Grid Data Mining by means of Learning Classifier Systems and Distributed Model Induction

Manuel Santos
Department of Information Systems
University of Minho
Guimarães, Portugal
+351253510306
mfs@dsi.uminho.pt

Wesley Mathew
Department of Information Systems
University of Minho
Guimarães, Portugal
+351253510264
wesley@dsi.uminho.pt

Henrique Santos
Department of Information Systems
University of Minho
Guimarães, Portugal
+351253510302
hsantos@dsi.uminho.pt

ABSTRACT

This paper introduces a distributed data mining approach suited to grid computing environments based on a supervised learning classifier system. Different methods of merging data mining models generated at different distributed sites are explored. Centralized Data Mining (CDM) is a conventional method of data mining in distributed data. In CDM, data that is stored in distributed locations have to be collected and stored in a central repository before executing the data mining algorithm. CDM method is reliable; however it is expensive (computational, communicational and implementation costs are high). Alternatively, Distributed Data Mining (DDM) approach is economical but it has limitations in combining local models. In DDM, the data mining algorithm has to be executed at each one of the sites to induce a local model. Those induced local models are collected and combined to form a global data mining model. In this work six different tactics are used for constructing the global model in DDM: Generalized Classifier Method (GCM); Specific Classifier Method (SCM); Weighed Classifier Method (WCM); Majority Voting Method (MVM); Model Sampling Method (MSM); and Centralized Training Method (CTM). Preliminary experimental tests were conducted with two synthetic data sets (eleven multiplexer and monks3) and a real world data set (intensive care medicine). The initial results demonstrate that the performance of DDM methods is competitive when compared with the CDM methods.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Induction, Knowledge acquisition

General Terms

Algorithms, Experimentation

Keywords

Grid Data Mining, Supervised Learning Classifier Systems, Model Merging Strategies.

1. INTRODUCTION

Recently, there is a significant progress in the research related to distribute data mining. Digital data stored in the distributed environments is doubling within a few years. For example, distributed organizations, e.g. large supermarkets, store chains, healthcare units that have different branches in all over the world should create and maintain individual data repositories. All the data that are stored in different databases are important for new decision making process. Supermarkets store large amounts of transactional data in their local databases. Local databases are important to induce local data mining models, but they are also important to induce global data mining models (e.g. predicting future sales volume).

More advanced and feasible distributed data mining algorithms and strategies are required in the current fast growing environment. The increasing necessity of distributed data mining applications is attracting more attention from researchers and software developers.

Learning Classifier System (LCS) is a concept formally introduced by John Holland as a genetic based machine learning algorithm [1]. Supervised Classifier System (UCS) is a LCS derived from XCS [1]. UCS adopted many features from the XCS that are suitable for supervised learning scheme. UCS algorithm was chosen to be applied in this implementation of grid based distributed data mining due to the supervised nature of the most part of the problems in this area. Substantial work has been done to parallelize and to distribute the LCS canonical model in order to improve the performance and to be suited to inherently distributed problems. Manuel Santos [2] developed the DICE system, a parallel and distributed architecture for LCS. In his work he attempted to parallelize the genetic algorithm and LCS message operations to increase system's performance. Interesting speedups were attained in the experimental work. A. Giani, Dorigo and Bersini also did significant research in the area of parallel LCS [3]. Their implementation also tried to increase the performance of the system. All implementations of parallel LCS consider a single data and generate a single model.

This work is part of a major project – the Gridclass project – whose main goal is to implement the UCS in a grid environment. Gridclass system does not paralyze any part of the UCS. Various instances of the UCS are executed in different distributed sites with different set of data. Recalling the supermarket application

example presented in the beginning of this introduction, each branch can generate local models running an UCS instance on the data available in that site. Using the conventional method of Centralized Data Mining (CDM), each distributed site has to send data to the central site. If on a daily basis each site $s \in \{1,..,NS\}$ generates R_s records, the total effort to generate a global model tends to:

$$Tcdm \approx Mcdm + \sum_{s} T(Rs)$$

Where *Tcdm* stands for the time needed to induce a global data mining model. *Mcdm* is the global modeling time. *T* stands for the communication time needed to transfer *Rs* records from the site *S* to the central site. Data security is another concern in sending data. This work adopts the pattern of Distributed Data Mining (DDM) approach where each distributed site sends an induced data mining model *Ms* to the central site. The key advantage of this method is it avoids sending large size of data from each distributed sites to the central site. The effort to induce a global model can be computed as:

$$Tddm \approx Mddm + \sum_{s} M(Ms) + \sum_{s} T(Ms)$$

Mddm corresponds to the global modeling time. *M* is the modeling time for the model M_s. When the volume of data rises *Tddm* tends to be much smaller than *Tcdm* (*Tddm* << *Tcdm*). The DDM can configure an attractive solution since the accuracies of the global models are similar.

