

An Agents and Artifacts Approach to Distributed Data Mining

Xavier Limón¹, Alejandro Guerra-Hernández¹, Nicandro Cruz-Ramírez¹,
and Francisco Grimaldo²

¹ Universidad Veracruzana, Departamento de Inteligencia Artificial, Sebastián
Camacho No 5, Xalapa, Ver., México 91000
xavier120@hotmail.com, {aguerra,ncruz}@uv.mx

² Universitat de València, Departament d'Informàtica, Avigunda de la Universitat,
s/n, Burjassot-València, España 46100
francisco.grimaldo@uv.es

Abstract. This paper proposes a novel Distributed Data Mining (DDM) approach based on the Agents and Artifacts paradigm, as implemented in CArtAgO [9], where artifacts encapsulate data mining tools, inherited from Weka, that agents can use while engaged in collaborative, distributed learning processes. Target hypothesis are currently constrained to decision trees built with J48, but the approach is flexible enough to allow different kinds of learning models. The twofold contribution of this work includes: i) JaCA-DDM: an extensible tool implemented in the agent oriented programming language Jason [2] and CArtAgO [10,9] to experiment DDM agent-based approaches on different, well known training sets. And ii) A collaborative protocol where an agent builds an initial decision tree, and then enhances this initial hypothesis using instances from other agents that are not covered yet (counter examples); reducing in this way the number of instances communicated, while preserving accuracy when compared to full centralized approaches.

Keywords: Multi-Agent System, Distributed Data Mining, CArtAgO, Jason, Collaborative Learning.

1 Introduction

As the amount of data produced by the everyday systems grows and distribute, the problems faced by Data Mining also grows. Being this the case, Data Mining as a research field needs to keep the pace. Distributed Data Mining (DDM) addresses the problem of mining huge amounts of distributed, even geographically, data. From the point of view of software engineering, DDM systems need to exhibit various desirable characteristics, such as scalability, configuration flexibility and reusability [7]. Multi-Agent Systems (MAS) are inherently decentralized and also distributed systems, being a good option to implement DDM systems that cope with the requirements. Nowadays, agent-based DDM is growing in popularity [14].

In this work we present JaCA-DDM, a novel approach to DDM based on the Agents and Artifacts paradigm, as implemented in CArtAgO [9]. Agents in the system are implemented in the well known agent oriented programming language Jason [2]. CArtAgO artifacts play a big role in our approach, for obtaining a modular, scalable, distributed Java based architecture, easy to design, implement and maintain. We also present a distributed learning strategy that borrows ideas from collaborative concept learning in MAS [3]. This strategy is an incremental collaborative protocol that tries to enhance a model created with few instances, by means of contradictory instances provided by the agents on the system. In this way, it is possible to reduce the number of instances communicated, while at the same time maintaining the accuracy of a centralized approach.

This paper presents a work in progress aimed to develop an agents & artifacts competitive approach for DDM. To this end, JaCA-DDM is used to create an experimental setting aimed to test the differences in accuracy and number of training examples used between our strategy and a traditional centralized strategy. Being this comparison our main concern, we put aside many efficiency aspects for the moment, but the main strategy and system architecture are open enough to allow further efficiency enhancements.

This paper is organized as follows. Section 2 introduces the background for this paper, this includes: DDM, agent based DDM, and CArtAgO environments. Section 3 introduces JaCA-DDM and describes the generalities of our leaning strategy. In section 4 the experimental setting and results obtained are addressed. Finally, section 5 closes with a conclusion and future work.

2 Background

Data mining is a discipline that merges a wide variety of techniques intended for the exploration and analysis of huge amounts of data, with the goal of finding patterns and rules somewhat hidden in it [13]. Since data mining is about data, it is important to know the origin and distribution of this data in order to exploit it efficiently. A traditional way of doing data mining is using a centralized schema. In this way, all the data and learned models are on the same site. With the ubiquitous presence of computational networks, it is common to find data belonging to the same system spreaded in various sites, even in sites that are geographically far away from each other. From the data mining point of view, some questions may arise in these distributed scenarios: Which is the best strategy for constructing learning models that take into account the data from all the sites?, What is the best way to face heterogeneous databases?, How can the communication of the data and the data mining process be optimized?, How can the privacy of the data be preserved?, Is there some efficient way to treat cases where the data changes and grows constantly?

