修　士　論　文

Using Prior Knowledge in Rule Induction

指導教官　　Ho Bao Tu　教授

北陸先端科学技術大学院大学
知識科学研究科知識システム基礎学専攻

150029 知識 Nguyen Dung Duc

審査委員：Ho Bao Tu　教授（主査）
石崎雅人　助教授
佐藤賢二　助教授
林 幸雄助　助教授

2003 年２月

Acknowledgements

I would like to express my deep gratitude to my supervisor, Prof. Ho Tu Bao, for providing me with kindly helps, supervision and motivation throughout the course of this work. His insight and breadth of knowledge have been invaluable to my training as a researcher. Without his care, supervision and friendship I would not be able to complete this work. I am also grateful to Dr. Nguyen Trong Dung for his comments and suggestions during the work.

My thank also goes to members of the Knowledge Creation Laboratory, especially to Mr. Saito Akinori, Ms. Kawasaki Saori, for providing their helps, a friendly and enjoyable environment while studying here; Mr. Nghiem Anh Tuan for his valuable discussions and comments.

I would like to express my appreciation to the Ministry of Education, Culture, Sports, Science, and Technology of Japan for providing me the scholarship for pursuing the master course at JAIST, and for the financial support for attending several conferences.

Finally, I am indebted to my parents for their forever affection, patience, and constant encouragement.

Using Prior Knowledge in Rule Induction

Abstract: One of the most expressive and human readable representations for learned knowledge is a set of if-then rules. Rule learning algorithms aim at finding a set of rules that best fits the training data according to some predefined criteria. Many of them use only data as the unique input, in other words they are purely inductive, or data-driven. There are also attempts to combine prior knowledge, in terms of domain theory, with inductive learning algorithms to take advantages of both inductive and analytical learning approaches.

In this thesis we describe a new approach to using prior knowledge in rule induction. The concept "prior knowledge" in this work is extended to a broader sense: it includes domain knowledge; prior rule set that has existed as result of previous learning process; and user's constraints. The objective of induction is also changed from finding a new hypothesis in hypotheses space, to finding a better hypothesis based on the existing one, and the data. To illustrate the effectiveness, the proposed approach is applied to improve the classification performance of the rule-based classifiers learned by two rule learning systems LUPC and See5. Experiment results show that the approach is effective in terms of improving the classification accuracy of rule-based classifiers.

Keywords: data mining, rule induction, prior knowledge, rule-based classifier, classification.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1. Research Context

Learning a set of rules from data is a problem that has attracted considerable interest because a rule provides a concise statement of potential useful information that is easily understood by end users [14]. Rule induction algorithms aim at finding a set of rules that satisfy some predefined criteria. Sequential covering algorithms like AQ [13], CN2 [4][5], search for a set of rules that covers all training samples. Simultaneous covering algorithms like ID3, C4.5 [21][22], extract rules from a decision tree built by dividing samples until some conditions are satisfied. In all of these covering algorithms, learning process will stop when all training samples are covered, or all of them satisfy some predefined conditions like belonging to the same class; so that only a small set of rules are discovered. Another common point of covering algorithms is that the training samples become fewer and fewer after rules are learned, or attributes are selected to build a decision tree. This causes a problem called *fragmentation*: the learned rules are locally important but globally insignificant [16][10]. There are also attempts to integrating domain knowledge into induction process to obtain the benefits from both purely inductive and analytical learning, the learning methods that use prior knowledge to derive general hypothesis deductively. One of the approaches is to use prior knowledge, in terms of domain theory, to augment search operators. The FOCL algorithm [17] uses the domain theory to increase the number of candidate specializations considered at each step of the search for a single Horn clause. Candidate hypotheses are then evaluated based on their performance over the training data. In this way, FOCL combines the greedy, general-to-specific inductive search strategy with the rule-chaining, analytical reasoning of analytical methods.

2. Research Objective

In this work we propose a new approach to use prior knowledge in rule induction. The concept "prior knowledge" is extended to a broader sense; it includes domain knowledge, prior rule set that has existed as the result of previous learning processes, and user's constraints. The objective of learning is also changed from finding a new hypothesis in the hypotheses space, to finding a better hypothesis based on the existing one, and data. The reason why we extend concept "prior knowledge" is that data mining is an iterative process, it comprises many steps repeated in multiple iterations [19]. But most rule induction algorithms are one-run process; they will produce the same result in any runs because they use the same search heuristic, the same search strategy on the same data. If we are given more resources like computational power, time, or users do not satisfy with the result of learning, they have no way to go further. Rule learning systems will throw away previous results, start the learning again with the old data, and produce the same thing. In the proposed approach, we use the learned rule set as an input factor to find new rules that are potentially useful. The newly learned rules, together with prior rules, are used to build a better rule set. By this way prior rule set, or prior knowledge, is reused, and enriched incrementally.

3. Research Content

Our approach consists of two main steps. In the first step prior knowledge, or prior rule set, is used to generate a set of simple rules called *rule seeds*. The role of these *rule seeds* is to direct the search for new rules that are different from those in the prior rule set. The second step is to specialize these simple, low confidence *rule seeds* to achieve more accurate rules. The newly learned rules are then used to improve the prior rule set to get a better set of rules. Because no example is removed from training data, the fragmentation is avoided; the newly learned rules are globally significant.

To illustrate the effectiveness, we have conducted two experiments. In the first experiment we apply the proposed approach to improve the classification accuracy of rule-based classifiers produced by LUPC, a sequential covering rule induction algorithm. Experiment result on 26 datasets downloaded from the UCI machine learning repository [15] shows that the proposed approach is effective: it can improve LUPC's classifiers on 7 out of 26 domains. In the second experiment, we apply the proposed approach to improve rule-based classifiers produced by the See5 system. Experiment on 9 datasets shows that the proposed approach effectively improves 3 of 9 classifiers: classification errors are reduced significantly while the size of classifiers is not changed.

4. Document Structure

The rest of this document is organized as follow. In chapter 2 we describe some representative rule induction algorithms: the ID3, CN2, and FOCL algorithms. Chapter 3 is for the main work of this thesis: the proposed approach to using prior knowledge in rule induction. Experiments for illustrating the effectiveness of the proposed approach in improving the classification performance of a prior rule-based classifier are described in chapter 4. In chapter 5 we would like to address some conclusions, and the future works will be described in chapter 6.

Chapter 2
Rule Induction Algorithms

1. Introduction

In many cases it is useful to learn the target function represented as a set of if-then rules that jointly define the function. A rule set in the form of "IF *conditions* THEN *conclusion*" is one of the most expressive and human readable representations for learned hypothesis [14]. In this chapter we will discuss about the three most famous rule induction algorithms: ID3, CN2, and FOCL. ID3 [14] is a representative for the simultaneous covering algorithms; it first learns a decision tree and then translates the tree into an equivalent set of rules, one rule for each leaf of the tree. CN2 [4][5] belongs to the family of sequential covering algorithms; it learns rules set based on the strategy of learning one rule, removing the data it covers, then iterating this process. FOCL [17], an extension of the purely inductive FOIL [20], learns a set of first order Horn clauses to cover the observed training examples. The difference between FOCL and FOIL is that FOCL uses domain theory to increase the number of candidates specializations considered at each step of the search.

2. The ID3 Algorithm

ID3 is a decision-tree learning algorithm that employs a top-down, greedy search through the space of possible decision tree. The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree. In the figure below, attribute *A* that best* classifies *Examples* in the ID3 algorithm is selected based on a statistical property called *information gain*, that measures how well a given attribute separates the training examples according to their target classification. ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.

In ID3, information gain of an attribute A relative to a collection of examples S is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Where $Values(A)$ is the set of all possible values for attribute A, S_v is the subset of S for which attribute A has value v. $Entropy(S)$ is the entropy of S, that characterizes the (im)purity of an arbitrary collection of examples. If the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Table1: The ID3 learning algorithm

ID3(*Examples*, *Target_attribute*, *Attributes*)
Input: *Examples* are the training examples. *Target-attribute* is the attribute whose value is to be predicted by the tree. *Attributes* is a list of other attributes that may be tested by the learned decision tree.
Output: a decision tree that correctly classifies the given Examples

- Create the *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = -
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
    - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
    - The decision attribute for *Root* $\leftarrow A$
    - For each possible value, v_i, of A
        - Add a new tree brand below *Root*, corresponding to the test $A = v_i$
        - Let $Examples_{vi}$ be the subset of *Examples* that have value v_i for A
        - If $Examples_{vi}$ is empty
            - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
            - Else below this new branch add the subtree **ID3**($Examples_{vi}$, *Target_attribute*, *Attributes* – {A})
- End
- Return *Root*

An alternative measure for selecting attributes is the gain ratio that has been used successfully in the C4.5 [21], the successor of ID3. Gain ratio is used to avoid the problem when information gain favors attributes with many values over those with few values. The gain ratio is defined as

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitInformation(S, A)}$$

Where split information is used to penalize attributes that have too many values.

$$SplitInformation(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

The next step to obtain a set of rules is to translate the tree into an equivalent set of rules. Every path from the root of an un-pruned tree to a leaf gives one initial rule. The left-hand side of the rule contains all the conditions established by the path, and the right-hand side specifies the class at the leaf. Each rules is then simplified by removing conditions that do not seem helpful for discriminating the nominated class from other classes, using a pessimistic estimate of accuracy of the rule [21]. This process leads to a production rule classifier that is usually about as accurate as a pruned tree, but more easily understood by people.

