "measure of the strength" with which the rule holds.

Sometimes, *contradictory rules*, which have the same condition but describe different concepts with nearly the same strength, may be generated. For example, the rules,

$$Weather\,(clear) \wedge Humidity\,(low) \to Wind\,(yes) \quad \text{with } P = 0.25,$$

and

$$Weather\,(clear) \wedge Humidity\,(low) \to Wind\,(no) \quad \text{with } P = 0.25,$$

are contradictory rules, because they have same conditions $Weather\,(clear) \wedge Humidity\,(low)$ and describe different concepts $Wind\,(yes)$ and $Wind\,(no)$ with same strength $P = 0.25$. The comments about rule strength will be further described in Sections 3.2 and 3.3.

3.2 The Discovery Process

We have developed an effective method for incremental rule discovery based on the network representation of the GDT described in Section 2.3. One advantageous feature of our approach is that every instance in a database is searched only once, and if the data in a database are changed (added to, deleted, or updated), we need only to modify the networks and the discovered rules related to the changed data; the database is not searched again. That is, unlike back-propagation networks, our approach does not need multiple passes over the training data set. We argue that this is a very important feature for discovering rules in very large databases. **Figure 4** gives an overall view of the basic process of discovering rules in databases. We can see that the process is divided into two main stages.

The *first* stage is that of deciding the form of the network representation and revising the

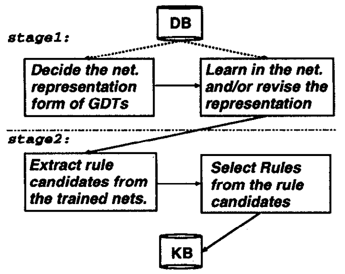

DB
stage1:
Decide the net. representation form of GDTs
Learn in the net. and/or revise the representation
stage2:
Extract rule candidates from the trained nets.
Select Rules from the rule candidates
KB

Fig. 4 Block diagram of the basic process of rule discovery.

weights by learning in every network in parallel. The main steps are as follows:

Step 1 : Create the initial networks according to the attribute values in a database. Before learning (i.e., at time $t = 0$), all the units and links are in the inhibitory state and

$$W^0(i,j) = \frac{1}{N^0_{il}(j)}, \tag{4}$$

where $N^0_{il}(j)$ is the number of inhibitory links related to the jth association unit.

Step 2 : Put an instance obtained from a database[☆] in the input unit layer and set $t = t + 1$. As a result of this operation, the stimulus units corresponding to the instance are activated.

Step 3 : Revise the weights of the corresponding excitatory and inhibitory links in Eqs. (5) and (6), respectively[☆☆]:

$$W^t_{exc}(i,j) = \frac{1 + \alpha N^t_{ins\text{-}new}(i)}{N^t_{il}(j) + N^t_{el}(j) + \alpha N^t_{ins\text{-}rel}(j)}, \quad (t > 0), \tag{5}$$

$$W^t_{inh}(k,j) = \frac{1 - \sum_{i:activated} W^t_{exc}(i,j)}{N^t_{il}(j)}, \quad (t > 0), \tag{6}$$

where $N^t_{el}(j)$ and $N^t_{il}(j)$ are respectively the number of *excitatory links* and *inhibitory links* related to the jth association unit over the time period from 0 to t; $N^t_{ins\text{-}rel}(j)$ is the number of instances that have already been obtained from the input unit layer and that are related to the jth association unit over the time period from

☆ Only the values belonging to condition attributes are considered at this time.

☆☆ Although in principle the weights of inhibitory links, $W^t_{inh}(k,j)$ need to be calculated and revised in Eq. (6), it is not necessary at present for our rule discovery process. However, for theoretical completeness, we describe here how to calculate and revise the weights of inhibitory links.

0 to t; $N^t_{ins\text{-}new}(i)$ is the number of instances that are the same as the newly input instance at time t and that are related to the ith stimulus unit over the time period from 0 to t; and α is a constant representing the learning rate for controlling the speed at which $W^t_{exc}(i,j)$ approaches 1.

Step 4 : If there are still instances that have not been processed, then go back to *Step* 2. Otherwise, regard the $W^t_{exc}(i,j)$ revised over the time period from 0 to t as the posterior distribution, $t = T$, and start the *second* stage of the process.

