An Inductive Learning Algorithm for Production Rule Discovery

Mehmet R. Tolun
Department of Computer Engineering
Middle East Technical University
Inonu Bulvari
Ankara
Turkey 06531
{tolun@ceng.metu.edu.tr}

Saleh M. Abu-Soud
Department of Computer Science
Princess Sumaya University College for Technology
Royal Scientific Society
P.O. Box 925819
Amman - Jordan
{abu-soud@risc.rss.gov.jo}

Abstract

Data mining is the search for relationships and global patterns that exist in *large databases*. One of the main problems for data mining is that the number of possible relationships is very large, thus prohibiting the search for the correct ones by validating each of them. Hence we need intelligent data mine tools, as taken from the domain of machine learning.

In this paper we present a new inductive machine learning algorithm called ILA. The system generates rules in canonical form from a set of examples. We also describe application of ILA to a range of data sets with different number of attributes and classes. The results obtained show that ILA is more general and robust than most other algorithms for inductive learning. Most of the time, the worst case of ILA appears to be comparable to the best case of some well-known algorithms such as AQ and ID3, if not better.

Keywords: *Machine Learning, Induction, Knowledge Discovery, Inductive Learning, Symbolic Learning Algorithm.*

1. Introduction

Using data mining or knowledge discovery techniques, automated tools can be designed for learning rules from databases. In the recent past, the application of data mining[Frawley, Piatetsky-Shapiro, and Matheus, 1991] has acquired considerable significance. Researchers have developed and applied machine learning techniques to automatically acquire knowledge from large databases and to learn rules for expert systems. The domains of data mining and machine learning intersect as they both deal with extracting interesting and previously unknown knowledge from databases[Deogun et al., 1997]. [Holsheimer and Siebes, 1994] state that, in fact, when a database is used as a training set, the learning process is called *data mining*.

One of the approaches to inductive machine learning that is often used is to form a *decision tree* from a set of training examples. Decision tree-based approaches to classification learning are typically preferred because they are efficient and, thus, can deal with a large number of training examples. However, the decision tree approaches do not always produce the most general production rules. Therefore, there are many algorithms which do not employ decision trees, for instance, AQ family of algorithms that utilize disjunction of features values covering positive examples[Michalski, 1983]. Furthermore several others use multiple learning algorithms within a single learning system as in FCLS system which combines rules with specific examples in a best-match framework[Zhang, 1990].

So far the best known algorithm which takes a set of examples as input and produces a decision tree which is consistent with examples has been Quinlan's ID3 algorithm[Quinlan, 1983]. This was derived from the *Concept Learning System(CLS)* algorithm described by [Hunt, Maria & Stone, 1966]. ID3 has two new features that improved the algorithm. First an information-theoretic splitting heuristic was used to enable small and efficient decision trees to be constructed. Second, the incorporation of windowing process that enabled the algorithm to cope with large training sets[Thornton, 1992]. With these advantages ID3 has become a mainstream of symbolic learning approaches and a number of derivatives are proposed by many researchers. For example, ID4 which incrementally builds a decision tree based on individually observed instances by maintaining positive and negative instance counts of every attribute that could be a test attribute[Schlimmer and Fisher, 1986], ASSISTANT 86 which handles induction bias caused by mutual information-theoretic measure that filters out irrelevant features [Cestnik, Kononenko, and Bratko, 1987], ID5 which provides an incremental method for building ID3 type decision trees but differs from ID4 in its method for replacing the test attribute [Utgoff, 1988], GID3 and GID3* which does not branch on each value of the chosen attribute to reduce the unnecessary sub-division of data [Irani, Cheng, Fayyad, and Qian, 1993], and C4.5 which handles uncertain data [Quinlan, 1993] with the expense of increasing classification error rate.

ID3 is a top-down, nonbacktracking decision tree algorithm. One of the problems with ID3 is that the decision tree produced overfits the training examples because it performs a stepwise splitting that attempts to optimize at each individual split, rather than on an overall basis [McKee, 1995]. This leads to decision trees that are too specific because they used unnecessary or irrelevant conditions. Hence this affects the ability to classify unknown examples or examples with incomplete attributes. Typically, to overcome overfit in decision trees, the tree is pruned [Breiman et al., 1984]. Though this method may not work adequately for an inconclusive data set which require probabilistic rather than categorical classification. [Uthurusamy et al., 1991] proposed an algorithm which improves on ID3 to make it applicable to inconclusive data sets. Another problem with ID3 relates to the fact that for applications involving a large number of training examples which cannot be kept in computer's main memory at once. The algorithm can work with a "representative" sample from the training set, called *windowing*, which however, cannot guarantee to yield the same decision tree as would be obtained from the complete set of training examples. In this case the decision tree would be unable to classify all examples correctly [Carter and Catlett, 1987].