A lot of systems devoted to DDM have been created. According to the strategy that they implement, those systems can be classified into two major categories [7]: centralized learning strategies, and meta-learning strategies. In the centralized strategies, all the data is moved to a central site, and when all data is

merged together, data mining algorithms are applied sequentially to produce a single learning model. In general, the centralized strategy is inapplicable because of the cost of data transmission. Meta-Learning refers to a general strategy that seeks to learn how to combine a number of separate learning processes in an intelligent way [4]. The idea behind Meta-Learning is to collect locally generated classification models and from there generate or simulate a single learning model. To accomplish this, it is necessary to collect the prediction of the local classification models on a validation data set, and then create a meta-level training data set from the predictions of the locally generated classification models. To generate the final meta-level classification model from the meta-level training data set, voting, arbitrating and combining techniques can be used [6]. Meta-learning is an efficient and scalable way to do DDM since the data transmitted is limited and the process is parallel, with a good load balance. Nevertheless, it is not as efficient as its centralized counterpart when a new instance needs to be classified. This is because the classification process is not direct, the classification query has to traverse a decision process that maybe has various classification models involved. Centralized learning is also simpler, to setup a meta-learning system can be more difficult. Another disadvantage of distributed meta-learning is that, because classifiers are not built globally on data, the model's performance may suffer as a result of incomplete information [11].

We propose an alternative learning strategy borrowed from SMILE [3], a setting for collaborative concept learning in MAS, designed for maintaining the consistency of the learned hypothesis when facing new evidence. A concept has to be consistent with respect to a set of examples that can be received from the environment or from other agents. The hypothesis is kept consistent through a series of incremental revisions. In this way, the hypothesis is incrementally enhanced through a process that involves the use of counter examples (examples not covered by the current hypothesis). We took this idea and translate it to DDM terms.

Data mining is applied to a variety of domains that have their own particularities, making it difficult to come with a general way of treating all scenarios. MAS are a straight and flexible way to implement DDM systems since they are already decentralized distributed systems. Each agent can do various tasks concurrently and it can be seen as an independent process. The location of the agents is in some degree irrelevant and transparent. The communication between agents is done in a high abstract level, that makes it easy to implement sophisticated protocols and behaviors. Some agent-based DDM systems [14] had been done in the past with successful results, e.g., JAM [12], a Meta-learning agent based DDM system, is one of the most influential. Our approach is closer to centralized distributed DDM systems. The challenges of agent based DDM are discussed in detail in [8].

A fundamental part of a MAS is the environment where it is deployed. It is important to adequately model the environment such that the agents can be able to perceive it, modify it, and exploit it. CArTAgO is an infrastructure and architecture based on artifacts used for modeling computational environments in

multi-agent systems. With CArtAgO the concept of environment is simplified, the environment is a set of artifacts.

An artifact is a first order abstraction used for modeling computational environments in MAS. Each artifact represents an entity of the environment, offering services and data structures to the agents in order for them to improve their activities, especially the social ones. Artifacts are also of great value in the design and reutilization of multi-agent systems since their structure is modular, based on object-oriented concepts. Artifacts are conceived as function-oriented computational devices, functionality that the agents may exploit in order to fulfill their individual and social activities. The vision proposed by CArtAgO impacts directly in the agent theories about interaction and activity. Under this vision a MAS is conceived as a set of agents that develop their activities in three distinct ways: computing, interacting with other agents and using shared artifacts that compose the computational environment.

Artifacts can be the objective of the agent activity as well as the tool the agents use to fulfill their activities, reducing the complexity of their tasks. Since the environment is composed by artifacts, the state of each artifact can be perceived by the interested agents. The infrastructure of CArtAgO was designed having in mind distributed environments. It is possible to define work-spaces to determine the context where an artifact exists and can be perceived and used. The distributed environment is transparent for the agent, the later is one of the most valuable characteristics of CArtAgO. In this work, CArtAgO plays an important role, and is one of the base technologies used to support JaCA-DDM.