All the experimental work was done using the Grid gain platform; a java based distributed computing middleware [4]. This middleware combines a computational grid with data grid and auto scaling on any managed infrastructure, moreover the applications are easy to implement.

The key objective of this work is to construct a global data mining model from different local models of the grid and compare DDM and CDM methods. Grid computing architecture is considered the best distributed framework for solving the distributed data mining task [5, 6]. Each node of the grid environment executes different UCS and those nodes send local data mining models to the central site for developing a global model. This paper introduces six different methods for merging local models from each distributed sites. The different strategies are: Specific Classifier Method (SCM), Weighted Classifier Method (WCM), Generalized Classifier Method (GCM), Majority Voting Method (MVM), Model Sampling Method (MSM) and Centralized Training Method (CTM). Three sets of data were used for testing all the strategies described in this paper: ICU data, 11mux problem and monks3 problem.

The remaining sections of this paper are organized as follows: Section 2 explains the construction methods of global model. Subsections of section 2 describe the six different construction strategies and the corresponding algorithms. Section 3 explains the experimental setup of ICU data, 11 multiplexer and monks3 problems. Section 4 discusses the performance of the different strategies used and related work. The final section presents conclusions and the road map for future efforts.

2. GLOBAL MODEL CONSTRUCTION

Gridclass is a project that uses the UCS for data mining proposes in a grid environment. Two levels of data mining models are generated in the Gridclass system. The first level is related to the models generated in each distributed sites and the second level correspond to the model generated in the central site. The first data mining models are known as local models. The second level is known as global model and is generated from all the local models in the first level. The global model represents all the data in the distributed environment.

During the training process, Gridclass system generates data mining models based on the training data and a predefined set of classifier [7]. If a predefined set of classifiers is provided, then the system can perform incremental learning. The incremental learning process improves the performance therefore the system can provide more generalized learning model. If a predefined set of classifiers is not provided, then the system generates the data mining models only from training data. Data mining models are maintained by genetic algorithm and covering operations in UCS system [8, 9, 10].

There are many challenges for constructing a global model, because wrong combination of the classifiers gathered from the local models, will affect negatively the performance of global model. The main difficulty is to derive the significance of each classifier and predict their values in the global model. All training data are completely independent even though there should be many similar classifiers with different sets of parameter values (benefits). Therefore the parameter evaluation of the classifiers in the global model is important.

Remaining sections demonstrate some solutions that are suitable for constructing the global model. Each strategy establishes different sort of combinations of local models in the global model. Those strategies help to understand the significance of availability of different sort of local classifiers in the global model. Each strategy has peculiar significance for the development of the global model. The performance of global model is evaluated from the testing accuracies of the global model.

2.1 Specific Classifier Method (SCM)

Specific Classifier Method (SCM) only preserves discrete classifiers in the global model [11]. SCM induce the global model without repeating similar classifiers and simultaneously keeping all the benefits of the local classifiers. The evaluation of the classifiers only considers the rule condition and the action part. SCM system doesn't consider the generality of the classifier. For example, 'c1' ="0#0" - >1, 'c2' =" 010" ->1, and 'c3' ="000"->1 are three classifiers. The action part of these classifiers is similar but condition part is different. In other sense the classifier 'c1' is more general than the other two classifiers but in SCM these three classifiers are considered as three different classifiers. If there are two or more classifiers that have similar conditions and actions, then system keeps only one classifier in the global model and updates the parameters of the classifier which is in the global model with the parameters of the repeated classifiers.

In SCM the initial process is to collect all the classifiers from the distributed sites and store them in a central location. The collected classifiers have to be evaluated based on the criteria of SCM and those classifiers that are eligible to be integrated the global model will be stored in the global model. While classifiers are evaluated, each classifier needs to be matched with all other classifiers in the collected local model. When one classifier finds another similar classifier in the collected local models then that classifier updates its parameters with parameters of matched classifier. Finally, the induced global model will be tested using a data set that was generated from the global data set.

The classifiers' parameters in SCM are updated using the expressions 1, 2 and 3. The expression 1 shows the modification of *number of match* and expression 2 explains the modification of *number of correct* and expression 3 shows the modification of *accuracy*. The parameter, *Last Time This Was in the GA* is modified with its maximum value among similar classifiers. Other parameters, *numerosity* and *correct set size* are not updated.

$$GNm = GNm + NNm \quad (1)$$

$$GNc = GNc + NNc \quad (2)$$

Where:

GNm = Number of Match of the current classifier in the global model; *NNm* = Number of Match of the new classifier; *GNc* = Number of Correct of the current classifier in the global model; and *NNc*= Number of Correct of the new classifier.

$$GAcc = \frac{(GNm*GNc*GAcc)+(NNm*NNc*NAcc)}{(GNm*GNc)+(NNm*NNc)} \quad (3)$$

The global model size of the SCM is dynamic. Algorithm 1 explains the functionality of the SCM.

