Supervised Learning Classifier Systems for Grid Data Mining

SANTOS, M.F ¹*, MATHEW, W ¹#, KOVACS, T ²+, SANTOS, H. ¹∞

¹ Department of Information System
University of Minho,
4800-058 Guimarães, Portugal.
* mfs@dsi.uminho.pt 
# wesley@dsi.uminho.pt, ∞ hsantos@dsi.uminho.pt

² Department of Computer Science
University of Bristol
Merchant Venturers Building, Bristol BS8 1UB
+ kovacs@cs.bris.ac.uk

Abstract: - This paper explores parallel and distributed implementation of the Learning Classifier System (LCS) technology. Specifically, the adaptation of supervised LCS to the grid data mining requisites, using the agent paradigm, is studied. The paper also examines the competitive data mining model induction possibility with homogeneous and heterogeneous data. A distributed framework is proposed using the grid computing architecture to make Supervised Classifier Systems (UCS) a more efficient distributed environment. This framework allows each local site (or agent) to employ a completely independent UCS. Local models (or UCS’s populations) are transmitted occasionally to the global model for combination. The global model then represents a complete knowledge base of the overall classification problem.

Key-Words: - LCS, UCS, Genetic Algorithm, Fitness, Accuracy, Data Mining, Grid computing, Cloud computing.

1. Introduction
With the rapid growth in computing power, machine learning and data mining have been receiving increasing attention from researchers as well as practitioners. Classification is a data mining technology that is of particular interest due to its wide range of applications including marketing, character recognition, robotics, resolution of control problems, medicine, social sciences, concept learning, optimization, artificial life and adaptive organisms behaviour modelling, and multi-agent system.[1] Classification can be defined as the process of assigning a class label to a given example, given a set of examples previously classified [2]. The approaches to classification include decision trees, instance-based learners, and neural networks.

This paper uses the Agent-Based Learning Classifier System for Grid Data Mining to induce global data mining models from distributed data. This approach will be compared, using benchmarking techniques, with other techniques such as, ensembles, parallel and distributed versions of other data mining algorithms (e.g., Artificial Neural Networks, Decision Trees). The proposed system is the consolidation of such knowledge, a computation medium that improves LCS’s performance and allows for the parallelization and distribution of classifiers in a modular and flexible ways, augmenting the scope of application of LCS.

Another objective of this paper is to implement a global model emerging from all local models in the grid structure. This system is built up from several distributed sites and a central global site. Each local site is responsible to extract learning models from local population. Data transmitted to the global model consists of the local model along with a set of data instances. A collection of local model at the server represents the knowledge of the whole distributed database.

The remainder of this paper is organized as follows. Section 2 describes some background of supervised learning classifier systems for grid data mining. This include learning classifier system, UCS, grid computing, cluster computing, data mining and distributed data mining. Section 3 gives details of proposed work. This section demonstrates working of the supervised learning classifier system for grid data mining. The section 4 includes basic level of discussion. Finally, section 5 summarizes the main conclusion for this work.
2. Background
The size of digital data has been increasing at a rapid rate in recent times, doubling every three years. These data are typically stored in geographically distributed repositories. Distributed data mining is a valuable technique to analyze such data. Classification is one of the technologies used for data mining. Common types of classifier systems include LCS, XCS, SB-XCS, ACS, and UCS. This paper addresses a supervised learning classifier system, which mainly includes LCS and UCS to perform distributed data mining classification tasks.

In distributed data mining, which implements data mining in a distributed fashion, the technologies available for distribution include distributed computing, cluster computing, grid computing and cloud computing. In this work, grid computing is employed, because of its usefulness in implementing data mining environments on grid platform by developing grid services. Since cloud computing is an important field in distributed environment, a brief description of cloud computing is also provided.