The *second* stage of the discovery process is that of generating *if-then* rules, which is based upon the posterior distribution learned in the *first* stage. The main steps are as follows:

Step 1 : Extract the rule candidates from the association units with excitatory links and compute the strengths of the rule candidates in Eq. (7). Then put them in the output unit layer.

$$O_c(j) = \left(\sum_{i:activated} W^T_{exc}(i,j) \right) \times \frac{N^T_{ins\text{-}rel_c}(j)}{N^T_{ins\text{-}rel}(j)}, \qquad (7)$$

where c is the class number, $N^T_{ins\text{-}rel}(j)$ is the number of instances related to the jth generalization over the time period from 0 to T, and $N^T_{ins\text{-}rel_c}(j)$ is the number of instances that are related to the jth generalization and belong to the class c over the time period from 0 to T.

Step 2 : Select *if-then* rules from the rule candidates generated in *Step* 1 by heuristic searching.

Here we would like to stress that our approach is very soft. There are several possible ways of selecting rules. For example,

- Selecting the rules that contain the most instances
- Selecting the rules in the highest possible levels
- Selecting the rules with the greatest strengths.

The method of rule selection used here can be briefly described as follows:

- Adjust the threshold value to distinguish contradictory rules and noise dynamically.
- Rules that contain fewer instances are deleted if a rule that contains more instances exists.
- Since we prefer simpler results of generalization (i.e., more general rules), we first consider the rules corresponding to an upper level of generalization, and delete redundant rules between different levels.
- Rules with greater strengths are first selected as the real rules, and contradictory rules are deleted.

In Section 4, we will further discuss how to use background knowledge as a bias for rule selection.

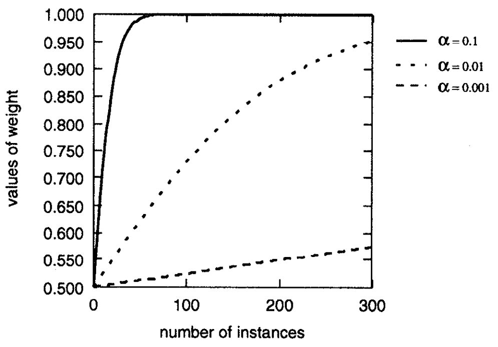


Fig. 5 Controlling the learning rate by adjusting α.

Here, we would like to further explain Eqs. (5) and (6). Equation (7) will be explained in Section 3.3. Equations (5) and (6) can be considered variants of the *Hebbian* learning rule [5]. The basic ideas can be summarized as follows:

- If an instance is observed, the weight that represents the probabilistic relationship between the observed instance and its possible generalizations should be strengthened in Eq. (5), and the greater the number of identical instances are observed, the more the weight is strengthened. In contrast, weights that represent probabilistic relationships between unobserved possible instances and their possible generalizations should be weakened in Eq. (6).
- α in Eq. (5) is a constant representing the learning rate. In our applications, we may in some cases need to approach 1 at a lower speed by setting a lower value of α, and in other cases, we may need to approach at a higher speed by setting a higher value of α. In other words, α is a bias for controlling the search and rule discovery. **Figure 5** shows an example of how to control the learning rate by adjusting α. For simplicity, in the rest of the paper, we assume that $\alpha = 1$.
- If we do not consider that changes in data (e.g., addition to, deletion of, or updat-