The following are some properties of the ID3 algorithms:

- ID3 is a purely inductive or data-driven algorithm; it searches for a decision tree that best fits the training data without any domain knowledge.
- ID3 is the classification-oriented learning algorithm; the central choice of the algorithm is to select the attribute that is the most useful for classifying examples [14]. Because the approximate inductive bias of ID3 is the shorter trees are preferred over larger tree then the rule set produced by ID3 is very simple and less significant (although all of them ensure a high classification accuracy)

- Incremental induction is one of the desirable additions of ID3 and C4.5 [21]. The algorithm proceeds directly from training cases to classifier. There is no role in the scheme of things for a previous or partially completed classifier. There are two situations in which the inability to make use of an existing classifier is unsatisfactory. The first one is the availability of new data after a classifier has been constructed, a common practical application of machine learning where the continual collection of data is the norm. There are two alternatives: ignore the new data or discard the previous classifier, add the new data to the training, and build a new classifier. The second of the concerns is that the greedy algorithms used in C4.5 (the successor of ID3) require a fixed amount of time to run; they cannot exploit more, and will produce nothing in less [21].

3. The CN2 Algorithm

Another way to learn a set of rules is to learn rules directly from data. In this section we will describe CN2 [11][12], a typical rule-learning algorithm belonging to the family of sequential covering algorithms.

Table 2: The sequential covering algorithm for learning a set of rules

Sequential-Covering(*Target_attribute*, *Atrributes, Examples*, *Threshold*)

- *Learned_rules* ← {}
- *Rule* ← **Learn-One-Rule**(*Target_attribute*, *Atrributes, Threshold*)
- while **Performance**(*Rule*, *Examples*) > *Threshold*, do
  - *Learned_rules* ← *Learned_rules* + *Rule*
  - *Examples* ← *Examples* – {examples correctly classified by *Rule*}
  - *Rule* ← **Learn-One-Rule**(*Target_attribute*, *Atrributes, Threshold*)
- *Learned_rules* ← sort *Learned_rules* accord to **Performance** over *Examples*
- return *Learned_rules*

Table 2 describes the general scheme of sequential covering algorithms for learning a disjunctive set of rules. Imagine we have a subroutine **Learn-One-Rule** that accepts a set of positive and negative training examples as input, and then outputs a single rule that cover many of the positive examples and few of negative examples. The procedure **Sequential-Covering** invokes **Learn-One-Rule** on all the available training examples, removes any positive examples covered by the rule it learns, then invokes it again to learn the second rule based on the remaining training examples. This procedure can be iterated as many times as desired to learn a disjunctive set of rules that together cover any desired fraction of the positive examples. This is called sequential covering algorithm because it sequentially learns a set of rules that together cover the full set of positive examples [14].

This sequential covering algorithm is one of the most widespread approaches to learn disjunctive sets of rules [14]. It reduces the problem of learning a disjunctive set

of rules to a sequence of simpler problems, each requiring that a single conjunctive rule be learned. Because it performs a greedy search, formulating a sequence of rules without backtracking, it is not guaranteed to find the smallest or best set of rules that cover the training examples.

One effective approach to implementing **Learn-One-Rule** is to perform a general-to-specific search through the space of possible rules. To reduce the risk that a sub-optimal choice will be made at any step of greedy search, CN2 conducts a beam search in which the algorithm maintains a list of *k* best candidates at each step, rather than a single best candidate. On each search step, descendants (specializations) are generated for each of these k best candidates, and the resulting set is again reduced to the k most promising members. Beam search keeps track of the most promising alternatives to the current top-rated hypothesis, so that all of their successors can be considered at each search step. This general-to-specific beam search is described in table 3.

The following are some properties of the CN2 algorithms:

- Like ID3, CN2 is a purely inductive or data-driven algorithm; it performs a greedy search for a set of rules that cover all training examples without using domain knowledge. The greedy search is not guaranteed to find the smallest or best set of rules [14].
- The common point of covering algorithms is that their heuristics are not based on the whole original training examples, except during the learning for the first rule. Training examples becomes fewer and fewer after iterations because the positive examples covered by the rule are removed from training data set. The rule that is output by the algorithm is the rule encountered during the search whose **Performance** is the greatest, but this measure is based on the remaining examples and this may lead to locally optimal result [10][26].

Table 3: One implementation for **Learn-One-Rule**, a general-to-specific used by the CN2

Learn-One-Rule(*Target_attribute*, *Atrributes, Examples*, *k*)

Returns a single rule that covers some of the Examples. Conducts a general_to_specific greedy search for the best rule, guided by the **Performance** metric.

- Initialize *Best_hypothesis* to the most general hypothesis ϕ
- Initialize *Candidate_hypotheses* to the set {*Best_hypothesis*}
- While *Candidate_hypotheses* is not empty, do
  1. *Generate the next more specific candidate_hypotheses*
     - *All_constraints* ← the set of all constraints of the form (*a* = *v*), where *a* is a member of *Attributes*, and *v* is a value of *a* that occurs in the current set of *Examples*
     - *New_candidate_hypotheses* ←
       for each *h* in *Candidate_hypotheses*
       for each *c* in *All_constraints*
       - Create a specialization of *h* by adding the constraint *c*
     - Remove from *New_candidate_hypotheses* any hypothesis that are duplicates, inconsistent, or not maximally specific
  2. *Update Best_hypothesis*
     - For all *h* in *New_candidate_hypotheses*, do
       - If (**Performance**(*h*, *Examples*, *Target_attributte*)) > **Performance**(*Best_hypothesis*, *Examples*, *Target_attributte*)
         Then *Best_hypothesis* ← *h*
  3. *Update Candidate_hypotheses*
     - *Candidate_hypotheses* ← the *k* best members of *New_candidate_hypotheses*, according to the **Performance** measure.
- Return a rule of the form
  "IF *Best_hypothesis* THEN *prediction*"
  where *prediction* is the most frequent value of the *Target_attribute* among those *Examples* that match *Best_hypothesis*.

Performance(*h*, *Examples*, *Target_attribute*)

- *h_examples* ← the subset of *Examples* that match *h*
- return -**Entropy**(*h_examples*), where entropy is with respect to *Target_attribute*

4. The FOCL Algorithm

FOCL is an extension of the purely inductive FOIL algorithm, an algorithm for learning first-order rules, or Horn clauses. FOIL [20] extends the sequential covering algorithm of CN2 [4] to handle the case of learning first-order rules. To learn each rule, FOIL performs a general-to-specific search, at each step adding a single new literal to the rule preconditions. The new literal may refer to variables already mentioned in the rule precondition or post-conditions, and may introduce new variables as well.

Table 4: The *Cup* learning task

Domain theory:

Cup ← *Stable, Liftable, OpenVessel*
Stable ← *BottomIsFlat*
Liftable ← *Graspable, Light*
Graspable ← HasHandle
OpenVessel ← HasConcavity, ConcavityPointsUp

Training examples:

	Cups				*Non-Cups*					
BottomIsFlat	√	√	√	√	√	√	√			√
CancavityPointsUp	√	√	√	√	√		√	√		
Expensive	√		√				√		√	
Fragile	√	√			√	√		√		√
HandleOnTop					√		√			
HandleOnSide	√			√					√	
HasConcavity	√	√	√	√	√		√	√	√	√
HasHandle	√			√	√		√		√	
Light	√	√	√	√	√	√	√		√	
MadeOfCeramic	√				√		√	√		
MadeOfPaper				√					√	
MadeOfStyrofoam		√	√			√				√

Both FOIL and FOCL learn a set of first-order Horn clauses to cover the observed examples. Both systems employ a sequential covering algorithm that learn single Horn clause, removes the positive examples covered by this new Horn clause, and then

iterates this procedure over the remaining training examples. In both systems, each new Horn clause is created by performing a general-to-specific search beginning with the most possible Horn clause [14](i.e., a clause containing no preconditions). Several candidate specializations of the current clause are then generated, and the specializations with the greatest information gain related to the training examples is chosen. This process is iterated, generating further candidate specializations and selecting the best, until a Horn clause with satisfactory performance is obtained.

The difference between FOIL and FOCL lies in the way in which candidate specializations are generated during the general-to-specific search for a single Horn clause. FOIL generates each candidate specialization by adding a single new literal to the clause precondition. FOCL uses this same method for producing candidate specializations, but also generates additional specializations based on the domain theory.

To illustrate the operation of FOCL we use the simple learning problem summaries in table 4. Here each instance describes a physical object in terms of material from which it is made, whether it is light, etc. The task is to learn the target concept *Cup* defined over such physical objects. Table below describes a set of training examples and domain theory for the *Cup* target concept. Notice the domain theory defines a *Cup* as an object that is *Stable*, *Liftable*, and *OpenVessel*. The domain theory also defines each of theses three attributes in terms of more primitive attributes, terminating in the primitives, operational attributes that describe the instances. Note that the domain theory is not perfectly consistent with the training examples.

There are two kinds of literal that appear in the domain theory and hypothesis representation. We will say a literal is *operational* if it is allowed to be used in describing an output hypothesis. For example, in the *Cup* examples, we allow output hypotheses to refer only 12 attributes that describe the training examples. Literals based on these 12 attributes are thus considered *operational*. In contrast, literals that occur only as intermediate features in the domain theory, but not as primitive attributes of the instances, are considered non-operational. An example of a non-operational attribute in this case is the attribute *Stable*.

At each point in its general-to-specific search, FOCL expands its current hypothesis *h* using the following two operators:

1. For each operational literal that is not part of *h*, create a specialization of *h* by adding this single literal to the preconditions. This is also the method used by FOIL to generate candidate successors. The solid arrows in figure 1 denote this type of specialization.
2. Create an operational, logically sufficient condition for the target concept according to the domain theory. Add this set of literals to the current preconditions of *h*. Finally; prune the preconditions of *h* by removing any literals that are unnecessary according to the training data. The dash arrow in figure 1 denotes this type of specialization.