BEGIN
Initialize Global Population
Collect all local models
Global_ size= 1
// update the parameters.
// Nm =number of match, Nc= number of correct, Gc= global model, Lc= local model

Repeat N
Repeat i
Repeat j
If Condition (Gc[j]) = Condition (Lc[i]) AND Ac (Gc[j]) = Ac (Lc[i]) Then
$Nm(Gc[j]) = Nm(Gc[j]) + Nm(Lc[i])$
$Nc(Gc[j]) = Nc(Gc[j]) + Nc(Lc[i])$
$Ac(Gc[j]) = \frac{(Ac(Gc[j]) * Nm(Gc[j]) * Nc(Gc[j])) + (Ac(Lc[i]) * Nm(Lc[i]) * Nc(Lc[i]))}{(Nm(Gc[j]) * Nc(Gc[j])) + Nm(Lc[i]) * Nc(Lc[i]))}$
$LastTimethiswas\ in\ the\ GA(Gm[j]) = Max(LastTimethiswas\ in\ the\ GA(Gm[j]), LastTimethiswas\ in\ the\ GA(Lm[i]))$
Flag=1
End if
Until j reached up to Global_ size
If flag != 1 then
Global_ Class [j+1] = Local_ Class[i]
Global_ size= Global_ size + 1
End if
Until i reached up to end of local models
Until N reached up to Node // node means number of nodes
END

Algorithm 1: The algorithm for Specific Classifier Method.

2.2 Generalized Classifier Method (GCM)

Generalized Classifier Method (GCM) only preserves more general classifiers in the global model [11]. The main intention of the GCM is to induce a global model with all *more general* classifiers. More general classifiers can represent all less general classifiers therefore in GCM. The system doesn't allow for less general classifiers into the global model. The parameter of the more general classifier which is already in the global model is updated with the value of the less general or similar classifier. In other case, if the new classifier is more general than the classifier that are already in the global model, then all less general classifiers have to be removed from the global model and the parameter of the new classifier are updated with the parameters of all removed classifiers.

Let's consider for example the three following classifiers: c1="0,0,0"->1 ,c2= "0,#,0"->1, c3= "0,1,0"->1; which come to the global model in that order, with 'c1' as the first classifier. In this case, the system only stores the classifier c2, because c2 is more general than the other two classifiers (c1, c3). Initially, the classifier c1 is stored in the global model, but when the classifier c2 arrives, classifier c1 is removed and c2 is stored in the global model. The parameters of the classifier c2 will be updated with the parameters of the classifier c1. Classifier c3 is less general than the classifier c2. So the system doesn't permit c3 to be stored in the global model. Here, the parameters of the classifier c2 are again updated with the parameters of c3. Normally, the global models generated by GCM are very small (compact) and the global model size are not determined by the user.

The initial process of GCM is to collect all local models from the distributed sites and store them in a global model. All classifiers whose condition and action part matches in the collected local models are stored and its parameters are updated with the parameters of other matched classifiers. All isolated classifiers are also stored in the global model. In a third step, each classifier (from the isolated classifiers) has to be evaluated based on the generality against to other classifiers that are available in the global model. If a classifier is less general than another classifier in the global model, then the less general classifier needs to be removed from the global model. Finally, last induced global model will be tested with a specific data that was collected from the global data set.

The classifiers' parameters are updated using expressions 4, 5, and 6. Expression 4 explains the modification of the parameter *number of match,* expression 5 explains how the *number of correct* is modified and expression 6 explains the modification of *accuracy*. The parameter *Last Time This Was In The GA* takes the maximum value among less general or similar classifiers. Other parameters such as numerosity and correct set size are not updated.

$$GNm = GNm + LNm \quad (4)$$

$$GNc = GNc + LNc \quad (5)$$

Expression 7, define the accuracy of new classifier.

$$Newacc = \frac{(GNm*GNc*GAcc)+(LNm*LNc*LAcc)}{(GNm*GNc)+(LNm*LNc)} \quad (6)$$

Where:

GNm = Number of Match of the more general classifier; *LNm* = Number of Match of the less general classifier; *GNc*= Number of Correct of the more general classifier; *LNc* = Number of Correct of less general classifier; *GAcc* = Accuracy of more general classifier; and *LAcc*= Accuracy of less general classifier.

Algorithm 2 explains the work flow of the GCM.

