Handling Inconsistent Symbolic Classifiers

Luiz Gustavo Moro Senko, Márcio Fuckner, Fabrício Enembreck

Pontifícia Universidade Católica do Paraná – PUCPR
Programa de Pós-Graduação em Informática – PPGIA
Av Imaculada Conceição, n. 1155 – Curitiba PR - Brasil
gustavosenko@brturbo.com.br, {marciofk,fabricio}@ppgia.pucpr.br

Abstract. This paper presents a technique for handling inconsistencies in symbolic classifiers. Very often, ensemble-based techniques are used to improve performance and classification accuracy. However such methods compromise the knowledge understandability and decisions explanation because the knowledge is partitioned among the classifiers. Furthermore, the classifiers generated from data samples are likely inconsistent, demanding a complex ensemble combination strategy. On the other hand, knowledge integration techniques are used to merge symbolic classifiers into an understandable and global classifier, merging and selecting good classification rules from the local models. In this paper Paraconsistent Logic concepts are used to gather concepts from local models, generating a global ruleset with good accuracy avoiding data exchange and rules evaluations, saving a lot of computational resources. Whenever local models are gathered together the rules are ordered using Paraconsistent Logic operators and other ones are used for classification of new instances. We could observe good results in the experiments performed on a number of benchmark datasets.

Index terms – Symbolic Classifiers, Paraconsistent Logic, Distributed Data Mining

1 Introduction

Several studies have been conducted to permit the analysis of large volumes of data generated daily in research centers, commercial and industrial organizations, etc. Such methods are usually based on distributed data mining. Data oriented distributed mining [12] constitute a noticeably important class of methods. Those methods basically split data into smaller subsets which are individually processed, yielding local classifiers. Later, those classifiers are merged into a unique global classifier. As a result, subsets are likely to include inconsistent concepts, causing the system to generate classifiers which represent opposite standpoints. Inconsistencies may either be introduced by the sampling method chosen or be inherent to databases distributed across several sites (for instance, databases from a shop chain).

The method proposed in this work uses knowledge acquired by algorithms for induction of rules of type "IF (*conditions*) THEN *class*", where the former (*conditions*) contains conjunctions of conditions derived from pairs attribute-value and the latter (*class*) contains the class assigned to a set of examples which satisfy the conditions exposed in the first part of the rule.

In a distributed data mining context, when models built by distinct classifiers are merged, inconsistent global knowledge may arise if mutually exclusive rules, extracted from different partitions of data, predict a common class. Consider, for instance, that an algorithm has discovered the following rule from partition *A*: IF age > 25 THEN class = man, whereas the following rule has been discovered from partition *B*: IF age < 12 THEN class = man. As the classifiers have been built upon data from the same application domain, such contradiction is not acceptable. Inconsistencies may also occur in the second part of the rule: for instance, if a classifier *A* has the rule "IF age > 15 THEN class = man" and the classifier *B* has the rule "IF age > 12 THEN class = woman", a contradiction between the concepts *man* and *woman* may be assumed, since a test instance whose attribute "age = 21" would be classified as both "man" and "woman". In this case, either a rule or classifier weighing technique should be used for selecting the final class.

The goal of this work was developing a methodology, based on Paraconsistent Logics [6], for supporting the acquisition of better interpretations of sets of rules where situations such as the ones described in the previous paragraph may occur without jeopardizing the performance of the system. The use of formalisms from Paraconsistent Logics allows the assignment of annotated evidential factors to rules which represent, respectively, belief (how much the rule is expected to be true) and disbelief (how much the rule is expected to be false) degrees. The evidential factors were used as the decision criterion for selecting the rule which satisfies the conditions in test instances and is the closest to the logic state *true* of Paraconsistent Logics.

2 Paraconsistent Logic

Contradictory information may be important for both reasoning and decision making processes. Their removal might cause negative impact on the search for solutions for some problems. The existence of inconsistent information in computational systems [6] can be easily recognized once, in several applications, inconsistencies are inherent to the problem.

