An Environment for Rule Extraction and Evaluation from Databases*

J. A. Baranauskas and M. C. Monard/ILTC

Institute of Mathematics and Computer Sciences,
Department of Computer Science and Statistics,
Laboratory of Computational Intelligence,
Av. Dr. Carlos Botelho, 1465 P.O. Box 668
São Carlos - São Paulo - Brazil - Zip Code 13560-970
Phone: +55 (16) 273-9638 Fax: +55 (16) 273-9633
{jaugusto, mcmonard}@icmc.sc.usp.br
University of São Paulo at São Carlos, Brazil.

Abstract Classification for very large databases has many practical applications in Data Mining. Thus, Machine Learning algorithms should be able to operate in massive datasets in order to extract symbolic classifiers. In this context, a symbolic classifier is the one that can be transformed into a set of rules. When a dataset is too big for a particular learning algorithm, there are other ways to make learning feasible such as dataset sampling: for each sample a classifier is extracted for further investigation, like accuracy evaluation. It is also possible to evaluate the performance of combining all extracted classifiers into an ensemble. However, combining symbolic classifiers into a single one by a majority (or other) vote mechanism does not result into a symbolic classifier any more. The approach adopted in this under development work consists in evaluating the induced knowledge as white-box *i.e.*, looking to the rules and trying to combine them using a computational environment. The environment is designed as a test-bed for Data Mining research, as well as a generic knowledge discovery tool for varied database domains. Flexibility is achieved by an open-ended design for extensibility, enabling integration of existing Machine Learning algorithms, support functions for pre-processing as well as new locally developed algorithm and functions.

Keywords: *Data Mining, Machine Learning.*

Topic: 7. Knowledge Discovery and Data Mining

* This research is partially supported by National Research Councils Finep and CAPES as well as FMRP-USP and FAEPA-HCFMRP-USP.

An Environment for Rule Extraction and Evaluation from Databases*

J. A. Baranauskas and M. C. Monard/ILTC

Institute of Mathematics and Computer Sciences,
Department of Computer Science and Statistics,
Laboratory of Computational Intelligence,
Av. Dr. Carlos Botelho, 1465 P.O. Box 668
São Carlos - São Paulo - Brazil - Zip Code 13560-970
{jaugusto, mcmonard}@icmc.sc.usp.br
University of São Paulo at São Carlos, Brazil.

Abstract Classification for very large databases has many practical applications in Data Mining. Thus, Machine Learning algorithms should be able to operate in massive datasets in order to extract symbolic classifiers. In this context, a symbolic classifier is the one that can be transformed into a set of rules. When a dataset is too big for a particular learning algorithm, there are other ways to make learning feasible such as dataset sampling: for each sample a classifier is extracted for further investigation, like accuracy evaluation. It is also possible to evaluate the performance of combining all extracted classifiers into an ensemble. However, combining symbolic classifiers into a single one by a majority (or other) vote mechanism does not result into a symbolic classifier any more. The approach adopted in this under development work consists in evaluating the induced knowledge as white-box *i.e.*, looking to the rules and trying to combine them using a computational environment. The environment is designed as a test-bed for Data Mining research, as well as a generic knowledge discovery tool for varied database domains. Flexibility is achieved by an open-ended design for extensibility, enabling integration of existing Machine Learning algorithms, support functions for pre-processing as well as new locally developed algorithm and functions.

Keywords: *Data Mining, Machine Learning.*

1 Introduction

Usual approaches for developing a knowledge base involve a manual formalization of expert's knowledge and encoding it in appropriate data structures. One important application of Machine Learning — ML — concerns the (semi) automatic construction of knowledge bases through inductive inference. Indeed, ML

* This research is partially supported by National Research Councils Finep and CAPES as well as FMRP-USP and FAEPA-HCFMRP-USP.

can provide an improvement of current techniques and a basis for developing alternative knowledge acquisition approaches.

One challenge in predictive Data Mining — DM — is to exploit and combine existing Machine Learning algorithms effectively. However, ML algorithms are not designed to deal with big data, an important issue in Data Mining. Another important point is related to the fact that, frequently, ML algorithms are interested in the predictive power of the induced knowledge on new instances *i.e.*, accuracy. Although prediction can be considered a strong goal of Data Mining, human understanding and evaluation of the induced knowledge also plays an important role, which is often neglected.