BEGIN
Initialize Global Population
Collect all local models
Global_ size= 1
Repeat N
Repeat i
Repeat j
If Condition (global_ Class[j]) = Condition (Local_ Class[i])
AND Action (global_ Class[j]) = Action (Local_ Class[i])
then
// update the parameters.
// Nm =number of match, Nc= number of correct,
//Gc= global model, Lc= local model

$$Nm(Gc[j]) = Nm(Gc[j]) + Nm(Lc[i])$$

$$Nc(Gc[j]) = Nc(Gc[j]) + Nc(Lc[i])$$

$$Ac(Gc[j]) = \frac{\left(Ac(Gc[j]) * Nm(Gc[j]) * Nc(Gc[j])\right) + \left(Ac(Lc[i]) * Nm(Lc[i]) * Nc(Lc[i])\right)}{(Nm(Gc[j]) * Nc(Gc[j])) + Nm(Lc[i]) * Nc(Lc[i]))}$$

$$LastTimethiswas\ in\ the\ GA(Gm[j]) = Max(LastTimethiswas\ in\ the\ GA(Gm[j]), LastTimethiswas\ in\ the\ GA(Lm[i]))$$

Flag=1
End if
Until j reached up to Global_ size
If flag != 1 then
Global_ Class [j+1] = Local_ Class[i]
Global_ size= Global_ size + 1
End if
Until i reached up to end of local models
Until N reached up to Node // node means number of nodes
Repeat i
Repeat j
If global_ Class[i] is more general global_ Class[j] then
// updates the parameters of Global_ class[i]
// Nm =number of match, Nc= number of correct,
// Gc= global model,Ac= Accuracy

$$Nm(Gc[i]) = Nm(Gc[i]) + Nm(Gc[j])$$

$$Nc(Gc[i]) = Nc(Gc[i]) + Nc(Gc[j])$$

$$Ac(Gc[j]) = \frac{\left(Ac(Gc[i]) * Nm(Gc[i]) * Nc(Gc[i])\right) + \left(Ac(Gc[j]) * Nm(Gc[j]) * Nc(Gc[j])\right)}{(Nm(Gc[i]) * Nc(Gc[i])) + Nm(Gc[j]) * Nc(Gc[j]))}$$

$$LastTimethiswas\ in\ the\ GA(Gm[j]) = Max(LastTimethiswas\ in\ the\ GA(Gm[j]), LastTimethiswas\ in\ the\ GA(Lm[i]))$$

Flag=1
End if
Until j reached up to Global_ size
Until i reached up to end of local models
Repeat i
Repeat j
If global_ Class[i] is more general global_ Class[j] then
Temp= j
Repeat Temp
Global_ Class[Temp] = Global_ Class[Temp +1]
Until Temp reached up to Global_ size -1
Global_ size = Global_ size -1
End if
Until j reached up to Global_ size
Until i reached up to Global_ size
END

Algorithm 2: The algorithm for Generalized Classifier Method.

2.3 Weighted Classifier Method (WCM)

Weighted Classifier Method (WCM) only maintains the highest weighted classifiers in the global population according to the global model size [11]. The purpose of the WCM is to calculate the quality of the classifiers from its parameters and eliminate all weightless classifiers from the global model. Global model size derived from the local model size. The accuracy of the classifiers is considered as the weight of a classifier. Classifier accuracy needs to be normalized because each local model may have a different background. Therefore, accuracy of a classifier needs to be multiplied to the ratio of the size of local training data set and the global training data set.

Initially, the system collects and sorts all the classifiers in the local model in descending order of the weights, then selects the classifiers that are in the range of global population size (to crowd the population). The global population in WCM cannot represent all the classifiers in the local models because the less weighted classifiers wouldn't be included in the global population. Algorithm 3 explains the workflow of the WCM.

BEGIN
Initialize Global Population
Collect all local training models
Repeat N // for each node
Repeat i
// calculate weights of each classifier
Weight (Local_ Class[i]) = number_ of_ match(Local_ Class[i])* number_ of_ correct (Local_ Class[i]) * accuracy (Local_ Class[i])
Until i reached up to POPMAX
// POPMAX is the maximum size of local model.
Until N reached up to Node
// node means number of nodes in the distributed site.
Repeat N
Repeat i
Repeat j
If weight (global_ Class[j]) < weight (Local_ Class[i]) then
Temp<- Global_ size
Repeat Temp
Global_ Class [Temp] = Global_ Class [temp -1]
Temp= Temp- 1
Until Temp = j
Global_ Class[j] =Local_ Class[i]
End if
Until j reach up to Global_ size
Until i reached up to end of local models
Until N reached up to Node
END

Algorithm 3: The algorithm for Weighted Classifier Method.

2.4 Majority Voting Method (MVM)

Majority Voting Method (MVM) is another strategy for constructing the global model from distributed local models. The goal of the MVM is to eradicate weak classifiers from the global model and construct a strong model in the central system (global model).

Initially, MVM gathers all local models and stores them in the central system, then goes on to find all discrete classifiers from the accumulated local models as SCM. Later, the system calculates a threshold value (*cut_ off_ threshold*) from the collected classifiers and uses it to benchmark the classifiers in the population [11]. If the accuracy of a classifier is greater than the *cut_ off_ threshold value* then that classifier will be stored in the global model. The threshold value is derived from the accuracies of the discrete classifiers, i.e. the average of the accuracies of the classifiers in the collected local models. The global population size of MVM is dynamic. Algorithm 4 describes the work flow of the MVM.