3 JaCA-DDM: A Jason Multi-Agent System for DDM

JaCA-DDM (Jason & CArtAgO for DDM) is a Multi-Agent System implemented in Jason and situated in a CArtAgO environment. JaCA-DDM is used to create and run distributed learning experiments that are based on the collaborative learning strategy explained later. Currently it supports J48 decision trees, but it can easily be extended to support other classification learning approaches. The artifacts provided by this environment encapsulate data mining tools as implemented by WEKA[5]. In what follows, the artifacts, agents, and the workflow are described in detail.

The MAS is composed by a coordinator agent and n workers. There are three main types of artifacts used by the agents: `Oracle`, `InstancesBase` and `ClassifierJ48`. The coordinator uses the `Oracle` to extract information about the learning set and split it among the workers and itself. Each agent stores its training examples in an `InstancesBase` artifact. Instances distribution is shown in figure 2 (page 344). The coordinator induces an initial model with its instances using `ClassifierJ48`. Then it asks for contradictory instances as shown in figure 3 (page 345). The interactions amongst the artifacts are shown in figure 1. In what follows, a more detailed account for each artifact is presented.

Since the main goal of JaCA-DDM is to experiment distributed learning scenarios, we are interested in partitioning existing training data sets in a controlled

way to enable comparisons with centralized scenarios. The single **Oracle** artifact creates random stratified data partitions and distributes them among the agents. An agent can use the **Oracle** artifact to: obtain the attributes information, as described in the ARFF file; restart the artifact for a new running of the system; recreate the artifact to run a new round in the cross-validation process; get the number of instances stored in the artifact; and reinitialize the artifact with a new training set. The **port1** is used to get other artifacts linked with this one. Usually **InstancesBase** artifacts are linked via this port to get set of instances.

ClassifierJ48 is a single artifact in charge managing and creating the learning model. An agent can execute a set of operations (\circ) on an instance of this artifact to: add a new training instance to the artifact; build a J48 classifier with the instances stored in the artifact; print the tree representation of the computed classifier; classify an instance; and restart the artifact for running a new experiment. An agent can also link other artifacts to this one, so that the linked artifacts can execute linked operations (\diamond) on the **Classifier48** for: getting the J48 classifier; and classifying an instance. Observe that the artifact is used to classify instances in two ways: i) An agent executes **classifyInstance** over a string representing the instance to be classified, obtaining an integer representing the instance class as defined in WEKA; ii) Another artifact executes **classifyInstance** to classify an instance stored in that artifact. The **port1** is used to link other artifacts linked to this one. Usually **InstancesBase** artifacts are linked via this port to classify instances.

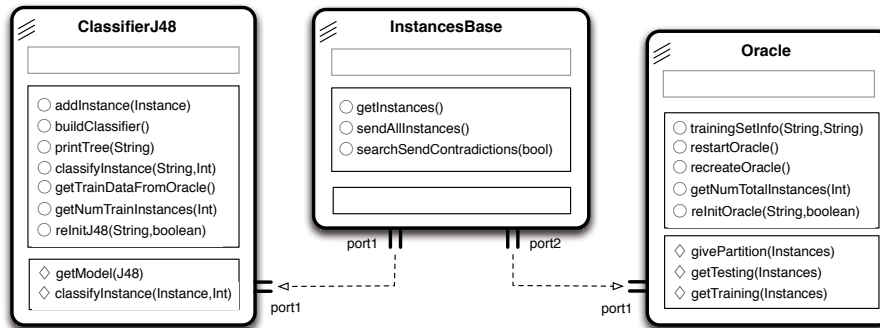


Fig. 1. The main artifacts used in JaCA-DDM

InstancesBase is an artifact class implementing local repositories of instances for the agents, so each agent has control of a **InstancesBase** artifact. Such an artifact can be linked with an **Oracle** artifact, via **port2**, in order to execute the linked operation **givePartition** to obtain a set of instances. It can be also linked to a **ClassifierJ48**, via **port1**, in order to search for a contradiction in the local repository and add it to the **ClassifierJ48**. A contradiction is an instance wrongly classified by the current model.

Other artifact related to the experimental setting provided by JaCA-DDM include: the **GUI** artifact is a front end for launching experiments and setting the different parameters for the experiment; the **Evaluator** artifact performs statistical operations with the results gathered, this operations include standard deviation, medias and paired T-test.

