Proceedings of the V Brazilian Conference on Neural Networks - V Congresso Brasileiro de Redes Neurais
pp. 139–142, April 2–5, 2001 - Rio de Janeiro - RJ - Brazil

A Neural Network Implementation for Data Mining High Performance Computing

Myrian C. A. Costa, Nelson F. F. Ebecken
COPPE/UFRJ, Brazil
E-mails: myrian@ntt.ufrj.br, nelson@ntt.ufrj.br

Abstract

This work discusses the implementation of a neural network algorithm within a data mining strategy for high performance computers. The study focuses an application comprising all the aspects related to the data preparation, the neural network implementation and training. The performance evaluation of the implementation is also addressed for two different computer architectures. Some conclusions and recommendations from this experience were done.

1. Introdution

In the last few years the development of methods and algorithms for Knowledge Discovery in Databases (KDD) has been growing very fast [1].

The KDD technology comprises several phases of transformation any kind of data that resides in a huge and complex database, frequently within high performance computers into useful and handled information

The preprocessing phase or data preparation plays a crucial role for the success of KDD process. This phase uses some automatic or semi-automatic data extraction techniques of databases and performs basic operations, like removal of noise and spurious data, treatment of missing data fields and effective reduce of number of variables under consideration. Almost always this phase reaches proper data for analysis even in a presence of poor data quality. (This technology involves the process of finding and interpreting patterns from data extracted of huge and complex databases). The Data Mining phase of KDD process involves efficient and intelligent algorithms and techniques that handle the preprocessed data, analyzing and discovering patterns from them. The last phase of this technology describes the interpretation and evaluation of discovered patterns into useful knowledge.

The overall KDD process is iterative, covers the repeated application of specific data mining method or algorithm. Neural Network is one of the most used data mining method to extract patterns in an intelligent and reliable way and has been greatly used to find models that describe data relationship.

1.1. Neural Network

Neural networks are multilayer structures of neurons (or basic units), as illustrated in figure 1, that maps input data in output data, using internal representations of hidden units. The repetitive presentation of a set of input/output pairs of data with the adjusting of the connection weights for minimization of the output error, is the learning phase of the procedure, which guides the search for a model that describes the data relationship. At each step of the process one data is presented to the input of the network, which computes the output and compares with the desired output. The error obtained with the difference of these two outputs is propagated back through the network adjusting internal network parameters, which are the connection weights between units of different layers.

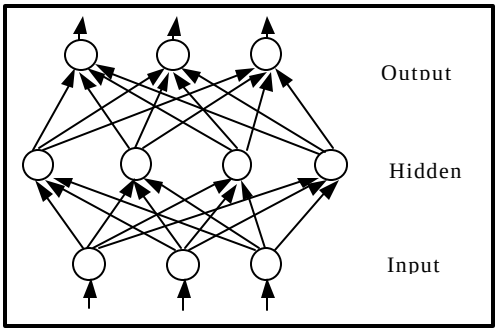

Figure 1 – Multilayer Neural Network

The back-propagation algorithm, proposed for Rumelhart, Hinton and Williams [2] in 1986, uses the generalized delta rule to minimize the output error function. The authors show that the derivative of the error measure with respect to each weight connection is proportional to a weight change with negative constant of proportionality. This correspond to performing steepest descent on a surface weight space height at any point in weight space is equal to the error measure. The rule for changing weight of the connection from unit i to unit j in different layers is represented by the Δw_{ji}, and is calculated by:

$$\Delta w_{ji}(n+1) = \eta \cdot \delta_j \cdot a_i + \alpha \cdot \Delta w_{ji}(n), \qquad (1)$$

where h is the constant of proportionality, the learning rate, d_j is the error at unit j, a_i is i-th input of unit j and a is the momentum term, that determines the effect of past weight changes on the current direction of movement in weight space. The error at unit j is calculated with two equations, one for the output layer units, equation2, and another for the hidden layers units, equation 3.

$$d_j = (t_j - a_j) \cdot f_j'(net_j) \quad (2)$$

$$\boldsymbol{d}_j = f_j'(net_j) \cdot \sum_k (\boldsymbol{d}_k \cdot w_{kj}) \quad (3)$$

where t_j is the desired output, a_j is the calculated output, the term f_j' is the partial derivative of the activation function and the factor S is determined recursively in terms of the error signals of the units to which it directly connects and the weights of those connections.