2.1 Data Mining
Data mining as a one step in the multi-step process of knowledge discovery in database (KDD). [1, 3] The KDD process starts with row data such as recorded financial or medical data and final produces some potentially useful pattern. The steps of KDD processes are selection, pre-processing, transformation, data mining, interpretation, evaluation [3]. During selection understanding the domain and goal of the end user and useful features extract from the row data. The pre-processing step cleans row data such as remove all noise and irrelevant data; fixing incorrect format of data, etc. The data transformation converts cleaned data to another form for better understanding of the relationship among the features. The data mining face search patterns of interest by using data mining algorithms. The interpretation and evaluation step interpret pattern from data mining to human understandable form.

Generally, data mining involves computer assisted analysis of large amount of data from different identities and converting it into useful information. The data mining process has many components such as machine learning, artificial intelligence, statistics, signal processing, mathematical optimization, and pattern recognition. Data mining software allows users to analyze data from many different angles or aspects to categorize data, and recapitulate the relationships identified. In principle, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

2.2 Distributed Data Mining
The main objective of distributed data mining is the parallel implementation of data mining in distributed sites to extract and combine useful knowledge [3]. There are two patterns in distributed data mining Distributed Mode (DMod) and Centralised Mode (CMod). In the DMod, data mining is applied to each distributed site and collects local models from each agent combining them in a central site where a final model is obtained. In the CMod, row data are collected from distributed locations and stored in a central location, and then data mining techniques are applied in the central location. The first pattern has the advantage that it needs to send only small amounts of data from local models thus reducing the bandwidth required. The second pattern needs to send row data that needs more bandwidth. Another advantage of the first pattern is that it is more secure than the second one.

As an example, consider a company that has different branches in different geographical locations and one central location, with each branch having a separate database. Figure 1 illustrates the distributed data mining process for this company. Data mining in the client extracts knowledge (local model) from that particular database and sent it to the central location. The central location combines all local knowledge from the results obtained in each local. The final model is derived from the combined model. Hence the central office of the company gets the over all knowledge.

A distributed environment involves large volumes of several sources of data. Analysing and monitoring these distributed data requires data mining techniques designed for distributed application. The quality of performance in data mining needs to be improved in distributed environments. The best example of a
distributed environment is the Internet, where growing more databases and data streams appear. Moreover internet consist of communication media for distributed information system, as for example the earth observing system of NASA [4].

2.3 Learning Classifier Systems

Learning Classifier Systems (LCSs) are a machine learning paradigm that can be used as a data mining technique [1, 5]. LCS uses the concept of evolutionary computation and reinforcement learning. The classifiers consist of a set of condition action rules, which is the knowledge of the LCS. The classifier evolves by learning from unknown environment.

LCSs were introduced by John H Holland in 1970 [4]. There was considerable research in the 1980s but in the early 1990s, LCSs introduced only a few successful applications [5]. In the mid 1990s, the field appeared almost at a dead end. But, during the last 10 years, new models have been developed and new applications have been presented which caused a great movement of this area.

The researches in the LCS area originated various types of LCS. We can differentiate two main approaches: the Michigan approach and the Pittsburg approach [4]. The main difference between these two approaches is that in the Michigan approach each individual is a single rule that represent a partial solution (the approach subscribed in this work), whereas in the Pittsburg approach, each individual is a set of rules that represents a complete solution to a learning problem [5]. The canonical LCS consists of four main components:

(i) The classifiers - which consists of condition action rules that represent the current knowledge of the system;

(ii) The performance component - which controls the interaction with the environment;

(iii) The reinforcement component (credit assignment component) - which receives reward from environment and delivers it to classifiers;

(iv) The genetic component - which is responsible for generating better rules and improving existing ones through a Genetic Algorithm (GA).

Classifiers have two associated measures: the strength and the fitness. Strength is the prediction of the classifier utility in terms of the amount of reward that the system will receive if the classifier is used. Fitness calculates the quality of the action about the problem that the classifier conveys, and it is performed by the discovery component for generating new population. The high fitness means that the classifier conveys more appropriate action about the problem and therefore it should generate more classifier through the genetic algorithm; a low fitness means that the classifier conveys little appropriate action about the problem and therefore should generate less classifier [5].