The collaborative learning strategy proposed has the following characteristics: there exists a central site, in this site the data is controlled by a special agent known as the coordinator. In this central site a base model is induced using the instances of the site, this base model serves as the first model that presumably needs enhancement since it maybe was induced with few instances. The base model can be shared between the different sites. The coordinator agent also is in charge of the experiment control, initialization of artifacts, control of the learning process, and managing the results. In each of the other sites, a worker agent resides, this agent manages the data of its corresponding site and runs a process with the purpose of finding contradictions between the base model and the instances of the site. A contradiction exists when the model does not predict the instance class correctly. The contradictory instances are sent to the central site enhancing the base model in a posterior induction. The process repeats itself until no contradictions are found.

To run the experiments we used a single computer to simulate different distributed scenarios consisting of various sites, the number of sites is configurable. Despite using a single computer for the experiments, the system architecture is flexible, it can also be applied in a true distributed environment without any major change.

Before an experiment begins, the parameters for the experiment are set through the GUI, those parameters include: database path, number of worker agents (in this manner simulating various sites) and type of model evaluation (hold-out or cross validation with its respective parameters). An experiment has the following general workflow: first, the coordinator determines which agents are going to participate in the experiment (currently all the agents participate). Then the coordinator creates the artifacts needed passing the relevant parameters. From there, each agent sends a request to **Oracle**, asking for its data partition. The coordinator sends all its examples to **ClassifierJ48** in order to create the base model. Next, the coordinator begins the social process, asking to each worker, one by one, if it has contradictory examples. If a worker finds a contradiction, the contradictory example is sent to **ClassifierJ48**. When a worker finishes sending all the contradictions, the coordinator may issue an induction request to **ClassifierJ48**, the frequency of this induction request can be tuned in order to increase efficiency. This process continues until no more contradictions are found.

The interaction diagrams in figures 2 (data distribution) and 3 (learning process) summarizes the workflow described earlier. These figures omit the **InstancesBase** artifacts for the sake of readability. Remembering that each agent has an **InstancesBase** associated for the storage and administration of its instances.

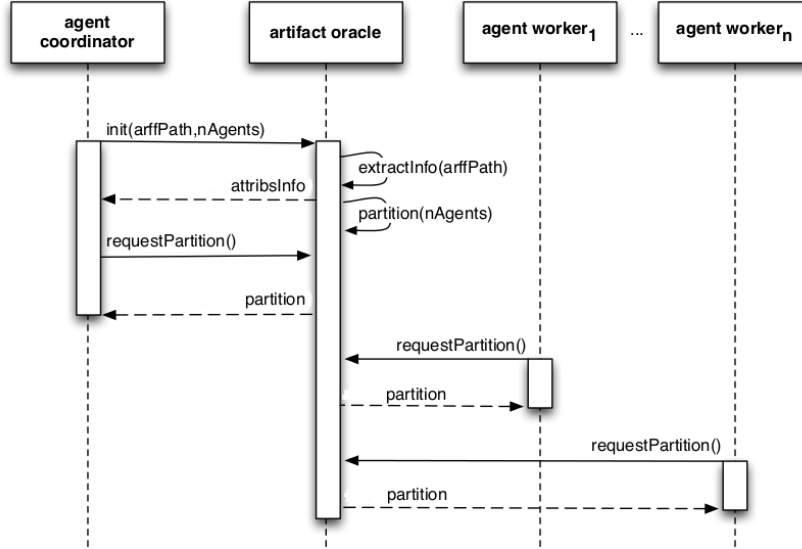


Fig. 2. Data distribution

Our current learning strategy is linear in the sense that only one worker agent at a time searches and sends contradictions. We are not exploiting yet the concurrent and parallel facilities that the architecture of JaCa-DDM provides.

4 Experiments and Results

JaCa-DDM was used to create a series of experiments to compare our collaborative learning strategy and a traditional centralized strategy. This comparison takes into account the number of examples used for training, classification accuracy and time. Since we ran the experiments in a single multi-core computer and not in a distributed system, the time results may not be fair because the cost of communication is not present, nevertheless, for the sake of completeness, we also show time results. A set of ten databases of the UCI repository [1] was used, giving consistent results. Table 1 lists the five most representative databases that are reported in the present.