The following are properties of the FOCL algorithm:

- FOCL uses domain theory to increase number of candidate specializations considered at each step of the search for a single Horn clause. If the domain theory is correct, the training data will bear out the superiority of this candidate over the others and it will be selected. If the domain theory is incorrect, the empirical evaluation of all the candidates should direct the search down an alternative path. In other words, the domain theory is used in the fashion that biased the learner, but leaves final search choices to be made based on performance over training data [14].
- FOCL combines the greedy, general-to-specific inductive search strategy with the rule-chaining, analytical reasoning of analytical methods. It has been shown that FOCL generalizes more accurately than purely inductive FOIL algorithm in a number of application domains in which imperfect domain theory available [14].

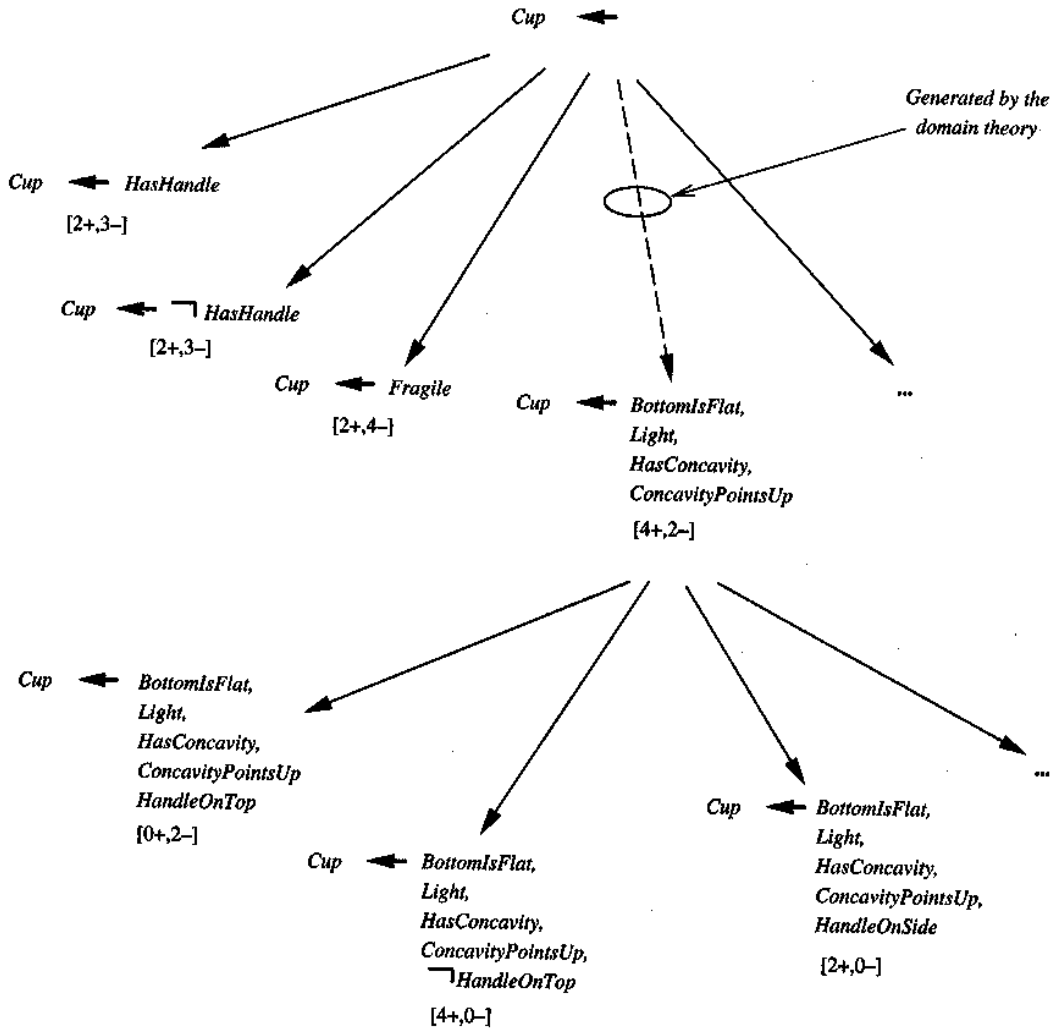

Figure 1: Hypothesis space search in FOCL. To learn a single rule, FOCL searches from general to increasingly specific hypotheses. Two kinds of operators generate specializations of the current hypothesis. One kind adds single literal (solid line in the figure). A second kind of operator specializes the rule by adding a set of literals that constitute logically sufficient conditions for the target concept, according to the domain theory (dash line in the figure). FOCL selects among all these candidate specializations, based on their performance over the data. There for, imperfect domain theories will impact the hypothesis only if the evidence supports the theory [14].

Chapter 3
Using Prior Knowledge in Rule Induction

1. Motivation

In previous chapters we have described three most well known algorithms that learn a set of rules from data. The ID3 and CN2 are two most famous algorithms representative for the simultaneous and sequential covering families. Their common point is that they are purely inductive or data-driven algorithms. They search in hypothesis space (rule sets) to find out one set of rules that coverers all training samples without prior knowledge. Different from ID3 and CN2, the FOCL algorithm uses domain theory to increase number of candidate specializations considered at each step of the search for a single Horn clause. This difference helps FOCL to generalize more accurately than purely inductive FOIL algorithm in a number of domains in which imperfect domain theory available [14]. More over, those above algorithms proceed directly from training samples to hypothesis; there is no role in the scheme of things for a previous or partially completed hypothesis [21]. One of the desirable additions of ID3, as well as of greedy algorithms, is incremental induction. Suppose that more computation time available after a classifier has been generated, the greedy algorithms require a fixed amount of time to run; they cannot exploit more, and will produce nothing in less [21]. An ideal algorithm would produce some classifier quickly, then use additional time available to it to improve the classifier [21].

In this chapter we describe a new approach to using prior knowledge in rule induction. Different from the purely inductive approaches or the approaches that use domain theory to effectively learn a new hypothesis in FOCL, our approach uses prior knowledge that includes existing rule set, domain theory, and user's constraints, to build

a new hypothesis, or rule set, based on the prior one. The improvement is done by searching for new rules and using them to replace prior rules, or to extern the prior rule set to achieve a better rule set. In the learning context, if the existing hypothesis is a rule-based classifier, the approach aims at building a better rule-based classifier that does classification more accurately. If the hypothesis is a set of rules that are already known by the user, the extension aims at finding new and potential useful rules that are different from existing rules. In this framework, the knowledge is re-used, enriched, and extended after learning iterations.

The rest of this chapter is organized as follow. In section 2 we will summary the learning problem in which prior hypothesis knowledge is used as an input factor. The concept prior knowledge will be discussed in more detailed in this section. Section 3 is for describing the general framework for learning new rules. Two main components of the framework are described.

2. The Learning Problem

To summary the learning problem is stated as follow:

Given:

- A set of training examples D
- A domain theory B, and user's constraint C
- An existing hypothesis (a set of rules) R
- A space of hypotheses H

Determine:

- A hypothesis R^+ that fits training examples D better.

A new point of the proposed approach is the prior knowledge includes an existing hypothesis, or rule set R. This set of rules may be the result of previous learning process, or provided by domain experts. If this rule set exists, the objective of induction is to find new rules and improve it. If rule set R does not exist, the objective of induction is to search for a new hypothesis that best fits the training data according to predefined

criteria, the same objective as common rule learning algorithms. The reason why we do use a set of rules as an input factor for rule induction is that we expect to find out new rules to improve the existing rule set. If we do not keep the rules, the search will give us the same result as we already have.

The second component of the prior knowledge is the domain knowledge *B*, exactly the same as described in the FOCL algorithm. Approximate prior knowledge, or domain theories, is available in many practical learning problems [14]. Purely inductive methods such as decision tree induction or neuron network fail to utilize such domain theories, and therefore perform poorly when data is scarce [14]. Purely analytical learning methods utilize domain theories, but produce incorrect hypotheses when given imperfect prior knowledge. Methods that blend inductive and analytical learning can gain benefits of both approaches: reduce complexity and the ability to overrule incorrect prior knowledge.

The last component of the prior knowledge is a set of user's constraints *C*. Users use constraints to narrow the search, for example, for patterns they are interested in; this plays a crucial role for the successful of many learning tasks. In data mining, discovering new and interesting patterns is still a hard problem [19]. Knowledge discovery in databases (KDD) is defined as "the nontrivial process of identifying valid, novel, potential useful, and ultimately understandable patterns in data" [6]. In many cases, it is possible to define measures of certainty (for example, estimated prediction accuracy on new data) or utility (for example, gain, perhaps in dollars saved because of better predictions or speedup in respond time of a system)). But the notions like novelty is much more subjective [18]. No one but the user can identify which pattern is new and interesting. The ways we use these constraints is quite different from the works that use constraints to rank the discovered rules [2][9][25] or to filter out not interesting ones [23][24]. In our approach the constraints are used during the search, not in the post-processing step. In section 3.1 we will introduce an illustrative example of how user's constraints could be integrated in discovering such kind of patterns.

3. General Framework

Figure 2 shows the general framework for using prior knowledge in rule induction.