A LCS learns by interacting with an environment from which it receives reward in the form of numerical compensation. Learning is accomplished by trying to maximize the amount of reward received. On each discrete time step (system life cycle) the environment present a situation or problem in the form of a string or a binary code, the system receives as input the current situation of the environment and builds a match set \( [M] \) that consist of all the classifiers in the population \( [P] \) whose condition satisfies the condition parts of input situation. Then, the system evaluates the usefulness of the actions appearing in the match set; an action \( A \) is selected from those in the match set \( [M] \), and sent to the environment to be performed. Depending on the current situation and on the performance of the action \( A \) the system then receives a reward (or a punishment). The credit assignment component distributes the reward value among the rules accountable of the incoming rewards. This can be either implemented with an algorithm specifically designed for LCS (e.g., the bucket brigade algorithm).

The genetic component in the learning classification system randomly selects two classifiers (in mono-step systems; for multi-step systems this value doubles the number of steps) from the population with probability proportional to their fitness’s. By applying crossover and mutation, two new child classifiers are generated and subsumed by parent for check the generality of the new classifiers. If new classifiers are more general, then new classifiers are appended to the population, while two others are deleted in order to keep the population size constant.
The objective of the learning is to generate more accurate rules from previously unknown situations. The classification problem environment is a training set of pre-classified examples (situations). Each example is a vector of attributes and class labels.

2.4 Supervised Classifier System

Supervised Classifier System (UCS) is a learning classifier system derived from XCS, one of the most successful implementation of LCS [2]. UCS is specifically designed for supervised learning problems, while XCS follows a reinforcement learning scheme. The basic format of the supervised learning is based on the fact that after the learner chosen an action, the environment indicates what the correct action was [3]. This means that the error can be measured and used to optimize the model to be learned. UCS is then more suitable for data mining applications.

Moreover, UCS develops classifiers by learning. Population contains set of classifiers. Each classifier consists of rules and a set of parameters. A classifier can be defined as a tuple:

\[
< \text{condition}, \text{action}, \text{fitness}, \text{accuracy}, \text{numerosity}, \text{number of correct}, \text{number of match}, \text{correct set size}, \text{last time this was in the GA} >
\]

Fitness and accuracy are the quality of classifier, numerosity is the number of copies of classifier in the set, “number of match” is the number of occurrences of the classifier in the match set, “number of correct” is the number of times that the classifier is in the correct set, “correct set size” is the average size of all the correct sets where the classifier participate.

Fitness of the UCS is based on the accuracy of the classifier [2, 3]. We can group classifiers into two categories: correct and incorrect classifiers. The correct classifiers are those that get highest payoff if they are fired, and incorrect classifiers are the ones that get lowest payoff. The existences of classifiers in the population have more chance to correct one because incorrect classifiers will receive less fitness. Therefore, they can not enter into the evolutionary process and these classifiers have more chance for deletion than correct one [6].

The UCS has two mode of operation: testing and training. During training, inputs (examples) coming from the training set are provided to UCS. Then, UCS forms a match set containing the classifiers in the population whose conditions match the condition part of the input example. From the match set, the classifiers that predict class c form the correct set [2]. If the correct set is empty, the covering operator is executed. The covering is used for generating new classifiers. The new classifiers condition and class same as in the example which matches with population. Then, the parameters of classifiers are updated and eventually, the GA is applied. The genetic algorithm is invoked when the average experience of classifiers in correct set is higher than a GA threshold defined by the user. In test mode, prediction is carried in match set for getting predicted class. For the prediction, the sum of fitness in each classifier in the match set is used. The class with highest value is chosen as predicted class. The test mode genetic operation is disabled.