A randomized stratified policy was used to distribute the data among the agents. Stratification ensures that each data partition conserves the ratio of class distribution. Stratified cross validation with 10 folds and 10 repetitions was applied. For each database, experiments were done with 1, 10, 30, 50 and 100 worker agents. For comparison, the same data partitions were used for both strategies. Two tailed paired T test with 0.05 degrees of significance are used to verify if there are significant differences between both strategies. The results are presented confronting the collaborative model against the centralized one (CollvsCen column in table 3) and the collaborative model against the base

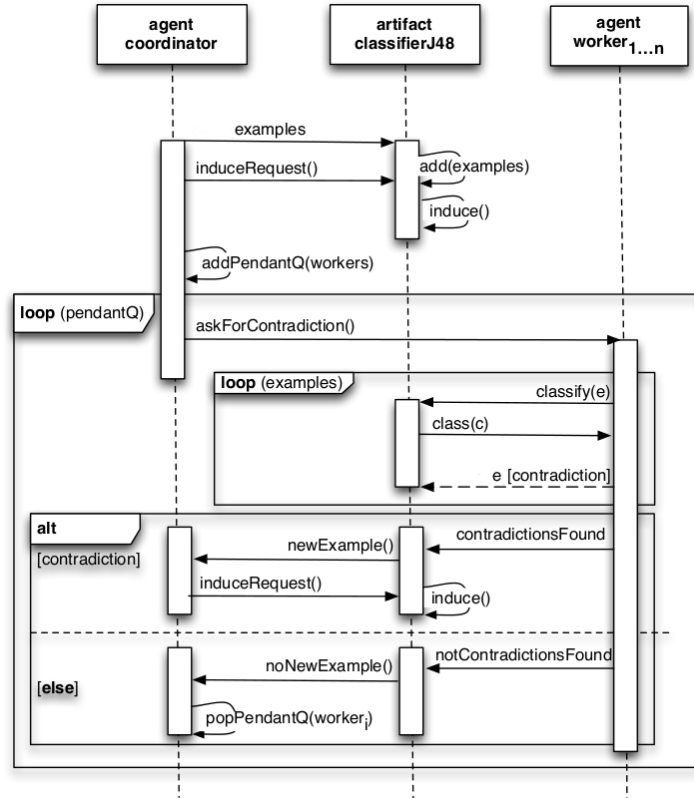


Fig. 3. Learning process

Table 1. Data Sets

Data Set	Instances	Attributes	Classes
adult	48842	15	2
german	1000	21	2
letter	20000	17	26
poker	829201	11	10
waveform	5000	41	3

model (CollvsBas column in the same table). 0 means there is not significant difference; 1 means that the first strategy paired won; and -1 means that the first strategy lost. J48 was used with pruning and the rest of the WEKA options set to default.

Table 2 shows the number of examples used to learn. For the collaborative model (Collab), the standard deviation is also shown, because of the variations in each experiment (100 runs). This table shows that our strategy definitely reduces the number of training examples used to induce the model. This can be seen for example in the results for the adult database (except for 1 worker agent) where

Table 2. Number of instances used to learn

Data Set	Wks	Total	Centralized	Base	Collab
adult	1	48842	43957	21978	27468.85 ± 107.53
adult	10	48842	43957	3996	16121.10 ± 147.73
adult	30	48842	43957	1417	15162.07 ± 142.62
adult	50	48842	43957	861	15403.52 ± 163.40
adult	100	48842	43957	435	16063.00 ± 221.61
german	1	1000	900	450	698.20 ± 15.68
german	10	1000	900	81	613.64 ± 13.94
german	30	1000	900	29	614.03 ± 16.20
german	50	1000	900	17	618.74 ± 15.94
german	100	1000	900	8	629.59 ± 16.10
letter	1	20000	18000	9000	11803.68 ± 164.30
letter	10	20000	18000	1636	8349.14 ± 217.90
letter	30	20000	18000	580	8259.17 ± 238.86
letter	50	20000	18000	352	8389.38 ± 227.43
letter	100	20000	18000	178	8628.10 ± 284.24
poker	1	829201	746280	373140	374100.00 ± 14.24
poker	10	829201	746280	67843	71419.50 ± 150.61
poker	30	829201	746280	24073	38988.50 ± 1750.09
poker	50	829201	746280	14632	48773.50 ± 994.89
poker	100	829201	746280	7388	81041.50 ± 1141.97
waveform	1	5000	4500	2250	3836.38 ± 33.40
waveform	10	5000	4500	409	3534.60 ± 34.54
waveform	30	5000	4500	145	3523.18 ± 34.12
waveform	50	5000	4500	88	3543.13 ± 34.50
waveform	100	5000	4500	44	3561.68 ± 37.20

only about 35% of the instances were used for training in our strategy. The standard deviation results show that our strategy is stable enough. Base reports the number of examples used to construct the initial model. Total reports the number of available examples.