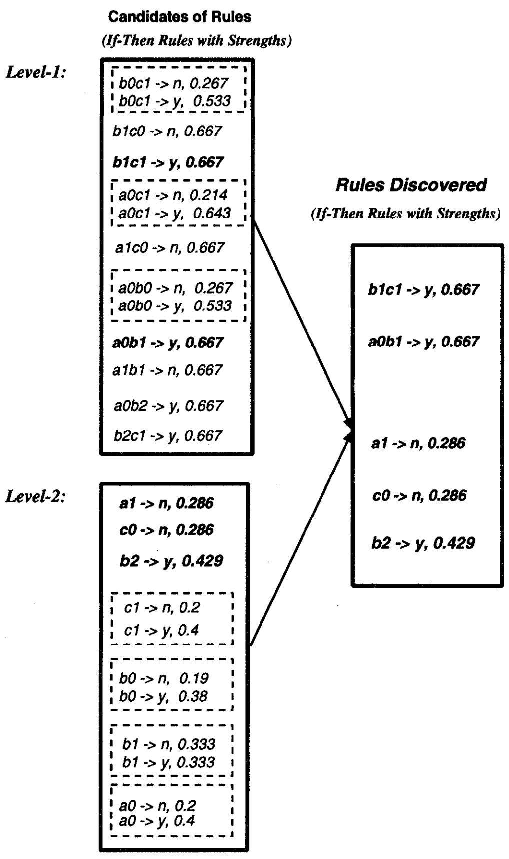


Fig. 6 Rules discovered in a sample database.

ing of the data in a database) may cause modification of the network representation, Eqs. (5) and (6) can be represented in an iterative form such as

$$W^t_{exc}(i,j) = \frac{W^{t-1}_{exc}(i,j) + \alpha W^{t-1}_{exc}(i,j)}{1 + \alpha W^{t-1}_{exc}(i,j)}, \quad (t > 0), \qquad (8)$$

$$W^t_{inh}(k,j) = \frac{W^{t-1}_{inh}(k,j)}{1 + \alpha W^{t-1}_{exc}(i,j)}, \quad (t > 0). \qquad (9)$$

However, the iteration relationship between W^t and W^{t-1} will not hold if the networks are dynamically modified along with data change. Hence, Eqs. (5) and (6) are used in our system.

Figure 6 shows examples of rules discovered in the sample database in Table 2 by using the above method. It includes the rule candidates extracted from the networks and *if-then* rules selected from the candidates. In this example, we prefer more general rules. Thus, we first consider selecting the rules from *Level-2* of the rule candidates. Since the rules in *Level-2* that are framed by dotted lines, such as

$B(b_1) \rightarrow D(n)$ with $P = 0.333$

and

$B(b_1) \rightarrow D(y)$ with $P = 0.333$,

are contradictory, they are rejected, and the rules, contained by the contradictory rules described above, in *Level-1*,

$r_{0.1}: B(b_1) \wedge C(c_0) \rightarrow D(n)$ with $P = 0.667$,
$r_{0.2}: B(b_1) \wedge C(c_1) \rightarrow (y)$ with $P = 0.667$,
$r_{0.3}: A(a_0) \wedge B(b_1) \rightarrow D(y)$ with $P = 0.667$,
$r_{0.4}: A(a_1) \wedge B(b_1) \rightarrow D(n)$ with $P = 0.667$,

are considered. Furthermore, since the rules $r_{0.1}$ and $r_{0.4}$ are respectively contained by rules in *Level-2*, namely,

$r_{0.5}: C(c_0) \rightarrow D(n)$ with $P = 0.266$

and

$r_{0.6}: A(a_1) \rightarrow D(n)$ with $P = 0.266$,

the rules $r_{0.1}$ and $r_{0.4}$ are deleted. In other words, $r_{0.2}$ and $r_{0.3}$ are selected as two of the discovered rules, as shown in Fig. 6.

3.3 Rule Strength and Unseen Instances

One of the main advantages of our approach is that it can predict unseen instances, because the search space based on the GDT considers all possible combinations of instances. Thus, according to the quantity and the quality of observed instances, and considering unseen instances in the discovery process, the uncertainty of a rule, including its ability to predict unseen instances, can be explicitly represented in the strength P of this rule given by Eq. (7). We can see that Eq. (7) can be divided into two parts. The first part is

$$\textstyle\sum_{i:activated} W^T_{exc}(i,j),$$

which is used to describe the strength of a rule, including its ability to predict unseen instances. For example, after learning from the database shown in Table 2 by using the method given in Section 3.2, we obtain the rule

$$r_{1.1}: A(a_0) \wedge B(b_1) \rightarrow D(y) \text{ with } P = 0.667.$$

This rule is acquired by generalizing the instance $a_0b_1c_1$. The reason why the strength of the rule is 0.667 is that we only observed one of two possible instances, which can be generalized into $A(a_0) \wedge B(b_1)$. In other words, another possible instance $a_0b_1c_0$ is not found in the database. Hence, we first revise the weights related to the generalization $A(a_0) \wedge B(b_1)$ into 0.667 and 0.333 in Eqs. (5) and (6) respectively, and then obtain the strength of the rule $r_{1.1}$ in the first part of Eq. (7). Furthermore, if the instance $a_0b_1c_0$ is added in the database shown in Table 2, then we can revise the weights into 0.5 in Eqs. (5) and (6) respectively, and then increase the strength of the rule to 1 in the first part of Eq. (7).