The parameters of classifiers are updated only in the training mode and only those belonging to the match set. The “number of match” (NM) updated corresponds to number of input examples correctly matched to the classifier condition part. The “number of correct” (NC) updated correspond to the number of the input class correctly matched to that classifiers’ class part.

The accuracy of a classifier is the ratio of number of correct by number of match:

\[
\text{Accuracy} = \frac{\text{NC}}{\text{NM}}
\] (1)

The fitness of a classifier depends on accuracy:

\[
\text{Fitness} = \text{Accuracy}^\nu
\] (2)

Where “\(\nu\)” is a pre-defined constant [2, 5]. A typical value is 20.

The genetic algorithm is used to generate classifiers in the population as the search mechanism. The GA is applied to the correct set [C]. The GA is triggered when the average of experience (last time this was in the GA) in the correct set is greater than a GA threshold defined by user. Genetic algorithm first selects two classifiers from the correct set with a probability proportional to fitness. These classifiers are considered as parent classifiers. The parent classifiers pass through crossover and mutation with probability “\(\lambda\)” and “\(\mu\)” respectively. The resulting offspring are checked for subsumption with their parents [2, 3]. If new offspring is more general and accurate than the parent then it’s added to population; otherwise the numerosity of parent is increased by one. The UCS population has a fixed size of N.
size of population is greater than N, then one classifier is deleted from the population.

2.5 Grid computing

Grid computing is a special kind of distributed computing system. In distributed computing, different computers within the same network share one or more resources; but in grid computing, every resource is shared [26]. Grid computing is a geographically-distributed computer network and heterogeneous computing, storage, and network resources, providing secure and pervasive access to their combined capabilities. Therefore, Grid platforms enable sharing, exchange, discovery, selection and aggregation of distributed heterogeneous resources such as computers, databases, visualization devices and scientific instruments.

Grid computing system connects all computer resources together in a way that gives the user advantages like a supercomputer [26] because user can access and leverage the collected power of all the computers in the system. Grid computing has the potential to support a wide range of applications that includes computer-intensive applications and applications requiring distributed services.

The problems found in the grid environment are not easily handled by executing a specific ad-hoc program. Instead, that needs several interacting software components, to run separately or concurrently over a given set of inputs. In this field, until now not much work has been done to build high-level design facilities for complex grid applications in which many programs and data sets are involved. These types of applications are most widely used in several domains, such as knowledge management, computational science, and e-business.

Data mining tools for use in grid environment is necessary due to databases that tend to have petabytes of stored data in geographically different places, which demands high processing power in a distributed way.

The increasing interest, application, and power demanding of data mining techniques in one side, and the grid technologies processing potential, it is very useful to develop data mining environments on grid platforms by developing grid services for the extraction of knowledge from distributed data repositories.

This paper proposes the development of an Agent-Based Learning Classifier System for Grid Data Mining.

2.6 Cloud computing

Cloud computing is an internet based development [28, 29]. For the computation over cloud, it combines different networks, software and services by the internet. The collection of networks and services are together called “the cloud” [29]. The cloud environment gives supercomputing access power to users. Using a thin client or other access point, like an iPhone, Blackberry or laptop, users can reach into the cloud for resources as they need them. For this reason, cloud computing has also been described as on demand computing.

Large processing ability is made possible though distributed cluster computing, often in concert with server virtualization software and parallel processing [30]. Cloud computing can process tens of trillions of computations in a second.

Cloud computing has many applications such as intelligence agencies, military, research labs, universities and major corporations to handle calculation and complex work like designing airplanes, and simulation of nuclear explosion. In addition, cloud computing can be used for sorting enormous amounts of data. For example: Google has an initial edge in cloud computing precisely because of its need to produce instant, accurate results for millions of incoming search inquires every day, parsing through the terabytes of Internet data cached on its servers [30].

3. Gridclass system

Gridclass is a project that aims to construct an UCS based grid/cloud environment to support distributed data mining tasks. At the first stage, only classification tasks are under consideration. In the next lines more details are presented.

