The Influence of Data Replication in the Knowledge Discovery in Distributed Databases Process

Valentin Pupezescu
Applied Electronics & Information Engineering Dept
Polytechnic University of Bucharest
Bucharest, Romania
vpupezescu@yahoo.com

Radu Rădescu
Applied Electronics & Information Engineering Dept
Polytechnic University of Bucharest
Bucharest, Romania
rradescu@gmail.com

Abstract – The Knowledge Discovery in Distributed Databases is the process of finding useful knowledge from mining data sets stored in real implementations of distributed databases. Distributed Databases represents a software system that allows a multitude of applications to access the data stored in local or remote databases. In this scenario, the distribution of data is achieved through the replication process. Nowadays many solutions for storing the data are available: relational distributed Database Management Systems (DBMS), NoSQL storing solutions, NewSQL storing solutions, graph oriented databases, object oriented databases, object-relational databases, etc. The present study analyzes the most commonly used storing solution: the relational model. The replication topology used in the related experiments was the classical publisher-subscriber topology. The distribution of data is made from the publisher system. In this paper we study the interaction between the most suited distributed data mining architecture (Distributed Committee Machines) for mining distributed data and real relational distributed databases. The chosen Data Mining task is the classification one. Distributed Committee Machines are a group of neural networks working in a distributed manner in order to achieve better classification results [3][4].

In this paper we will study the interaction between DCM architectures (Fig. 1) and a real implementation of relational distributed database.

Keywords: Knowledge Discovery in Distributed Databases, Knowledge Management, Data Mining, Distributed Databases, Distributed Learning, Neural Networks, Distributed Committee Machines

I. INTRODUCTION
The Knowledge Discovery in Relational Distributed Databases process represents the process of finding useful information in relational distributed databases [1]. The entire process usually consist in these steps: understanding the application domain, creating the target data set from raw data stored in databases, data cleaning and preprocessing, data transformation, choosing the data mining algorithm for the given data mining task, applying the data mining algorithm on the processed data set, analyzing the obtained patterns, discovered knowledge approval [1][2].

Data Mining (DM) represents a set of specific methods and algorithms aimed solely at extracting patterns from raw data [1]. The DM tasks are: classification, regression, clustering, association rules, summarization, dependency modeling, change, and deviation detection [1]. In our experiments we used classical neural structures (multilayer perceptrons) in order to achieve the classification task.

A. Distributed Committee Machines
The most suited architecture for mining data sets stored in real implementations of relational databases are the Distributed Committee Machines (DCM). This architectures consists in more than one neural network that work in a distributed manner in order to achieve better classification results [3][4].

In this paper we will study the interaction between DCM architectures (Fig. 1) and a real implementation of relational distributed database.
The entire input data set is identical for all distributed multilayer perceptrons (MLP-1, MLP-N). The data availability is achieved through the replication process. Every neural structure (MLP unit, see Fig. 2) will run on a different computing machine and will obtain its local result. In our experiments we choose “the winner takes it all” policy. After each multilayer perceptron ends its training and testing epochs, in the final step, the combiner module will choose the best classification result.

The neural structure units (MLP) were trained with the classical backpropagation algorithm (BKP) [5]. For this experiment we used the following activation function:

\[ y = f(v) = \frac{\tanh(\lambda v)}{2} \]  

\[ v = \sum_{i=0}^{n} w_i x_i \]  

The classification problems that were analyzed with the BKP algorithm were the standard iris1 [6], wine1 [7] and conc1 (in this set there are two classes of stimuli arranged in a concentric way – one class inside the circle and the other one outside the circle). The activation function is divided by two because the output data of the analyzed data are scaled in the interval \([-0.5, 0.5]\). For the output layer of the multilayer perceptron (from the \(k\) layer – see Fig. 2) we have the next set of equations (\(\eta\) was experimentally set to a value of 0.1) for adjusting the weights [5]:

\[ e_k = d_k - y_k \]  

\[ \delta_k = e_k \varphi'(v_k) \]  

\[ w_{kj}(t + 1) = w_{kj}(t) + \eta \delta_j y_i \]  

The data is arranged as follows (Fig.2):

\[ TR = \left( (x_1, d_1), \ldots, (x_n, d_n) \right) \]  

\[ TR \]  

\[ TS = \left( (x_1, d_1), \ldots, (x_n, d_n) \right) \]  

\[ TS \]  

Figure 3. Training and testing data

The analyzed datasets are kept in the database in the following configuration:

<table>
<thead>
<tr>
<th></th>
<th>iris1</th>
<th>trr</th>
<th>tsr</th>
<th>trs</th>
<th>tss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Columns</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>wine1</th>
<th>trr</th>
<th>tsr</th>
<th>trs</th>
<th>tss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines</td>
<td>90</td>
<td>88</td>
<td>90</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>Columns</td>
<td>3</td>
<td>3</td>
<td>13</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>conc1</th>
<th>trr</th>
<th>tsr</th>
<th>trs</th>
<th>tss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines</td>
<td>200</td>
<td>100</td>
<td>200</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Columns</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

### B. Distributed Databases

All distributed databases implementations rely at the base on the replication process. The replication process represents the duplication of data from the publisher server on the subscriber type systems. We worked in this manner because it is very important to have the same input data sets for all distributed neural structures from the DCM architecture.

The Relational Database Management Systems used in our experiment was Microsoft SQL Server. We wanted to experimentally determine the influence of replication types available in SQL Server on the overall performance of the DCM structure.