BEGIN
Initialize Global Population
Collect all local models
Global_ size= 1
Repeat N
Repeat i
Repeat j
If Condition (global_ Class[j]) = Condition (Local_ Class[i]) AND Action (global_ Class[j]) = Action (Local_ Class[i]) then
// update the parameters.
// Nm =number of match, Nc= number of correct,
// Gc= global model, Lc= local model
$Nm(Gc[j]) = Nm(Gc[j]) + Nm(Lc[i])$
$Nc(Gc[j]) = Nc(Gc[j]) + Nc(Lc[i])$
$$Ac(Gc[j]) = \frac{(Ac(Gc[j]) * Nm(Gc[j]) * Nc(Gc[j])) + (Ac(Lc[i]) * Nm(Lc[i]) * Nc(Lc[i]))}{(Nm(Gc[j]) * Nc(Gc[j])) + Nm(Lc[i]) * Nc(Lc[i]))}$$
$LastTimethiswas\ in\ the\ GA(Gm[j]) = Max(LastTimethiswas\ in\ the\ GA(Gm[j]), LastTimethiswas\ in\ the\ GA(Lm[i]))$
Flag=1
End if
Until j reached up to Global_ size
If flag != 1 then
Global_ Class [j+1] = Local_ Class[i]
Global_ size= Global_ size + 1
End if
Until i reached up to end of local models
Until N reached up to Node // node means number of nodes
AvgAcc = 0
Repeat j
AvgAcc = AvgAcc + Accuracy(Global_ Class[j]
Until j reached up to Global_ size
$$AvgAcc = \frac{AvgAcc}{Global\_size}$$
Repeat j
If Accuracy (Global_ Class [j]) < AvgAcc then
Temp = j
Repeat Temp
Global_ Class [Temp] = Global_ Class [Temp + 1]
Until Temp up to Global_ size -1
Global_ size =Global_ size - 1
Until j reached up to Global_ size
END

Algorithm 3: The algorithm for Majority Voting Method.

2.5 Model sampling Method (MSM)

Model Sampling Method (MSM) is another strategy for constructing the global model from distributed local models. The main intension of MSM is to replicate the classifiers depending on the experience of each classifier. Each time a classifier is correctly matched with an example (training data), the value of *number of match* of that classifier will be increased by one. Therefore the experience of a classifier is equivalent to the *number of match* of a classifier.

The system replicates the classifier proportionally to the value of experience of a classifier. During sampling, all don't care symbols in the rule condition are replaced with other suitable values. But parameters of the replicated classifiers received the same values from the base classifiers. After sampling, replicated classifiers have to be filtered based on some quality criteria. Quality of a classifier is defined from the accuracy of that classifier. In MSM, the system will filter the classifier based on the user defined quality level. After the filtering, system will provide a final global model. Fig 1, describes the structure of MSM and Algorithm 5 explains the MSM operations.

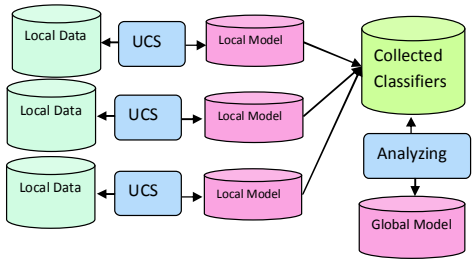

Figure 1. Basic structure of MSM.

BEGIN
Initialize Global Population
Initialize GloAccTh
// global accuracy threshold
Collect all local models
Sum_num_match<-0
Repeat N
Repeat i
// update the parameters.
// Nm =number of match, Nc= number of correct,
// Num= numerosity, NCorr = Number of correct,
//Gc= global model, Lc= local model
Condition (Gc[j,])= Condition (Lc[j,])
$Nm(Gc[j]) = Nm(Lc[i])$
$Nc(Gc[j]) = Nc(Lc[i])$
$Ac(Gc[j]) = (Ac(Lc[i])$
Num(Gc[j])= Num(Lc[j])
NCorr(Gc[j])= NCorr(Lc[j])
$LastTimethiswas\ in\ the\ GA(Gm[j]) = LastTimethiswas\ in\ the\ GA(Lm[i]))$
j=j+1
Until i reached up to end of local models
Until N reached up to Node
// node means number of nodes
Global_ size=j
//Replication process based on number of match of the classifier
//NewAc =new Accuracy, NewNm= New Number of Match,
// NewNc= New number of correct, NewNum= new numerosity,
// NewNcorr= New number of correct,