The grid environment consists of physically distributed computer systems. The current work deals with a physically distributed agent environment and merging of all agent classifiers into a non distributed environment (global model). Both parts are responsible for building complete knowledge as a whole from local mining systems.

Figure 2 displays the architecture of a grid classifier system. The grid environment consists of multiple individual autonomous systems. The second environment (nondistributed) is responsible for combining all knowledge from the grid environment. This architecture corresponds to a many-to-one relationship in which all agents in the grid environment are required to communicate with the global model occasionally in a parallel fashion. For the development of the grid version of the supervised learning classifier system, it is expected a grid
infrastructure like the Globus toolkit middleware [27]. The toolkit of a given architecture gives input instance to the local UCS systems. This toolkit is connected to all local UCS in the parallel structure. Therefore, all local UCS get the same input instance at same time.

![Figure 3 – Grid class System.](image)

### 3.1 The Grid Environment

The grid environment contains distributed agents, corresponding to an agency. Agents might or might not communicate with each other for exchanging their knowledge. The communication between agents might help to speed up the learning at each agent [7].

Each agent is an independent component, which is able to operate on its own right. The parameter settings of each agent are, in principle, the same. A complete distribution is employed at each agent in order to obtain local patterns coming from local data sources. Data arrive at each agent in stream fashion from the toolkit, so its learning model updates the knowledge on the fly.

Each agent is responsible for updating its knowledge at the global model. Input instance of each agent must be the same but the output instance may be different. Outputs of each agent depend on the population of the individual. So, the size of each local model (knowledge) may vary. The communication frequency is decided by the user. In general, more frequent communication provides faster updates at the global model, but it results in more traffic load.

### 3.2 Global Model

Global model contains all the local models in its memory. Three local models are maintained at the global model, as shown in Figure 3. These models represent local knowledge of agents. From the manageability point of view, a company might want to build a single knowledge source rather than several independent sources of information.

Each local model consists of set of individuals. Each and every individual consists of an IP address of the local system and an input instance ID, classifier and their parameters. The memory of the global site is equally accessed by all the local agents and the global system. The agents are responsible for updating their models regularly. Suppose M different local agents in the distributed site provide M different set of models at the global model. The responsibility of the global model is to integrate these local models and choose a final model from the integrated one.

The global model has two main phases: the first one is the testing phase and the other is voting phase. The testing events directly contact with all local models. During the testing, it applies cut-off among the local models. It is mainly for keeping a minimum grade (standard) in all classifiers in the global model. It is assumed that the total classifier in the M models have N classifiers. The cut-off threshold (T) can be defined by the average fitness of local models:

$$T = \frac{\sum_{c=1}^{M} F_c}{N}$$

(3)

Here GM is the global model, c is the classifier in the global model and $F_c$ is the fitness of classifier.

If $F_c$ is greater than and equal to T then that represent 1 (classifier exists in the global model); otherwise 0 (classifier is removed from global model). This assists to improve the efficiency of the global model. In the first stage, the value of cut-off threshold is defined by the user. During the each iteration, the system can update its value depending of the average fitness of local models. After the testing phase subsets of local models are appended to the global model. Then, the final model is taken from this global model. In the current work the voting technique is employed to find the final model. By the voting approach it is possible to combine all local models to form a single and coherent model. The combination of local models is not a physical process but logical because we consider all local models are in a single structure by using input example ID. Any conflict in the output of local models is resolved through voting [3].

