Handling Inconsistency in Distributed Data Mining with Paraconsistent Logic

Simone N.M. Ferreira[1,2] Alex A. Freitas[3] Braulio C. Avila[4]

[1] CEFET-PR, Av. Mont. Lobato, Km 04, s/n, C.P 20. Ponta Grossa, PR. 80215-901. Brazil.
[2] ITA, Praca Mal. Eduardo Gomes, 50, Sao Jose dos Campos, SP, 12228-900. Brazil.
[3] Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK.
[4] PPGIA, PUC-PR, R. Imaculada Conceicao, 1155, Curitiba, 8025-901, Brazil.

Abstract. This paper addresses the problem of inconsistent rule subsets in distributed data mining. In this scenario, *N* rule subsets are independently discovered from *N* different data subsets. This can result in inconsistent rules – i.e. rules with the same antecedent but different class predictions – across the *N* rule subsets. In order to handle these rule inconsistencies, this paper proposes a paraconsistent logic-based method for post-processing different rule subsets discovered by a rule induction algorithm in a distributed data mining scenario. The proposed method produces a global inconsistency-free rule set by using principles and concepts of paraconsistent logic, a relatively novel kind of logic developed specifically for inconsistency handling.

1 Introduction

At present many computing environments are inherently distributed. Some typical examples are the world wide web and distributed databases. There are several driving forces for distributed databases [18], [15]. First, real-world applications often involve computational constrains, such as the fact that the database might be too large to be kept into a single site. Second, there might be legal constraints and/or privacy concerns that require the data to be distributed, rather than stored at a single site. Third, distributed databases can offer significantly better performance than centralized databases. Finally, data distribution can be a natural approach for a company whose business activities are inherently decentralized.

Hence, it is important to develop data mining algorithms that can effectively mine distributed data [13], [10]. One of the challenges of distributed data mining is that two or more data items stored in different sites might be inconsistent with each other in the context of the data mining task being solved. For instance, suppose the target task is classification, the class attribute is the *Credit* of the customer, one site stores the tuple <*Gender = male, Salary = high, Loan = No, Credit = good*> and another site the tuple <*Gender = male, Salary = high, Loan = No, Credit = bad*>. These two tuples are inconsistent, because they have the same values for all predictor attributes but different values of the class attribute. In a distributed database scenario, the two main causes for this inconsistency are as follows.

First, inconsistency can be caused by different attribute value definitions or, more generally, differences in the schema of the local data subsets. Continuing the previous example, it is possible that the two data sites have different definitions for the attribute value *Salary = high*. For instance, the first data site could contain data about customers based in London, whereas the second data site could data about customers located in a small English town where the salary tends to be much smaller than in London. Hence, the value of salary used as a threshold to determine a high salary would be different in the two local data sets.

Second, even if the data sites have exactly the same database schema, there can still be data inconsistencies due to the fact that different data sites have different probability distributions. This is the

case, for instance, when data is independently collected and stored in each data site by a largely independent unit of a highly decentralized organization or group of organizations.

In addition, there are also other possible causes of data inconsistency that are not specifically related to the scenario of distribute databases, but they can occur in either distributed or centralized databases. Some examples are the presence of noise and the problem of missing relevant attributes, as follows. In essence, noise refers to random errors in the data. Note that, although noise can lead to inconsistency, they are different problems. A data set can be noisy but inconsistency-free, or noise-free but inconsistent. To illustrate the importance of this difference, the noise filtering method proposed by Gamberger et al. [11] requires that the data be inconsistency free.

The problem of missing relevant attributes, or inconclusive data [20], is characterized by the fact that the available predictor attributes are not enough to discriminate between the classes. This is a direct cause of data inconsistency, by definition. The relevance of this problem is clearly shown by this quote from Uthurusamy [20]: "*After studying several diagnostic databases at General Motors, we are convinced that inconclusiveness [inconsistency] is a characteristic that rule induction algorithms must be able to deal with if they are to have any practical application in industry.*"

