Using Distributed Data Mining and Distributed Artificial Intelligence for Knowledge Integration

Ana C.M.P. de Paula, Bráulio C. Ávila, Edson Scalabrin, and Fabrício Enembreck

Pontifícia Universidade Católica do Paraná – PUCPR
PPGIA – Programa de Pós-Graduação em Informática Aplicada
Rua Imaculada Conceição, 1155, CEP 80215-901
Curitiba PR, Brasil
Tel./Fax: +55 41 3027-1669
{avila,scalabrin,fabricio}@ppgia.pucpr.br

Abstract. In this paper we study Distributed Data Mining from a Distributed Artificial Intelligence perspective. Very often, databases are very large to be mined. Then Distributed Data Mining can be used for discovering knowledge (rule sets) generated from parts of the entire training data set. This process requires cooperation and coordination between the processors because inconsistent, incomplete and useless knowledge can be generated, since each processor uses partial data. Cooperation and coordination are important issues in Distributed Artificial Intelligence and can be accomplished with different techniques: planning (centralized, partially distributed and distributed), negotiation, reaction, etc. In this work we discuss a coordination protocol for cooperative learning agents of a MAS developed previously, comparing it conceptually with other learning systems. This cooperative process is hierarchical and works under the coordination of a manager agent. The proposed model aims to select the best rules for integration into the global model without, however, decreasing its accuracy rate. We have also done experiments comparing accuracy and complexity of the knowledge generated by the cooperative agents.

Keywords: Distributed Data Mining, Distributed Artificial Intelligence, Cooperative Agents, Learning Agents.

1 Introduction

Data Mining [1] [3] permits efficient discovery of valid, non-obvious information in large collections of data, and it is used in information management and decision making, enabling an increase in business opportunities.

Despite the augmentation of computer processing power, much attention is given to speeding up the data mining process. Basically, speeding up approaches can be either (i) data-oriented or (ii) algorithm-oriented. In (i) the dataset is processed and the learning instances space is reduced by discretization, attribute selection or sampling. In (ii) new search strategies are studied or data are mined in a distributed or parallel fashion.

M. Klusch et al. (Eds.): CIA 2007, LNAI 4676, pp. 89–103, 2007.

According to Freitas and Lavigton [12], Distributed Data Mining (DDM) [4] [5] consists of partitioning the data being mined among multiple processors, applying the same or different data mining algorithms to each local subset and then combining the local knowledge discovered by the algorithms into a global knowledge. The authors also discuss that such global knowledge is usually different (less accurate as the number of subsets increases) from the knowledge discovered by applying the mining algorithm on the entire dataset formed from the union of the individual local datasets. Unfortunately, subsets are often too large to be merged into a single dataset and then, some distributed learning approach must be used.

DDM can deal with public datasets available on the Internet, corporate databases within an Intranet, environments for mobile computation, collections of distributed sensor data for monitoring, etc. Distributed Data Mining offers better scalability, possibility of increase in data security and better response time when compared with a centralized model.

Techniques from Distributed Artificial Intelligence [6] [7], related to Multi-Agent Systems (MAS), can be applied to Distributed Data Mining with the objective of reducing the necessary complexity of the training task while ensuring high quality results.

Multi-Agent Systems [8] [9] are ideal for the representation of problems which include several problem solving methods, multiple points of view and multiple entities. In such domains, Multi-Agent systems offer the advantages of concurrent and distributed problem solving, along with the advantages of sophisticated schemes of interaction. Examples of interaction include cooperative work towards the achievement of a shared goal.

This paper discuss about the use of Multi-Agent Systems [10] [11] in Distributed Data Mining technique taking as example a previous work developed for knowledge integration [26]. Model integration consists in the amalgamation of local models into a global, consistent one. Agents perform learning tasks on subsets of data and, afterwards, results are combined into a unique model. In order to achieve that, agents cooperate so that the process of knowledge discovery can be accelerated. The proposed model previously aims to select the best rules for integration into the global model without, however, causing a decrease in its accuracy rate.

This paper is organized as follows: section 2 discusses the relationship between Distributed Data Mining and Distributed Artificial Intelligence. The knowledge integration protocol approach is discussed in section 3 and its results are displayed and discussed in section 4. In section 5 we include some related work and conclusions are exposed in section 6.

2 Distributed Data Mining and Distributed Artificial Intelligence

Distributed Data Mining (DDM) resulted from the evolution of Data Mining as it incorporated concepts from the field of Distributed Systems. DDM deals with data which is remotely distributed in different, interconnected locations.

According to Freitas [12], distributed mining is a 3-phase process:

- Split the data to be mined into p subsets, where p is the number of processors available, and send each subset to a distinct processor;
- Each processor must apply a mining algorithm to the local dataset. Processors may run the same mining algorithm or different ones;
- Merge the local knowledge discovered by each mining algorithm into a consistent, global knowledge.