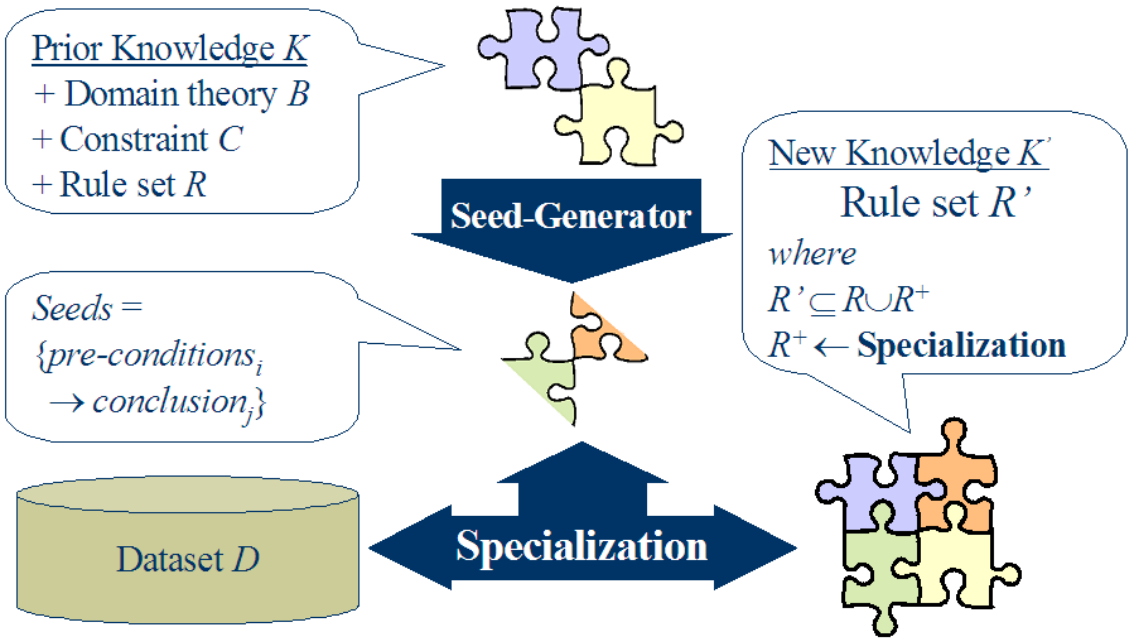

Figure 2: A general framework for utilizing prior knowledge in rule induction.

There are two important components in this framework. First, the *seed generator* takes prior knowledge as input and outputs a set of *rule seeds*. Second, the *rule specialization* process specializes the *seeds* to achieve accurate rules. These two components are described in more detail in the next sections.

3.1 Seed generator

The *seed generator* is the most important component in the framework. It is a procedure that takes prior knowledge as input, outputs a set of simple rules that will guide the search in the specialization process. Suppose that *s* is a *rule seed* produced by the *seed generator*, *s* has to satisfy two conditions:

i) Rules specialized from *s* are different from existing rules in *R*.
ii) Rules specialized from *s* are potentially useful.

The first condition is to ensure that newly learned rules are new with respect to existing rule set *R*. The concept "*new*" is a relative concept because it is based on at least two objects belonging to the same category, and of course, is very complicated. In this context we simplify it as follow: a rule *r'* in the form of "IF *conditions* THEN *conclusion*" where *conditions* is a conjunctive of conditions and *conclusion* is a target concept; rule *r'* is called *new* with respect to a rule set *R* if for every rule *r* of *R*, one of the two following conditions is satisfied:

i) $conclusion(r') \neq conclusion(r)$
ii) $conditions(r') \not\subseteq conditions(r)$

According to this simplification a rule is said new if it refers to a different concept; or its conclusion consists of a new factor, or a new condition.

Note that the *seed generator* is application-oriented procedure. If the application is to build a rule-based classifier, the generator should generate the seeds that rules specialized from them could improve the accuracy of a classifier. If the application is to discover interesting patterns, the seeds will direct the search to rules that makes people surprised, for example. Building the generator will be described in more detailed in the next section and in chapter 4 when the proposed approach is used in improve a rule based classifier.

To illustrate how the *seed generator* works, lets take an example on a real application. The stomach cancer data collected at the National Cancer Center in Tokyo in 30 years from 1962 to 1991 contains data of 7,520 patients. Each patient is described by 83 attributes about personal information, symptoms, pre-operative and post-operative complication, etc. The knowledge doctors want to know is the symptoms related to the death or alive of patients within 90 days, after 90 days, after 5 years of operation, or to

be alive. After doing necessary pre-processing step we have used rule induction systems, including See5 [22] from RuleQuest, and our system called LUPC [8] to discover rules for above concepts. The result is See5 discover one rule for the concept "*the patient die within 90 days*" and LUPC discover 7 rules, with minimum accuracy 90% and cover at least 5 samples. We introduce this result to the doctor and ask them about their comments. Unfortunately they said that all of them are believable, but not new. Five of seven rules have the symptoms "*level 3 of liver metastasis*" in the condition part of the rules for concept "*death within 90 days*", this is not new. One question raised to us is what should we do next? If we run the See5 or LUPC again we will get the same result because See5 and LUPC discover a rule set on the same data set, using the same search strategy with the same stop condition that the rule sets cover all samples. Our solution is first, we consider again the preprocessing step because if this step is not done well, there will be no interesting pattern hiding in the data; we will find no gold from rock. The second direction is to continue the search, but for new rules that different from existing ones that are obvious to the doctors. We visualize these five rules (in figure 3) and discovered that symptom "*level 3 of liver metastasis*" relates only to concept "*death within 90 days*". It means that there is no relation between symptom "*level 3 of liver metastasis*" and concept "*alive*" or "*death after 5 years*". Our question is there any such kind of relation? And these relations are potentially interest? And the problem now is turned out to proposed framework:

Given:

- A data set: the Stomach Cancer data
- A set of existing rules: the rule sets discovered by See5 and LUPC

Determine:

- New and interesting rules

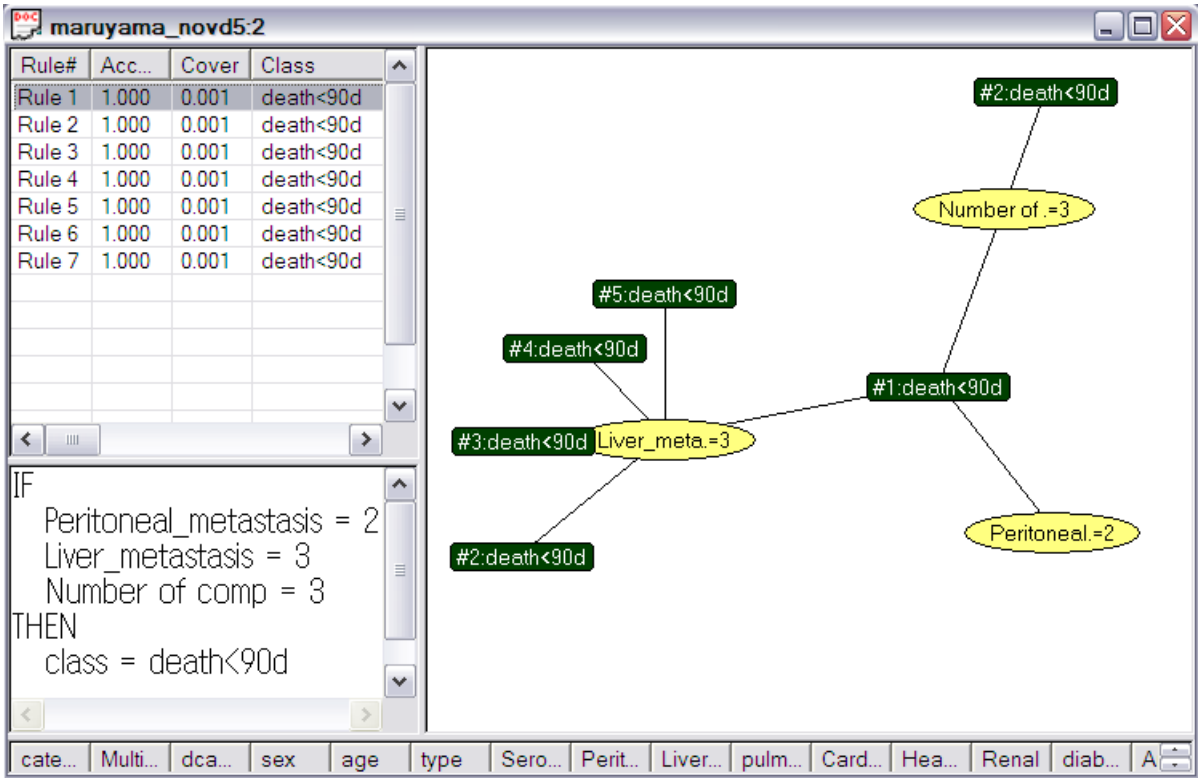

Figure 3: Visualization of the concept "*death within 90 days*"

The objective of the application, finding interesting patterns about relationship among symptoms and the death or alive of patients after operation, is not changed, but the objective of learning is changed from searching for a rule set covering all samples, to searching for new and potential interesting rules that different from existing ones (in fact, LUPC discovered 1130 rules for all classes). In this situation, the seed generator generates two seeds:

Seed #1: IF *Liver_metastasis = 3* THEN *alive*
Seed #2: IF *Liver_metastasis = 3* THEN *death > 5 years*

These two seeds will guide the search for rules of the two concepts "*alive*" and "*death after five years*", and the result is that the following rule is discovered:

IF *type = B1 ^ Liver_metastasis = 3 ^ Number of complications = 1*
THEN *alive*

From this example, we can see that rule seeds are generated from prior knowledge and in this case prior knowledge is rule sets learned by See5 and LUPC, and from use's constraint about relation between symptom "*level 3 of liver metastasis*" and the two concepts. This constraint comes from the interaction between user and data mining system while exploring discovered knowledge. The rule seeds are generated to direct the search in a narrower space that closer to user's interest.

Table 5: The **Specialization** procedure

Specialization(*RuleSeeds, Examples*, *Threshold*)

- *Learned_rules* ← {}
- For all *Seed* in *RuleSeeds*
  - *Rule* ← **Learn-One-Rule**(*Seed*, *Atrributes, Threshold*)
  - If **Performance**(*Rule*, *Examples*) > *Threshold*, do
    - *Learned_rules* ← *Learned_rules* + *Rule*
- return *Learned_rules*

3.2 Rule specialization

The task of the *rule specialization* process is to specialize the rule seeds to achieve more accurate rules. The procedure to do this task is described in table 5. In comparing with sequential covering algorithms the role and scheme of the **Specialization** and **Sequential-Covering** procedure (described in [14]) are similar: they conduct a subroutine called **Learn-One-Rule** to learn rules from data. But there are three crucial differences as follow:

(1) The inputs of the two procedures are different. Procedure **Sequential-Covering** takes only training samples (of course there are also attributes list and parameters) as input factor. This is because sequential covering algorithms are purely inductive; they do not use any prior knowledge in learning. Procedure **Specialization** takes two inputs: training samples and a set of rule seeds. In comparing with analytical learning methods or inductive-analytical learning like FOCL, the set of rule seeds may be consider as domain knowledge and the

Specialization procedure could be seen as in context of the analytical learning approach. The difference is the rule seeds or the prior knowledge is generated from not only domain theory, but also from prior rule set.