AQ is another well-known inductive learning algorithm. The original AQ does not handle uncertainty very well. Existing implementations, such as AQ11[Michalski and Larson, 1978], AQ15[Michalski et al., 1986] handle noise with pre and post-processing techniques. The basic AQ algorithm however, heavily depends on specific training examples during search(the algorithm actually employs a beam search). The AQ algorithm when generating a conjunction of attribute value conditions (called a *complex*), also performs a general-to-specific search for the best complex. The algorithm only considers specializations that exclude some particular covered negative example from the complex while ensuring some particular ‘seed’ positive example remains covered, iterating until all negative examples are excluded. As a result , AQ searches only the space of complexes that are completely consistent with the data.

CN2 algorithm[Clark and Niblett, 1989], which is an adaptation of the AQ algorithm, retains the same heuristic search method of the AQ algorithm but on the other hand, removes its dependence on specific examples during search and also extends AQ’s search space to include rules that do not perform perfectly on the training data. Both AQ and CN2 are rule induction systems that are regarded as non decision tree approaches.

Other algorithms include OC1[Murthy, Kasif, and Salzberg, 1994] which is a system for induction of oblique decision trees suitable for domains where attributes have numeric values, and RULES[Pham and Aksoy, 1995] which is a rule induction algorithm with an ability to classify unseen examples. The disadvantage of RULES lies in the increased number of rules generated to handle such data.

We present a production rule induction system called ILA(Inductive Learning Algorithm) which produces IF-THEN rules directly from a set of training examples in a general-to-specific way (i.e. starting off with the most general rule possible and producing specific rules whenever it is deemed necessary). ILA eliminates all unnecessary and irrelevant conditions from the extracted rules and therefore its rules are more simple and general than those obtained from ID3 and AQ. ILA also produces rules fewer in number than ID3 and AQ most of the time. The generality of rules increases the classification capability of ILA. A rule becomes more general as the number of conditions on its IF-part becomes fewer. A general rule also help in classifying incomplete examples in which one or more attributes may be unknown. They also embody the general patterns within the database. The rules can be used to interpret and understand the active mechanisms underlying the database.

We describe the application of ILA to a range of problems demonstrating the performance of the algorithm on three domains from UCI repository[1]. The results of ILA are compared to those of ID3 and AQ.

[1]University of California Irvine Repository of Machine Learning Databases and Domain Theories via anonymous ftp to charlotte.ics.uci.edu : pub/machine-learning-databases.

2. The Inductive Learning Algorithm(ILA)

Now that we have reviewed ID3 and AQ we can turn to ILA, a new inductive algorithm for generating a set of classification rules for a collection of training examples. The algorithm works in an iterative fashion, each iteration searching for a rule that covers a large number of training examples of a single class. Having found a rule, ILA removes those examples it covers from the training set by marking them and appends a rule at the end of its rule set. In other words our algorithm works on a rules-per-class basis. For each class, rules are induced to separate examples in that class from examples in all the remaining classes. This produces an ordered list of rules rather than a decision tree. The advantages of the algorithm can be stated as follows:

- The rules are in a suitable form for data exploration; namely a description of each class in the simplest way that enables it to be distinguished from the other classes.
- The rule set is ordered in a more modular fashion which enables to focus on a single rule at a time. Decision trees are hard to interpret, particularly when the number of nodes is large.

Feature space selection in ILA is stepwise forward. ILA also prunes any unnecessary conditions from the rules.

ILA is quite unlike ID3 or AQ in many respects. The major difference is that ILA does not employ an information theoretic approach and concentrates on finding only relevant values of attributes, while ID3 is concerned with finding the attribute which is most relevant overall, even though some values of that attribute may be irrelevant. Also ID3 divides a training set into homogeneous subsets without reference to the class of the subset, ILA must identify each specific class.

ILA to be described in section 2.2 starts processing the training data by dividing the example set into sub-tables for each different class attribute value. Afterwards it makes comparisons between values of an attribute among all sub-tables and counts their number of occurrences. ILA is designed for handling discrete and symbolic attribute values in an attempt to overcome the attribute selection problem. Continuous-valued attributes can be discretized during decision tree or rule generation by partitioning their ranges using *cut points*[Fayyad and Irani, 1994]. But most of the time the motivation for discretization is to improve the learning speed of the algorithm when continuous(numeric) attributes are encountered[Ching, Wong and Chan, 1995].