DDM systems handle different components: mining algorithms, subsystems of communication, resource management, task planning, user interfaces and others. They provide efficient access to data and distributed computational resources, while they still permit monitoring the mining procedure and properly presenting results to users. A successful DDM system must be flexible enough to fit a variety of situations. It must also dynamically identify the best mining strategy according to the resources at hand and provide a straightforward manner of updating its components.

Client-server and agent-based architectures are examples of solutions which have been proposed in the literature. All agent-based DDM systems contain one or more agents in charge of each dataset. Such agents are responsible for analyzing local data and deliberately communicate with others during the mining stage, exchanging local knowledge until a global, coherent knowledge is reached. Due to the complexities involved in maintaining complete control over remote resources, many agent-based systems utilize a supervisor agent, called facilitator, which controls the behavior of local agents. Java Agents for Meta-learning (JAM) [13] and the BODHI system [14] follow this approach.

As previously mentioned, some of the techniques proposed in Distributed Artificial Intelligence, such as Multi-Agent systems, can be applied to Distributed Data Mining as to reduce the complexity needed for training while ensuring high quality results.

Distributed Artificial Intelligence (DAI) is the union of Artificial Intelligence (AI) and techniques from Distributed Systems. A definition for DAI which is more suitable to the concept of agency is given by Jennings [15], who declared that the object of investigation for DAI lies on knowledge models and techniques for communication and reasoning, necessary for computational agents to be able to integrate societies composed of computers and people. Jennings also splits DAI into two main research areas:

Distributed Problem Solving (DPS) – divides the solution of a particular problem amongst a number a modules which cooperate by sharing knowledge about the problem and the solutions involved.

Multi-Agent Systems (MAS) – studies the behavior of a set of (possibly pre-existing) autonomous agents whose shared goal is the solution of an arbitrary problem.

A Multi-Agent System, according to works by Jennings, Sycara and Wooldridge [16] and Wooldridge [17], is a computer program with problem solvers located in interactive environments, which are capable of flexible, autonomous, socially organized actions which can, although not necessarily, be directed towards predetermined goals or targets. Multi-Agent systems are ideal for the representation of problems which include several problem solving methods, multiple points of view

and multiple entities. In such domains, Multi-Agent Systems offer the advantages of concurrent and distributed problem solving, along with the advantages of sophisticated schemes of interaction. Examples of interaction include cooperation towards the achievement of a shared goal, coordination when organizing activities for problem solving, avoidance of harmful interactions and exploitation of beneficial opportunities, and negotiation of constraints in sub-problems, so that a satisfactory performance be achieved. On the flexibility of these social interactions lies the distinction between Multi-Agent systems and traditional programs, providing power and attractiveness to the paradigm of agents.

There are a number of application domains where Multi-Agent-based problem solving is appropriate, such as: manufacture, automated control, telecommunications, public transport, information management, e-commerce and games.

3 A Multi-agent System for Distributed Data Mining

This section discuss a Distributed Data Mining technique based on a Multi-Agent environment, called SMAMDD (Multi-Agent System for Distributed Data Mining) presented previously in [26]. In this work, a group of agents is responsible for applying a machine learning algorithm to subsets of data.

Basically, the process involves the following steps: (1) preparation of data, (2) generation of individual models, where each agent applies the same machine learning algorithm to different subsets of data for acquiring rules, (3) cooperation with the exchange of messages, and (4) construction of an integrated model, based on results obtained from the agents' cooperative work.

The agents cooperate by exchanging messages about the rules generated by each other, searching for the best rules for each subset of data. The assessment of rules in each agent is based on accuracy, coverage and intersection factors. The proposed model aims to select the best rules to integrate the global model, attempting to increase quality and coverage while reducing intersection of rules.

3.1 The SMAMDD Architecture

In the distributed learning system proposed, the agents use the same machine learning algorithm in all the subsets of data to be mined. Afterwards, the individual models are merged so that a global model is produced. In this approach, each agent is responsible for a subset of data whose size is reduced, focusing on improving the performance of the algorithm and considering also the physical distribution of data. Thus, each dataset is managed by an autonomous agent whose learning skills make it capable of generating a set of classification rules of type *if-then*. Each agent's competence is implemented with the environment WEKA (Waikato Environment for Knowledge Analysis), using the machine learning algorithm RIPPER [18]. The agents also make use of a validation dataset. Training and validation percentages can be parameterized and their default values are 90 and 10%, respectively. 0 presents the general architecture of the system.

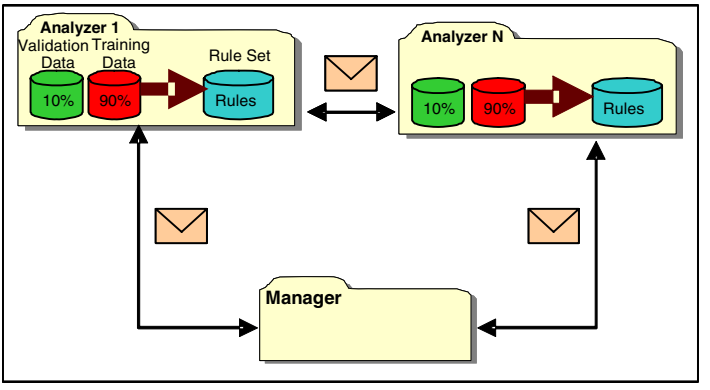

Fig. 1. System architecture: Analysers and Manager communicating. Communication can occur through the Manager or not.

In order to compare rules and rulesets, some measurements are made on the validation dataset. *Support*, *Error*, *Confidence* and *Quality* are specific measures to a rule depicted formally as the following way to a rule *R* predicting class *Class*:

- *Support*: number of examples covered by *R*;
- *Error*: number of examples covered by *R* but with class != *Class*;
- *Confidence = 1 – (Error / Support);*
- *Quality = Support * Confidence*;

Coverage and Intersection are specific measures to a ruleset depicted formally as the following way to a ruleset $RS = \{R_1^{E1}, R_2^{E2}, \ldots, R_r^{Er}\}$, where E_i is the set of examples covered by rule R_i:

- Coverage = |E1| + |E2| + … + |Er|;
- Intersection = |E' ∩ E1| + |E' ∩ E2| + ... + |E' ∩ E3|, to a Rule $R^{E'}$.

The two last measures are computed ignoring the *default rule*[1]. Otherwise the coverage of a ruleset RS' is always |DB| for any dataset DB.

Support and confidence factors are assigned to each rule, yielding the rule quality factor. In this work, the quality factor is being proposed as an evaluation metric for rules. Each *Analyzer Agent* will still maintain *intersection* and *coverage* factors, representing the intersection level among rules and the amount of examples covered by rules in the agent, respectively. At the end of the process, rules are maintained in the following format:

rule(Premises, Class,
[Support, Error, Confidence, Coverage, Intersection, Quality, self_rule]).

The term self_rule indicates whether the rule has been generated in the agent or incorporated from another, in which case it is named external_rule. There is no need

[1] A default rule has not conditions and covers all the examples not covered previously by the other rules.

Table 1. (a) Exchange of messages among the agents. (b) Description of Messages.

(a)

Message ID	Sender	Receiver
load	Analyzer	Self
manageEvaluation	Manager	Self
waitRulesToTest	$Analyzer_i$	$Analyzer_j$
testRules	Manager	Analyzer
evaluateRules	Manager	Analyzer
startEvaluation	Analyzer	Self
test	$Analyser_i$	$Analyser_j$
add	$Analyzer_i$	Self or $Analyzer_j$
remove	$Analyzer_i$	Self or $Analyzer_j$
evaluate	Analyser	Self
finishEvaluation	Analyser	Manager
calculateRightnessTax	Manager	Self
calculateRightnessTax	Manager	Analyser
getRules	Manager	$Analyser_i$
generateReport	Manager	Self

(b)

Message ID	Action
load	Generate rules
manageEvaluation	Start the process to coordinate cooperation
waitRulesToTest	Await delivery of rules
testRules	Start the process of analysis of external rules. For each rule, calculate the following factors: support, confidence, quality, intersection and coverage. Such factors allow the generation of a *StateValue* for each rule in the agent. Each *StateValue* is stored in *NewState* and sent to the source agent in a message of type *response_test[NewState:Rule])*. The procedure *testRules* continues until all rules in the agent are tested.
evaluateRules	Request that the process of analysis of self rules not yet analyzed be started
startEvaluation	Start the analysis of local rules
test	Calculate confidence, error, support, rule quality and intersection and coverage for the rule received.
add	Add a rule to the set of selected rules
remove	Delete a rule from the set of selected rules
evaluate	Calculate confidence, coverage and intersection for the set of rules
finishEvaluation	Return to the process manageEvaluation to verify whether there exists another rule in the agent yet to be tested or if another agent can start the evaluation of its rules.
calculateRightnessTax	Request accuracy rate for remaining local rules
getRules	Request the agent's rules with best accuracy rate
generateReport	Calculate final accuracy rates, quantity and complexity of rules, and present them to the user

to test rules with the parameter value external_rule. After rules are generated and the factors mentioned are found, the process of cooperation for discovery of best rules starts. This cooperative process occurs by means of exchange of messages and is hierarchically coordinated due to the existence of a manager agent.

Agents hold restricted previous knowledge about each other, namely agent's competences and information for communication. An interface for communication with users is also embedded in the Manager agent. In addition to that, it is responsible for coordinating the process of interaction among agents.