In SQL Server we mainly have three types of replication: merge replication, snapshot replication and transactional replication [8]. This DBMS also support a variant for the transactional replication – transactional replication with queued updating. Merge replication offers the highest level of autonomy and accepts the highest levels of latencies [8] [9].

Snapshot replication makes the copy of the entire database to all the subscribers [8] [9]. This replication guarantees transactional consistency because all changes are made on the publisher system [8] [9].
In the transactional replication the transactions are sent from publisher to subscribers [8] [9]. Changes are made only at the publisher [8] [9]. In our previous work [9] we analyzed the internal functioning and latencies for each type of replication. In Fig.3 we presented the used architecture for our experiments (Fig. 4) – here we used the publisher-subscriber replication topology (master-slave):

In the following section we will present the classification results and the benefits of using DCM structures instead of individual neural networks. In the experiments we used 2 neurons, 4 neurons, 6 neurons and 8 neurons on the hidden layer of the multilayer perceptrons. We used a maximum of 6 distributed computing machines. On each system will run a multilayer perceptron in the following manner: each MLP is trained (with the training data set) and then tested (with the testing data set) for 1000 epochs. After each testing, we computed the misclassification rate for the testing data set (PCICtest). Each MLP will have its best misclassification rate and the combiner module will choose the best result. The entire application was written in Java. On each system we had TCP Java servers that sent the best local results to the combiner module. The application also allows all the configuration parameters for the MLP units to be set and then sent for each computing system.

For our initial experiments that show the advantage of working in a distributed manner we used the merge replication. All writing operations were done locally because this type of replication permits this way of working.

In Fig.5, Fig.6 and Fig.7 we presented the classification results for the DCM:

As we can see in Fig.5, Fig.6 and Fig.7 it is an advantage to work in a distributed manner with multiple neural networks. In all cases we obtained good results when working with DCMs with multiple multilayer perceptrons executed in a distributed manner. The same classification results can be achieved when working with many MLP units that run in a sequential order – the only disadvantage in this case would be that the entire system would be slow.

II. THE INFLUENCE OF THE REPLICATION PROCESS ON DISTRIBUTED COMMITTEE MACHINES

The measured parameters for DCMs are distributed acceleration (Sd) and distributed efficiency (Ed) [10]:

\[
S_d = \frac{T_d}{T_{d,\text{max}}}
\]

\[
T_d = \max \{t_1, t_2, ..., t_n\}
\]

\[
\overline{t}_i = \frac{1}{N} \sum_{j=1}^{N} t_j
\]
The distributed efficiency is given by the following ratio [10]:

$$E_d = \frac{S_d}{n}$$  \hspace{1cm} (12)

In our experiment $n$ represents the total number of subscriber systems (we had a maximum of 6 distributed subscribers).

The following experimental results were done for all types of SQL Server types of replication available in merge replication, snapshot, transactional replication (transact1) and transactional replication with queued updating (transact2). The distributed multilayer perceptrons had on their hidden layers a number of 2, 4, 6 and 8 neurons.

Figure 8. Distributed speedup – conc1, 2 neurons

Figure 9. Distributed efficiency – conc1, 2 neurons

Figure 10. Distributed speedup – conc1, 4 neurons

Figure 11. Distributed efficiency – conc1, 4 neurons

Figure 12. Distributed speedup – conc1, 6 neurons

Figure 13. Distributed efficiency – conc1, 6 neurons

Figure 14. Distributed speedup – conc1, 8 neurons
The Influence of Data Replication in the Knowledge Discovery in Distributed Databases Process

In Figure 15 we have an example of a distributed execution on 6 distributed systems (IP0 – IP5) for the conc1 classification problem.

In Figure 16 we have an example of a distributed execution for 6 distributed MLP units with 8 neurons on the hidden layer.

In Figure 17 we have an example of a sequential execution on 6 MLP units executed in a sequential manner on one computing system (IP0) for the conc1 classification problem.

In these experiments all MLP units made writing operations after each testing epoch. In the merge replication we did the inserts in the local result databases for each distributed system.

The best performance was obtained for snapshot replication and for transactional replication (both transactional versions, see Fig. 8 – Fig. 15). In the snapshot replication all the writing operations were made on the publisher system. The same behavior applies for the classical transactional replication (transact1).

Although the snapshot replication has the best performance it should be noted that this type of replication will copy the entire database from the publisher to all the subscribers for every modification on the database from the publisher. This is a big workload on the network inside the DCM architecture so it should be avoided.

In the transactional replication with queued updating (transact2), all the writings are made on the local result database but these operations will be made automatic also on the publisher server by the database management system [8].

The most recommended type of replication when working in this configuration is the transactional replication (both versions are indicated). Further optimization for this architecture can be made by minimizing the number of writing operations. This can be achieved by making the insert in the database of the classification result only if the obtained misclassification rate (PCICtest) has a lower value than the previous optimum.

III. CONCLUSIONS

In this paper we analyzed the influence of the replication process on the knowledge discovery in distributed relational databases process. We worked with Microsoft SQL Server relational database management system for storing all data sets. The most suitable type of replication for our experiment was the transactional replication. This research is very close to real implementation of distributed committee machines because we did not wanted to store the input data and the results in simple files. None of the real data mining applications should work with data sets stored in files because of the security issues that might appear. Another advantage of databases management systems is also the transaction and data recovery support [11]. Distributed data mining research should take into account all these implementation factors. This work is useful in many research areas such as medical research, medical diagnosis, big data implementations, adaptive e-Learning [12], astronomy, gaming, business, knowledge management, artificial intelligence.

REFERENCES