//NewLastTimethiswas in the GA= New LastTimethiswas in the GA ,
//Rgs=size of repeated global population.
Rgs<-1
Repeat i
Repeat j
Repeat k
If Condition (Gc[i,k])== # then
NewCondition [Rgs, k]=random value of value of that position
End if
Until k reached up to size of condition
NewAc[Rgs]= $Ac(Gc[i]) / (Localmodelsize * Globalpopulation size)$
NewNm[Rgs]=Nm(Gc[i])
NewNc[Rgs]=Nc(Gc[i])
NewLastTimethiswas in the GA[Rgs]= LastTimethiswas in the GA(Gc[i])
NewNum[Rgs]= NewNum[i]
NewNcorr[Rgs]= NewNcorr[i]
Rgs<-Rgs+1
Until j reached up to Nm(Gc[i])
Until i reached up to Global_size

// filter the repeated classifier
Repeat i
If NewAc[i] < GloAccTh Then
Remove i^{th} *classifier from the population.*
Until i reached up to Rgs (Global_ size)
END

Algorithm 5: The algorithm for Model sampling Approach.

2.6 Centralized Training Method (CTM)

Global Model Method (CTM) is based on the training of the replicated local models into the central system. In CTM, classifiers in the local models are replicated like in MSM, converting those replicated rule conditions and actions to the central training data. The parameters of the classifiers in the local models are ignored while constructing the centralized training data. The central training data is obtained using UCS to form a final global model. The global model size of the CTM is the maximum population size of the UCS in the central node. CTM structure is depicted in Figure 2 and its operation is explained in the Algorithm 6.

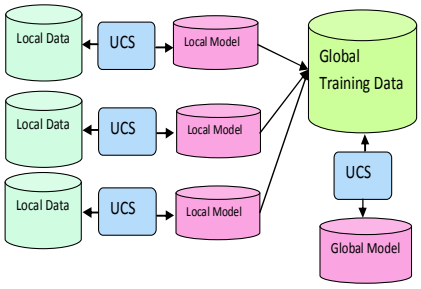

Figure 2. Basic structre of CTM

BEGIN
Initialize Global Population
Initialize GloAccTh // global accuracy threshold
Collect all local models
Repeat N
Repeat i
//, Gc= global model, Lc= local model, Ac is accuracy
Condition (Gc[j,])= Condition (Lc[j,])
Ac(Gc[j])=Ac(Lc[i])
j=j+1
Until i reached up to end of local models
Until N reached up to Node
// node means number of sites in the distributed system
Global_ size=j
//Replication process is based on number of match of the classifier
//Rgs=size of repeated global population.
Rgs<-1
Repeat i
Repeat j
Repeat k
If Condition (Gc[i,k])== # then
NewCondition [Rgs, k]=random value that is suitable for that position
End if
Until k reached up to size of condition
NewAc[Rgs]= $Ac(Gc[i]) / (Localmodelsize * Globalpopulation size)$
Rgs<-Rgs+1
Until j reached up to Nm(Gc[i])
Until i reached up to Global_size
// Training the replicated population,
//UCS = supervised learning classifier system
UCS(NewCondition, NewAc)
// UCS will return the final global model
END

Algorithm 6: the algorithm for Centralized Training Method.

3. EXPERIMENTAL WORK

Three sets of data were taken for this experimental work: 11mux problem; monks3problem; and Intensive Care Unit (ICU) data. 11mux and Monks are synthetic data while ICU data is a real world data that was collected from the Oporto Hospital Center. 11mux problem has 2048 ordered cases (2^{11}). The first three bits of the 11mux is considered as the address bits while the final eight bits are considered as the answer bits. Monks3 problem has 8 attributes and 432 instances. The permitted values for the first, the second, and the fourth positions is from 1to 3, third and sixth position can have values 1 and 2, and the fifth position can have values between 1 and 5. The ICU data is concerning to the prediction of organ failure about six different organic systems [12]. There are 31 fields of data that was collected from three different sources such as the electronic health record, ten bed side monitors, and paper based nursing records. The ICU data set has a total of 3566 records of data, from 32 patient's information of first five days. The three datasets have only two classes (1 and 0).

Four nodes were considered for the distributed sites and a same number of training and testing data sets were created from the main data set. Random generated data was provided to each training and testing data sets. For the induction of local data mining models 5000 training iteration were considered for each execution. Holdout sampling has been applied to all data sets partitioning them into training (70 %) and testing data sets (30%). Seventy percent of the training data was equally distributed into 4 data sets, one for each node in the distributed sites; similarly thirty

percent of the testing data was equally distributed into 4 data sets among the four nodes in the distributed sites. In the experiment, 400 classifiers were considered in each local model. The configuration parameters used in the UCS are: *ProbabilityOfClassZero* = 0.5, *V* = 20, *GaThreshold* = 25, *MutationProb* = 0.05, *CrossoverProb* = 0.8, *InexperienceThreshold* = 20, *InexperiencePenalty* = 0.01, *CoveringProbability* = 0.33, *ThetaSub* = 20, *ThetaSubAccuracyMinimum* = 0.99, *ThetaDel* =20, *ThetaDelFra* = 0.10.