According to Enembreck [8], there are basically two ways of handling data inconsistencies: i) providing the learning algorithm with ability for adequately handling contradictory information during the learning process in order to generate consistent and reliable concepts; or ii) applying a reasoning method which permits reliable inference to the knowledge acquired by an ordinary learning algorithm: transforming the knowledge base so it becomes consistent, generating a new knowledge base out of consistent data from the original base, or applying uncertainty handling techniques which permit reasoning over inconsistent data. In this work, the last option was chosen. Such choice enables reasoning over any set of IF-THEN-like rules, what makes the method independent of symbolic learning algorithm.

Furthermore, it is possible to infer conclusions out of sets of rules coming from different locations, originated in distinct databases, even in cases for which there are variations in distribution. The inference model proposed is based on the Paraconsistent Logics model [7] [3].

As opposed to Classical Logics, Paraconsistent Logics permits representing and performing inferences over contradictory information and also distinguishes situations where an arbitrary proposition is indeed false from another where there is not enough evidence to determine a conclusion. It allows contradictory information p and $\neg p$ to be simultaneously present and provides mechanisms for reasoning over information with such characteristics. The conclusion drawn is that either there is not enough information or they are contradictory.

Paraconsistent Evidential Logics (PEL) [3] is a formalism derived from Paraconsistent Logics. The truth-values used in PEL are composed of two evidential factors which belong to a lattice $\{x \in \Re \mid 0 \le x \le 1\} \times \{x \in \Re \mid 0 \le x \le 1\}$. Evidential factors *belief* and *disbelief* are associated to every knowledge item in a system or, in this case, to each rule in a subset. The belief degree represents the weight associated to the truth of the evidence whereas disbelief indicates the weight associated to the falsehood of the evidence. Both factors belong to the interval [0.0, 1.0], i.e., infinite values may be associated to the premises in the system. Therefore, it is possible to define an infinite lattice $\tau = \langle |\tau|, \le \rangle$, so that:

$$|\tau| = \{x \in \Re \mid 0 \le x \le 1\} \times \{x \in \Re \mid 0 \le x \le 1\}$$

The lattice τ has a maximum point $[1.0,1.0]$ which represents maximum inconsistency and indicates a statement would be considered both true and false simultaneously. A statement is true if it is represented by the evidential factors [1.0,0.0], since their evidences indicate complete truth and falsehood unknown. On the other hand, false is represented as [0.0,1.0], indicating unknown truth and complete falsehood. Besides being able to represent an inconsistent state, PEL, as opposed to Classical Logics, is also able to represent the value unknown or non-determined – [0.0, 0.0]. In this case, there is no information at all about truth or falsehood. The difference between falsehood and non-determination may provide important information about inferences in knowledge bases and the decision-making process. For example, consider a system which assesses a person's involvement in a crime. If the answer retrieved by the system is no ([0.0,1.0]), the person is said not to be involved with the crime and, therefore, innocent. However, no conclusion is possible with an answer ([0.0,0.0]) and more information is needed for further judgments. In conventional logic systems, such distinction may not be obtained directly as the only two possible interpretations are *true* and *not true*. The retrieval of an answer *not true* does not allow one to know whether it is associated to falsehood or to lack of knowledge about the truth-value of the statement related (closed-world assumption).