In this work we describe a under development computational environment for extraction and evaluation of rules from databases. The emphasis in our work is not centered in classification performance but in understanding for explanation since it is often necessary that a potential user can make an appreciation of the methods and rationale behind a classification rule. In fact, in some contexts, such as credit scoring, this is a legal requirement.

This work is organized as follows. Section 2 outlines the architecture of the computational environment currently being developed, sketching its main components. The basic steps taken by our system are provided from Sections 2.1 through 2.3 and Section 2.4 shows a running example of our approach. Finally, Section 3 presents some concluding remarks.

2 Basic System Overview

A general overview of the system can be seen in Figure 1. It is assumed that the original database has already been processed in such a way that it holds critical information that can be used for further analytical processing and decision making. In other words, the database holds the sort of data typically found in a Data Warehouse [16]. Furthermore, it is assumed that the data extracted from this database has already been transformed to a plain text specialized standard form (Text Dataset in Figure 1) explained in Section 2.1.

The Text Dataset is transformed into a Database Table structure inside the $\mathcal{M}\mathrm{y}\mathcal{SQL}$ database server [11], dividing the original data into two disjunct subset *i.e.*, Training Set and Testing Set in Figure 1. Afterwards, it is possible for the user to pick up samples and construct classifiers using different ML algorithms (which description languages can be translated into a set of if-then rules). As each extracted classifier differs from each other the induced knowledge is then converted into a standard rule form. Inside the $\mathcal{M}\mathrm{y}\mathcal{SQL}$ database server the complete rule set can be evaluated for interestingness criteria [7, 8] or using a covering algorithm that tries to increase the predictive power of the final rule set that represents the knowledge of the original database. The next sections describe briefly relevant points of our system.

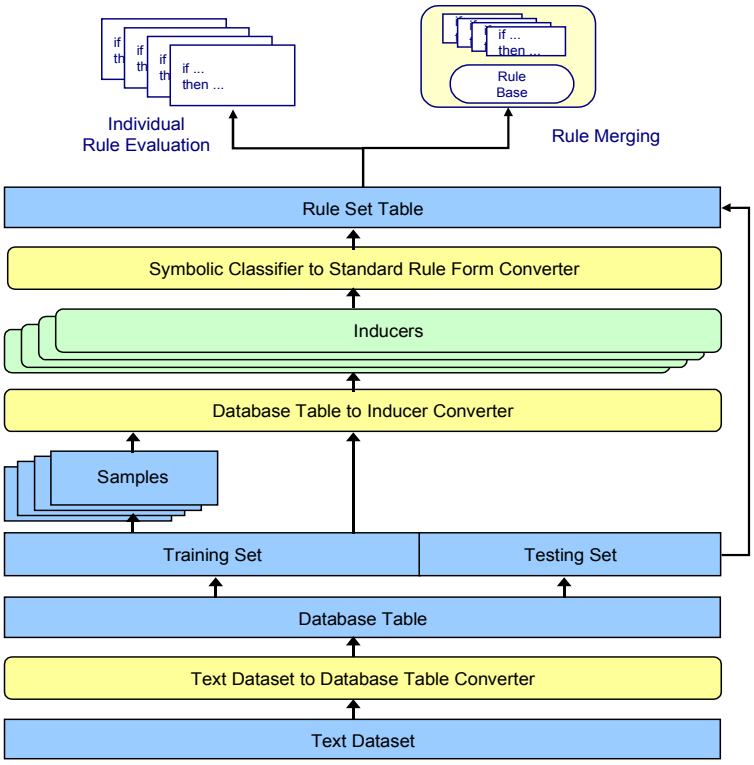

Figure1. The Proposed Computational Environment

2.1 From Text Datasets to Database Tables

Our environment assumes a dataset in the feature-value format. A *dataset* is a set of classified (labeled) instances. Table 1 shows the general format of a dataset T with n instances and m features. In this table a row i refers to the i-th instance ($i = 1, 2, \ldots, n$) and column entries x_{ij} refer to the value of the j-th ($j = 1, 2, \ldots, m$) feature X_j of instance i.