We argue that the ability to predict unseen instances is an important function for discovering rules in real-world databases. In most cases, the set of instances collected in a database represents only a subset of all possible instances. This is reasonable, because we expect to learn rules without first collecting every possible instance (just as physicians learn how to diagnose diseases without first having seen every possible patient)[16]. However, it also means that the learning task is ill-posed, because for previous inductive approaches, without some other source of constraint, there is no way of knowing instances of a concept that has never been observed. Our approach, based on the GDT and its network representation, provides a capability for predicting unseen instances and for explicitly representing the strength of a rule that includes the prediction. In other words, our approach tries to find descriptions of concepts not only from the instances observed during learning but also from unseen instances.

The second part of Eq. (7) is

$$\frac{N^T_{ins\text{-}rel_c}(j)}{N^T_{ins\text{-}rel}(j)}.$$

This part is significant only when a rule is acquired from the instances or their generalizations with different classes. It shows the quality of classification, that is, how many identical instances or generalizations of them as conditions that the rules must satisfy can be categorized as belonging to the same class. We distinguish the following three cases according to the ratio of the second part of Eq. (7)[☆]:

- If the ratio is 1, there are no noisy data. That is, all instances as conditions that the rules must satisfy are categorized as belonging to the same class. In this case, the second part of Eq. (7) does not influence the strength of a rule.
- If the ratio is close to 0.5, then the rules are contradictory or the attributes now available are insufficient for describing some concepts. That is, we may guess that there is a *hidden attribute* for describing the concepts, which has not been collected.
- If the ratio is neither 1 nor near to 0.5 then we conclude that there are noisy data in the current database for the classification.

We again use the example of the first part of Eq. (7). If we only observed one of two possible instances that are generalized into $A(a_0) \wedge B(b_1)$ (i.e., the instance $a_0b_1c_1$) from the database shown in Table 2, and we know that its classification is y, then we can acquire the rule $r_{1.1}$. In other words, at this time, since another possible instance $a_0b_1c_0$ is not found, we do not know what its classification is. However, if the instance $a_0b_1c_0$ is added to the database, and its classification is y, then all possible instances for the generalization $A(a_0) \wedge B(b_1)$ are categorized as belonging to the same class y. In this case, the ratio of the second part of Eq. (7) is 1. Conversely, if the classification of the new added instance is n, then the ratio of the second part of Eq. (7) will be changed to 0.5, and the rule candidates will be changed from $r_{1.1}$ to $r_{2.1}$ and $r_{2.2}$:

$r_{2.1}: A(a_0) \wedge B(b_1) \rightarrow D(y)$ with $P = 0.5$,
and

$r_{2.2}: A(a_0) \wedge B(b_1) \rightarrow D(n)$ with $P = 0.5$.
This means that both of the rules $r_{2.1}$ and $r_{2.2}$ must be deleted, because they are contradictory.

On the other hand, if 40 identical instances $a_1b_0c_1$ are added, and only one of them is classified as belonging to n, while the other 39 instances are classified as belonging to y, then the following rules,

$r_{3.1}: B(b_0) \rightarrow D(y)$ with $P = 0.872$,
and

$r_{3.2}: B(b_0) \rightarrow D(n)$ with $P = 0.04$,
can be generated. We can see that the ratio of the noise affects the strengths of these rules. In this case, we infer that the instance $a_1b_0c_1$ with the classification n is a noisy data, because the strength of $r_{3.2}$ is much smaller than that of $r_{3.1}$. Hence, rule $r_{3.2}$ is deleted.