For the selection of final model, two patterns can be used. First pattern is the system prediction employed...
in separate models based on IP address and takes the separate class from each. Voting can be applied among these classes. In the above example there are 3 local models in the global model. All the individuals in the local model-1 have the same IP address. Select one class from local model-1 by system prediction using the same methodology. The separate class are formed for other two local models. From the three classes we can select one as final class. Alternatively, in the second pattern, a single model is formed from all models, based on input example ID and system prediction applied over these models. The main advantage of such arrangement is that if any system has failed during the iteration that does not affect overall system performance. In the current project, the second method is preferred because it is more efficient and it has short processing time. To calculate the System Prediction (SP) firstly the summation of fitness (F) and numerosity (Num) of each class should be calculated by:

\[ F_{\text{class}}(i) = \sum_{e \in [\text{NN}]} F_c \]  
\[ \text{Num}_{\text{class}}(i) = \sum_{e \in [\text{NN}]} \text{Num}_e \]  

Then find the ratio of “F” by “Num”:

\[ \text{SP}_{\text{class}}(i) = \frac{F_{\text{class}}(i)}{\text{Num}_{\text{class}}(i)} \]  

These values are taking for voting. The majority value of these votes i.e. Max (\( \text{SP}_{\text{class}}(i) \)) is considered as the final class.

3.3 Majority Voting

The majority voting is a simple but quite efficient approach for combining knowledge from multiple models. In this approach, there is no requirement of training and an acceptable performance is observed. [5] The final outcome of global model is the effect of all local models impending from the testing process in the global model. The global model applies voting among all predicted output classes. Let’s assume grid environment has a set \( L = \{l_1; l_2; \ldots; l_m\} \). Each testing instance gets new sets of local models. Such new set of local models consider a single model for the voting process. Voting applies to the output class of system prediction. The winning class is the one whose vote is majority among result set.

It is worth mentioning that the paper reports only a preliminary approach. More aspects will be included in the future work by considering other possible solutions.

4. Discussion

The option on how the data mining models should be induced in a distributed environment is not yet clear. Both approaches present advantages and limitations as resumed in the Table 1, where:

- **DMod** - Stands for the Distributed Model induction. Each site has a local model that contributes to the global model;
- **CMod** – Refers to the Centralized Model induction. The data is sent to a central repository and then used to induce the global model.

Table 1 – Advantages and limitations of the model induction approaches.

<table>
<thead>
<tr>
<th></th>
<th>Advantages</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMod</td>
<td>• Speed up on the global model induction</td>
<td>• The need of efficient merging methods.</td>
</tr>
<tr>
<td></td>
<td>• The co-existence of local and centralized</td>
<td>This is where the LCS technology can play an</td>
</tr>
<tr>
<td></td>
<td>models</td>
<td>important role</td>
</tr>
<tr>
<td></td>
<td>• Accuracy of the models</td>
<td></td>
</tr>
<tr>
<td>CMod</td>
<td>• The global model represents all the data</td>
<td>• Communication costs to transfer big quantities</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Computational costs to process a bigger</td>
</tr>
<tr>
<td></td>
<td></td>
<td>volume of data</td>
</tr>
</tbody>
</table>

Majority voting can be seen as a contribution in favour of the DMod approach allowing an efficient global model construction from local models during the training phase. The main objection against this approach is the parameter setting (e.g., the threshold \( T \)) which tuning can be difficult.

5. Conclusion and Future work

In this paper, an attempt has been made to study the applicability of supervised learning classifier system in grid data mining applications. The following conclusions can be derived:

- UCS was chosen as a base system for investigation because supervised learning is more suited for data mining applications;
A comprehensive analysis of UCS in grid computing for data mining indicate that it is potentially useful;

It is potentially very useful to implement a data mining environment on a grid platform by developing grid services for learning classifier systems.

Two methods have been identified to induce the global data mining model and a set of advantages and limitations for each one of them. A majority voting approach was proposed to incrementally induce a global model. A comparative study is ongoing in order to obtain definitive answers for the open questions.

Current system only considers homogeneous data. In the future work heterogeneous data by more dynamic learning behaviour integration will be included.

Acknowledgement
The authors would like to express their gratitudes to FCT (Foundation of Science and Technology, Portgal), for the financial support through the contract GRID/GRI/81736/2006.

Reference:


