Multiagent-based Model Integration

Ana Carolina M. Pilatti de Paula, Bráulio C. Ávila, Edson Scalabrin, Fabrício Enembreck
Pontifical Catholic University of Paraná – PUCPR
Graduate Program in Applied Computer Science – PPGIA
Rua Imaculada Conceição, 1155, CEP 80.215-901 Curitiba PR – Brazil
{pilatti, avila, scalabrin, fabricio}@ppgia.pucpr.br

Abstract

This paper presents a Distributed Data Mining technique based on a multiagent environment, called SMAMDD (MultiAgent System for Distributed Data Mining), which uses model integration. As databases grow and distributed ones are required, new techniques are developed with the aim of accelerating the process of knowledge extraction. Model Integration consists in the amalgamation of local models into a global, consistent one. In each subset, agents perform mining tasks locally and, afterwards, results are merged into a global model. In order to achieve that, agents cooperate by exchanging messages, aiming to accelerate the process of knowledge discovery. 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. The agents responsible for analyzing local data exchange messages concerning their own local viewpoints until they reach a coherent, global knowledge. The multiagent system for Distributed Data Mining proposed in this paper has been compared with classical machine learning algorithms which are based on model integration as well, simulating a distributed environment. The results obtained show that SMAMDD can produce highly accurate data models.

1. Introduction

Recently, technological advances have enabled the extraction of knowledge from stored data. The increase in the use of bar code on goods, online business transactions and the advent of Internet generate large quantities of data every day. Such growth has stimulated the development of new techniques and tools to deal with the intelligent and automatic transformation of processed data into useful information and knowledge. Consequently, Data Mining has become an extremely important research area.

Data Mining [1] [2] 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.

The expansion of databases and the existence of distributed datasets have intensified the search for techniques which could accelerate the mining process in large databases. Distributed Data Mining (DDM) [3] [4] [11] is the subfield of Data Mining which focuses on analyzing data remotely distributed in different, interconnected locations. 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. DDM offers better scalability, possibility of increase in data security and better response time when compared with a centralized model.

Techniques from Distributed Artificial Intelligence [5] [6], related to multiagent systems (MAS), can be applied to DDM with the objective of reducing the necessary complexity of the training task while ensuring high quality results. MAS [7] [8] are ideal for the representation of problems which include several problem solving methods, multiple points of view and multiple entities. In such domains, MAS 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 presents a DDM technique based on a multiagent environment [9] [10] which uses model integration. 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 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: Our approach is presented in section 2 and its results are displayed and discussed in section 3. In section 4 we include some related work and conclusions are exposed in section 5.

2. A Multiagent System for DDM

This paper presents a DDM technique based on a multiagent environment, called SMAMDD (Multiagent System for DDM), which uses model integration. 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.

2.1 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), from the machine learning algorithm RIPPER [14]. The system also makes use of a validation dataset. Training and validation percentages can be parameterized and their default values are 90 and 10%, respectively.

Support (number of examples covered) and precision (number of examples correctly covered) factors are assigned to each rule, yielding the rule quality factor (quality = support x precision). 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, Precision, 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 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.

$$state\_value = \frac{((quality \times 1{,}0) + (coverage \times 0{,}5) + (intersection \times 0{,}1))}{3}$$

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 dataset. An 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. All results obtained are compared through the use of bagging [15] and boosting [16] techniques, along with the algorithm RIPPER [14].

Table 1. Messages and actions of each agent.

Message ID	Sender	Receiver	Action
manageEvaluation	Manager	Self	Start the process to coordinate cooperation
waitRulesToTest	Analyzer$_i$	Analyzer$_j$	Await delivery of rules
testRules	Manager	Analyzer	Start the process of analysis of external rules. For each rule, calculate the following factors: support, precision, quality, intersection and coverage. Such factors allow the generation of a S*tateValue* for each rule in the agent. Each S*tateValue* 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	Manager	Analyzer	Request that the process of analysis of self rules not yet analyzed be started
test	Analyser$_i$	Analyser$_j$	Calculate precision, error, support, rule quality and intersection and coverage for the rule received.
add	Analyzer$_i$	Self or Analyzer$_j$	Add a rule to the set of selected rules
remove	Analyzer$_i$	Self or Analyzer$_j$	Delete a rule from the set of selected rules
evaluate	Analyser	Self	Calculate precision, coverage and intersection for the set of rules
finishedEvaluation	Analyser	Manager	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	Manager	Self	Request accuracy rate for remaining local rules
calculateRightnessTax	Manager	Analyser	Request accuracy rate for remaining local rules
getRules	Manager	Analyser$_i$	Request the agent's rules with best accuracy rate
generateReport	Manager	Self	Calculate final accuracy rates, quantity and complexity of rules, and present them to the user

2.2 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 Figure 1. Figure 1 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 Table 1.

3. Results

With the intention of evaluating the work proposed, the following public datasets from the UCI Repository [24] have been utilized: Breast Cancer, Vote, Zoo, Lymph, Soybean, Balance-scale, Audiology, Splice, Kr-ys-kp and Mushroom. The results obtained with the SMAMDD system were compared to the results obtained with the techniques bagging and boosting and the RIPPER algorithm [14].

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%. 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 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.

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

Table 3 presents the accuracy rates obtained by the SMAMDD system with the techniques bagging and boosting and the RIPPER algorithm. The best results are displayed in bold. 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.

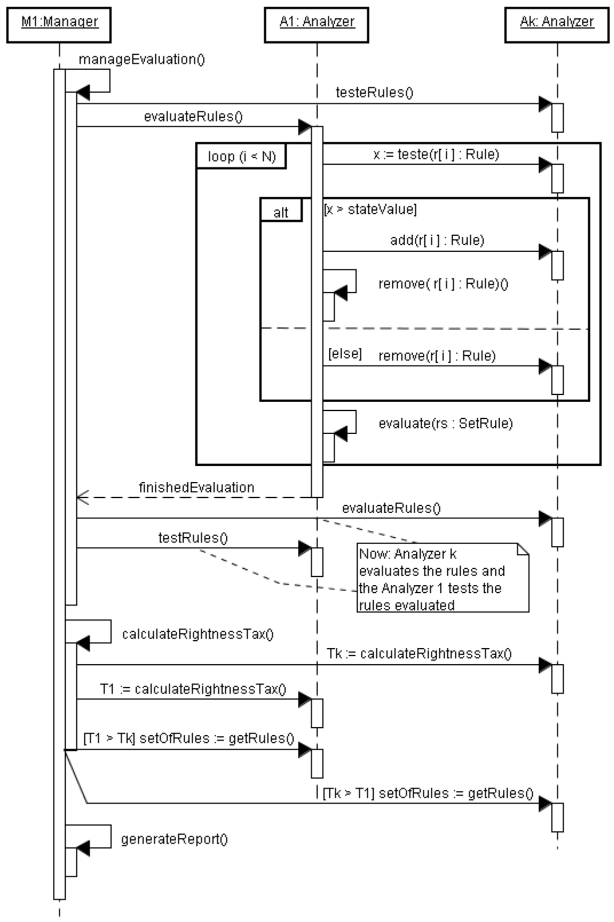

Figure 1: Diagram for cooperation among agents

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.

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**

From the statistics presented in Table 4, it can be noticed that the system SMAMDD has an inclination to producing a more complex model in relation to the 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.

4. 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 [13], 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 [17] 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 [12] is a Java-based MAS 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 [18] is a Java-based system for DDM over clusters from sites with heterogeneous data and metaclusters. In [19], 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 precision 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 [19]. Although it demands more processing time, the risk of removing a concept important for the global model is reduced.

5. Conclusion

From the results above referenced, it can be concluded that the SMAMDD system, which performs model integration, yielded 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.

Some questions have yet to be better studied in future work: more tests need to be performed in databases with different characteristics and from different domains; distinct classification algorithms can be used in the generation of local models and studies concerning alternative rule evaluation metrics are yet to be done. An important issue consists in developing new strategies which permit reducing the number of rule evaluations. Currently, each rule is assessed by all agents so that the one which is mostly satisfied can be distinguished, resulting in high computational cost. The use of a criterion to choose only a few agents for evaluation would significantly reduce the processing time needed by the whole process.

References

[1] Hand D., Mannila H., Smyth P. *Principals of Data Mining*. MIT Press, Cambridge, Mass, 2001.

[2] Han J., Kamber, M. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, San Francisco, CA, 2001.

[3] Kargupta ,H. Sivakumar, K. Existential pleasures of distributed data minig. In *Data Mining: Next Generation Challenges and Future Directions*, MIT/ AAAI Press, 2004.

[4] Park, B., Kargupta, H., Distributed Data Mining: Algorithms, Systems, and Applications, In *The Handbook of Data Mining*, N. Ye (ed.), Lawrence Erlbaum Associates, pp: 341-358, 2003.

[5] Wittig, T. ARCHON : *On Architecture for Multi-Agent Systems*, Ellis Horwood, 1992.

[6] Sridharam, N.S. Workshop on Distribuited AI (Report) *AI Magazine*, 8 (3), 1987.

[7] Jennings, N., Sycara, K., Wooldridge, M. *A Roadmap of Agent Research and Development, Autonomous Agents and Multi-Agent Systems,* 1:7-38, 1998.

[8] Wooldridge, M. J. *Reasoning about Rational Agents*. MIT Press, 2000.

[9] L. F. Schroeder and A. L. C. Bazzan. A multi-agent system to facilitate knowledge discovery: an application to bioinformatics. In *Proc. of the Workshop on Bioinformatics and Multi-Agent Systems*, pages 44{50, Bologna, Italy, 2002.

[10] H. Viktor and H. Arndt. Combining data mining and human expertise for making decisions, sense and policies. *J. of Systems and Information Technology*, 4(2):33-56, 2000.

[11] Freitas, A.; Lavington, S. H. *Mining very largedatabases with parallel processing* Kluwer Academic Publishers The Netherlands, 1998.

[12] 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*, 1997, p. 74– 81. Menlo Park, CA: AAAI Press.

[13] Kargupta, H. et. al. Collective data mining: A new perspective towards distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery* p. 133–184. Cambridge, MA: AAAI/MIT Press, 2000.

[14] Cohen, W. W. Fast effective rule induction. In: *Proc. of the Twelfth Intl. Conf. on Machine Learning*, pp: 115-123. 1995.

[15] Breiman, L. . Bagging Predictors. *Machine Learning*, n. 2, vol. 24, p.p. 123-140, 1996.

[16] Schapire, R. E.; Freund, Y.. The Boosting Approach to Machine Learning: An Overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA. 2001

[17] Kargupta, H. et. al. Scalable, distributed data mining using an agent-based architecture. In D. Heckerman, H. Mannila, D. Pregibon, R. Uthurusamy, eds*: Proc 3rd International Conference on Knowledge Discovery and Data Mining*, Newport Beach, California, USA, AAAI Press, 1997.

[18] 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, 1999.

[19] Santos, C.; Bazzan, A. Integrating Knowledge through cooperative negotiation - A case study in bioinformatics, In.: Autonomous Intelligent Systems: Agents and Data Mining: International Workshop, AIS-ADM 2005, St. Petersburg, Russia, June 6-8, 2005.