3.1 DDM Experiments

Table 1 shows the global model testing accuracies for the six DDM strategies considering the three different data sets. The data set of the monks 3 problem is very small and simple, on the opposite the ICU data set is a bigger and more complex data set.

Table 1. Testing accuracies of global models generated using 6 different strategies.

Data Set	Strategy	Accuracy	Global model size
ICU	GCM	0.84	1382
11mux	GCM	0.79	238
Monks3	GCM	0.72	17
ICU	SCM	0.85	1466
11mux	SCM	0.93	775
Monks3	SCM	1	276
ICU	MVM	0.89	1416
11mux	MVM	0.91	507
Monks3	MVM	1	240
ICU	WCM	0.73	400
11mux	WCM	0.94	400
Monks3	WCM	1	400
ICU	MSM	0.85	2755
11mux	MSM	0.87	14,901
Monks3	MSM	0.89	42905
ICU	CTM	0.63	400
11mux	CTM	0.84	800
Monks3	CTM	1	400

Table 2 compares the results obtained crossing the different data sets and the DDM global model construction strategies considered. For monks 3 problem SCM, MVM, WCM, and CTM achieved best accuracy. For 11 multiplexer problem WCM received best accuracy but SCM and MVM also have good accuracies. For ICU data MVM reached the best accuracy but GCM, SCM and MSM also have almost similar accuracies. Based on the strategies, three datasets in SCM and MVM have good accuracies. But for 11 multiplexer and monks3 problem WCM received good accuracies.

Table 2. Results by data set and DDM strategies.

Strategies	ICU	11MUX	MONKS3
GCM	**0.84**	**0.79**	**0.72**
SCM	**0.85**	**0.93**	**1**
MVM	**0.89**	**0.91**	**1**
WCM	**0.73**	**0.94**	**1**
MSM	**0.85**	**0.87**	**0.89**
CTM	**0.63**	**0.84**	**1**

3.2 CDM Experiments

For the CDM experiments, a training data set was created by combing all four training data sets in the distributed sites. So the sizes of the centralized training data sets are: 2496 for the ICU, 1360 for the 11 multiplexer problems, and 576 for the monk's problem. The number of training iterations considered was 10000 because the data sets are larger. Table 3 shows the testing accuracies attained by the CDM method with the three different datasets.

Table 3. Testing accuracies for the CDM method.

Data	Accuracy	Model size
ICU	0.62	400
11mux	0.97	400
Monks3	1	400

Taking as an example for comparison the ICU dataset, CDM attained an accuracy of 0.62. For the same dataset, excepting the CTM method, all the other DDM methods attained higher accuracies. For the 11multiplexer problem, all the DDM methods attained a lower accuracy than CDM. In Monks3 problem, only MSM and GCM have less accuracy then the CDM.

4. DISCUSSION AND RELATED WORK

The main goal of this work is to induce global data mining models and compare the performance of CDM versus the DDM methods. Six strategies described above were able to construct the global model from the distributed local models. The global model in the CDM method is obviously representing the overall problem (dataset) in the distributed site because that model is generated from global data without any intervention. Though table 2 and 3 shows that DDM and CDM attained similar accuracies. SCM and MVM strategies could generate good global models than the other four methods, since SCM and MVM have good accuracies in the given three datasets. Another advantage assignable to the DDM method is that it avoids sending large size of data from different sites to a central site. DDM data is processed at each distributed sites and generate learning models. As mentioned in the introduction the size of the training data is always very large than the data mining model size (classifiers population) and the computational and communicational times associated to DDM tend to be very much lower the required for CDM. This way of processing has two main advantages: 1) privacy of the data; and 2) less communication costs [8].

It should be stressed that those strategies are not based on any specific domain. The main idea of behind these different strategies is to understand the behavior of a global model constructed with classifiers copied from the local modes. The first two strategies (SCM and GCM) shape the global model based on the rules (Condition and Action of the classifier), next two strategies (MVM and WCM) shape the global model based on the classifiers' parameter values. The last two strategies (MSM and CTM) shape the global model based on the replication function.

Therefore the best way of comparing these strategies is based on these groups. GCM method can generate more compact global models but the accuracies attained are always poor. The SCM attains good accuracies in all global models but the population size is higher than the GCM method. Every classifier in local model has at least one representative classifier in the global model. This is the key advantage of the SCM and GCM strategies. MVM and WCM strategies find the best classifier in the global model from the classifiers in the local models based on its parameters (*classifier experience and accuracy*). The WCM has a fixed size of population but in MVM the global model population size is not defined by user. Testing accuracies of the MVM and WCM are almost similar. WCM presented a better operational efficiency, since it follows a smaller algorithm for constructing the global model. MSM and CTM correspond to the more complicated methods to induce the global model. The main disadvantage of MSM is the large population size of the global model. For example, MSM generated 42905 classifiers in the population of the global model for the monks3 problem. But MSM presented better global model testing accuracies than CTM. CTM's limitation is the needing for a high number of training iterations, since the replication process will generate large size of training data in the central system from the collected local models. Also the testing accuracies of the CTM are poor.