Although some methods have been proposed to remove some kinds of data inconsistency in a pre-processing step [12], [7], [17], this "data cleaning" approach for inconsistency removal might not be very cost-effective. This approach can be difficult, expensive and slow. In addition, this approach usually requires a considerable amount of human intervention. Actually, it is interesting to note that the database community has recently started to advocate a different approach for the problem of inconsistency handling where, instead of removing inconsistency from the database, they keep the inconsistent data in the database and modify the queries in such a way that they retrieve only consistent information [3], [16]. In any case, the data cleaning approach is intended to cope only with data inconsistency arising from differences in database schema or related problems. In principle it cannot solve, for instance, the problem of inconsistencies in different data sets with exactly the same database schema but different probability distributions, which is an important problem in distributed data mining. Finally, removing inconsistency in a pre-processing step might even be undesirable from a data mining point of view, since the presence of inconsistency can be a valuable clue for the user and can lead to interesting discoveries. As a simple example, inconsistencies in a taxpayer's records can be very useful for detecting a possible fraud [22].

To summarize, data inconsistency is a challenging problem in distributed data mining, and it is important to develop methods that can effectively solve this problem in a principled way. In this spirit, this paper proposes a *paraconsistent logic*-based method for handling inconsistency in distributed data mining.

The knowledge representation used in this work consists of IF-THEN classification rules, where the antecedent (IF part) contains a conjunction of attribute-value pairs and the consequent (THEN part) contains the class predicted for an example (record or data item) that satisfies all the conditions in the rule antecedent. It is well-known that this representation has the advantage of being intuitively comprehensible for the user [21], which was our main motivation to use it in this work.

In essence, the method proposed in this paper works as follows. First, a rule induction algorithm is applied to each of the local data subsets of a distributed data set, discovering a local rule set from each of those local data subsets. Then inconsistencies between rules of different local rule sets are handled. Two or more rules are said to be inconsistent if they have the same rule antecedent (IF part) but make different class predictions – i.e., have different rule consequents (THEN part). Inconsistent rules are handled by using concepts and principles of paraconsistent logic [5], [19], a relatively novel kind of logic specifically developed to cope with inconsistency. As a result, a global, inconsistency-free rule set is produced, which can then be used to classify unseen examples in the test set.

Therefore, the proposed method can be regarded as a method for post-processing different rule subsets discovered by a rule induction algorithm in a distributed data mining scenario, in order to produce a global inconsistency-free rule set. Although this was the original motivation for development of the method, it turns out that it can also be applied to other data mining scenarios where different rule sets are discovered from different data subsets, such as some ensemble scenarios, as will be discussed later.

To the best of our knowledge, the only previous work using paraconsistent logic in data mining is the work of Enembreck et al. [9], which is very different from our work. Enembreck at al. have developed a paraconsistent version of a decision tree algorithm, with no mention of distributed data mining. In particular, their method for computing belief and disbelief degrees (terms to be defined later) is entirely different from ours, since they address a different kind of data mining problem.

2 A Distributed Data mining Scenario

The method for inconsistency handling proposed in this paper is based on a distributed data mining framework in which the different data sites have the same database schema but possibly different probability distributions. More precisely, this scenario can be described as follows.

We are given N data sets, each of them with M predictor attributes and a class attribute. Each of the M attributes has the some domain in all the N data sets. The knowledge representation used in this work consists of IF-THEN classification rules, as mentioned in the Introduction.

The distributed data is mined in two phases: a) generating a rule set for each local data set; and b) combining the multiple local rule sets into a single global rule set. This basic two-phase scenario is often used in the distributed data mining literature – see, e.g., [14] – and its basic idea is illustrated in Figure 1.

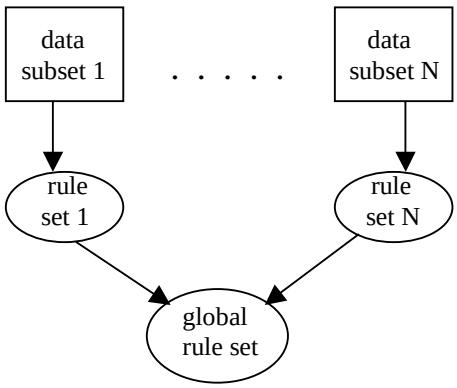

Figure 1: generating a global rule set from N local data subsets

We emphasize that each local prediction model is generated using only a subset of the entire data. Therefore, it is quite possible that the local models are somewhat inconsistent with each other. This can be due, for instance, to the fact that different data subsets have different probability distributions, as mentioned earlier.

As mentioned earlier, the method proposed in this paper can be regarded as a post-processing method applied to the output of a distributed rule induction algorithm. That is, it generates a single global, inconsistency-free rule set out of the N local rule sets discovered by N runs of a rule induction algorithm – each run applied to a different data subset. The method will be described in section 4, after an overview of paraconsistent logic (in which the method is based) in section 3.

3 An Overview of Paraconsistent Logic

One definition of paraconsistent logic is as follows [5], [2]. A deductive system *S* is said to be inconsistent if there is a formula *a* of *S* such that both *a* and its negation *(¬a)* are theorems of *S*. If all formulas of *S* are theorems then *S* is said to be trivial, otherwise it is said to be non-trivial. In classical logic *S* is trivial if and only if it is inconsistent. A paraconsistent logic is a logic where *S* is inconsistent and non-trivial. A paraconsistent logic can also be defined as one that rejects the principle of *ex-contradiction sequitur quod libet (EC) – i.e., "from a contradiction, everything follows"*. This princi-

ple can be formalized as follows: "For any theory *T*, formulas *a*, *b*: *T*, *a*, ¬*a* ⊢ *b*. Hence, a logic is paraconsistent if and only if there exists a theory *T* and formulas *a*, *b* such that *b* cannot be proved from *T*, *a*, ¬*a*.

In general, paraconsistent logics are useful because they allows us to *restrict the effects of inconsistency* in a deductive system *S*. Intuitively, the *EC* principle (which is followed by classic logic) is undesirable in many practical data mining situations. As a simple example, suppose that for each patient in a hospital database we have one or more sources of evidence for diagnosing whether or not the patient has a given disease. It is possible that all the evidence about a given patient p_1 suggests that p_1 does not have the disease, whereas we have contradictory evidence about patient p_2 – i.e., one evidence suggests it has the disease, whilst another evidence suggests the opposite. Intuitively, since the contradiction involves only p_2, this contradiction should not affect our conclusion that p_1 does not have the disease. However, the *EC* principle would allows us to conclude the opposite as well, which is undesirable. By rejecting that principle a paraconsistent logic would allow us to conclude only that p_1 does not have the disease, and not the opposite.

Among the many paraconsistent logics proposed in the literature, this work is based on evidential paraconsistent logic with a two-valued annotation [19], [6], hereafter denoted 2vA-PL. The basic idea is that each proposition p is associated with a two-valued annotation <μ, ν>, where μ is the degree of belief in the proposition p and ν is the degree of disbelief in p. Both μ and ν take on values in the range [0..1]. In other words, μ and ν are interpreted as the (normalized) amount of evidence for and evidence against p, respectively.

One motivation for choosing this kind of paraconsistent logic in this work is its evidential reasoning interpretation, which is intuitively desirable in a data mining context. Belief is different from truth, and the classification task of data mining (addressed in this paper) essentially involves induction from data, where the degree of belief in a rule should be proportional to the available evidence – corresponding to the data being mined. In addition, the use of this kind of paraconsistent logic seems suitable for distributed data mining. As discussed earlier, in distributed data mining it is quite possible that local rule sets – each one discovered from a different local data subset – are somewhat inconsistent with each other. Hence, a given rule can have a high *belief* degree with respect to the data subset where it was discovered from but at the same time a high *disbelief* degree with respect to other data subsets (since the latter can, for instance, have a probability distribution quite different from the former).

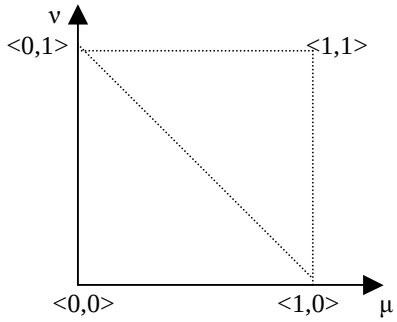

Figure 2: Two-valued annotation of a proposition represented in a Cartesian plan

We emphasize that, unlike probability theory, in 2vA-PL there is no requirement that μ + ν = 1. Actually, in 2vA-PL the belief and disbelief degrees are independent from each other, that is, μ and ν are computed from different sources of evidence – different data subsets, in the case of distributed data mining. Hence, the two-valued annotation of a proposition can be represented as a point in the Cartesian plan shown in Figure 2, with the belief degree represented as the horizontal axis and the disbelief degree represented as the vertical axis.

The point at coordinates <1,0> (i.e., μ =1, ν = 0) represents the case where we have total belief in the truth of the proposition, and the point at coordinates <0,1> represents the case where we have total

disbelief in the truth (or, equivalently, total belief in the falsity) of the proposition. Intuitively, these two points correspond to the truth degrees true and false of classic logic, respectively. The point at coordinates <1,1> represents the case where one source of evidence leads us to have a total belief in the truth of the proposition, whereas another source of evidence leads us to have a total belief in its falsity. Hence, we have maximum inconsistency between the two sources of evidence. Finally, the point at coordinates <0,0> represents the case where there is no evidence for nor against the proposition. Hence, we have a situation of maximum indeterminacy.

Note that in Figure 2 there is a dashed diagonal line linking the point at coordinates <0,1> to the point at coordinates <1,0>. This line is called the "perfectly-determined" line. All points belonging to this line have an interesting property: they satisfy the equation $\mu + \nu = 1$. Points "above" this line satisfy the inequality $\mu + \nu > 1$, and they represent propositions that are "over-determined". By contrast, points below the perfectly-determined line satisfy the inequality $\mu + \nu < 1$, and they represent propositions that are "under-determined". The degree of over-determinacy or under-determinacy of a proposition can be precisely quantified by computing its degree of indeterminacy, denoted by *ind*, as follows: $ind = |\mu + \nu - 1|$, where $|x|$ denotes the absolute value of x. Note that *ind* varies in the range [0..1]. The larger the value of *ind*, the larger the degree of indeterminacy. The measure of indeterminacy is interpreted either as a measure of over-determinacy, if $\mu + \nu > 1$, or as a measure of under-determinacy, if $\mu + \nu < 1$. The method proposed in this paper, described in the next section, uses the complement of *ind*, called the degree of determinacy (*det*), defined as $det = 1 - ind$. Another concept of 2vA-PL used by the method proposed in this paper is the truth degree of a proposition, denoted *tr*, which is defined as $tr = \mu - \nu$, that is, the belief degree minus the disbelief degree. Hence, *tr* varies in the range [-1..1]. Of course, the closer the value of *tr* is to 1, the more we believe the proposition is true, and the closer the value of *tr* is to -1, the more we believe the proposition is false.

4 A Paraconsistent Logic-based Method for Handling Inconsistency in Distributed Rule Sets

As mentioned earlier, in the context of this paper distributed data is mined in two phases: a) generating a rule set for each local data set; and b) combining the multiple local rule sets into a single global rule set. The proposed method was designed for performing the second phase. Hence, it accepts, as input, a set of N rule sets – one for each of the N data sets – which have been discovered by a classification algorithm.

At a high level of abstraction, the method consists of 3 steps, as follows. First, it computes the degree of belief and disbelief for each rule of each rule set. Second, it detects rules from two or more different rule sets which have the same antecedent, putting them into the same group. Third, it merges the rules within each group into a single rule. These steps are now described in more detail.

4.1 First Step: Computing a Rule's Degree of Belief and Disbelief

The degree of belief of each rule is simply the confidence factor of that rule, which is essentially the number of examples correctly classified by the rule divided by the total number of examples covered by the rule. More precisely, let μ be the degree of belief of a given rule discovered from a data subset D, C be the class predicted by the rule, and A be the rule antecedent (a conjunction of conditions). Then $\mu = |C \,\&\, A|_{(D)} / |A|_{(D)}$, where $|C \,\&\, A|_{(D)}$ is the number of training examples in D having class C and satisfying all conditions in A, and $|A|_{(D)}$ is the number of training examples in D satisfying all conditions in A.

Note that the degree of belief of a rule is computed by using only the local data subset from which the rule was discovered. At first glance the degree of disbelief, denoted by ν, could be defined as the complement of the degree of belief, i.e., $\nu = 1 - \mu$. However, this would be wrong. Recall that in evidential paraconsistent logic 2vA-PL the degrees of belief and disbelief must be independent from each other. In other words, the degree of disbelief must be computed from a different data source than the degree of belief, which corresponds to using a different kind of evidence. Therefore, we compute the degree of disbelief of a rule by using all the data subsets *except* the one from which the rule was discovered. More precisely, let D' be the multi-set (possibly containing duplicates) of examples consist-

ing of all examples that belong to any data subset different from D, which is the data subset from which the rule was discovered. Then $\nu = |C' \& A|_{(D')} / |A|_{(D')}$, where $|C' \& A|_{(D')}$ is the number of training examples in D' having class different from C and satisfying all conditions in A, and $|A|_{(D')}$ is the number of training examples in D' satisfying all conditions in A. In other words, the degree of disbelief of a rule is essentially the number of examples incorrectly classified by the rule in D' divided by the total number of examples covered by the rule in D'.

It should be noted that, although the computation of ν uses data from $N - 1$ data subsets, there is no need to move such a large amount of data among the distributed sites. The only thing that needs to be moved among the distributed sites are summarized statistics. More precisely, in order to compute ν for each rule, each site just has to compute two numbers – the number of examples incorrectly classified by the rule and number of examples covered by the rule in its local data subset – and send those two numbers to the site storing the rule, which simply adds up the results to compute $|C' \& A|_{(D')}$ and $|A|_{(D')}$. All the $N - 1$ sites (i.e., the sites different from the one storing the rule) can do this in parallel, and this activity can also be done in parallel with the computation of μ in the site storing the rule. Hence, the computation of the belief and disbelief degrees for each rule is entirely distributed and parallel, and it involves just a minor amount of "data communication" among the distributed sites.

4.2 Second Step: Identify Rules with the Same Antecedent in Different Sites

This step is simple. Each data site sends its discovered rules – with μ and ν computed as in the previous step – to a coordinator site. The coordinator divides the entire rule set into groups of rules, in such a way that each group contains all the rules with the same antecedent. Hence, the number of rules in each group varies from 1 to N (the number of data subsets), assuming that at most one rule with a given antecedent can be discovered from each data set – a very natural assumption in the classification task.

4.3 Merging the Rules within Each Group into a Single Rule

In this step the rules within each group of rules formed in the previous step are merged into a single rule. Of course, if a group contains just one rule, there is no merging to be done, and the rule is considered a discovered rule, without any modification. If a group contains two or more rules, this step uses the evidential paraconsistent logic 2vA-PL concepts that take into account the belief and disbelief degrees of all rules within the group in order to merge them into a single rule. To simplify our discussion, suppose a group contains two rules – the arguments below can be easily generalized to the case with more than two rules within a group.

There are two cases to be considered: (i) the rules predict the same class in their consequent, or (ii) the rules predict different classes in their consequent. In case (i) there is no inconsistency in the predictions of the rules, and the rule resulting from the "merging" – denoted by r – trivially contains the same antecedent and consequent as the two rules being merged. The only problem is to compute μ and ν for rule r, denoted μ_r and ν_r. There are two sub-cases to consider. First, if the two rules being merged – denoted r_1 and r_2 – have the same values of μ and ν, the merged rule trivially has the same values of μ and ν, i.e., $\mu_r = \mu_{r1} = \mu_{r2}$ and $\nu_r = \nu_{r1} = \nu_{r2}$. Second, if the rules have a different value of μ or ν, we need to compute μ_r and ν_r as aggregated values of μ_{r1}, μ_{r2}, ν_{r1}, ν_{r2}, respectively. In this work we use the supreme operation of paraconsistent logic, so that $\mu_r = \max(\mu_{r1},\mu_{r2})$ and $\nu_r = \max(\nu_{r1},\nu_{r2})$, where μ_{r1}, μ_{r2}, ν_{r1}, ν_{r2} are computed using the procedure described in step 1 (subsection 4.1). Other approaches to compute the aggregated values μ_r and ν_r will be explored in future research.

Consider case (ii), where clearly there is an inconsistency in the predictions of rules r_1 and r_2, since they have the same antecedent but predict different classes. To cope with this inconsistency we use the paraconsistent logic's concepts of truth degree and determinacy degree, as explained in section 3. More precisely, the system first computes $tr_{r1} = (\mu_{r1} - \nu_{r1})$, $tr_{r2} = (\mu_{r2} - \nu_{r2})$, $det_{r1} = 1 - |\mu_{r1} + \nu_{r1} - 1|$, and $det_{r2} = 1 - |\mu_{r2} + \nu_{r2} - 1|$. Then it computes a single measure of quality (Q) for each rule. This measure favours rules with a larger truth degree and with a larger determinacy degree, being defined as the product of these two degrees. That is, $Q_{r1} = tr_{r1} * det_{r1}$, $Q_{r2} = tr_{r2} * det_{r2}$. The motivation for this quality measure is that we want to maximize both the truth degree and the determinacy degree of a rule. For instance, the ideal rule would have $\mu = 1$ and $\nu = 0$, which would result in tr = 1 and det = 1, corresponding to the maximum possible value of Q.

Finally, the rule r resulting from the merging of rules r_1 and r_2 is defined as follows. First, r contains the same antecedent as r_1 and r_2 (recall that these two rules always contain the same antecedent, as a result of the grouping process performed in step 2). Second, if $Q_{r1} > Q_{r2}$ then r contains the same consequent as r_1, and we make $Q_r = Q_{r1}$. Otherwise r contains the same consequent as r_2, and we make $Q_r = Q_{r2}$.

4.4 Classifying Examples in the Test Set

Once the previous three steps have been performed, the system has produced a set of rules where each rule has a different antecedent and each rule is assigned a quality measure Q. We are now ready to classify examples in the test set, which is done by using a procedure which is commonplace in the rule induction literature, as follows.

For each new test example, the system determines the rules covering that example. If there is a single rule covering that example, the example is trivially assigned to the class predicted by that rule. If there is no rule covering that example, it is simply assigned to the majority class – corresponding to a "default rule". Finally, if there are two or more rules covering that example, it is classified by the rule with the largest value of the Q measure.

5 A "Proof-of-Concept" Computational Experiment

In order to evaluate the proposed method, we have performed a preliminary experiment with the tic-tac-toe data set, a public domain data set obtained from the well-known UCI data repository (http:// www.ics.uci.edu/~mlearn/MLRepository.html), and a simple simulation of a distributed environment with two data subsets. We emphasize that this experiment is intended to be a "proof-of-concept" experiment. In order to effectively evaluate the performance of the proposed method we would need to use more data sets and preferably data that is really distributed in the real-world, which is left for future research. The tic-tac-toe data set contains 958 examples and 9 predictor attributes, corresponding to the 9 squares in a game of tic-tac-toe. The class attribute indicates whether or not a given position of the board is a win for a given player. The data set was first randomly divided into a training set and a test set, with 638 examples and 320 examples, respectively. The training set was then divided into two "local" training subsets with 319 instances each, to simulate a distributed environment.

Recall that the method proposed in the previous section has the goal of integrating the rule sets discovered by different runs of a rule induction algorithm (or even runs of different rule induction algorithms), in a post-processing phase. Hence, any rule induction algorithm could be used to discover rules from each data subset, including decision tree induction algorithms – since a decision tree can be directly converted into a set of rules. In this work we use the well-known PRISM rule induction algorithm [4], more precisely the public-domain implementation of this algorithm available from the WEKA data mining tool [21].

After applying PRISM to each of the two training subsets, the proposed paraconsistent logic-based method was applied to post-process the two discovered rule sets, producing a global rule set which was used to classify examples in the test set. The classification error rate obtained in the test set was 6.88%.

We also measured the classification error rate associated with PRISM only, as a baseline method – i.e., without using the proposed paraconsistent logic-based post-processing method. This baseline method consists of applying PRISM separately to each training subset and simply create a global rule set with all discovered rules, ignoring inconsistencies between the two local rule sets. At a high level of abstraction, the procedure used to classify examples in the test set is the same as the one used with the proposed post-processing method, as described in section 4.4. One difference, at a lower level of abstraction, is that the quality measure of the rules – used to choose the "winner" rule when two or more rules cover the test example – is different in the two methods. When using the proposed paraconsistent logic-based post-processing method, rule quality is given by the product of truth degree and determinacy degree, as discussed earlier. When using the baseline method, rule quality is given sim-

ply by the quality measure computed by the original PRISM algorithm. This baseline method obtained an error rate of 7.41%.