The interaction process arises from the need to test the rules generated by an arbitrary agent against the validation set of others, where rule quality on their datasets and intersection and coverage factors obtained with the insertion of rules in their datasets are verified. Such factors are used to quantify the degree of agent satisfaction, represented by the scalar state_value, obtained with Equation 1.

$$state\_value = ((quality \times 0{,}33) + (\mathrm{cov}erage \times 0{,}17) + (intersection \times 0{,}033)) \quad (1)$$

Based on weights assigned to each factor in the formula above, the algorithm searches for rules with high quality and coverage factors and whose intersection factor is minimal. Weights used in this work have been determined after experiments in an attempt to obtain the highest accuracy level. Future work may approach a deeper study on values to be defined for each factor.

Thus, one must find the agent which holds the highest degree of satisfaction (state_value) for a given rule. For any given rule, the agent whose state_value is the highest must incorporate that rule into its rules set; it must also inform the agent where it came from as well as the other agents which also hold it to exclude it from their rules set. After all rules in all agents have been analyzed, they must be tested against each agent's validation set (10%). The agent's rules whose accuracy against its validation set is the highest will integrate the global model. In addition to the rules obtained at the end of that process, the accuracy of rules against the test set, the number of rules generated and their average complexity are calculated.

3.2 Cooperation Model

As seen in the previous section, there are N analyzer agents and one manager agent in SMAMDD. Their main components are: an individual knowledge base and a communication module which permits the exchange of asynchronous messages with each other. The Manager agent possesses also a module which eases the coordination tasks amongst the analyzer agents.

The human operator uses a graphical interface to start the system and visualize intermediate results generated by agents, as shown in 0. 0 partially illustrates the exchange of messages among agents in the form of a UML 2.0 diagram. Both messages and respective actions executed by each agent along the process of interaction are detailed in Tables 1a and 1b. Whenever an analyzer agent receives a *start* message it generates its ruleset based on local data running a rule-based algorithm. Then the manager chooses one analyzer A_1 to start evaluating the local rules (*evaluateRules*) and warns any other analyzer on it (*waitRulesToTest*). Then, A_1 chooses a rule r_i and asks the other analyzers for evaluation of r_i (*test*). The agent answering with the higher *stateValue* receives a confirmation to keep the rule (*add*)

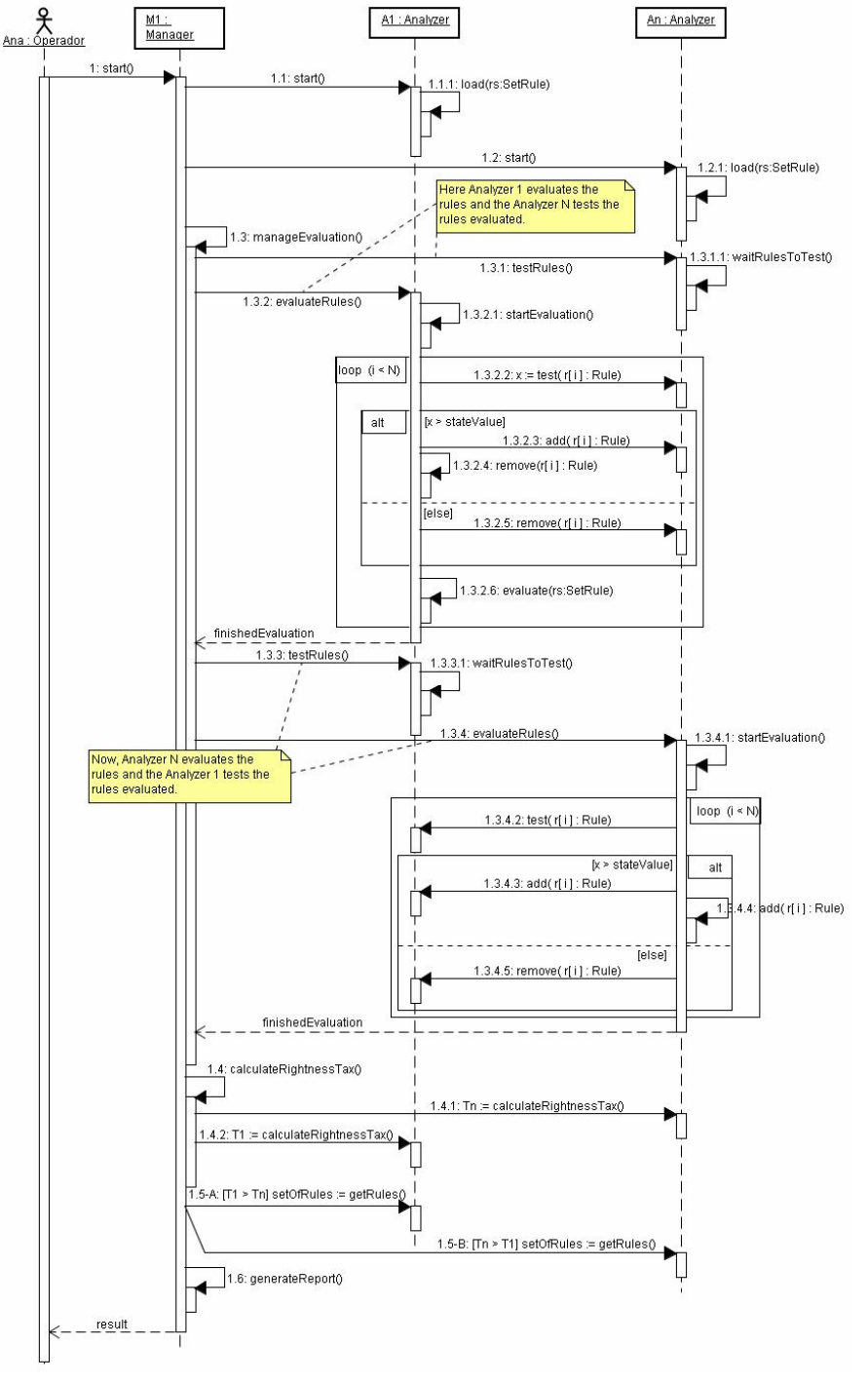

Fig. 2. Sequence Diagram for cooperation among agents

and the other agents must remove the rule from the local ruleset (*remove*). Agent A_1 continues the process sequentially with the next rules. When the evaluation of A_1 is over, it warns the manager (*finishedEvaluation*). Afterwards, manager starts a new evaluation process with A_2, A_3 and so on.

4 Results

In this section we complete the initial results presented in [26] with results regarding the complexity of the knowledge discovered with SMAMDD. With the intention of evaluating the work proposed, the following public datasets from the UCI Repository [1] have been utilized: Breast, Cancer, Vote, Zoo, Lymph, Soybean, Balance-scale, Audiology, Splice, Kr-ys-kp and Mushroom. Some combination techniques are not commonly used for learning in distributed data sets but are useful to improve the performance of the base classifiers on a centralized training data set. Such techniques generally use some heuristic for selecting instances or partitioning data. Since in distributed data mining we do not have control on the data organization and distribution, the performance of these methods are not foreseen. However, studies show that the combination of classifiers generated from several samples of a centralized data set can significantly improve the accuracy of the predictions [23] [24] [25]. We briefly discuss in the next paragraphs two well-known approaches (*bagging* and *boosting*) used in the experiments.

4.1 Bagging

Quite often, training sets are not good enough to describe a given problem. In this case, the available data represent a partial view of the entire problem. Consequently, a learning algorithm will not be able to generate a good classifier because it is based on a local model. Breiman [23] discusses a procedure able to improve the performance of any algorithm by combining several classifiers: *bagging*. Bagging solves the problem of local models by selecting equal sized samples of training subsets and generating classifiers for such subsets based on a learning algorithm. Each training subset is made up of instances selected randomly but *with replacement*. This means a given instance may repeatedly appear or not at all in any training subset. Finally, the classification of a test instance is given by a vote strategy [23]. According to Breiman, bagging works very well for unstable learning algorithms like decision trees or neural networks. In such methods small changes into the training set will produce very different classifiers. However, bagging is not so efficient for stable algorithms as nearest neighbor. Spite of this, Ting and Witten contested such ideas showing that Bagging can generate also good results for stable algorithms like Naïve Bayes in large databases [25].

4.2 Boosting

We have said in the former paragraph that bagging generally works for unstable learning algorithms. This happens because learning algorithms generate very different

models that are possibly complementary, even for similar inputs. Thus, the more different the data models, the larger the covered space of training instances is. However, with bagging, we do not guarantee complementary models because the instances are selected randomly. Boosting exploits this problem by providing a hill-climbing-like algorithm, guaranteeing the models to be as complementary as possible [25] [24]. There are many versions of boosting and here we discuss the general idea. To guide the generation of models, instances are initially weighed with an initial value. The weights are used for the classifier error estimation. The error is given by the sum of the weights of the misclassified instances divided by the sum of the weight of all instances. The weighing strategy forces the algorithm to pay more attention to misclassified instances (high weight). First, an equal error is assigned to all training instances. Then the learning algorithm generates the classifier and weights in all training instances are updated, increasing the weights of misclassified instances and decreasing the weights of well-classified instances. The global error of the classifier is also computed and stored. The process is repeated interactively until the generation of a small error. This procedure generates the classifiers and the weight of the training instances, where the weights represent the frequency the instances have been misclassified by the classifiers. To classify a new instance, the decision of a classifier is taken into account by assigning the weight of the classifier to the predicted class. Finally, the class with the largest weight (sum of the weights) is returned.

4.3 Empirical Results

Datasets were randomly split into *n* subsets each for cross-validation, yielding a total of 10 steps, iteratively, for each dataset. In SMAMDD, every partition is divided into training and validation, where the former constitutes 90 and the latter 10%. SMAMDD generates rules using only the Ripper algorithm. At the end of each step, accuracy rates, quantity of rules and complexity of rules are calculated. Table 2 presents some statistics about the datasets used in the experiments.

Table 3 presents the accuracy rates obtained by the SMAMDD and the techniques Bagging, Boosting and the RIPPER algorithm. The best results are displayed in bold.

Table 2. Statistics on datasets

Index	Dataset	Attributes	Classes	Examples
1	Breast C.	9	2	286
2	Vote	16	2	435
3	Zoo	17	7	101
4	Lymph	18	4	148
5	Soybean	35	19	683
6	Balance	4	3	625
7	Audiology	69	24	226
8	Splice	61	3	3190
9	Kr-vs-kp	36	2	3196
10	Mushroom	22	2	8124

Table 3. Average accuracy rates[2]

Datasets	SMAMDD	Ripper	Bagging	Boosting
Breast C.	**80,23 ±4,4**	71,74 ±3,8	71,39 ±3,3	72,55 ±4,8
Vote	**97,56 ±1,83**	94,88 ±2,50	95,41 ±1,12	93,99 ±1,39
Zôo	91,29 ±6,28	88,71 ±4,87	89,03 ±5,09	**94,84 ±4,36**
Lymph	**84,22 ±11,01**	74,67 ±5,36	80,44 ±7,16	83,78 ±2,78
Soybean	**95,36 ±2,14**	91,12 ±2,17	92,19 ±1,89	92,24 ±1,72
Balance	**85,37 ±4,53**	73,62 ±4,45	79,41 ±3,56	79,15 ±4,15
Audiology	**81,32 ±6,44**	73,97 ±5,53	73,23 ±7,03	75,59 ±7,54
Splice	**98,12 ±1,85**	93,31 ±0,74	95,08 ±0,66	94,67 ±0,23
Kr-vs-kp	97,04 ±2,51	98,99 ±0,32	99,21 ±0,24	**99,37 ±0,41**
Mushroom	**100,00 ±0,00**	99,93 ±0,16	100,00 ±0,00	99,93 ±0,16

Results from Table 3 are quite encouraging, as the SMAMDD system has achieved high accuracy rates in the majority of the datasets. The best results were obtained with the Balance-scale dataset, where the number of attributes is reduced.

By comparison with the boosting technique, the SMAMDD system experienced a reasonably lower accuracy performance in the Zoo dataset only. We believe that such performance loss was due to a reduced number of examples combined with a proportionally large number of attributes. Such characteristic often affects significantly the performance of non-stable algorithms such as RIPPER, producing quite different models for each subset. Consequently, concepts discovered locally rarely improve the satisfaction of neighbors and are doomed to remain limited to their original agents, having a negative impact on the agents' cooperative potential.

Even having obtained an accuracy performance higher than those of the other algorithms, the SMAMDD produced a high standard deviation in the dataset Lymph. Such distortion is due again to the large number of attributes in comparison with the number of examples.

It can be observed that some datasets presented a high standard deviation, revealing the instability of the algorithm upon the proportion of number of attributes to number of examples. With these results, it can be noticed that the SMAMDD system presents better accuracy performances when the number of attributes is small in comparison with the number of examples, i.e., the tuples space is densely populated (large volumes of data).

Table 4 compares the amount of rules generated by the SMAMDD system in comparison with the techniques bagging and boosting and the RIPPER algorithm.

From the statistics presented in Table 4, it can be noticed that the system SMAMDD has an inclination of producing a more complex model (high number of rules) when compared to the other techniques. It is also noticeable that the increase in the size of datasets, in general, causes a considerable increase in the number of rules along with higher standard deviations.

Table 5 presents the complexity of rules generated by the SMAMDD system, by the techniques bagging and boosting and by the RIPPER algorithm. The complexity

[2] Extracted from [**Error! Reference source not found.**].

of a rule is proportional to the number of conditions it has. Simple rules (few conditions) are preferable in relation to complex rules.

According to the statistics presented in Table 5, the complexity of the rules generated by the SMAMDD system is similar to those obtained with the other techniques. Even the largest datasets have not resulted in more complex models, demonstrating that number of attributes and volume of data have both been unable to significantly affect the complexity of rules.

Table 4. Average number of rules

Datasets	SMAMDD	Ripper	Bagging	Boosting
Breast C.	3,9 ±1,20	2,7 ±1,03	4,3 ±0,90	**2,3 ±1,40**
Vote	3,2 ±1,03	3,2 ±1,03	**2,7 ±0,48**	4,0 ±2,36
Zoo	**5,7 ±0,48**	**5,7 ±0,48**	6,4 ±0,84	6,2 ±1,40
Lymph	5,2 ±2,30	5,2 ±1,87	5,4 ±1,26	**5,0 ±2,11**
Soybean	49,3 ±18,86	25,1 ±1,29	26 ±2,54	**22,1 ±7,11**
Balance	17,2 ±7,61	11,6 ±3,24	12,4 ±3,75	**10,1 ±3,97**
Audiology	17,7 ±5,19	**13,3 ±1,64**	13,6 ±1,17	16,10 ±3,25
Splice	29,67 ±10,07	**13,67 ±5,51**	17,67 ±4,51	29,0 ±3,46
Kr-vs-kp	38,5 ±7,94	14,5 ±1,38	14,5 ±2,26	**10,0 ±4,82**
Mushroom	12,8 ±2,68	**8,6 ±0,55**	8,8 ±0,84	**8,6 ±0,55**

Table 5. Average complexity of rules

Datasets	SMAMDD	Ripper	Bagging	Boosting
Breast C.	1,84 ±0,14	**1,10 ±0,16**	1,49 ±0,12	1,41 ±0,16
Vote	1,31 ±0,60	1,31 ±0,60	**1,22 ±0,56**	1,36 ±0,65
Zoo	1,08 ±0,17	1,08 ±0,17	**1,00 ±0,16**	1,03 ±0,14
Lymph	1,24 ±0,24	**1,18 ±0,23**	1,44 ±0,23	1,27 ±0,55
Soybean	1,86 ±0,21	**1,59 ±0,06**	1,72 ±0,07	1,89 ±0,32
Balance	1,76 ±0,16	**1,71 ±0,20**	1,85 ±0,19	1,84 ±0,42
Audiology	1,61 ±0,11	**1,50 ±0,12**	1,57 ±0,14	1,66 ±0,21
Splice	3,95 ±0,16	3,52 ±0,44	**3,36 ±0,37**	3,75 ±0,31
Kr-vs-kp	3,39 ±0,24	**2,78 ±0,10**	2,89 ±0,24	3,53 ±1,86
Mushroom	1,51 ±0,11	1,37 ±0,10	**1,36 ±0,10**	1,41 ±0,11

5 Discussion and Related Work

From the collection of agent-based DDM systems which have been developed, BODHI, PADMA, JAM and Papyrus figure in the list of the most prominent and representative. BODHI [14], a Java implementation, was designed as a framework for collective DM tasks upon sites with heterogeneous data. Its mining process is distributed among local and mobile agents, the latter of which move along stations on demand. A central agent is responsible for both starting and coordinating data mining tasks. PADMA [20] deals with DDM problems where sites contain homogeneous data only. Partial cluster models

are generated locally by agents in distinct sites. All local models are amalgamated into a central one which runs a second level clustering algorithm to create a global model. JAM [13] is a Java-based Multi-Agent system which was designed to be used for meta-learning in DDM. Different classification algorithms such as RIPPER, CART, ID3, C4.5, Baves and WEPBLS can be applied on heterogeneous datasets by JAM agents, which either reside in a site or are imported from others. Agents build up classification models using different techniques. Models are properly combined to classify new data. Papyrus [21] is a Java-based system for DDM over clusters from sites with heterogeneous data and meta-clusters.

In [22], an agent-based DDM system where agents possess individual models, which are built up into a global one by means of cooperative negotiation, is proposed. Only rules with confidence greater than a predetermined threshold are selected along the generation stage; the negotiation process starts just afterwards.

The SMAMDD system proposed has been designed to perform classification. An important characteristic of the system is the degree of independence, for it does not require configuration parameters and thresholds as do most of the ones aforementioned. Moreover, local models are completely analyzed, as opposed to being pruned as in [22]. Although it demands more processing time, the risk of removing a concept important for the global model is reduced.

6 Conclusion

Distributed Data Mining and Distributed Artificial Intelligence have many shared and complementary issues involving the generation of global points-of-view based on local ones. Coordination, collaboration, cooperation and specific evaluation measures are important challenges in both the areas. This paper has discussed some works that use such concepts in different ways for the accomplishment of different tasks. SMAMDD, for instance, performs model integration and yields results comparable to the ones obtained with other machine learning techniques; besides, it exhibited superior performance in some cases. The system permits discovering knowledge in subsets of data, located in a larger database. High quality accuracy rates were obtained in the tests presented, corroborating the proposal and demonstrating its efficiency.

Although promising, the use of Distributed Artificial Intelligence in Distributed Data Mining and vice-versa is a quite recent area. Particularly, some questions have yet to be better studied in future work: consolidation of the techniques developed with evaluations in databases with different characteristics and from different domains; the use of distinct classification algorithms in the generation of local models and studies concerning alternative knowledge quality evaluation metrics are yet to be done. An important issue consists in developing new strategies which permit reducing the number of interactions. Techniques of *game theory* and *coalition formation* can be useful for the detection of groups and hierarchies between learning agents. Such information can often improve the coordination, reducing the number of messages, the amount of processing and the general quality of the interaction.

References

1. Blake, C., Merz, C.J.: UCI Repository of machine learning databases - Irvine. CA: University of California. Department of Information and Computer Science (1998), http://www.ics.uci.edu/ mlearn/MLRepository.html.
2. Hand, D., Mannila, H., Smyth, P.: Principals of Data Mining. MIT Press, Cambridge, Mass (2001)
3. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufman Publishers, San Francisco, CA (2001)
4. Kargupta, H., Sivakumar, K.: Existential pleasures of distributed data minig. In: Kargupta, H., Joshi, A., Sivakumar, K., e Yesha, Y.: (eds.) Data Mining: Next Generation Challenges and Future Directions, MIT/ AAAI Press (2004)
5. Park, B., Kargupta, H.: Distributed Data Mining: Algorithms, Systems, and Applications. In: Ye, N. (ed.) The Handbook of Data Mining, pp. 341–358. Lawrence Erlbaum Associates, Mahwah (2003)
6. Wittig, T.: ARCHON: On Architecture for Multi-Agent Systems. Ellis Horwood (1992)
7. Sridharam, N.S.: Workshop on Distribuited AI (Report) AI Magazine, 8 (3) (1987)
8. Jennings, N., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems 1, 7–38 (1998)
9. Wooldridge, M.J.: Reasoning about Rational Agents. MIT Press, Cambridge (2000)
10. Schroeder, L.F., Bazzan, A.L.C.: A multi-agent system to facilitate knowledge discovery: an application to bioinformatics. In: Proc. of the Workshop on Bioinformatics and Multi-Agent Systems, Bologna, Italy, pp. 44–50 (2002)
11. Viktor, H., Arndt, H.: Combining data mining and human expertise for making decisions, sense and policies. J. of Systems and Information Technology 4(2), 33–56 (2000)
12. Freitas, A., Lavington, S.H.: Mining very largedatabases with parallel processing. Kluwer Academic Publishers, The Netherlands (1998)
13. Stolfo, S., et al.: Jam: Java agents for meta-learning over distributed databases. In: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pp. 74–81. AAAI Press, Menlo Park, CA (1997)
14. Kargupta, H., et al.: Collective data mining: A new perspective towards distributed data mining. In: Advances in Distributed and Parallel Knowledge Discovery, pp. 133–184. AAAI/MIT Press, Cambridge, MA (2000)
15. Jennings, N.R.: Coordination tecniques for distributed artificial intelligence. In: O'hare, G.M.P., Jennings, N.R (eds.) Foundations of distributed artificial intelligence, pp. 187–210. John Wiley & Sons, New York (1996)
16. Jennings, N., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems 1, 7–38 (1998)
17. Wooldridge, M.J.: Reasoning about Rational Agents. MIT Press, Cambridge (2000)
18. Cohen, W.W.: Fast effective rule induction. In: Proc. of the Twelfth Intl. Conf. on Machine Learning, pp. 115–123 (1995)
19. Schapire, R.E., Freund, Y.: The Boosting Approach to Machine Learning: An Overview. In: MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA (2001)
20. Kargupta, H., et al.: Scalable, distributed data mining using an agent-based architecture. In: Heckerman, D., Mannila, H., Pregibon, D., Uthurusamy, R. (eds.) Proc 3rd International Conference on Knowledge Discovery and Data Mining, AAAI Press, Newport Beach, California, USA (1997)
21. Bailey, S., et al.: Papyrus: a system for data mining over local and wide area clusters and super-clusters. In: Proc. Conference on Supercomputing. ACM Press, New York (1999)

22. Santos, C., Bazzan, A.: Integrating Knowledge through cooperative negotiation - A case study in bioinformatics. In: Gorodetsky, V., Liu, J., Skormin, V.A. (eds.) AIS-ADM 2005. LNCS (LNAI), vol. 3505, Springer, Heidelberg (2005)
23. Breiman, L.: Bagging Predictors. Machine Learning 24(2), 123–140 (1996)
24. Shapire, R.E.: The Boosting Approach to Machine Learning: An Overview, MSRI Workshop on Nonlinear Estimation and Classification (2002)
25. Ting, K.M., Witten, I.H.: Stacking Bagged and Dagged Models. In: ICML, pp. 367–375 (1997)
26. de Paula, A.C.M.P., Scalabrin, E.E., Ávila, B.C., Enembreck, F.: Multiagent-based Model Integration. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2006, Hong Kong. International Workshop on Interaction between Agents and Data Mining (IADM-2006) (2006)