As can be seen, instances are tuples $T_i = (x_{i1}, x_{i2}, \ldots, x_{im}, y_i) = (\mathbf{x}_i, y_i)$ also referred as $(\mathbf{X}, Y)$ where the last column, Y, is what we try to predict given the other $\mathbf{X}$ features *i.e.*, $Y = f(\mathbf{X})$. Each $\mathbf{X}$ is an element of the set $X_1 \times X_2 \times \ldots \times X_m$ where X_j is the domain of the j-th feature and Y belongs to one of the k classes *i.e.*, $Y \in \{C_1, C_2, \ldots, C_k\}$. The ML algorithms known by our system assume the instances are given in this feature-value format.

In general, the feature-value format of a dataset is represented by two text files with extensions .names and .data. The .data file contains the data itself whereas the .names file describes the dataset schema (or structure). For instance, $\mathcal{C}4.5$, $\mathcal{C}5.0$ [14] as well as the $\mathcal{MLC}$++ library [9] share this sort of representation.

Table1. Dataset in the feature-value (or spreadsheet) format

	X_1	X_2	$\cdots$	X_m	Y
T_1	x_{11}	x_{12}	$\cdots$	x_{1m}	y_1
T_2	x_{21}	x_{22}	$\cdots$	x_{2m}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_n	x_{n1}	x_{n2}	$\ldots$	x_{nm}	y_n

Having the dataset represented in the text form is useful when transferring data across different platforms. However, simple statistics such as counting different feature values can not be performed directly by the user.

Our system recognizes datasets in the $\mathcal{MLC}$++ format which are then scanned by a PERL script [15] that converts its schema (the .names file) into a table structure inside the $\mathcal{M}$y$\mathcal{SQL}$ database server. Once created, the table is populated with data from the original .data text file. All conversions are performed automatically, including conversion from unknown values (represented as '?' in the $\mathcal{MLC}$++ .data text file) to NULL values in the database table. As can be seen, this step needs to be performed only once in order to make a dataset available in the $\mathcal{M}$y$\mathcal{SQL}$ table inside our system.

One immediate advantage of having instances inside a database table is that simple statistics like counting values, finding minimum/maximum, computing averages as well as find class distribution among feature values, beyond other measures, can be easily performed through SQL statements. In fact, we have already developed a tool for extracting such basic information [12].

2.2 From Database Tables to Symbolic Classifiers

With the dataset now inside a $\mathcal{M}$y$\mathcal{SQL}$ table, the system allows the user to split the data into training and testing sets. The system also allows the automatic extraction of several samples from the training data. We will refer further to samples from the training set although a simple (unique) sample that contains all training set can be used instead.

Once samples are extracted, the user is able to choose which inducer (or inducers) are to be used. In the simple case, it is possible to (i) select just one inducer with default parameters. The complexity can be increased from (ii) selecting one inducer but with several different user defined parameters to (iii) different inducers with default parameters or even several to (iv) inducers with default and user defined parameters. The effect of choosing different inducer's parameters can help to find more general or specific rules in the final classifier.

After defining both samples and inducers to be used, the induction process can take place. For each sample and inducer, the system converts the sample into the specific inducer's input, using PERL scripts. Then, the system runs the inducer with its associated parameters in order to obtain the corresponding classifier. As can be seen, this process can generate a large amount of classifiers. For example, for 10 samples and 2 inducers executed with default parameters

as well as once with user defined parameters, $10 \times 2 \times 2 = 40$ classifiers are extracted at the end of this step.

Recall that in supervised classification an instance is a pair $(\mathbf{X}, f(\mathbf{X}))$ where $\mathbf{X}$ is the input and $f(\mathbf{X})$ is the output. The task of an inducer is, given a set of instances, to induce a function h that approximates f. In this case, h is called an *hypothesis* over f.