Starting off with the maximum number of occurrence combinations it then immediately begins generating rules until it marks all rows of a sub-table classified. ILA then repeats this process for all values of each attribute of each sub-table. Finally, all possible IF-THEN rules are derived when there are no unmarked rows left for processing.

2.1 **General Requirements**

1. The examples are to be listed in a table where each row corresponds to an example and each column contains attribute values.
2. A set of *m* training examples, each example composed of *k* attributes and a class attribute with *n* possible decisions.
3. A rule set, R, with an initial value of ϕ.
4. All rows in the table are initially unmarked.

2.2 **The Inductive Learning Algorithm(ILA)**

Step1: Partition the table which contains *m* examples into *n* sub-tables. One table for each possible value of the class attribute.

(* steps 2 through 8 are repeated for each sub-table *)

Step2: Initialize attribute combination count j as j = 1.

Step3: For the sub-table under consideration, divide the attribute list into distinct combinations, each combination with *j* distinct attributes.

Step4: For each combination of attributes, count the number of occurrences of attribute values that appear under the same combination of attributes in unmarked rows of the sub-table under consideration but at the same time that should not appear under the same combination of attributes of other sub-tables. Call the first combination with the maximum number of occurrences as max-combination.

Step5: If max-combination = ϕ, increase *j* by *1* and go to Step 3.

Step6: Mark all rows of the sub-table under consideration, in which the values of max-combination appear, as classified.

Step7: Add a rule to R whose left hand side comprise attribute names of max-combination with their values separated by AND operator(s) and its right hand side contains the decision attribute value associated with the sub-table.

Step8: If all rows are marked as classified, then move on to process another sub-table and go to Step 2. Otherwise(i.e., if there are still unmarked rows) go to Step 4. If no sub-tables are available, exit with the set of rules obtained so far.

3. A Description of the Inductive Learning Algorithm

ILA is a rather simple algorithm for extracting production rules from a collection of examples. An example is described in terms of a fixed set of attributes, each with its own set of possible values. In describing ILA we shall make use of three different training example sets(i.e. object, weather and season classifications).

As an illustration of the operation of ILA, let us consider the training set for object classification given in Table 1, consisting of seven examples (i.e. m=7) with three attributes (k=3) and one decision(class) attribute with two possible values, {yes, no}, (n=2). In this example, “Size”, “Color” and “Shape” are attributes with sets of possible values {small, medium, large}, {red, blue, green}, and {brick, wedge, sphere, pillar} respectively.

TABLE 1. Object Classification Training Set[Thornton, 1992].

Example no.	Size	Color	Shape	Decision
1	medium	blue	brick	yes
2	small	red	wedge	no
3	small	red	sphere	yes
4	large	red	wedge	no
5	large	green	pillar	yes
6	large	red	pillar	no
7	large	green	sphere	yes

Since *n* is two, the first step of the algorithm generates two sub-tables which are shown in Table 2.

TABLE 2. Sub-Tables of The Training Set Partitioned According to Decision Classes.

Sub-Table 1					
Example no.		**Size**	**Color**	**Shape**	**Decision**
old	**new**				
1	1	medium	blue	brick	yes
3	2	small	red	sphere	yes
5	3	large	green	pillar	yes
7	4	large	green	sphere	yes
Sub-Table 2					
Example no.		**Size**	**Color**	**Shape**	**Decision**
old	**new**				
2	1	small	red	wedge	no
4	2	large	red	wedge	no
6	3	large	red	pillar	no

Applying the second step of the algorithm, we consider the first sub-table in Table 2:
For j=1, the list of attribute combinations comprises: {size}, {color}, and {shape}.

For the combination {size} the attribute value “medium” appears in sub-table 1 but not in sub-table 2, so the value of max-combination becomes “medium”. Since other available attribute values “small” and “large” appear in both sub-table 1 and sub-table 2 they are not considered at this step. The occurrence of {size} attribute value “medium” is noted as one times and next combination is evaluated with max-combination set to “green”. For combination {color} we have “blue” with an occurrence of one times and “green” with an occurrence of two times. Continuing further with the combination {shape}, we have “brick” with one occurrence and “sphere” with two occurrences. At the end of step 4, we have {color} attribute value “green” and {shape} attribute value “sphere” marked with maximum number of occurrences. Here either of the attribute values can be selected, because both of them can classify the same number of training examples. The algorithm always selects the first one(i.e. “green” in this case) by default, and this will make max-combination to keep its current value of “green”. Rows 3 and 4 are marked as classified in sub-table 1, since the value of max-combination is repeated in these two rows, the following production rule(Rule 1) is extracted:

<u>Rule1</u>
IF color is green THEN the decision is yes.

Now, ILA algorithm repeats step 4 through step 8 on the rest of the unmarked examples in sub-table 1(i.e. rows 1 and 2). By applying these steps again we have “medium” attribute value of {size}, “blue” attribute value of {color}, “brick” and “sphere” attribute values of {shape} occurring once. Since the number of occurrences are the same, the algorithm applies the default rule and selects the first one considered(i.e. “medium” attribute value of {size}). Then the following rule(Rule 2) is added to the rule set:

<u>Rule2</u>
IF size is medium THEN the decision is yes.

The first row in sub-table 1 is marked as classified and steps 4 through 8 are applied again on the remaining row(i.e. the second row). Here we have “sphere” attribute value of {shape} occurring once, so the third rule is extracted:

<u>Rule3</u>
IF shape is sphere THEN the decision is yes.

By marking the second row as classified all of the rows in sub-table 1 are now marked as classified and we proceed on to sub-table 2. The “wedge” attribute value of {shape} occurs twice in the first and second rows in sub-table 2. So, these two rows are marked as classified and Rule 4 is appended to the rule list.

<u>Rule4</u>
IF shape is wedge THEN the decision is no.

In the remaining row in sub-table 2(i.e. the third row) we have {size} attribute with a value of “large” that appears also in sub-table 1. So according to the algorithm this cannot be considered. The same applies to “red” value of {color} and “pillar” value of {shape} attributes. In this case, ILA increases *j* by *1*, and generates 2-attribute combinations, {size and color}, {size and shape}, and {color and shape}. The first and third combinations satisfy the conditions as they both appear in sub-table 2 but not in sub-table 1 for the same attributes. The “large pillar” value of {size and shape} combination is ignored because it already appears in sub-table 1. According to this, we can choose either the first or the third combination but the default rule allows us to select the first one. The following rule(Rule 5) is extracted and the third row in sub-table 2 is marked as classified:

Rule5
IF size is large AND color is red THEN the decision is no.

Now, since all of the rows in sub-table 2 are marked as classified and no other sub-table is available, the algorithm terminates.

3.1 Comparison of ILA and ID3

Several distinctions between ILA and ID3 are pointed out earlier in Section 2. For comparison purposes, the rules resulting from applying ID3 on the same training set and the ones produced by ILA are presented in Table 3.

TABLE 3. A Comparison Between Rules Generated by ID3 and ILA.

Algorithm	Rule No.	Rule

ID3
ILA
	1	IF color=green AND shape=pillar THEN yes
		IF color=green THEN yes

ID3
ILA
	2	IF shape=brick THEN yes
		IF size=medium THEN yes

ID3 ILA	3	IF shape=sphere THEN yes IF shape=sphere THEN yes
ID3 ILA	4	IF shape=wedge THEN no IF shape=wedge THEN no

ID3
ILA
	5	IF color=red AND shape=pillar THEN no
		IF size=large AND color=red THEN no

It is evident from Table 3 that the two algorithms generate the same number of rules but Rule 1 extracted by ILA is simpler than the same rule generated by ID3 because the latter has an unnecessary condition(i.e. shape = pillar). Clearly rules 2, and 5 are also different in both sets of rules but with the same level of complexity. However, ILA could generate these same two rules, as for example, attribute value “brick” was one of the choices. But we gain nothing if we change this choice since in both algorithms the two rules have the same level of specificity and classify the respective examples correctly.

Let us consider another training set from [Quinlan, 86] in Table 4:

TABLE 4. Weather Training Examples.

Example	Outlook	Temperature	Humidity	Windy	Class
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P

11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

where P = Positive and N = Negative.

Applying ILA on the training set given in Table 4 we obtain the following rules:

Rule1: IF outlook is overcast THEN the decision is Positive.
Rule2: IF outlook is sunny AND humidity is high THEN the decision is Negative.
Rule3: IF outlook is rain AND windy is true THEN the decision is Negative.
Rule4: IF outlook is rain AND windy is false THEN the decision is Positive.
Rule5: IF outlook is sunny AND humidity is normal THEN the decision is Positive.

For this example, these are the same rules generated by ID3. In this case, extracted rules do not contain any unnecessary conditions. This is actually the worst case of ILA. The worst case of ILA happens when it generates rules that do not contain unnecessary conditions to eliminate.

To compare ILA with a much recent rule extraction system called RULES[Pham and Aksoy, 95] and also with ID3 let us consider the training example set for classifying the seasons given in Table 5.

TABLE 5. The Training Set for Season Classification Problem[Pham and Aksoy, 95].

Example	Weather	Trees	Temperature	Season(Class)
1	rainy	yellow	average	autumn
2	rainy	leafless	low	winter
3	snowy	leafless	low	winter
4	sunny	leafless	low	winter
5	rainy	leafless	average	autumn
6	rainy	green	high	summer
7	rainy	green	average	spring
8	sunny	green	average	spring
9	sunny	green	high	summer
10	sunny	yellow	average	autumn
11	snowy	green	low	winter

The rules resulting from applying ID3 and RULES on the same training set and the ones produced by ILA are presented in Table 6. Again ILA generates the same number of rules but one rule (Rule 3) being simpler than that has been generated by

ID3. The unnecessary condition that ID3 generated is “temperature is average”, which is eliminated as described in Table 6.

On the other hand we note that RULES generates seven rules from the same training set, the first five of them being the same as the rules generated by ILA while rule 6 and rule 7 are generated neither by ID3 nor by ILA.

TABLE 6. A Comparison Between Rules Generated by ID3, RULES and ILA.

Algorithm	Rule No.	Rule
ID3 RULES ILA	1	IF temperature = low THEN winter IF temperature = low THEN winter IF temperature = low THEN winter
ID3 RULES ILA	2	IF temperature = high THEN summer IF temperature = high THEN summer IF temperature = high THEN summer

ID3
RULES
ILA
	3	IF trees = yellow AND temperature = average THEN autumn
		IF trees = yellow THEN autumn
		IF trees = yellow THEN autumn

ID3 RULES ILA	4	IF trees = leafless AND temperature = average THEN autumn IF trees = leafless AND temperature = average THEN autumn IF trees = leafless AND temperature = average THEN autumn
ID3 RULES ILA	5	IF trees = green AND temperature = average THEN spring IF trees = green AND temperature = average THEN spring IF trees = green AND temperature = average THEN spring
ID3 RULES ILA	6	— IF weather = snowy THEN winter —
ID3 RULES ILA	7	— IF weather = sunny AND trees = leafless THEN winter —

4. Evaluation of Inductive Learning Algorithm(ILA)

The evaluation of learning systems is a complex task. One way it can be assessed is in terms of its performance on specific tasks which are assumed to be representative of the range of tasks which the system is intended to perform[Cameron-Jones and Quinlan, 1994].

For evaluation purposes of ILA we have mainly used two parameters: number of rules generated and average number of conditions. Number of rules has been included as an evaluation parameter because the aim here is to produce the minimum number of rules as possible that classify the examples in the training set successfully. But a good algorithm should produce rules that not only classify the cases in the training set but also classify the unseen examples. So, the second parameter, that is the average number of conditions, helps to give indication whether the algorithm can classify more unseen examples or not. It can be easily realized that a rule with fewer number of conditions can classify more examples, thus making the average number of conditions a suitable parameter for the assessment of induction algorithms.

ILA extracts rules in canonical form, i.e. in the most general and simple form. This is because ILA eliminates all unnecessary conditions from the rules and generates the minimum number of rules that some other systems fail to produce, such as RULES(cf. Table 6). The generality of rules extracted increases the classification capability of an algorithm. A rule becomes more general as the number of conditions on its LHS becomes fewer, in other words, as the number of attributes becomes fewer. A general rule also help in classifying incomplete examples in which one or more attributes are unknown. For example in the Season Classification Problem, if an unknown example that has an attribute-value pair "Trees are yellow" but has no value for {Temperature} will be classified correctly by ILA, but not by ID3 even though both algorithms produce the same number of rules as seen in Table 7. Using the proposed algorithm the opportunity to classify unknown examples(examples not listed in the training set) therefore, becomes very high.

In this section we first describe the characteristics of training sets used in evaluating ILA against ID3 and AQ algorithms. Next we outline experiments followed by a discussion of evaluation parameters and a summary of results obtained. Finally, elimination of unnecessary conditions and classification of unseen examples are described.

4.1 Training Sets

We used three different training sets, namely Balloons, Balance and Tic-tac-toe in our experiments with ILA.

Table 7 summarizes the characteristics of the three different domains used in the experiments. We have obtained those training sets from the University of California Irvine Repository of Machine Learning Databases and Domain Theories via anonymous ftp to charlotte.ics.uci.edu : pub/machine-learning-databases.

TABLE 7. Description of the Domains.

Domain Characteristic	**Balloons**	**Balance**	**Tic-tac-toe**
Number of attributes	4+1	4+1	9+1
Number of examples	16	625	958
Average Values per attribute	2	5	3
Number of Class Values	2	3	2
Distribution of Examples Among Class Values	1. 25% are T 2. 75% are F	1. 46.08% are L 2. 07.84% are B 3. 46.08% are R	1. 65.3% are P 2. 34.7% are N

4.2 Experiments

The algorithm outlined in Section 2 has been embodied in a new rule induction system which takes as input a set of training examples entered as a file of ordered sets of attribute values, each example being terminated by a decision attribute. The results are output as individual rules for each of the classifications listed in terms of the described attributes.

The results of applying ILA on these training sets are compared with three well-known algorithms in inductive learning, namely ID3, and AQ. We conducted two different sets of experiments on the ILA. In the first experiment set performance of ILA is assessed using two criteria-number of rules generated by the algorithm and the average number of conditions on the IF-parts of rules. While in the second experiment set we specifically measured relative performances of ID3, AQ, and ILA on partitioned data to observe classification capability of the algorithms on unseen example data.

4.2.1 Discussion

The number of rules is considered as an evaluation parameter because the main aim is to produce the minimum number of rules as possible that can classify all of the examples in the training set. The second parameter that has great significance in the evaluation process of inductive learning systems is the capability of the system to classify as much unseen examples as possible. As discussed in section 3.1, the average number of conditions can be used for this purpose successfully since a system that produces fewer number of conditions can classify more examples

TABLE 8. Summary of the Results Obtained.

Training Set	Algorithm	No. of Rules	Average no. of Conditions
Balloons	**ID3**	3	1.67
	AQ	3	1.33
	ILA	3	1.33
Balance	**ID3**	401	3.85
	AQ	312	3.53
	ILA	303	3.41
Tic-tac-toe	**ID3**	218	5.78
	AQ	86	4.92
	ILA	32	3.5

Table 8 shows the number of rules and average number of conditions in the resulting rules for the four algorithms for each training set. It is clear that ILA can produce less number of rules and less number of conditions on the IF-part of the rules than those generated by ID3 and AQ algorithms. So it is expected that ILA classifies more unseen examples than above mentioned algorithms, as we shall discuss later.

A closer look at the figures in Table 8 shows that the results are almost the same for the smallest data set tested, namely balloons. However, it is noted that as the training sets get larger, ILA gives better results for both parameters in comparison to ID3 and AQ algorithms.

4.2.2 Elimination of Unnecessary Conditions

From the previous discussion it is clear that ID3 and AQ algorithms produce rules that contain unnecessary conditions. ILA, on the other hand, eliminates such conditions. It is also clear from Table 8 that ILA produces rules which are significantly less in number than those produced by these algorithms. In order to see the reason, let us consider Tic-tac-toe training set, for which ID3 produces 218 rules while ILA produce only 32 rules. Let us consider the following set of rules produced by ID3:

IF P_1 = x & **P_3 = x & P_5 = x & P_7 = x** & P_9 = o THEN Class is Positive
IF P_1 = o & **P_3 = x & P_5 = x & P_7 = x** & P_9 = x THEN Class is Positive
IF P_1 = o & **P_3 = x & P_5 = x & P_7 =** x & P_9 = o THEN Class is Positive
IF P_1 = x & **P_3 = x & P_5 = x & P_7 = x** & P_9 = b THEN Class is Positive
IF P_1 = b & **P_3 = x & P_5 = x &** P_6 = x & **P_7 = x** & P_9 = o THEN Class is Positive

All of these rules are correct and classify the examples in the training set correctly, but all of them contain unnecessary conditions. ILA eliminates these conditions and produces only the following rule instead of five:

IF P_3 = x & P_5= x & P_7 = x THEN Class is Positive

In this case, ID3 produced 5 rules with 5.2 as the average number of conditions, while ILA produced only one rule with 3 conditions which leads to low error rates for classifying unseen examples as shown in Table 9. In fact, this is the reason why ILA produces fewer number of rules with fewer average number of conditions in the rules. This particular rule given above can classify 90 out of 958 examples from tic-tac-toe data base while the five rules produced by ID3 can classify 73 examples. From above discussion we can assume that ID3 is also affected by *small junction problem*. As each conjunct supports fewer training examples it has a rather poor predictive accuracy in unseen examples to be shown later in Section 4.2.3.

Similar situations can easily be found in rules produced by AQ and ID3 algorithms especially for large training data sets. It is clear that when ILA eliminates unnecessary conditions from the rules, the number of rules and average number of conditions decrease significantly. This situation illustrates the difference in the values of these two parameters between ILA on one side and the other algorithms on the other side.

4.2.3 Classification of Unseen Examples

Concept learning systems often describe a decision as a disjunction or conjunction of conditions(attributes). Recently it is noted that small junctions(disjuncts), i.e., those supported by few training examples, typically have poor predictive accuracy in unseen examples. Several approaches are proposed to overcome this problem, for example by [Ali and Pazzani, 1993]. All of the algorithms tested were affected by the small junction problem with varying degrees, ID3 being the most affected one while ILA was the least affected algorithm.

In order to test the three algorithms for the ability of classifying unseen examples, each training set has been divided into two sets, the first set containing a sub set of the training examples on which the algorithms are run, while the second set contains rest of the examples which are selected randomly to form the unseen examples on which the generated rules from all algorithms are tested.

Tests are conducted on different sizes of data as follows:

Partition I : about 2/3 of the original set is kept as the training set and 1/3 as the set of unseen examples.
Partition II: about 1/2 of the original set is kept as the training set and 1/2 as the set of unseen examples.
Partition III: about 1/3 of the original set is kept as the training set and 2/3 as the set of unseen examples.

To enhance the generality of the results, these tests have been conducted on the above cases for five times, each time with different (randomly selected) examples in both sets that contain the training examples and the unseen examples as well.

Table 9 lists the average number of rules generated from the five tests of applying the four algorithms on the three different training sets for the cases mentioned above.

Table 9. Number of Rules Generated.

Training Set	Partition	ID3	AQ	ILA
Balloons	I	3	3	3
	II	3	3	3
	III	4	4	2
Balance	I	212	178	160
	II	164	134	121
	III	139	107	99
Tic-tac-toe	I	129	53	18
	II	116	57	28
	III	74	38	27

From Table 9, it is clear that ILA produces the fewest number of rules compared with ID3 and AQ. For the small data set, Balloons, the results are almost the same with only a small difference among them. However, as the size of the training sets increases, the difference between ILA and other algorithms becomes obvious, as in the case of Balance and Tic-tac-toe sets.

Table 10, on the other hand, shows the powerful aspects of ILA. It shows the average of error rates of applying the four algorithms on the training sets of the same cases in Table 9, also for the five tests. It is apparent that the error rates of ILA is the best among all compared with ID3 and AQ

Table 10. Error Percentages for Classifying Unseen Examples**.**

Training Set	**Partition**	**ID3**	**AQ**	**ILA**
Balloons	I	0.0%	0.0%	0.0%
	II	0.0%	0.0%	0.0%
	III	30.8%	30.8%	30.8%
Balance	I	64.4%	41.5%	40.4%
	II	53.0%	40.6%	34.6%
	III	54.7%	51.1%	47.7%
Tic-tac-toe	I	29.7%	13.8%	4.1%
	II	31.2%	18.4%	6.8%
	III	42.5%	4.4%	2.4%

5. Conclusions

ILA is a supervised, simple but powerful inductive algorithm for classifying symbolic data. In particular it deals with discrete and symbolic attribute values. The results obtained so far indicate that ILA is comparable to other well-known algorithms. In this paper, ILA has been applied to several domains to derive IF-THEN rules from training examples. The results obtained are compared with results obtained from applying two well-known algorithms in the domain, namely ID3 and AQ on the same training sets. It has been shown that in all of the tests the generality of the extracted rules is achieved. This is due to the fact that ILA eliminates the unnecessary condition problem. ILA's accuracy of rules induced from an unseen training set are better than the accuracy of a decision tree induced by ID3 and rules generated by AQ.

As a further research, two new improvements to the algorithm are being added. The first is the ability to deal with noisy and incomplete examples, where some of the attribute values are wrong or unknown. The second improvement is to convert the algorithm in a way to be able to treat continuous attribute values.

Acknowledgments

The authors would like to thank Hayri Sever for helpful comments on an earlier draft of this article. We would also like to thank the following people for supplying data sets: R.S. Siegler for balance scale weight and distance database and to Tim Hume for the donation, Michael Pazzani for balloons data and David W. Aha for supplying and donating tic-tac-toe endgame database. All of the data sets were obtained from the University of California-Irvine's repository of machine learning databases and domain theories, managed by Patrick M. Murphy. Finally, we acknowledge Raymond J. Mooney for Common Lisp implementations of ID3 and AQ algorithms.

References

Ali, K.M., and Pazzani, M.J. (1993). “HYDRA: A Noise-tolerant Relational Concept Learning Algortihm”, *Proceedings of 13th International Joint Conference on Artificial Intelligence*, (Ed. R. Bajcsy) Philadelphia, PA: Morgan Kaufmann 1064-1070.

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984). Classification and Regression Trees. Monterey, Calif.:Wadsworth and Brooks.

Cameron-Jones, R.M., and Quinlan, J.R. (1994). “Efficient Top-down Induction of Logic Programs”, ACM Sigart Bulletin, 5(1), 33-42.

Carter, C., and Catlett, J. (1987). “Assessing Credit Card Applications Using Machine Learning”, IEEE Expert, 2(3), 71-79.

Cestnik, B., Kononenko, I., & Bratko, I. (1987). “ASSISTANT 86:A Knowledge-Elicitation Tool for Sophisticated Users”, in I. Bratko & N. Lavrac(eds.), Progress in Machine Learning, Wilmslow, UK: Sigma Press, 31-45.

Ching, J.Y., Wong, A.K.C., and Chan, K.C.C. (1995). “Class-Dependent Discretization for Inductive Learning from Continuous and Mixed-Mode Data”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(7), 641-651.

Clark, P. & Niblett, T.(1989). “The CN2 Induction Algorithm”. Machine Learning, 3, 261-283.

Deogun, J. S., Raghavan, V. V., Sarkar, A., and Sever, H. (1997). “Data Mining: Research Trends, Challenges, and Applications”, chapter in a book (Ed. T. Y. Lin), Kluwer Academic Publishers.

Fayyad, U.M., and Irani, K.B. (1993). “Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning”, *Proceedings of 13th International Joint Conference on Artificial Intelligence*, (Ed. R. Bajcsy) Philadelphia, PA: Morgan Kaufmann 1022-1027.

Frawley, W.J., Piatetsky-Shapiro G., and Matheus, C.J. (1991).”Knowledge Discovery in Databases: An Overview”, in Knowledge Discovery in Databases, (Eds. Frawley, W.J., Piatetsky-Shapiro G., and Matheus, C.J.), MIT Press, Cambridge, MA, 1-27.

Holsheimer, M., & Siebes, A. (1994). “Data Mining-The Search for Knowledge in Databases”, (Report No. CS-R9406). CWI, Amsterdam, The Netherlands.

Hunt, E.B., Marin J., & Stone, P.J. (1966). Experiments in Induction. New York, London : Academic Press.

Irani, Cheng, Fayyad, and Qian, (1993).”Applying Machine Learning to Semiconductor Manufacturing”, IEEE Expert, 8(1), 41-47.

McKee, T.E. (1995). “Predicting Bankruptcy via Induction”, Journal of Information Technology, 10, 26-36.

Michalski, R.S., & Larson, J.B. (1978). “Selection of most representative training examples and incremental generation of VL1 hypothesis: The underlying methodology and the descriptions of programs ESEL and AQ11 (Report No. 867)”. Urbana, Illinois: Department of Computer Science, University of Illinois.

Michalski, R.S. (1983). “A Theory and Methodology of Inductive Learning”. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell, *Machine Learning, an Artificial Intelligence Approach*, Palo Alto, CA: Tioga.

Michalski, R.S., Mozetic, I., Hong, J., & Lavrac, N. (1986). “The Multipurpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains”, *Proc. of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA: Morgan Kaufmann, 1041-1045.

Murthy, S.K., Kasif, S., & Salzberg, S. (1994). “A System for Induction of Oblique Decision Trees”, Journal of Artificial Intelligence Research, 2, 1-32.

Pham, D.T. & Aksoy, M.S.(1995). “RULES: A Simple Rule Extraction System”, Expert Systems with Applications, 8(1), 59-65.

Quinlan, J.R.(1983). “Learning Efficient Classification Procedures and their Application to Chess End Games”. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell, *Machine Learning, an Artificial Intelligence Approach,* Palo Alto, CA: Tioga, 463-482.

Quinlan, J.R.(1986). “Induction of Decision Trees”, Machine Learning, 1, 81-106.

Quinlan, J.R.(1993). C4.5: Programs for Machine Learning. Philadelphia, PA: Morgan Kaufmann.

Schlimmer, J.C. & Fisher, D. (1986). “A Case Study of Incremental Concept Induction”. *Proc. of the Fifth National Conference on Artificial Intelligence,* Philadelphia, PA: Morgan Kaufmann, 496-501.

Thornton, C.J. (1992). Techniques in Computational Learning-An Introduction, London: Chapman & Hall.

Utgoff, P.E. (1988). “ID5: An Incremental ID3”, *Proc. of the Fifth National Conference on Machine Learning*, Ann Arbor, MI, University of Michigan, 107-120.

Uthurusamy, R., Fayyad, U.M. and Spangler, S. (1991). "Learning Useful Rules from Inconclusive Data", Knowledge Discovery in Databases, (Eds.G. Piatetsky-Shapiro and W.J. Frawley), AAAI/MIT Cambridge, MA, 141-157.

Zhang, J. (1990). “A Method that Combines Inductive Learning with Exemplar-Based Learning”, *Proc. of the Second International Conference on Tools for Artificial Intelligence*, San Jose, CA, 31-37.