4. An Example

We have tested or are testing our approach with some databases containing data on postoperative patients, weather, breast cancer, and so on. We would like to use the postoperative patient database as an example[17]. This database

[☆] For convenience, the classification is limited to two classes.

consists of several condition attributes such as

(1) L-CORE (patient's internal temperature): high (>37), mid (C$\geq$36 and $\leq$37), low (<36);

(2) L-SURF (patient's surface temperature in C): high (>36.5), mid ($\geq$36.5 and $\leq$35), low (<35);

(3) L-O2 (oxygen saturation in %): excellent ($\geq$98), good ($\geq$90 and <98), fair ($\geq$80 and $\leq$90), poor (<80);

(4) L-BP (last measurement of blood pressure): high (>130/90), mid ($\leq$130/90 and $\geq$90/70), low (<90/70);

(5) SURF-STBL (stability of patient's surface temperature): stable, mod-stable, unstable;

(6) CORE-STBL (stability of patient's core temperature): stable, mod-stable, unstable;

(7) BP-STBL (stability of patient's blood pressure): stable, mod-stable, unstable;

(8) COMFORT (patient's perceived comfort at discharge): high, mid, low.

and a decision attribute,

ADM-DECS (discharge decision): (patient sent to Intensive Care Unit); S (patient prepared to go home); A (patient sent to general hospital floor).

The discovery task in this database is to determine where patients in a postoperative recovery area should be sent to next. Because hypothermia is a significant concern after surgery[17], the attributes correspond roughly to body temperature measurements. From the postoperative patient database, the following rules,

$r_{4.1}$: *L-BP* (*low*) $\rightarrow$ *ADM-DECS* (*A*)
with $P = 0.002058$.

$r_{4.2}$: *CORE-STBL* (*mod − stable*)
$\rightarrow$ *ADM-DECS* (*A*)
with $P = 0.000446$.

$r_{4.3}$: *CORE-STBL* (*stable*)
$\wedge$ *COMFORT* (*mid*)
$\rightarrow$ *ADM-DECS* (*A*)
with $P = 0.172296$.

$r_{4.4}$: *L-O2* (*excellent*)
$\wedge$ *CORE-STBL* (*stable*)
$\rightarrow$ *ADM-DEC* (*A*)
with $P = 0.160787$.

$r_{4.5}$: *L-CORE* (*mid*)
$\wedge$ *CORE-STBL* (*stable*)
$\rightarrow$ *ADM-DECS* (*A*)
with $P = 0.002058$.

......

$r_{4.32}$: *L-BP* (*high*) $\wedge$ *COMFORT* (*high*)
$\rightarrow$ *ADM-DECS* (*S*)
with $P = 0.002056$.

$r_{4.33}$: *BP-STBL* (*stable*)
$\wedge$ *CORE-STBL* (*high*)
$\rightarrow$ *ADM-DECS* (*S*)
with $P = 0.002056$,

can be generated by using the method described in Section 3.

Here we would like to describe how to use background knowledge as a bias in the discovery process described in Section 3.2. First, if we use the background knowledge,

"when the surface temperature of a patient is high, it is not possible that the internal temperature of the patient is low",

then we do not consider the possible instances that contradict this background knowledge in all possible combinations of different attribute values in a database for creating a GDT and its network representation. Thus, we can get a more refined result by using background knowledge in the method described in Section 3. Furthermore, in the *second* stage of the discovery process described in Section 3.2, if we also use the background knowledge,

"the decision is difficult if we use the rule in which there is only one attribute value as a condition",

then $r_{4.1}$ and $r_{4.2}$ should be replaced by the new rules:

$r_{4.1'}$: *L-BP* (*low*) $\wedge$ *COMFORT* (*mid*)
$\rightarrow$ *ADM-DECS* (*A*)
with $P = 0.162394$.

$r_{4.2'}$: *CORE-STBL* (*mod − stable*)
$\wedge$ *L-BP* (*mid*)
$\rightarrow$ *ADM-DECS* (*A*)
with $P = 0.173625$.

This example shows that our approach is a *soft* one that can use background knowledge as biases for controlling the creation of a GDT (and its network representation) and the discovery process.

5. Conclusions

In this paper, we presented a new approach to rule discovery in data with uncertainty and incompleteness, using soft techniques. The main features of our methodology can be summarized as follows:

- It is basically an incremental, bottom-up learning approach, but it also has some of the advantages of the top-down style;
- It can predict unseen instances and repre-

sent explicitly the uncertainty of a rule, including the ability to predict possible instances, in the strength of the rule;
- It can flexibly select biases for search control;
- It can use background knowledge for dynamically revising and changing the network representation of the GDT in the discovery process;
- It can discover *if-then* rules in an evolutionary, parallel-distributed cooperative mode.