To summarize, the use of the proposed paraconsistent logic-based method (applied as a post-processing method to two local rule sets discovered by PRISM) led to a reduction of the error rate (by comparison with the baseline use of PRISM) from 7.41% to 6.88%, which can be considered a valid "proof-of-concept" result.

It is also interesting to note that the baseline version of PRISM discovered 81 rules from the two data subsets. When the two rule subsets were handled by our paraconsistent logic-based method, the system detected that 10 out of those 81 were inconsistent. This confirms that indeed inconsistent rules arise when independently mining distributed data subsets. Those 10 inconsistent rules were then post-processed by our method, so that each pair of inconsistent rules (with the same antecedent but predicting different classes, each rule in the pair coming from a different data subset) was merged into a single rule, as discussed earlier. This produced a global set with 76 rules, which led to a reduction of the error rate in the test set.

6 Discussion

This paper has proposed a paraconsistent logic-based method for post-processing different rule subsets discovered by a rule induction algorithm in a distributed data mining scenario, in order to produce a global rule set where rule inconsistencies have been properly handled. Although the original motivation for developing the method was to tackle the problem of inconsistency in distributed data mining, it turns out that it can also be applied to other scenarios where different rule sets are discovered from different data subsets. In particular, it can be applied in some ensemble scenarios (a popular scenario in the machine learning and data mining literature), as follows.

The basic idea of many ensemble methods consists of two phases. In the first phase the system learns N base-level rule subsets from the training set. This can be achieved in several different ways [8], [1]. For instance, one can run N different rule induction algorithms on the training set, one can use training set resampling to learn N rule sets from N different subsets of the training set, etc. In any case, the N base-level rule subsets might contain inconsistent rules, since they were independently discovered from each local data subset. In the second phase the rules of the base-level rule subsets have to be combined into a single rule set. Methods to perform this second phase can be roughly divided into two broad kinds of approach, namely approaches without meta-learning and approaches with meta-learning.

A typical representative of the no-meta-learning approach is plurality voting. A typical representative of the meta-learning approach is stacking, where a meta-learning algorithm is applied to the predictions made by the N base-level classifiers (e.g., local rule subsets), in order to generate a meta-classifier (e.g., a global rule set). In the context of an ensemble of rule sets, the method proposed in this paper is directly applicable if no meta-learning approach is used. However, it is not very suitable for the meta-learning approach, where inconsistencies in the predictions of the base-level rule subsets are already handled by the meta-learning algorithm.

To summarize, the proposed method has the advantage of being quite generic. It can be applied to virtually any data mining scenario where several subsets of classification rules have been independently discovered from different data subsets. In these scenarios a simple union of those rule subsets would produce a global rule set prone to rule inconsistencies. In order to solve this problem, the proposed method can be used to handle those rule inconsistencies and produce a global inconsistency-free rule set, using principles and concepts of evidential paraconsistent logic. This is a principled approach, since this kind of logic was specifically developed to handle inconsistency.

In addition, it should be noted that the proposed method is well suited for parallel processing because the most computationally expensive step of the method, the computation of the degrees of belief and disbelief for each rule (section 4.1), is entirely distributed and parallel. The data subsets themselves are never moved – only very summarized statistic information is moved among the distributed sites. The other two steps of the method (sections 4.2 and 4.3) involve a central coordinator site, so they cannot exploit parallelism across different data sites, but this is not a serious problem because

these two steps are computationally fast. In parallel data mining (as in other parallel processing applications, for that matter) the most important thing is to parallelize the most computationally expensive part of the algorithm [10], which is accomplished in the proposed method.

The main limitation of this work is that the reported computational results are still preliminary. Hence, future research should involve more extensive computational experiments – with more data sets and larger data sets. It would also be interesting to investigate new ways of using paraconsistent logic in data mining.

The method proposed in this paper is the first attempt to apply paraconsistent logic to distributed data mining, and so it offers many opportunities for further research in this area. We hope that it will generate enough interest in the data mining community to do this further research, which would probably lead to more advanced paraconsistent logic-based algorithms for data mining.

References

1. K. Ali. On the link between error correlation and error reduction in decision tree ensembles. Dept. of Information and Computer Science, Technical Report 95-38. University of California at Irvine, CA, USA. 1995.
2. J.-Y. Beziau. What is paraconsistent logic? In: D. Batens, C. Mortensen, G. Priest, J.-P. Van Bendegem (Eds.) *Frontiers of Paraconsistent Logic*, pp. 95-111. Baldock, Hertfordshire, UK : Research Studies Press, 2000.
3. A. Celle and L. Bertossi. Querying inconsistent databases: algorithms and implementation. *Computational Logic: Proc. 1st Int. Conf. (CL-2000). Lecture Notes in Artificial Intelligence 1861*, pp. 942-956. Springer-Verlag, 2000.
4. J. Cendrowska. PRISM : an algorithm for inducing modular rules. *International Journal of Man-Machine Studies, 27*, pp. 349-370. 1987.
5. N.C.A. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic, Vol. XV, No. 4*, pp. 497-510. Oct. 1974.
6. N.C.A. da Costa, J.M. Abe, J.I.S. Filho, A.C. Murolo, C.F.S. Leite. *Logica paraconsistente aplicada (In Portuguese)*. Sao Paulo, Brasil: Atlas, 1999.
7. W.W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. In: *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD-2000)*, pp. 255-259. ACM, 2000.
8. T. Dietterich. Machine learning: four current directions. *AI Magazine 18(4)*, pp. 97-136. 1997.
9. F. Enembreck, B.C. Avila and R. Sabourin. Decision tree-based paraconsistent learning. *Proc. XIX Int. Conf. of the Chilean Computer Society*, pp. 43-52. IEEE Computer Society Press, 1999.
10. A.A. Freitas & S.H. Lavington. *Mining Very Large Databases with Parallel Processing*. Kluwer, 1998.
11. D. Gamberger, N. Lavrac, and C. Groselj. Experiments with noise filtering in a medical domain. *Proc. ICML-99*. Morgan Kaufmann, 1999.
12. W. Hsu, M.L. Lee, B. Liu and T.W. Ling. Exploration mining in diabetic patients databases: findings and conclusions. In: *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD-2000)*, pp. 430-436. ACM, 2000.
13. H. Kargupta and P. Chan (Eds.) *Advances in Distributed and Parallel Knowledge Discovery*. AAAI/MIT, 2000.
14. A. Lazarevic and Z. Obradovic. The distributed boosting algorithm. In: *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD-2001)*, pp. 311-316. ACM, 2001.
15. R. Pairceir, S. McClean, B. Scotney. Discovery of multi-level rules and exceptions from a distributed database. *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD-2000)*, pp. 523-532. ACM, 2000.
16. S. Rice and J.F. Roddick. Lattice-structured domains, imperfect data and inductive queries. In: *Database and Expert Systems Applications (Proc. DEXA-2000), LNCS 1873*, pp. 664-674. Springer, 2000.

17. S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In: *Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD-2002)*. ACM, 2002.
18. S. Stolfo, A.L. Prodromidis, S. Tselepis, W. Lee, D.W. Fan and P.K. Chan. JAM : Java agents for meta-learning over distributed databases. *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD-97)*. AAAI, 1997.
19. V.S. Subrahmanian. Towards a theory of evidential reasoning in logic programming. *Logic Colloquium'87 – The European Summer Meeting of the Association for Symbolic Logic*. Granada, Spain, July 1987.
20. R. Uthurusamy, U.M. Fayyad, S. Spangler. Learning useful rules from inconclusive data. In: G. Piatetsky-Shapiro & W.J. Frawley (Eds.) *Knowledge Discovery in Databases*, pp. 141-157. AAAI Press/MIT Press, 1991.
21. I.H. Witten and E. Frank. *Data Mining: practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann, 2000.
22. P. Wong. Inconsistency and preservation. *Topics in Artificial Intelligence (Proc. PRICAI-2000), LNAI 1866*, pp. 50-60. Springer, 2000.