Table 3 reports the accuracy of the obtained models. This table also shows, as mentioned, the results for paired T-test. Standard deviation is due to data random distribution for each experiment. Our approach (Collab) maintains a similar accuracy when compared with the centralized strategy. Even in the cases where our approach loses, the difference is very small, e.g., the poker database. There are significant differences in those cases because the standard deviation of the centralized strategy is small.

Finally, table 4 shows the mean time (ms) for inducing a model. From this results it is obvious that our strategy has its process overhead, this is more noticeable in small databases like german. Nevertheless, as the database grows, the advantages of our strategy begin to show up. This is specially true for the poker database, where the time efficiency actually improves. This boost in the time efficiency occurs because as the data grows it is more efficient to do multiple inductions with a small amount of data, rather than doing a single induction with

Table 3. Accuracy results

Data Set	Wks	Centralized	Base	Collab	CollvsCen	CollvsBas
adult	1	86.00 ± 0.44	85.78 ± 0.48	86.32 ± 0.45	1	1
adult	10	85.97 ± 0.44	84.75 ± 0.57	86.22 ± 0.56	1	1
adult	30	85.99 ± 0.43	83.84 ± 0.73	86.25 ± 0.57	1	1
adult	50	86.02 ± 0.44	83.54 ± 0.89	86.28 ± 0.51	1	1
adult	100	85.98 ± 0.43	82.20 ± 1.58	86.30 ± 0.52	1	1
german	1	72.05 ± 3.73	71.33 ± 4.05	71.82 ± 4.02	0	0
german	10	71.57 ± 3.74	68.38 ± 3.81	71.73 ± 3.78	0	1
german	30	71.83 ± 4.11	68.14 ± 3.89	71.18 ± 4.00	0	1
german	50	71.75 ± 4.0	66.56 ± 5.56	71.51 ± 3.96	0	1
german	100	72.50 ± 3.73	65.36 ± 7.94	71.79 ± 4.09	-1	1
letter	1	87.98 ± 0.76	83.74 ± 0.87	88.18 ± 0.74	1	1
letter	10	88.07 ± 0.70	69.28 ± 1.35	88.26 ± 0.70	1	1
letter	30	87.96 ± 0.80	57.86 ± 1.69	88.23 ± 0.84	1	1
letter	50	88.09 ± 0.73	51.26 ± 2.23	88.26 ± 0.80	1	1
letter	100	88.02 ± 0.67	40.35 ± 3.11	88.26 ± 0.76	1	1
poker	1	99.78 ± 0.01	99.76 ± 0.010	99.79 ± 0.01	0	0
poker	10	99.78 ± 0.01	99.06 ± 0.11	99.74 ± 0.01	-1	0
poker	30	99.79 ± 0.01	96.47 ± 0.25	99.76 ± 0.01	-1	0
poker	50	99.79 ± 0.01	92.22 ± 1.36	99.33 ± 0.02	-1	0
poker	100	99.79 ± 0.01	87.99 ± 0.40	98.99 ± 0.79	-1	1
waveform	1	75.35 ± 1.87	74.77 ± 2.03	75.24 ± 1.75	0	1
waveform	10	75.36 ± 1.99	70.89 ± 2.22	75.08 ± 1.88	0	1
waveform	30	75.35 ± 1.90	67.52 ± 3.03	74.69 ± 2.09	-1	1
waveform	50	75.05 ± 1.74	65.44 ± 3.26	74.85 ± 1.94	0	1
waveform	100	75.21 ± 1.99	62.76 ± 4.54	74.99 ± 2.04	0	1

a big amount of data. Since our strategy pretends to be applied in scenarios where the amount of data is really big, this result is very promising.