3 Related Work

A similar work has been developed by Simone Ferreira [10], who proposed a method for dealing with the inconsistent rules problem in distributed mining. Her method assumes that *n* subsets of rules generated are independent for *n* subsets of data, what might result in inconsistent rules. Rules are inconsistent whenever they share the same conditions while predicting distinct classes. The methodology proposed by Ferreira uses Paraconsistent Logics for determining the most adequate rule for classifying a new example. Each rule belonging to the first subset is compared against the *n* subsets of data available. The belief degree is obtained by dividing the number of examples correctly covered by the rule (where the attribute values in the example satisfy the conditions of *n* rules, which share the same target class) by the number of examples covered (where the attribute-values in the example satisfy the conditions of a rule). Conversely, the disbelief degree is obtained by dividing the number of examples incorrectly covered by the rule (where the attribute values satisfy the conditions of *n* rules while predicting distinct classes) by the number of examples covered (where the attribute-values in the example satisfy the conditions of a rule).

After assigning both belief and disbelief degrees to each rule, the subsets are checked for rules which share a common set of conditions. When both set of conditions and target class are identical in a pair of rules, the algorithm returns their supreme[1]; on the contrary, if sets of conditions are equal and classes are distinct, then the rule whose multiplication value (an operation which multiplies the truth degree by the determination degree) is greatest will be selected. Unfortunately, such technique is not realistic since it requires the existence of completely opposite concepts (equal set of conditions and distinct classes) for the occurrence of inconsistencies.

According to Chan [4], algorithms for induction of rules usually build classifiers with high precision rates; even so, several factors reduce the quality of the learning process. For instance, the use of one algorithm only against distinct distributions from the training set may generate classifiers with distinct accuracy rates and the natural distribution of classes may not result in good performance.

Hall et al. [13] built a unique model of rules for a distributed data set. The model is obtained by means of meta-learning techniques, where all rules sets are merged into one set only. Hall defined that conflicting and low performance rules need to be removed because the final model requires as much accuracy as a model generated from all training sets available would exhibit. A research on a number of distributed mining techniques is found in [9].

Although interesting, such techniques generate large information flow among processors responsible for the execution of the learning algorithm and also require a high number of calculations, as the evaluation of a rule requires access to all validation instances in all processors involved. In addition, the majority of those techniques apply only to non-ordered sets of rules. As described by Prati [14], the sets of rules may be defined *as non-ordered rule sets* and *ordered rule sets*. For ordered sets, the order of application of rules is fundamental since an instance needs to be evaluated iteratively from the first rule on until one which covers the example is

[1] Supreme: maximum value obtained from the comparison of belief and disbelief degrees in two rules which are merged if and only if their classes are identical.

found. The example needs to be classified exclusively by that rule, even though others might cover it. With non-ordered rule sets, all rules might be applied to a single example in order to find measurements to assessing them, as the order of the rules is irrelevant. This work presents a technique which handles ordered sets of rules, once most symbolic learning algorithms yield knowledge in this format.

4 Methodology

Researches indicate that there may be a great number of causes for uncertainty in both information and AI systems, such as the existence of inaccurate or inconsistent information. Inconsistencies may arise, for instance, when information from distributed sources is amalgamated so inferences may be performed.

Aiming to provide adequate reasoning for such situations, this work proposes the application of Paraconsistent Evidential Logics [3] [6] [7] to distributed mining. The first stage for the development of a method on top of the Paralog_e language [1] is database preparation and mining, as depicted in Fig 1.

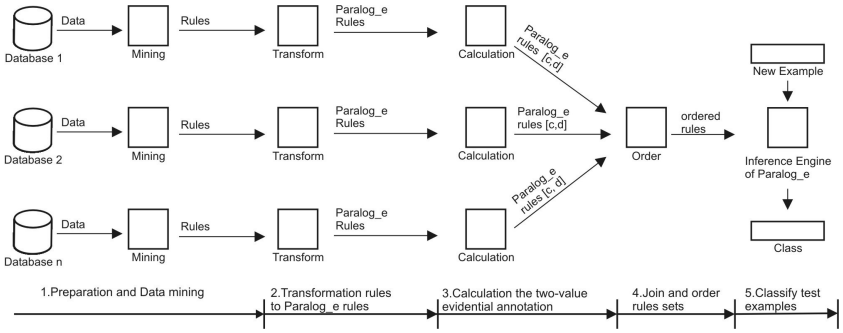

Fig. 1. Obtain rules using the classic data mining process (Step 1), Transform this rules to Paralog_e rules (Step 2), Calculation the two-value evidential annotation (Step 3), Join and order rules sets (Step 4) and Classification (Step 5).

Distinct sets of rules are retrieved with the application of the algorithm RIPPER [5] on distinct subsets of data. RIPPER is available in WEKA [11] and applies to ordered rule sets, therefore allowing a single example to be classified by more than one rule. It uses incremental pruning criteria to minimize misclassification and yield high quality rules even in noisy domains. The second step consists in mapping the rules obtained in the previous step to the format required by Paralog_e [1], which is an inference engine based on evidential logic programming and is able to perform inferences over annotated rules. Belief and disbelief degrees are assigned to each rule.

Thus, all rules of type IF (conditions) THEN (class) are represented as Paralog_e rules, whose format is *Head* ← *Body*. *Head* represents the conclusion of the rule and contains its class along with evidential factors belief and disbelief, associated to that rule. *Body* is composed of conjunctions, represented by "&", of conditions. The set of conjunctions consists in the conditions which integrate the rule. Each condition

contains an evaluator predicate which allows further application when test instances are submitted to inference. To illustrate the procedure of mapping rules to the Paralog_e format, the following rule represented in the RIPPER [5] format has been created from a hypothetical database:

```
(tear_production = normal) and
(astigmatism = yes) and
(spectropy = hypermetropy) => class=none (2.0/1.0)
```

Upon submitting the rule above to transformation, the following output is returned:

```
class('none'):[2.0,1.0] <--
        evaluator(tear_production, V_0):[1.0,0.0] &
        V_0 = normal &
        evaluator(astigmatism, V_1):[1.0,0.0] &
        V_1 = yes &
        evaluator(spectropy, V_2):[1.0,0.0] & V_2 = hypermetropy.
```

It is important to notice, at this stage, that the evidential factors associated do not correspond to the interval [1.0,0.0] yet; those values will be modified later in the next stage. In the third stage, as demonstrated in Fig 1, the evidential factors are modified according to a linear rule weighing function, which consists in an arithmetic progression. Such update occurs locally, i.e., before amalgamation of the sets of rules.

The positive evidential factor (c) (belief degree) is updated with the value which results of the linear weighing function of the subset of rules. The negative evidential factor (d) (disbelief degree) is the complement of the linear weighing function of the subset of rules (d = 1 – c). The common difference of the arithmetic progression is obtained with Equation 1.

$$r = \frac{1}{number\_of\_rules\_in\_subset} \qquad (1)$$

The belief degree assigned to the i^{th} rule corresponds to the arithmetic progression of i with common difference r from the last rule to the first. The last rule is characterized by $c = r$ whereas the first one invariably exhibits $c = 1.0$.

In Paraconsistent Logics, the values of both belief and disbelief degrees are independent of one another; they are not complementary. Since belief (c) and disbelief (d) obtained thus far are complementary, they need to be modified in order to be comparable to the absolute truth, denoted by the ordered pair (1.0, 0.0). $Value_1$ and $value_2$ are determined out from belief and disbelief degrees as follows:

- $value_1$ is named Certainty Degree and is derived from the difference between the belief degree and the disbelief degree in the rule, ($value_1$ = c – d);
- $value_2$ is the result of the multiplication of the disbelief degree of the rule by 2, ($value_2$ = 2 x d).

The certainty degree $value_1$ specifies the certainty associated to the truth of the proposition. Nevertheless, $value_2$ has been empirically defined as twice the disbelief degree associated to the rule. We believe all rules would produce greater error rates (therefore, greater disbelief degrees) in case they were evaluated over distinct subsets.

As there are no clues on the extent of that propagation, the disbelief has been doubled for empirical usage.

After calculating evidential factors, the subsets of rules are merged into one unique set (Fig. 1). At this stage, rules are sorted according to the least Euclidian distance – according to Equation 2 – to the evidential factors (1.0, 0.0) which represent truth.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \qquad (2)$$

where x_i is the ordered pair composed of ($value_1$, $value_2$) in a rule and x_j is (1.0,0.0).

The smaller the distance between ($value_1$, $value_2$), which represent the evidential factors of the rule, and the logical state which represents the truth (1.0,0.0), the more the rule approaches the truth in quantitative terms.

In the very last stage, every instance test is mapped to a set of facts which are tested against the knowledge base formed by the set of rules. By representing the rules as Horn clauses, it is possible to verify whether their conditions are true when applied to the test instance provided. The *Paralog_e* language works in a way analogous to the *Prolog* language, using a SLD Resolution procedure to determine truth or falsehood for each condition in a rule.

Next, a query Q is triggered for each class belonging to the database domain. The answers are the evidences obtained for each class. Because there is a query for each class in the domain, the decision criterion selects the rule capable of covering the example whose evidential factors yield the smallest Euclidian distance to the logical state truth – ordered pair (1.0, 0.0). To evaluate the method based on the Paralog_e language, the selected class is compared against the class which labels the test example; if they are equal, the classification is correct.

5 Experiments and Results

Ten public databases, collected at the directory of databases for machine learning and data mining, maintained by the University of California [2], have been used in the experiments.

Each base used in the experiments was randomly split into training and test sets. The training set received 80% of the examples from the original base, whereas the test set included the remaining 20%. The training set was further split into 10 training partitions, where each sample had 10% of the instances with replacement. Therefore, each rule set was generated in partitions with $(0{,}8 \times 0{,}1) \times 100 = 8\%$ of the original data. Such percentage has been carefully chosen for the probability of conflicts to be increased, once the learning algorithm uses only a subspace of the *tuple* set. Each classifier generated in a partition is evaluated over the 20% test examples remaining. That procedure was iteratively repeated 10 times for each database.

The characteristics of each base utilized, as well as the number of instances present, the existence or absence of missing values in the examples included in the base, amongst other things, have been detailed in Table 1.

Table 1. Characteristics of databases used.

	Database	#Example.	Missing Values	#Attr.	#Nominal Attributes	#Numeric Attributes	#Class	Class Distribution
1	Zoo	101	no	17	16	1	7	unbalanced
2	Audiology	226	yes	69	69	–	24	unbalanced
3	Monk 1	432	no	6	6	–	2	unbalanced
4	Monk 2	432	no	6	6	–	2	unbalanced
5	Soybean	683	yes	35	35	–	19	unbalanced
6	Vehicle	846	no	18	–	18	4	unbalanced
7	Tic-Tac-Toe	958	no	9	9	–	2	unbalanced
8	Vowel	990	no	13	3	10	11	balanced
9	Car	1728	no	6	6	–	4	unbalanced
10	Segment	2310	no	19	–	19	7	balanced

The characteristics of the bases used are highly diverse, such as: domain, number of examples, presence or absence of missing values in the examples and the total number of attributes. In Table 1, the column *Number of Attributes* does not include the target-attribute associated to the example and the column *Class Distribution* characterizes a non-uniform distribution of instances in relation to the associated class. The overall average of the results obtained with Paralog_e has been compared against the overall average of results obtained with the algorithm RIPPER along all ten iterations using 80% of training examples (see Table 2). On average, the Paralog_e method outperformed the RIPPER method in 7 out of 10 experiments carried out.

Table 2. Comparison of results among Paralog_e method and RIPPER algorithm.

Database	Paralog_e (%)	Ripper (%)	Percentual Difference between Methods	Relation *#Attr/#Examples*
Audiology	**48,69 ± 2,94**	32,65 ± 3,88	1,49	0,305
Zoo	**61,43 ± 4,73**	49,14 ± 4,14	1,25	0,168
Soybean	**63,06 ± 7,59**	51,21 ± 2,32	1,23	0,051
Vowel	**43,89 ± 4,35**	33,29 ± 1,14	1,32	0,013
Segment	**87,87 ± 3,83**	82,83 ± 1,43	1,06	0,008
Monk2	**63,70 ±10,83**	60,48 ± 2,25	1,05	0,014
Vehicle	**52,35 ± 4,12**	52,24 ± 2,18	1,00	0,021
Monk1	55,23 ± 5,66	**55,50 ± 4,25**	1,00	0,014
Tic-Tac-Toe	65,22 ± 0,94	**67,76 ± 2,27**	0,96	0,009
Car	58,24 ± 4,45	**71,98 ± 1,36**	0,81	0,003

Upon analyzing the bases which presented the most significant difference between the average results of the two methods applied, we concluded that the reason why the Paralog_e method had a better performance may be related to the total number of attributes in the base. For instance, in base Audiology (see Table 2), composed of 69 attributes, Paralog_e had an accuracy rate of 48,69% whereas RIPPER had 32,65%. In percentage terms (performance of Paralog_e divided by performance of RIPPER), the result is 1,49% in favor of Paralog_e.

Considering the results obtained with base Zoo, present in Table 2, the relation (*number of attributes / number of examples*) is 0,168. In this case, it is possible to notice that the proposed method outperforms local classifiers generated by RIPPER.

It is important to consider that, according to Freitas [12], in induction algorithms,

the Cartesian product of variables and the number of values that the attributes may assume increase exponentially the *tuple space* and, consequently, the rules space to be searched. Despite that, the number of attributes as well as their possible values, contributed to the generation of more expressive sets of rules, possibly composed of a greater number of conditions. Thus, those rules possibly better represent the characteristics of the database, covering specific regions of the *tuple space* and generating fewer intersections among rules. When the subsets of rules are analyzed individually, the intersections and inconsistencies are ignored and as a result too many misclassifications occur. Nonetheless, the proposed method is capable of recognizing that situation and selecting the rule most adequate to classify the instance.

It should also be noticed that as the number of attributes decreases, the *tuple space* is drastically reduced. The databases Tic-Tac-Toe and Car are densely populated (in terms of number of instances). Thus, the act of sampling data does not cause a strong impact on the performance of the local symbolic classifiers. On the other hand, when those classifiers are merged, the system is not able to identify the best rule, what results in misclassifications. Therefore, we believe that the technique introduced in this work is indicated to mining distributed data sets which represent partial views of the *tuple space*.

6 Discussions

This paper introduced a method for integration of symbolic classifiers based on *Evidential Paraconsistent Logics*. The method is capable of reliable decision-making merging a set of classifiers based on rules, even when contradictory information is present.

In the distributed data mining domain, several data-oriented approaches use meta-learning [4] [15] and sampling, splitting the databases into smaller subsets, which are processed individually and later all classifiers are amalgamated into a global one. A drawback of such approach is that the existence of inconsistent across the rule sets may jeopardize performance and generate classifiers with contradictory views.

The aim of the proposed method was handling possible inconsistencies, arising from the merge of *N* subsets of rules which resulted from data segmentation, avoiding large communication flow among processors, decreasing processing time and, more importantly, ensuring an acceptable accuracy rate. The use of Paraconsistent Logics permitted assigning evidential factors belief and disbelief to the rules, indicating the degree of importance of a rule in relation to all subsets. By sorting the set of rules, they could be compared against the logical state which represents absolute truth, thereby resulting in a decision criterion for selecting candidate rules.

The analysis of results made it possible to identify that the relation between the number of attributes which constitute the database and the number of instances may have a significant impact on the performance of the method. Noticeably, the method exhibits better performance when the relation between the number of attributes and the number of instances is high (the *tuple space* is large), because the local models produced cover distinct regions of the *tuple* space.