Considerable related work could be found in parallel and distributed implementations of LCS. The experimental work is mainly oriented to compare the speed-up attained. Our work points to a different direction. We are primarily concerned with the induction of global models based on local models. Similarities can be established with meta-learning approaches. The goal of the *meta-learning* is to construct the global population of classifiers from a collection of inherently distributed data sources [13]. GALE (Genetic and Artificial Life Environment) is another related work in the distributed data mining area. GALE is a fine grained parallel genetic algorithm based on a classification system [14]. Learning classifier system ensembles with rule sharing is another associated work relating to in the parallel and distributed LCS [15].

5. CONCLUSIONS AND FURTHER WORK

This paper presented six different strategies to induce global data mining models in a distributed environment (e.g., grid our cloud environments). Based on preliminary results, some final conclusions can be made:

- The performance in terms of accuracy of DDM seems to be similar to the performance of the CDM. Therefore DDM approach is more convenient and economical for the implementation of the distributed data mining problems;
- Among the various strategies of DDM, WCM accrue superior accuracy when compared to the other strategies.

Further experimentation is required to corroborate the results presented in this work and to reach more robust conclusions (e.g. considering more configurations in terms of the data sets, number of iterations). Next work will also be focused on real world data and include more dynamic methods to construct global model from the distributed local learning models.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

[1] M. F. Santos, W. Mathew, T. Kovacs, H. Santos, A grid data mining architecture for learning classifier system. WSEAS TRANSACTIONS on COMPUTERS Volume 8, 2009 ISSN: 1109-2750

[2] M. F. Santos. Learning Classifier System in Distributed environments, University of Minho School of Engineering Department of Information System. PhD Thesis work 1999.

[3] A. Giani, Parallel Cooperative classifier system. Dottorato di ricerca in informatica Universita di Pisa, PhD Thesis TD-4/ 99.

[4] http://www.gridgain.com/key_features.html. Consulted on 8 - 2 - 2011.

[5] .M.Cannataro, A. Congiusta, A. Pugliese, D.Talia, P. Trunfio, Distributed Data Mining on Grid: Services, Tools, and Applications. *IEEE TRANSACTIONS ON SYSTEM, MAN, AND CYBERNETICS- PART B: CYBETNETICS, VOL. 34 NO6,* DECEMBER 2004

[6] J. Luo, M. Wang, J. Hu, Z. Shi, Distributed data mining on Agent Grid: Issues, Platform and development toolkit. *Future Generation computer system 23 (2007) 61-68*

[7] J. Luo, M. Wang, J. Hu, Z. Shi, Distributed data mining on Agent Grid: Issues, Platform and development toolkit. *Future Generation computer system 23 (2007) 61-68.*

[8] http://www.idsia.ch/~juergen/icmlkolmogorov/node9.html. Consulted on 1/7/ 2010.

[9] H. H. Dam, A scalable Evolutionary Learning Classifier System for Knowledge Discovery in Stream Data Mining, M.Sci. University of Western Australia, Australia, B.Sci. (Hons) Curtin University of Technology, Australia. *Thesis work 2008.*

[10] A. Orriols-Puig, A Further Look at UCS Classifier System. *GECCO'06,* July 8–12, 2006, Seattle, Washington, USA

[11] M. F. Santos, W. Mathew, and H. Santos: GridClass: Strategies for Global Vs Centralized Model Construction in Grid Data Mining, Proceeding of the workshop on ECAI, Lisbon 2010.

[12] M. Vilas-Bous, M. F. Santos, F. Portela, A. Silva, F. Rua, Hourly prediction of organ failure and outcome in intensive care based on data mining techniques. ICEIS 2010 conference, 2010.

[13] E. Cesario, A. Congiusta, D. Talia, P.Trunfio, Data analysis services in the knowledge Grid in DATA MINING TECHNIQUES in Grid Computing Environments, Ed. Dubbitzky,W., Wiley-Blackwell, UK, 2008.

[14] X. Llora, Joseph M. Garrell, Knowledge- Independent Data Mining with Fine Grained Parallel evolutionary Algorithm. In proceeding of the Genetic and Evolutionary Computation Conference (GECCO 2001).

[15] L. Bull, M. Studley, A. Bagnall, I. Whittley, Learning Classifier System Ensembles With Rule Sharing. IEEE 1089-778x, 2006.