As we continue to develop our approach, a more in depth analysis about the results and consequences of our collaborative learning strategy will be done. For the present paper, we limited our analysis to the most noticeable facts as an evidence of the plausibility of the approach.

5 Conclusions and Future Work

In this paper we presented JaCa-DDM, an extensible tool that we proposed to run a series of experiments of DDM. The principles entailed by JaCa-DDM make it easy to extend and improve it. This is due to the modular nature of the system and the fact that agents and artifacts raise the level of abstraction, so we can think naturally in terms of shared services, communication and workflow.

As the results in the previous section show, our learning strategy is promissory. Our initial expectation of reducing the number of training instances used to train the model while conserving the accuracy of a traditional centralized strategy was fulfilled. Now we have the challenge to improve the learning strategy to

Table 4. Processing time in milliseconds

Data Set	Wks	Centralized	Collab	Data Set	Wks	Centralized	Collab
adult	1	1393.97	5913.78	letter	1	795.76	5435.10
adult	10	1419.80	14191.26	letter	10	816.49	17850.55
adult	30	1435.85	15167.84	letter	30	813.13	18120.53
adult	50	1441.34	12626.48	letter	50	826.68	14723.98
adult	100	1465.65	9720.67	letter	100	848.36	11643.36
german	1	10.14	68.16	poker	1	143236.00	180256.00
german	10	7.70	264.52	poker	10	147610.00	120582.00
german	30	6.70	385.76	poker	30	145595.00	53229.00
german	50	6.73	402.97	poker	50	148476.00	54364.50
german	100	6.89	546.88	poker	100	147646.00	54837.00
waveform	1	372.84	3330.27				
waveform	10	370.02	9056.13				
waveform	30	377.05	9371.32				
waveform	50	390.79	6669.84				
waveform	100	399.83	6933.90				

enhance efficiency as well as to do a more in depth analysis of the benefits and consequences of this approach. This analysis has to take into account more databases with a wide range of characteristics as well as more classification techniques, and not only J48. As it was mentioned earlier, we ran the experiments in a single computer, simulating various distributed sites. In the future, we hope to do experiments in a true distributed setting. In this way, we can have a better account of time results that will help us to move forward in the efficiency enhancements that we want to implement.

Acknowledgements. First author was supported by the Conacyt scholarship 320544. Second author was supported by the Conacyt project 78910.

References

1. Bache, K., Lichman, M.: UCI machine learning repository (2013)
2. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in Agent Speak using Jason, vol. 8. Wiley-Interscience (2007)
3. Bourgne, G., El Fallah Segrouchni, A., Soldano, H.: SMILE: Sound multi-agent incremental learning. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, p. 38. ACM (2007)
4. Chan, P.K., Stolfo, S.J.: On the accuracy of meta-learning for scalable data mining. *Journal of Intelligent Information Systems* 8(1), 5–28 (1997)
5. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter* 11(1), 10–18 (2009)
6. Prodromidis, A., Chan, P., Stolfo, S.: Meta-learning in distributed data mining systems: Issues and approaches. *Advances in Distributed and Parallel Knowledge Discovery* 3 (2000)

7. Rao, V.S.: Multi agent-based distributed data mining: An overview. *International Journal of Reviews in Computing* 3, 83–92 (2009)
8. Rao, V.S., Vidyavathi, S., Ramaswamy, G.: Distributed data mining and agent mining interaction and integration: A novel approach (2010)
9. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems* 23(2), 158–192 (2011)
10. Ricci, A., Viroli, M., Omicini, A.: Construenda est CArtAgO: Toward an infrastructure for artifacts in MAS. *Cybernetics and Systems* 2, 569–574 (2006)
11. Secretan, J.: An Architecture for High-Performance Privacy-Preserving and Distributed Data Mining. PhD thesis, University of Central Florida, Orlando, Florida (2009)
12. Stolfo, S., Prodromidis, A.L., Tselepis, S., Lee, W., Fan, D.W., Chan, P.K.: Jam: Java agents for meta-learning over distributed databases. In: *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pp. 74–81 (1997)
13. Witten, I.H., Frank, E.: *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann (2005)
14. Zeng, L., Li, L., Duan, L., Lu, K., Shi, Z., Wang, M., Wu, W., Luo, P.: Distributed data mining: a survey. *Information Technology and Management* 13(4), 403–409 (2012)