In spite of the good results observed, more experiments and research need to be carried out to improve the method. One could work on identifying the best rules for classification, designing a pruning method in order to improve the comprehensibility of the final model. The technique needs yet be evaluated with different symbolic learning algorithms so as to determine the impact of their *bias* on the final model. It is known that, for instance, some algorithms are more stable than others. The system needs also adaptation for dealing with non-ordered sets of rules and evaluations with different percentages of training and test data.

7 References

1 Ávila, B. C. *Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar de Exceções em Sistemas de Frames com Múltipla Herança.* Tese de Doutorado, Escola Politécnica da Universidade de São Paulo. São Paulo, 1996.

2 Blake, C.L.; Newman, D.J.; Hettich, S.; Merz, C.J. *UCI Repository of machine learning databases*. [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

3 Blair, H.A.; Subrahmanian, V. S. *Paraconsistent Foundations for Logic Programming*. Journal of Non-Classical Logic, 5, 2, pag. 45-73, 1988

4 Chan, P. K.; Stolfo, S. J. *Toward Scalable Learning with Non-Uniform Distribution: Effects and a Multi-Classifier Approach*. KDDM. pg.164-168,1998.

5 Cohen, W. W. *Fast Effective Rule Induction*. In Proceedings of the 12th Int. Conference in Machine Learning (ICML '95). Pages 115-123. 1995.

6 da Costa, N.C.A. et al, N. A. *Lógica Paraconsistente Aplicada.* Atlas, São Paulo, 1999.

7 da Costa, N. C. A; J. M. Abe. *Paraconsistência em Informática e Inteligência Artificia*l. Revista Estudos Avançados n 14, vol 39 - USP. São Paulo, Maio/Agosto 2000.

8 Enembreck, F. *Um Sistema Paraconsistente para Verificação Automática de Assinaturas Manuscritas*. Dissertação de Mestrado PPGIA – PUCPR. Curitiba, 1999.

9 Enembreck, F. ; Ávila, B. C. *KNOMA: A new Approach for Knowledge Integration*. In: 11th IEEE Symposium on Computers and Communications, 2006, Sardinia. IEEE Symposium on Computers and Communications, 2006.

10 Ferreira, S. M. N.; Ávila, B. C.; Freitas, A. A. *Handling Inconsistency in Distributed Data Mining with Paraconsistent Logic*. In: Turkish Symposium on Artificial Intelligence and Neural, Tainn, 2004.

11 Frank, E. *WEKA Machine Learning Software*. [http://www.cs.waikato.ac.nz/ml/weka]

12 Freitas, A.; Lavington, S. H. Approaches to Speed Up Data Mining. *Mining Very Large Databases with Parallel Processing*. ISBN 0-7923-8048-7. Pages 89-108. Kluwer Academic Publishers, The Netherlands, 1998.

13 Hall, O. L.; Chawla, N.; Bowyer, K. W.; Kegelmeyer, P. *Learning Rules from Distributed Data*. Department of Computer Science and Engineering, University of South Florida. USA. ISBN 3-540-67194-3. 1999

14 Prati, R. C. *Novas abordagens em Aprendizado de Máquina para Geração de Regras, Classes desbalanceadas e Ordenação de Casos.* Tese apresentada ao Instituto de Ciências Matemáticas e de Computação ICMC-USP, São Carlos, 2006.

15 Prodomidis, A. L., Stolfo, S. J.; Chan, P. K. *Meta-Learning in Distributed Data Mining Systems: Issues and Approaches.* In Advances in Distributed and Parallel Knowledge Discovery, Chapter 3. 2000.

16 Subrahmanian, V. S. *On the Semantics of Quantitative Logic Programs*. Proceedings of 4th IEEE Symposium on Logic Programming, San Francisco, September, pp. 173-182, 1987.