(2) The stopping conditions are different. Covering algorithms stop learning process when learned rules cover all training samples; even for the analytical-inductive rule induction algorithms like FOCL also stop when all training samples are covered, or when no sample remains. In procedure **Specialization** the main loop will finish when all rule seeds are specialized, it doesn't matter if the training samples are covered or not because the learning has not to satisfy the completeness condition that all training samples are covered.

(3) After one rule is learned the sequential covering algorithms remove all training samples that covered by this rule. This causes the number of training samples becomes fewer and fewer after iterations; and the rule's performance estimated by function **Performance** becomes more and more locally; except for the first rule. The procedure **Specialization** does not remove any training samples, so that rule's performance is globally significant because it is based on the whole training samples.

Table 6: Comparison of the **Sequential-Covering** and the S**pecialization** procedures

	Sequential-Covering	**Specialization**
Input	+ Training samples + Attributes list	+ Training samples + Attribute list + Rule seeds
Output	A set of rules	A set of rules
Stopping condition	+ All training samples are covered	+ All rule seeds are specialized
Heuristics	Based on remaining samples	Based on all samples

Table 7: The subroutine for learning one rule conducted by **Specialization**

Learn-One-Rule(*Seed*, *Atrributes, Examples*, *k*)

Returns a single rule that covers some of the Examples. Conducts a general_to_specific greedy search for the best rule, guided by the **Performance** metric.

- Initialize *Best_hypothesis* to the rule *Seed*
- Initialize *Candidate_hypotheses* to the set {*Best_hypothesis*}
- While *Candidate_hypotheses* is not empty, do
    3. *Generate the next more specific candidate_hypotheses*
    4. *Update Best_hypothesis*
    5. *Update Candidate_hypotheses*
- Return *Best_hypothesis*

Performance(*Rule*, *Examples*)

- return **LaplaceAccurace**(*Rule*, *Examples*)

Both **Sequential-Covering** and **Specialization** procedure invoke subroutine **Learn-One-Rule** to continually search for a new rule, but the difference is the sequential covering algorithms start the search for new rules from the most general hypothesis, or a null hypothesis. The *Best_hypothesis* of sequential covering algorithms is initialized with no conditions, so all the *Candidate_hypotheses* are specialized from an empty hypothesis. In the proposed approach, similar to the analytical learning approaches, the *Best_hypothesis* is a simple non-empty rule generated from prior knowledge. In searching for a new rule the **Learn-One-Rule** procedure conducts a beam search with rule's Laplace expected accuracy estimate as search heuristic. The Laplace estimate is used to avoid selecting specific rules instead of entropy. This expected accuracy is given by the formula:

$$\text{LaplaceAccuracy} = (n_c + 1)/(n_{tot} + k)$$

Where

k is the number of classes in the domain

n_c is the number of examples in the predicted class c covered by the rule

n_{tot} is the total number of examples covered by rule

Chapter 4

Improving a Rule-based Classifier

1. Problem Statement

Suppose that a rule-based classifier R is produced by some rule learning algorithms, LUPC or See5 in our experiments, our attempt is to apply the proposed approach to improve this classifier. In this context the prior knowledge consists only the existing hypothesis, or learned classifier R, no domain theory B, and no user's constraint C. The learning problem could be stated as follow:

Given:

- A set of training examples D
- A rule-based classifier R

Determine:

- A rule-based classifier R^+ that does classify better than R

It is clear that the learning task here is quite different from traditional learning task. The learning task of ID3, CN2, or FOCL is to search for a set of rules that do classify well, given a training data set with (FOCL) or without (ID3, CN2) domain knowledge. The task in this situation is to improve an existing rule set to achieve a new rule set that does classification better. Following sections describe in more detail how this work could be done.

2. Updating a Rule-based Classifier

In this section we will describe how to improve an existing rule set, or an existing classifier, to achieve a better classifier. Especially about the prior knowledge used in our experiments, how to generate rule seeds to bias the search for potential useful rules, and how to use newly learned rules to update preceding classifier.

From now we will use the following terms and definitions. We consider a set of variables $x = \{x_1, x_2, \ldots x_n\}$, $n > 0$ with domain $X = \{X_1, X_2, \ldots X_n\}$ called attributes and a single variable y with domain Y called class attribute or just class. A rule is in the form of "IF *conditions* THEN *conclusion*" where *conditions* is a conjunctive of attribute-value pairs, and *conclusion* is one of the values of Y. For example given a rule r:

$$\text{"IF } x_1 = X_{1i} \text{ and } x_2 = X_{2j} \text{ THEN } Y_k\text{"}$$

The conditions part is a conjunctive of two attribute-value pairs belonging to two attributes X_1 and X_2; the conclusion part is the value Y_k belonging to class attribute Y. We say that this rule uses two attributes in the conclusions part.

2.1 Prior knowledge

In our experiments, prior knowledge, or rule set, is learned by one of the two rule induction algorithms: the LUPC or See5 algorithm.

The LUPC [8], standing for Learning in Unbalanced Positive Class, is a rule induction algorithm that aims at learning for rare class in unbalanced datasets. It follows the general scheme of sequential covering algorithms with the following properties:

- **Search strategy**: LUPC conducts a beam search for finding rules in rule space. Many learning algorithms like AQ (Michalski et al., 1986), CN2 (Clack and Niblett, 1989), *m*FOIL (Dzeroski and Bratko, 1992), and BEXA

(Theron and Cloete, 1996) use this strategy to alleviate the myopic behavior of hill-climbing search. In addition to remembering the best rule found so far, beam search also keeps track of a fixed number of alternatives, the so-called *beam*. While hill-climbing search has to decide upon a single refinement at each step, beam search can defer some of the choices until later by keeping the b best rules in its beam. In general, beam search effectively maintains hill-climbing's efficiency (reduced by a constant factor), but can yield better results because it explorer a larger portion of the hypothesis space.

- **Search heuristic**: LUPC uses rule's purity as its search heuristic. Search heuristic is the most influential bias in searching, which estimates the quality of rules found in the search space and ideally guides the search algorithms into the right regions of the hypothesis space. As the goal of the **Learn_One_Rule** is to find a rule that covers as many positive samples while covering as few negative samples as possible, most heuristics try to find a trade-off between the two conditions. The most commonly used search heuristics are accuracy, purity, entropy, Laplace estimate …Rule's purity measure will attaint its optimal value when no negative samples are covered. The reason why LUPC use rule's purity as search heuristic is LUPC focus in learning for rare classes in unbalanced datasets. In many applications rare class is important but very difficult to learn [8]. Purity is used as search heuristic in the GREEDY3 (Pagallo and Haussler, 1990), SWAP-1 (Weiss and Indurkhya, 1991). The purity of a rule is given by the following formula

$$Purity(r) = \frac{p}{p+n}$$

Where p is the number of positive samples covered by rule r, and n is the number of negative samples covered by r.

- Constraints: To overcome the disadvantage of the purity measure that does not aim at covering many positive samples, LUPC uses minimum accuracy and minimum cover of a rule as constraints in the search. In our experiments minimum accuracy is set to 90% and minimum cover is set to cover at least 2 samples.

The See5 system [22] is the successor and the commercial version of the most well-know algorithm C4.5[21]. See5 builds a decision tree first, and then extracts rules from this tree.

Both LUPC and See5 use the same way of using learned rule set to do classification. It may happen that several rules are applicable (that is, all their conditions are satisfied) when classifying a new sample. If the applicable rules predict different classes, there is an implicit conflict that could be resolved in two ways: we could believe the rule with the highest confidence, or we could attempt to aggregate the rule's prediction to reach the verdict. Both See5 and LUPC adopt the late strategy: each applicable rule votes for its predicted class with a voting weight equal to its confident value, the votes are totted up, and the class with the highest total vote is chosen as the final decision. The confidence of a rule is given by the below formula

$$Confidence(r) = \frac{PositiveCover(r) + 1}{Cover(r) + 2}$$

When a test sample is not covered by any rule, it is assigned to a default class. The default class is the class that has dominated distribution in the training data.

2.2 The ***Seed-Generator*** procedure

The task of the *seed generator* is to generate rule seeds that will guide the search for useful rules. In our experiments, we use attributes that have not been used by prior rule set, to generate the seeds. Table 8 gives detail of the procedure.

Table 8: The *seed generator* procedure used in improving a rule-based classifier

Seeds_Generator(*RuleSet*, *Attributes*, *TargetAttribute*)

Takes an existing hypothesis, or a set of rules, as an input factor to produce a set of rule seeds in the form of simple rules that will be used to guide the search for new and useful rules in the next **Specialization** process

- *Seeds* ← {}
- For all attribute *a* in *Attributes*
  - If *a* is not used in *RuleSet*
    - For all values *v* of attribute *a*
      - For all values *c* of *TargetAttribute*
        - *Seeds* ← *Seeds* + {"IF *a* = *v* THEN *c*"}
- Return *Seeds*

In table 8, an attribute *a* is said not to be used by rule set *R* if there is no rule *r* of *R* that consists a condition $a = a_i$ in the *conditions* part of rule *r*.

To improve the classification accuracy of a rule-based classifier learned by a covering algorithm, the seed generator in our experiment is relatively simple: the generator uses attributes that are not used by in prior rule set. The using of what other algorithms do not use seem to be trivial because learning algorithms use their heuristics, or their "spirits", to select best attributes in searching for the best result. This means that unused attributes seem to be not valuable, according to their heuristics. The reason why we build a so simple seed generator is due to one important shortcoming of covering algorithms: the fragmentation of training data [26][10]. Covering algorithms remove all

covered samples (belonging to the positive class or in the whole training data) after one rule is learned, this causes number of training samples becomes fewer and fewer after iterations and leads to the generation of many locally important but globally insignificant rules [10]. Our seed generator generates the seeds that will lead the search to these globally significant rules.

Figure 4 illustrates the fragment problem of covering algorithms. Supposed that in training data D there exist three patterns P_1, P_2, and P_3 listed by the order of their significant. Sequential covering algorithms discovered pattern P_1 first, and then removes the (positive) samples covered by P_1. Due to this moving the pattern P_2 becomes less significant than pattern P_3 in the remaining data, and the search will result pattern P_3, instead of P_2. After iterations the learned rules will become more and more locally important and the chance of missing globally significant rules becomes bigger and bigger. Our approach could avoid the fragmentation problem because no training samples are removed and the search heuristic is based on the whole training data.

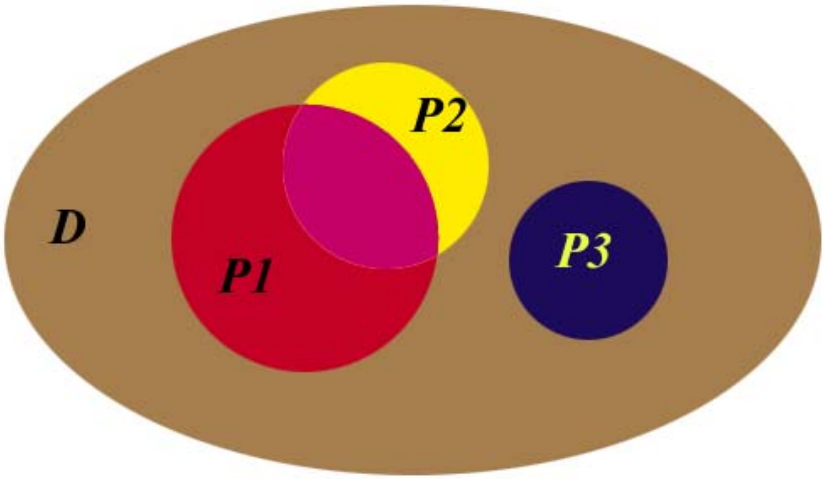

Figure 4: Covering algorithms may loss many globally significant rules

2.3 Updating prior rule sets

After the *seed generator* generates a set of rule seeds, they are then specialized by adding more conditions into the *conditions* part to achieve a necessary accuracy. The specialization process is actually a search, not in the whole space of possible rules, but the space is narrowed by the condition part of rule seeds. Result of this search is a set of new rules that will be used to upgrade prior rule-based classifier.

The newly learned rules will be used to improve the prior rule set. Note that a classifier formed by many rules does not guarantee an accurate classification, so that not all of the rules are used to improve the prior rule set. In our experiment, a newly learned rule *r'* is used to upgrade the prior rule set *R* to the new rule set *R'* by two ways:

i. *r'* will be used to replace a rule *r* in *R*:

$$R' = R \setminus \{r\} \cup r', r \in R$$

or

ii. *r'* will be added to *R*:

$$R' = R \cup r'$$

If the replacement or extension helps to improve the classification accuracy on the training data *D*:

$$\text{ClassificationAcc}_D(R') > \text{ClassificationAcc}_D(R)$$

To avoid the over fitting problem caused by too specific rules, the new rules are forced to satisfy a constraint on coverage ratio. In our experiment this constraint is 5% of the population of positive class.

3. Experiments

This section describes the experiment of applying the proposed approach to improve classification accuracy of a rule-based classifier learned by a covering algorithm. The evaluation will be based on the classification accuracy on testing data. Sections below will describe the datasets, the prior rule-based classifier used in our experiments, and the results obtained from them.

3.1 Datasets

In the first experiment, we use 26 datasets downloaded from the UCI machine learning repository [15] described in table 9. Because the LUPC algorithm cannot run with numerical data, so all these datasets are discretized by the CBA system [11][12]. The first column of table 2 is for indexes of these 26 datasets; second column describes their name. Each data set is described by number of samples (column 3), number of attributes (column 4), and number of classes (column 5).

In the second experiment, we use 9 other datasets also from the UCI. Each data set consists of two different parts used for learning and testing purposes. In table 3, columns from number 2 to number 6 describe name, number of training samples, number of testing samples, number of attributes, and number of classes of the 9 datasets. Like the first experiment, all these datasets are discretized by the CBA system.

3.2 Methodology

We have conducted the experiments by two ways. In the first experiment we use classifiers produced by LUPC with default constraints (90% of minimum accuracy and 2 samples of minimum cover) as prior knowledge, and apply the proposed approach to improve them. The performances are evaluated based on running 10 times of 10-fold cross validation test, a common way of testing a learning algorithm. In the second experiment, we use the classifiers produced by the See5 system with default parameters,

denoted by See5 in table below. The performances are then compared with the upgrade version of See5, denoted by See5+, by applying the proposed method.

Table 9: Comparison between classification performance of LUPC and LUPC+

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Index	**Dataset**	**#Samples**	**#Attributes**	**#Classes**	**#LUPC Rules**	**#LUPC+ Rules**	**LUPC Error(%)**	**LUPC+ Error(%)**
1	anneal	898	39	6	34.9	49.4	4.9	4.4
2	australian	690	15	2	89	89	13.8	13.8
3	auto	205	23	7	45.2	48.9	18.5	18.3
4	breast-w	699	10	2	19.8	19.8	4.7	4.7
5	cleve	303	12	2	45.2	45.2	17.6	17.6
6	crx	690	16	2	81.1	81.1	14.1	14.1
7	diabetes	768	7	2	84.6	84.6	23.1	23.1
8	german	1000	16	2	265.1	265.1	26.7	26.7
9	glass	214	8	7	34.6	34.6	25.9	25.9
10	heart	270	10	2	35.2	35.2	17	17
11	hepatitis	155	17	2	20.9	22.2	17.9	17.7
12	horse	368	19	2	72.5	72.9	16.2	16.2
13	hypo	3163	24	2	16.3	30	3.5	2.7
14	ionosphere	351	34	2	32.5	48	8.6	8.3
15	iris	150	5	3	6.1	7.1	6.4	6.5
16	led7	3200	8	10	168.2	168.2	26.9	26.9
17	labor	57	13	2	12.4	15.4	13.9	12.1
18	lymph	148	16	4	25.1	25.5	17.7	17.5
19	pima	768	7	2	84.9	84.9	23.2	23.2
20	sick	2800	27	2	27.2	37.5	4.6	4.6
21	sonar	208	22	2	34.3	34.3	16.4	16.4
22	tic-tac-toe	958	10	2	63.4	63.4	5.6	5.6
23	vehicle	846	19	4	238.5	238.5	28.9	28.9
24	waveform21	5000	20	3	1030.1	1030.1	16.3	16.3
25	wine	178	14	3	9.3	17.7	2.1	1.6
26	zoo	101	17	7	9.6	18.2	4.6	4.4
	Average	*930.31*	*16.46*	*3.31*	*99.46*	*102.57*	*14.58*	*14.40*

3.3 Results

Column number 6 and 7 in table 9 show number of rules used by LUPC and LUPC+ classifiers, column 8 and 9 show error rate on testing data. Result from table 2 shows that proposed approach could reduce error rate on 9 of 26 datasets. On the *iris* data set, classification error is not improved, but becomes worse. This is due to the fact that the classifier is upgraded by using training data. It may happen that a classifier run well on

Table 10: Comparison of six classifiers on classification error rate (%)

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Index	**Dataset**	**LUPC**	**LUPC+**	**See5Tree**	**See5Rule**	**CBA**	**CBA (2)**
1	anneal	4.9	4.4	8	7.4	3.6	2.1
2	australian	13.8	13.8	13.7	13.9	13.4	14.6
3	auto	18.5	18.3	18.5	20.7	27.2	19.9
4	breast-w	4.7	4.7	4.8	4.2	4.2	3.7
5	cleve	17.6	17.6	22.1	21.1	16.7	17.1
6	crx	14.1	14.1	13.2	13.1	14.1	14.6
7	diabetes	23.1	23.1	21.9	23.3	25.3	25.5
8	german	26.7	26.7	27.5	26.8	26.5	26.5
9	glass	25.9	25.9	24.8	25.6	27.4	26.1
10	heart	17	17	18.4	17.3	18.5	18.1
11	hepatitis	17.9	17.7	18.8	17.2	15.1	18.9
12	horse	16.2	16.2	15	15.1	18.7	17.6
13	hypo	3.5	2.7	0.8	0.8	1.7	1
14	ionosphere	8.6	8.3	10.4	8.7	8.2	7.7
15	iris	6.4	6.5	6.1	6.1	7.1	5.3
16	led7	26.9	26.9	26.6	26.6	27.8	28.1
17	labor	13.9	12.1	16.1	15.5	17	13.7
18	lymph	17.7	17.5	22.7	23.4	19.6	22.1
19	pima	23.2	23.2	22.3	23.6	27.6	27.1
20	sick	4.6	4.6	2.1	2.1	2.7	2.8
21	sonar	16.4	16.4	18.9	18.1	21.7	22.5
22	tic-tac-toe	5.6	5.6	14.2	3.4	0.1	0.4
23	vehicle	28.9	28.9	29	28.8	31.3	31
24	waveform21	16.3	16.3	24.9	22.2	20.6	20.3
25	wine	2.1	1.6	7.4	4.1	8.4	5
26	zoo	4.6	4.4	7.6	7.4	5.4	3.2
	Average	*14.58*	*14.40*	*15.99*	*15.25*	*15.77*	*15.19*

training data but worse than when applying on testing data. Table 3 shows the relative comparison on classification accuracy of six classifiers LUPC, LUPC+, See5 tree, See5 rule, CBA, and CBA2, the latest version of CBA. Experiment result shows that LUPC+ has a very competitive performance in comparing with the state-of-the-art rule mining systems. Evaluated on 26 datasets by running 10 times of 10-fold cross validation tests, LUPC+ wins 7, See5 tree wins 6, CBA2 wins 6 on total 26 datasets. In this experiment the results of CBA and CBA2 are drawn from the author's papers [11].

Table 11: Comparison of six classifiers on classification error rate (%)

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Index	**Dataset**	**LUPC**	**LUPC+**	**See5Tree**	**See5Rule**	**CBA**	**CBA (2)**
1	anneal	4.9	4.4	8	7.4	3.6	2.1
2	australian	13.8	13.8	13.7	13.9	13.4	14.6
3	auto	18.5	18.3	18.5	20.7	27.2	19.9
4	breast-w	4.7	4.7	4.8	4.2	4.2	3.7
5	cleve	17.6	17.6	22.1	21.1	16.7	17.1
6	crx	14.1	14.1	13.2	13.1	14.1	14.6
7	Diabetes	23.1	23.1	21.9	23.3	25.3	25.5
8	German	26.7	26.7	27.5	26.8	26.5	26.5
9	glass	25.9	25.9	24.8	25.6	27.4	26.1
10	heart	17	17	18.4	17.3	18.5	18.1
11	hepatitis	17.9	17.7	18.8	17.2	15.1	18.9
12	horse	16.2	16.2	15	15.1	18.7	17.6
13	hypo	3.5	2.7	0.8	0.8	1.7	1
14	ionosphere	8.6	8.3	10.4	8.7	8.2	7.7
15	iris	6.4	6.5	6.1	6.1	7.1	5.3
16	led7	26.9	26.9	26.6	26.6	27.8	28.1
17	labor	13.9	12.1	16.1	15.5	17	13.7
18	lymph	17.7	17.5	22.7	23.4	19.6	22.1
19	pima	23.2	23.2	22.3	23.6	27.6	27.1
20	sick	4.6	4.6	2.1	2.1	2.7	2.8
21	sonar	16.4	16.4	18.9	18.1	21.7	22.5
22	tic-tac-toe	5.6	5.6	14.2	3.4	0.1	0.4
23	vehicle	28.9	28.9	29	28.8	31.3	31
24	waveform21	16.3	16.3	24.9	22.2	20.6	20.3
25	wine	2.1	1.6	7.4	4.1	8.4	5
26	zoo	4.6	4.4	7.6	7.4	5.4	3.2
	Average	*14.58*	*14.40*	*15.99*	*15.25*	*15.77*	*15.19*

In the second experiment, the result shows that the proposed approach can improve the classification accuracy of the See5's rules on 3 of 9 datasets. From table 12 we can see that the size of classifiers is not changed; it means that some rules of the prior classifier are replaced by the newly learned rules.

Table 12: Comparison between classification performance of See5 and See5+ rule sets

(1)	(2)	(3)	(4)	(5)	(6)	(7)			(8)	(9)		
						See5			#	See5+		
		#	#	#	#		Error		New		Error	
ID	Dataset	Training	Testing	Attrs.	Classes	Size	Training	Testing	Rules	Size	Training	Testing
1	adult	32561	16281	14	2	112	65	21	0	-	-	-
2	annealing	798	100	39	6	25	30	4	17	25	12	2
3	audiology	200	26	70	24	22	20	5	3	22	20	5
4	image segmentation	210	2100	16	7	16	9	190	15	16	8	187
5	labor negotiations	40	17	13	2	2	5	4	0	-	-	-
6	letter image recognition	1600	400	16	26	2062	1208	840	0	-	-	-
7	sick	2800	972	27	2	15	48	21	7	15	48	21
8	soybean-big	307	376	36	19	35	14	50	7	35	14	50
9	thyroid disease	2800	972	27	5	10	18	12	20	10	17	7

Let's examine a classifier to see how it is upgraded. In the annealing application the See5 system found 25 rules to establish the classifier. The classifier classifies wrongly 30 samples on the training data and 4 samples when performing on the testing data. By applying the proposed approach a set of 20 rules are newly discovered, and 5 of them are used to replaces 5 others in the prior rule set. This replacement reduces the number errors on training data from 30 samples to 12 samples, and on testing data from 4 samples to 2 samples. These 5 rules are listed in table 14.

Table 13: Differences between See5 and See5+ classifiers on the *annealing* domain.

See5

Replaced rules:

Rule 5: (27/31)
steel = R
thick(continuous) = 0.7995-0.8005
→ class 2

Rule 6: (2/2)
steel = V
thick(continuous) = 1.5495-1.5995
→ class 2

Rule 11: (124/124)
hardness(continuous) = 22.5-75
→ class 3

Rule 12: (188/190)
thick(continuous) = 0.6005-0.7995
→ class 3

Rule 20: (190/215)
formability = 2
width(continuous) = - 1167.5
→ class 3

Evaluation on training data:

(a)	(b)	(c)	(d)	(e)	(f)	<-classified as
5	2	1	0	0	0	(a): class 1
0	76	12	0	0	0	(b): class 2
0	9	597	0	0	2	(c): class 3
0	0	0	0	0	0	(d): class 4
0	0	2	0	58	0	(e): class 5
0	0	2	0	0	32	(f): class U

Evaluation on testing data:

(a)	(b)	(c)	(d)	(e)	(f)	<-classified as
0	0	0	0	0	0	(a): class 1
0	9	2	0	0	0	(b): class 2
0	2	74	0	0	0	(c): class 3
0	0	0	0	0	0	(d): class 4
0	0	0	0	7	0	(e): class 5
0	0	0	0	0	6	(f): class U

See5+

Newly learned rules:

Rule 5: (249/249)
condition = S
surface-quality = E
→ class 3

Rule 6: (56/56)
surface-quality = E
bw/me = B
→ class 3

Rule 11: (33/33)
hardness(continuous) = - 22.5
bw/me = M
→ class 3

Rule 12: (167/167)
condition = S
thick(continuous) = 0.6005-0.7995
→ class 3

Rule 20: (69/69)
steel = A
bl = Y
→ class 3

Evaluation on training data:

(a)	(b)	(c)	(d)	(e)	(f)	<-classified as
5	2	1	0	0	0	(a): class 1
0	80	8	0	0	0	(b): class 2
0	1	605	0	0	0	(c): class 3
0	0	0	0	0	0	(d): class 4
0	0	0	0	60	0	(e): class 5
0	0	2	0	0	32	(f): class U

Evaluation on testing data:

(a)	(b)	(c)	(d)	(e)	(f)	<-classified as
0	0	0	0	0	0	(a): class 1
0	9	2	0	0	0	(b): class 2
0	0	76	0	0	0	(c): class 3
0	0	0	0	0	0	(d): class 4
0	0	0	0	7	0	(e): class 5
0	0	0	0	0	6	(f): class U

Chapter 5
Conclusions

In this work we propose an approach to using prior knowledge in rule induction. The concept knowledge in our work includes not only domain knowledge, user's constraints, and the existing knowledge. The objective of learning is also changed from searching for a new hypothesis fitting well training data, to improving a prior hypothesis to achieve a better one. By this means knowledge is reused, enriched, and extended after learning iterations.

The approach consists of two main components. The first component is the *seed generator* that takes prior knowledge as an input factor and generates a set of rule seeds. These seeds are used to narrow the search for new rules in the second component, the *rule specialization* process. The task of the specialization is to search for new rules starting from rule seeds in rule space. The newly learned rules are then used to update the prior rule set.

To illustrate the effectiveness, the proposed approach is applied to improve rule-based classifiers learned by two rule learning algorithms. In the first experiment we use the results of LUPC, a homemade sequential covering rule induction algorithm, as prior knowledge. Experiment on 26 datasets from the UCI machine learning repository shows that the proposed approach reduces the error rate of LUPC on 10 of 26 domains. Comparing with the state-of-the-art rule learning systems See5 and CBA, the proposed approach helps LUPC to achieve a very competitive performance: LUPC+ wins 7, See5 tree wins 6, CBA2 wins 6 on total 26 datasets. In the second experiment, we use the rule sets produced by the See5 system, a commercial product of the most famous learning algorithm ID3 and C4.5. Achieving improvements on 3 of 9 datasets is the result of this experiment.

Chapter 6
Future Works

This research opens a number of research directions to go further in the future.

First, the seed generator used in the above experiments is quite simple. It uses only attributes that are not referred to in the prior rule sets. Although the result of experiments are interesting: with very simple techniques, the proposed approach can improve effectively rule-based classifiers learned by state-of-the-art algorithms, but in several cases when all attribute are used by prior rule set, the proposed approach can do nothing. In such a case, the set of *rule seeds* is empty and no rule is discovered. More over, with this way of building *seed generator*, learning process is iterated at most twice because after applying the proposed approach all attributes are considered and the next iterations will produce nothing more.

Second, the way of using newly learned rules is also need to be improved. A newly learned rule is used when it effectively replaces a prior rule, or extends the prior rule set. But in rule-based classifier, a new sample may be classified by the vote of several rules when it is fired by more than one rule. This means that classification result depends on the whole rule set, not on the individual. In our experiment the newly learned rules are individually considered in improving the prior rule set. This problem needs to be improved in the future.

The third direction concerns about the availability of new data after a classifier has been constructed, a common occurrence in practical applications [21]. It may be that the existing classifier deals correctly with all the new data, so we don't have to do nothing. If we are not so fortunate, there are two alternatives: ignoring the new data, or discard the previous classifier, add the new data to the training set, and build a new classifier. Neither of these alternatives is attractive if the classifier seems likely to benefit from new data, yet the computational cost of developing a classifier from scratch is high [21].

Adapting the new data by extending the prior rule set is an open direction of this research.

The fourth direction concerns about the problem of discovering interesting rules, one of the areas of slow progress in data mining [19]. One important criterion of an interesting rule is the novelty, a subjective measure that cannot be drawn from data. There are many works trying to deal with this problem like ranking discovered rules according to some interestingness measure, or filter out what is not interesting. All of these works are done in the post processing step rule sets are discovered from data. They give users no way to participate into the mining process. The proposed approach can help the user to do data mining incrementally, discover new rules that are different from existing ones, and interactively narrow the search in rules space through the seed generator while exploring the prior rule set.

References

[1] R. Agrawal, T. Imielinski, A. Swami. *Mining Associations between Sets of Items in Massive Databases*. Proc. of the ACM-SIGMOD 1993 Int'l Conference on Management of Data, Washington D.C., pages 207-216, May 1993.

[2] R. J. Bayardo and R. Agrawal. *Mining the Most Interesting Rules*. In Proc. of the 5th Int. Conf. on Knowledge Discovery and Data Mining, pages 145--154, San Diego, CA, USA, Aug. 1999.

[3] R. J. Bayardo Jr., R. Agrawal, and D. Gunopulos. *Constraint-based rule mining in large, dense databases*. Data Mining and Knowledge Discovery, 4(2/3): pages 217-240, 2000.

[4] P. Clark and T. Niblett. *The **CN2** induction algorithm*. Machine Learning, 3(4): pages 261--284, March 1989.

[5] P. Clark and R. Boswell. *Rule induction with CN2: Some recent improvements*. In Proceedings of the Fifth European Working Session on Learning, pages 151--163. Springer, 1991.

[6] U. Fayyad, G. **Piatetsky**-Shapiro, and P. Smyth. *From data mining to knowledge discovery: An overview*. In Advances in Knowledge Discovery and Data Mining, pp. 1--34. AAAI Press, Menlo Park, CA, 1996.

[7] R.J. Hilderman and H.J. Hamilton. *Knowledge discovery and interestingness measures: A survey*. Technical Report CS 99-04, Department of Computer Science, University of Regina, October 1999.

[8] T.B. Ho and D.D. Nguyen. *Chance Discovery and Learning Minority Classes.* Journal of New Generation Computing, Springer, Vol. 21, No. 2, pages 147-160, February 2003.

[9] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. *Finding interesting rules from large sets of discovered association rules*. In CIKM-94, 401 -- 407, November 1994.

[10] J. Li and L. Wong. *Solving the Fragmentation Problem of Decision Trees by Discovering Boundary Emerging Patterns*. In proceedings of the 2002

IEEE International Conference in Data Mining, pages 653-656. 9-12 December 2002, Maebashi City, Japan.

[11] Bing Liu, Yiming Ma, C-K Wong. *Classification Using Association Rules: Weaknesses and Enhancements*. To appear in Vipin Kumar, et al, (eds), Data mining for scientific applications, 2001.

[12] Bing Liu, Wynne Hsu, Yiming Ma. *Integrating Classification and Association Rule Mining*. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98, full paper), New York, USA, 1998.

[13] Ryszard S. Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac. *The multipurpose incremental learning system AQ15 and its testing application to three medical domains*. In AAAI-86, 1041 -- 1045, 1986.

[14] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. http://www-2.cs.cmu.edu/~tom/mlbook.html.

[15] P. M. Murphy and D.W. Aha. *UCI repository of Machine Learning Databases*. University of California, Department of Information and Computer Science, Irvine CA, 1994.

[16] G. Pagallo and D. Haussler. *Boolean feature discovery in empirical learning*. Machine Learning, 5:71-99, 1990.

[17] M. Pazzani, C. Brunk, and G. Silverstein. *A knowledge-intensive approach to learning relational concepts*. Proceedings of the Eighth International Workshop on Machine Learning (pp. 432-436). Evanston, IL-Morgan Kaufmann, 1991.

[18] Gregory Piatetsky-Shapiro and Christopher J. Matheus. *The interestingness of deviations*. In KDD-94, 25 -- 36, July 1994.

[19] Gregory Piatetsky-Shapiro. *Knowledge Discovery in Databases: Ten years after*. SIGKDD Explorations 1(2): 59-61 (2000).

[20] J. R. Quinlan and R. M. Cameron-Jones. *FOIL: A midterm report*. In Pavel B. Brazdil, editor, Machine Learning: ECML-93, Vienna, Austria, 1993.

[21] J. R. Quinlan. C4.5: *Programs for machine learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[22] J. R. Quinlan. *C5.0: An Informal Tutorial*. RuleQuest, 1998. http://www.rulequest.com/**see5**-unix.html.

[23] Sigal Sahar. *Interestingness via what is not interesting.* In Fifth International Conference on Knowledge Discovery and Data Mining (S. Chaudhuri and D. Madigan, eds.), (San Diego, CA, USA), pp. 332-336, ACM Press, 1999.

[24] Sigal Sahar. *On Incorporating Subjective Interestingness Into the Mining Process*. Proc. of IEEE Int'l Conference on Data Mining, page 681-684. Maebashi, Japan, 9-12 December 2002.

[25] A. Silberschatz and A. Tuzhilin. *On subjective measures of interestingness in knowledge discovery*. Proceedings of the First International Conference on Knowledge Discovery and Data Mining, pages 275-281, 1995.

[26] R. Vilalta, G. Blix, and L. A. Rendell. *Global data analysis and the fragmentation problem in decision tree induction*. In 9th European Conference on Machine Learning, pages 312--326. Lecture Notes in Artificial Intelligence, Vol. XXX, Springer-Verlag, Heidelberg.

List of Contributions

1. Ho T.B., Nguyen, D.D. (2003). Chance Discovery and Learning Minority Classes, *Journal of New Generation Computing*, Springer, Vol. 21, No. 2, February 2003, 147-160.
2. Ho, T.B., Nguyen, T.D., Nguyen, D.D., Kawasaki, S. (2001). Visualization Support for User-Centered Model Selection in Knowledge Discovery and Data Mining, *International Journal of Artificial Intelligence Tools*, World Scientific, Vol. 10, No. 4, 691-713.
3. Ho, T.B., Nguyen, T.D., Nguyen, D.D. (2002). A User-Centered Visual Approach to Data Mining. The system D2MS, *Intelligent Information Processing*, M. Musen, B. Neumann, R. Studer (Eds.), Kluwer Academic Publishers, 213-224.
4. Ho, T.B., Nguyen, T.D., Nguyen, D.D., Kawasaki, S. (2002). Visualization of Data and Knowledge in the Knowledge Discovery Process, *Active Mining: New Directions of Data Mining*, H. Motoda (Ed.), IOS Press, 229-238.
5. Ho, T.B., Nguyen, D.D., Nguyen, T.D., Kawasaki, S. (2002). Extracting Knowledge from Hepatitis Data with Temporal Abstraction, *IEEE Conference on Data Mining, Workshop on Active Mining*, Maebashi, December 9-12, 91-96.
6. Kawasaki, S., Saitou, A., Nguyen, D.D., Ho, T.B. (2002). Mining from Medical Data: Case-Studies in Meningitis and Stomach Cancer Domains, *KESS 2002, 6th International Conference on Knowledge-based Intelligent Information & Engineering Systems*, Crema, September 16-18, 2002, 547-551.
7. Ho, T.B., Nguyen, D.D., Kawasaki, S. (2002). Learning Minority Classes in Unbalanced Datasets, *Third International Conference on Parallel and Distributed Computing*, Kanazawa, September 3-6, 196-203.
8. Ho, T.B., Nguyen, D.D., Kawasaki, S., Nguyen, T.D. (2002). Extracting Knowledge from Hepatitis Data with Temporal Abstraction, *ICML/PKDD 2002 Discovery Challenge, 6th European Conference on Principles of Data Mining and Knowledge Discovery PKDD 2000*, Hensinki, 19-23 August.
9. Ho, T.B., Nguyen, T.D., Nguyen, D.D. (2002). Visualization Support for a User-Centered KDD Process, *ACM International Conference on Knowledge Discovery and Data Mining KDD-02*, Edmonton, 23-26 July, 519-524.

10. Ho, T.B., Saito, A., Kawasaki, S., Nguyen, D.D., Nguyen, T.D. (2002). Failure and Success Experience in Mining Stomach Cancer Data, *International Workshop Data Mining Lessons Learned*, Inter. Conf. Machine Learning 2002, Sydney, 8-12 July, 40-47.
11. Kawasaki, S., Nguyen, D.D., Nguyen, T.D., Ho, T.B. (2002). Study of Hepatitis Data by Visual Data Mining System D2MS, *JSAI SIG-KBS-A201 Workshop Active Data Mining*, Pusan, 23-24 May, 43-48.
12. Nguyen, T.D., Ho, T.B., Nguyen, D.D. (2002). Data and Knowledge Visualization in the Knowledge Discovery Process*, 5th International Conference Recent Advances in Visual Information Systems,* Taiwan, 11-13 March, *Lecture Note in Computer Science 2314*, Springer, 311-321.
13. Ho, T.B., Nguyen, D.D., Kawasaki, S. (2001). Mining Prediction Rules from Minority Classes, *14th International Conference on Applications of Prolog (INAP2001), International Workshop Rule-Based Data Mining RBDM 2001*, Tokyo, 20-22 October, 254-264.
14. Ho, T.B., Kawasaki, S., Nguyen, D.D. (2001). Extracting Predictive Knowledge from Meningitis Data by Integration of Rule Induction and Association Mining, *International Workshop Challenge in KDD*, 22 May, Shimane, Japan, 25-32, *Lecture Notes in Artificial Intelligence 2253,* Springer 2001, 508-515.