We showed that the problems and disadvantages of both version-space and back-propagation, described in Section 1, can be systematically solved or overcome through the methodology proposed in this paper.

The ultimate aim of the research project is to create an *agent-oriented* and *knowledge-oriented* hybrid intelligent model and system for knowledge discovery and data mining in an evolutionary, parallel-distributed cooperative mode. In this model and system, the typical methods of symbolic reasoning such as deduction, induction, and abduction, as well as the methods based on soft computing techniques such as rough sets and fuzzy sets, can be cooperatively used by taking the GDT, the transition matrix in stochastic process, and their network representation as the media[8),19)~21)]. The work presented in this paper is one step toward such a model and system.

Acknowledgments The authors would like to thank the anonymous reviewers for their valuable comments on the first submitted version of this paper.

This research has been supported by the Kayamori Foundation of Informational Science Advancement.

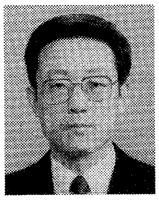
References

1) Cendrowska, J.: PRISM: An Algorithm for Inducing Modular Rules, *Inter. J. of Man-Machine Studies*, Vol.27, pp.349–370 (1987).
2) Chen, M.S., Han, J. and Yu, P.S.: Data Mining: An Overview from a Database Perspective, *IEEE Trans. Knowl. Data Eng.*, Vol.8, No.6, pp.866–883 (1996).
3) Gordon, D.F. and DesJardins, M.: Evaluation and Selection of Biases in Machine Learning, *Machine Learning*, Vol.20, pp.5–22 (1995).
4) Han, J., Cai, Y. and Cercone, N.: Data-Driven Discovery of Quantitative Rules in Relational Databases, *IEEE Trans. Knowl. Data Eng.*, Vol.5, No.1, pp.29–40 (1993).
5) Hebb, D.O.: *The Organization of Behavior*, Wiley (1949).
6) Hinton, G.E.: Connectionist Learning Procedures, *Artificial Intelligence*, Vol.40, pp.185–234 (1989).
7) Hirsh, H.: Generalizing Version Spaces, *Machine Learning*, Vol.17, pp.5–46 (1994).
8) Lin, T.Y. and Cercone, N. (Eds.): *Rough Sets and Data Mining: Analysis of Imprecise Data*, KAP (1997)
9) Mitchell, T.M.: Version Spaces: A Candidate Elimination Approach to Rule Learning, *Proc. 5th Int. Joint Conf. Artificial Intelligence*, pp.305–310 (1977).
10) Mitchell, T.M.: Generalization as Search. *Artificial Intelligence*, Vol.18, pp.203–226 (1982).
11) Quinlan, J.R.: Induction of Decision Trees, *Machine Learning*, Vol.1, pp.81–106 (1986).
12) Ohsuga, S.: Symbol Processing by Non-Symbol Processor, *Proc. 4th Pacific Rim Inter. Conf. on Artificial Intelligence (PRICAI' 96)*, pp.193–205 (1996).
13) Rumelhart, D.E., Hinton, G.E. and Williams, R.J.: Learning Internal Representations by Back-Propagation Errors, *Nature*, Vol.323, pp.533–536 (1986).
14) Rumelhart, D.E. and McClelland, J.L.: *Parallel Distributed Processing*, The MIT Press (1986).
15) Towell, G.G. and Shavlik, J.W.: Extracting Refined Rules from Knowledge-Based Neural Networks, *Machine Learning*, Vol.13, pp.71–101 (1993).
16) Shavlik, J.W. and Dietterich, T.G. (Eds.): *Readings in Machine Learning*, Morgan Kaufmann (1990).
17) Woolery, L., Grzymala-Busse, J., Summers, S. and Budihardjo, A.: The Use of Machine Learning Program LERS-LB 2.5 in Knowledge Acquisition for Expert System Development in Nursing, *Computers in Nursing*, Vol.9, pp.227–234 (1991).
18) Zhong, N. and Ohsuga, S.: Discovering Concept Clusters by Decomposing Databases, *Data & Knowledge Engineering*, Vol.12, No.2, pp.223–244 (1994).
19) Zhong, N. and Ohsuga, S.: Using Generalization Distribution Tables as a Hypothesis Search Space for Generalization, *Proc. 4th Inter. Workshop on Rough Sets, Fuzzy Sets, and Machine Discovery (RSFD-96)*, pp.396–403 (1996).
20) Zhong, N., Fujitsu, S. and Ohsuga, S.: Generalization Based on the Connectionist Networks Representation of a Generalization Distribution Table, *Proc. First Pacific-Asia Conf. on Knowledge Discovery and Data Mining*