A *symbolic classifier* is an hypothesis whose description language can be transformed into a set of rules — for instance induction of decision trees or unordered decision rules. Although inducers can produce classifiers that differ in their syntactic form, our system is able to deal with them if it is possible to convert them into the following *rule* form

$$\textbf{if } <\text{complex}> \textbf{ then } <\text{class} = C_i>$$

where C_i belongs to the set of possible k classes $\{C_1, C_2, \ldots, C_k\}$. The <complex> part is also denominated rule *conditions* and $<\text{class} = C_i>$ is denominated rule *conclusion*. The <complex> is a disjunction of conjunctions of feature tests in the form

$$X_i \ op \ Value$$

where X_i is a feature, *op* is an operator in the set $\{=, \neq, <, \leq, >, \geq\}$ and *Value* is a valid feature X_i constant value, according to feature domain.

Instances that satisfy the <complex> part of the rule composes its *covered set*, or in other words, those instances are *covered* by the rule. Instances that satisfy both the <complex> and the conclusion $<\text{class} = C_i>$ are *positively covered* by the rule. On the other hand, instances satisfying the <complex> but with $<\text{class} \neq C_i>$ are called *negatively covered*.

2.3 From Symbolic Classifiers to the Standard Rule Form

After inducing symbolic classifiers the following step is to convert them into our standard rule form. Table 2 describes the grammar $G =$ (`T, NT, S, RR`) where:

- `T` is the set of terminal symbols (represented by bold face symbols) as well as relational operators;
- `NT` is the set of nonterminal symbols represented by words inside angular brackets;
- `S` is the start symbol <rule>;
- `RR` is the set of rewrite grammar rules of the form $LHS \rightarrow RHS$, where *LHS* is a nonterminal, and *RHS* is a sequence of zero or more symbols either terminal or nonterminal.

Converters, implemented using PERL scripts, are responsible for translating the symbolic classifier induced by each specific ML algorithm (provided as a text output by inducers) to the standard rule form required by our system [2]. Current available converters provide translation for inducers $\mathcal{ID}3$ [13], $\mathcal{C}4.5$ [14]

Table2. BNF grammar of rules

S = <rule>	→ <rule-number> **IF** <complex> **THEN** <class>
<rule-number>	→ R0001 | R0002 | ...
<complex>	→ <factor> | <factor> **OR** <complex>
<factor>	→ <term> | <term> **AND** <factor>
<term>	→ <feature> <operator> <value>
<operator>	→ < | <= | > | >= | = | <>
<class>	→ **CLASS** = <value>
<feature>	→ $X_1 \mid X_2 \mid \ldots \mid X_m$
<value>	→ $x_{11} \mid x_{12} \mid \ldots \mid x_{nm} \mid y_1 \mid y_2 \mid \ldots \mid y_n$

and $\mathcal{CN}2$ [5]. It should be observed that additional converters can easily be included into the system.

During the standardization, additional information provided by the inducer for each rule — like number of correct and incorrect covered instances or any other accuracy measure — is discarded. This is motivated by the fact that measures attached to each rule differ for each inducer and our system aims to use a standard measure for all rules.

The next step translates the rules already in the standard form into a $\mathcal{M}y\mathcal{SQL}$ table. This table, besides the inducer and sample from where the rule came from[1], holds also the following information:

rule-id the rule number;
rule-part-order the relative position inside the rule of the expression '<feature> <op> <value>';
feature the feature's name;
op a relational operator;
value the value for the associated feature;
position a value 'p' indicates that row is a rule premise (left-hand side) while a value 'q' indicates the row is a conclusion (right-hand side), following the $p \Rightarrow q$ logical implication terminology.

For instance, Table 3 shows the conversion of the rule 'R0001 **IF** $X_1 < 2$ **AND** $X_3 =$ yes **THEN** CLASS = good' which has two conditions (premises) into a database table form.

Importing rules into the relational database component of our system aims to facilitate rule evaluation, since each rule can be easily converted into SQL statements. Our system allows rule evaluation using a sample which was not used during the training phase for the specific symbolic classifier the rule came from, thus computing measures that are less biased than measures taken from the

[1] In fact, this table contains foreign keys poiting to an inducer table as well as a sample table.

Table3. Database table form for rule 'R0001 **IF** $X_1 < 2$ **AND** $X_3 =$ yes **THEN** CLASS = good'

rule-id	rule-part-order	feature	op	value	position
R0001	1	X_1	$<$	2	p
R0001	2	X_3	$=$	yes	p
R0001	1	CLASS	$=$	good	q

training set. Consequently, using this strategy it is possible to answer questions like:

- Which original features are present in one specific classifier?
- How often each feature is used in the rules?
- What is the average number of premises in rules?
- How many positive (negative) instances are covered by one specific rule?

After evaluating each individual rule, the user can decide which ones are better for the problem at hand, based on his/her experience in the domain. However, for a large number of rules, this manual evaluation becomes infeasible. In order to try to overcome this difficulty, we are currently working on rule set merging, trying to generate a final rule base from all extracted symbolic classifiers. One approach we are testing consist on a greedy covering algorithm, similarly the one proposed in [6].

2.4 Example

This section shows a toy running example, adapted from [14] by [1], for better understanding of how a symbolic classifier is translated into the standard rule form. For this purpose, we shown the results using the $\mathcal{C}4.5$ decision tree inducer although, as mentioned before, other inducers are currently known by the system.

Let us suppose a dataset containing day-by-day measures from weather conditions, where each instance is composed by the following features:

- outlook: assumes discrete values "sunny", "overcast" or "rain";
- temperature: a numeric value indicating the temperature in Celsius degrees;
- humidity: also a numeric value indicating the percentage of humidity;
- windy: assumes discrete values "yes" or "no" indicating if it is a windy day.

Furthermore, for each day (instance), someone has labeled each day-by-day measure as "go" if weather was nice enough for taking a trip to the farm or "dont_go" if this is not the case. This *voyage* data could look just like the one shown in Table 4.

The decision tree output produced by $\mathcal{C}4.5$ is shown in Figure 2. After processing by PERL scripts, the standard rule format is shown in Figure 3 [12]. Finally, rules are imported into a relational database in the form shown in Table 5.

Table4. The voyage data

Instance No.	Outlook	Temperature	Humidity	Windy	Voyage?
T_1	sunny	25	72	yes	go
T_2	sunny	28	91	yes	dont_go
T_3	sunny	22	70	no	go
T_4	sunny	23	95	no	dont_go
T_5	sunny	30	85	no	dont_go
T_6	overcast	23	90	yes	go
T_7	overcast	29	78	no	go
T_8	overcast	19	65	yes	dont_go
T_9	overcast	26	75	no	go
T_{10}	overcast	20	87	yes	go
T_{11}	rain	22	95	no	go
T_{12}	rain	19	70	yes	dont_go
T_{13}	rain	23	80	yes	dont_go
T_{14}	rain	25	81	no	go
T_{15}	rain	21	80	no	go

```
outlook = overcast: go (5.0/1.0)
outlook = sunny:
|   humidity <= 78 : go (2.0)
|   humidity > 78 : dont_go (3.0)
outlook = rain:
|   windy = yes: dont_go (2.0)
|   windy = no: go (3.0)
```

Figure2. The $\mathcal{C}4.5$ voyage classifier

```
Standard Rules Converter    Copyright (c) Ronaldo C. Prati
Inducer: C4.5               Input File: voyage.tree
Date: Thu Mar  2 15:35:43 2000

R0001   IF outlook = overcast
    THEN CLASS = go
R0002   IF outlook = sunny
        AND humidity <= 78
    THEN CLASS = go
R0003   IF outlook = sunny
        AND humidity > 78
    THEN CLASS = dont_go
R0004   IF outlook = rain
        AND windy = yes
    THEN CLASS = dont_go
R0005   IF outlook = rain
        AND windy = no
    THEN CLASS = go
```

Figure3. The $\mathcal{C}4.5$ voyage standard form classifier

Table5. The $\mathcal{C}4.5$ voyage standard rule form

rule-id	rule-part-order	feature	op	value	position
R0001	1	outlook	=	overcast	p
R0001	1	CLASS	=	go	q
R0002	1	outlook	=	sunny	p
R0002	2	humidity	<=	78	p
R0002	1	CLASS	=	go	q
R0003	1	outlook	=	sunny	p
R0003	2	humidity	>	78	p
R0003	1	CLASS	=	dont_go	q
R0004	1	outlook	=	rain	p
R0004	2	windy	=	yes	p
R0004	1	CLASS	=	dont_go	q
R0005	1	outlook	=	rain	p
R0005	2	windy	=	no	p
R0005	1	CLASS	=	go	q

3 Concluding Remarks

In this work we described a system under development which addresses the problem of predictive DM whenever the result of learning can be expressed in the form of classification rules [3]. Our proposed framework assumes that the dataset is a normal relational database table, which consists of n records described by m distinct features that have been classified into k known classes.

Although many work have been published proposing new learning systems as well as modifications to existing ones, little attention has been given to the extracted symbolic knowledge. One of the possible reasons for this phenomena is the extensive use of ready-to-learn datasets for ML (as those found in data repositories [4]) and the extensive use of accuracy to evaluate the induced knowledge. The emphasis of our environment and its software architecture is on integration of DM operations, such as database access and selection, ML algorithms that induce classification rules, focusing on human understanding and evaluation of the induced knowledge, as well as on extensibility.

There are many different methods for constructing classification rules although none of them is universally the best since different application domains lead to different problems, requiring different solutions [10]. This being the case, it is expected that a combination of these methods, as proposed in this work, may yield better classification rules. Such combinations can be made in various ways. We consider simplicity of rules as a goal in itself, although simplicity does not necessarily lead to greater accuracy.

Acknowledgments: We would like to thank Jaqueline Brigladori Pugliesi for helpful comments on a draft of this paper.

Bibliography

[1] Baranauskas, J. A. & Monard, M. C. (2000a). Reviewing some machine learning concepts and methods. Technical Report 102, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_102.ps.zip.

[2] Baranauskas, J. A. & Monard, M. C. (2000b). An unified overview of six supervised symbolic machine learning inducers. Technical Report 103, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_103.ps.zip.

[3] Baranauskas, J. A., Monard, M. C., & Batista, G. E. A. P. A. (2000). A computational environment for extracting rules from databases. In *Proceedings of the Second International Conference on Data Mining*, Cambridge, UK. (in print).

[4] Blake, C., Keogh, E., & Merz, C. J. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[5] Clark, P. & Boswell, R. (1991). Rule induction with $\mathcal{CN}2$: Some recent improvements. In Kodratoff, Y., editor, *Proceedings of the 5th European Conference (EWSL 91)*, pages 151–163. Springer-Verlag.

[6] Fayyad, U. M., Djorgovski, S. G., & Weir, N. (1996). From digitized images to on-line catalogs: Data mining a sky survey. *AI Magazine*, 17(2):51–66.

[7] Freitas, A. A. (1998a). A multi-criteria approach for the evaluation of rule interestingness. In *Proceedings of the International Conference on Data Mining*, pages 7–20, Rio de Janeiro, RJ.

[8] Freitas, A. A. (1998b). On objective measures of rule surprisingness. In *Principles of Data Mining & Knowledge Discovery: Proceedings of the Second European Symp. Lecture Notes in Artificial Intelligence*, volume 1510, pages 1–9.

[9] Kohavi, R., Sommerfield, D., & Dougherty, J. (1994). *$\mathcal{MLC}$++: A Machine Learning Library in C++*. IEEE Computer Society Press.

[10] Kohavi, R., Sommerfield, D., & Dougherty, J. (1996). Data mining using $\mathcal{MLC}$++: A machine learning library in $\mathcal{C}$++. *Tools with Artificial Intelligence*, pages 234–245.

[11] MySQL (2000). The $\mathcal{MySQL}$ server. http://www.mysql.com.

[12] Prati, R. C., Baranauskas, J. A., & Monard, M. C. (2000). Proposta de formato padrão de regras & obtenção de informações básicas. Technical report, ICMC-USP. (in progress).

[13] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106. Reprinted in Shavlik and Dieterich (eds.) Readings in Machine Learning.

[14] Quinlan, J. R. (1988). *C4.5 Programs for Machine Learning*. Morgan Kaufmann, CA.

[15] Wall, L., Christiansen, T., & Schwartz, R. L. (1996). *Programming Perl*. O'Reilly & Associates, Inc.

[16] Weiss, S. M. & Indurkhya, N. (1998). *Predictive Data Mining: A Practical Guide*. Morgan Kaufmann, San Francisco, CA.