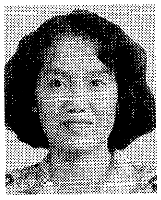
(*PAKDD-97*), World Scientific, pp.183–197 (1997).

21) Zhong, N., Dong, J.Z. and Ohsuga, S.: Discovering Rules in the Environment with Noise and Incompleteness, *Proc. 10th Florida AI Inter. Conf.* (*FLAIRS-97*), edited in the *Special Track on Uncertainty in AI*, pp.186–191 (1997).

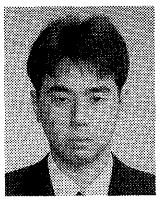
(Received April 30, 1997)
(Accepted June 5, 1998)



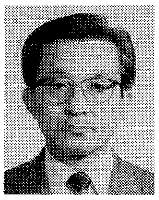
Ning Zhong is currently an associate professor in Department of Computer Science and Systems Engineering, Yamaguchi University, Japan. He graduated at Beijing Polytechnic University in 1982 and has been a lecturer in the Department of Computer Science, Beijing Polytechnic University. He received Ph.D. degree in the Interdisciplinary Course on Advanced Science and Technology from The University of Tokyo. His research interests include knowledge discovery and data mining, rough sets and granular-soft computing, intelligent agents and databases, knowledge-based systems and hybrid systems. He is an editor of Knowledge and Information Systems: an international journal (Springer). He is member of the advisory board of International Rough Set Society, coordinator of a Special Interest Group on Granular Computing in Berkeley Initiative in Soft Computing (BISC/SIG-GrC). He has served or is currently serving on the program committees of international conferences and workshops, including IEEE KDEX-97, RSDMGrC'98, PAKDD'98, PKDD'98, RSCTC'98, ISMIS'99, PAKDD'99 (Program Chair), RSFDGrC'99 (Program Chair).



Juzhen Dong is currently a doctoral student in the graduate school, Yamaguchi University. She graduated at Beijing Polytechnic University in 1982 and has been a lecturer in the Computing Center, Beijing Polytechnic University. She worked as a system engineer at several software companies in Japan. Her research interests include knowledge discovery and data mining, machine learning, and intelligent information systems.



Shinichi Fujitsu is currently a system engineer at INS Engineering Corporation. He graduated at the graduate school, Yamaguchi University in 1997. His research interests include knowledge discovery and data mining, multimedia database and systems.



Setsuo Ohsuga is currently a professor in Department of Information Science and Computer Science, Waseda University, Japan. He received his Ph.D. from The University of Tokyo. He has been professor and director of Research Center for Advanced Science and Technology (RCAST) at The University of Tokyo. He is adviser and was president of Japanese Society for Artificial Intelligence (JSAI). He is vice president of International Society of Applied Intelligence, and is president of International Society of Multimedia and Virtual Systems. He is a member of the steering committee of International Rough Set Society, the advisory board of a Special Interest Group on Granular Computing in Berkeley Initiative in Soft Computing (BISC/SIG-GrC). He is an editor of several scientific journals including Knowledge-Based Systems (Elsevier), Applied Intelligence (Kluwer), Intelligent Information Systems (Kluwer), Knowledge and Information Systems (Springer), and CAD. He has served or is currently serving as a chair of international conferences and workshops, including PRICAI'90, ALT'90, EJCIMKB'91, IEA-AIE'96, MVS'96, IJCAI'97, PAKDD'99, RSFDGrC'99. His research interests include artificial intelligence, knowledge information processing, databases, knowledge discovery and data mining, hybrid systems, automatic programming, and CAD.