The activation function requires a continuos and non-linear function. One of the most used is the logistic function, with the formulation:

$$a_j = \frac{1}{1 + e^{-\left(\sum_i (w_{ji} \cdot a_i) + \boldsymbol{q}_j\right)}}, \quad (4)$$

where q_j is a bias similar in function to a threshold. The partial derivative of this function is:

$$f_j'(net_j) = a_j(1 - a_j). \quad (5)$$

The Back-Propagation algorithm has one phase for training and one phase for testing. During training phase it repeats the above procedure until it reaches an acceptable error and during testing phase it produces the error model with the presence of data never seen before. The data set for each phase is different that assures proper overall error.

Some points have to be considered using the back-propagation algorithm. The initial values of weight connections must be small, positive and negative, unequal and random set otherwise the network encounters a local maximum and never learns. Gallant [3] proposes the choice of weights in range [-2/z, 2/z], where z is the number of inputs of a unit. The convergence to a local minimum is not guaranteed even with improvements of the method. Some works have been reported with improvements in this method accelerating the convergence [4], [5], or trying to resolve the problem of local minima [6] [7].

The algorithm requires long training times on a serial machine, leading to a study of its parallel implementation. The use of distributed memory architectures with powerful computing nodes interacting with each other via message passing is encouraged.

In order to find the best configuration (number of neurons per layer) of the neural network two approaches are usually used. The empirical approach starts considering few neurons at hidden layer, adding more neurons as the training gets better. In spite of the good results this procedure becomes tedious and long. With the use of genetic algorithm this procedure can be improved, making several configurations as the initial population and using the training error as the individual configuration fitness. As the population evolves, new generation of configurations, resulting from mutation and crossover of antecedent individuals, were created with better fitness than the last one. The best fitness individuals will survive, showing the best configurations for the network.

2. Parallel Neural Network Implementation

As mentioned before neural network model is an inherently parallel structure with independent units performing local calculation [8]. The main issues under consideration in the study of parallel implementation of conceived sequential algorithms are the partitioning algorithm schema and the target machine. These two topics are presented here in the next sections.

2.1. Partitioning Algorithm Schema

For the partitioning algorithm schema the study considers the use of the data parallelism approach, as illustrated in figure 2, that keeps a copy of the entire neural architecture, with the internal variables and functions, in each processing node partitioning training data set among nodes. This approach ensures that all values needed during the training phase, like output calculation and partial error back propagation, are locally available, reducing the number of messages passing among nodes and the algorithm synchronization. In fact, all nodes perform only one communication after each complete presentation of a training data subset. In distributed memory machines this approach seems to be very efficient leading to great gain of performance.

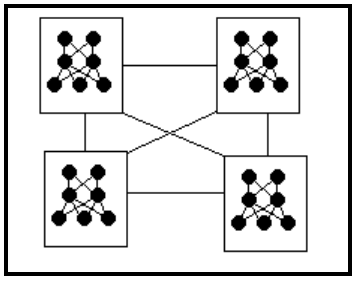

Figure 2 – Data Parallelism

This implementation of data parallelism uses a control node (node 0) that gets training data set, normalizing and distributing them among nodes. All

nodes are started with same internal parameters but with different subsets of training data. During training phase of the neural net, each node passes its training data subset producing partial errors that must be combined with partial errors produced by other nodes, generating an overall error. This error is used to update the connection weights in each node, until the procedure reaches the minimal acceptable overall error. It can be pointed out that each node only broadcasts its partial error to other nodes at this time. All other calculations involve local data and can be made without synchronization. This drastically reduces the communications needs.

The application uses few and simple MPI standard primitives and can be portable for several machines.

2.1. Target Machines

Two machines with different architectures were available for this project: IBM RS/6000 SP with 12 nodes and SGI Origin 2000 with 128 nodes.

The IBM RS/6000 SP is a high performance machine with MIMD (Multiple Instruction Multiple Data) architecture connected via crossbar switch allowing direct connection between any two nodes. The IBM Parallel Environment (PE) allows the development and execution of parallel applications. It conforms to MPI (Message Passing Interface) standard for communication among parallel process running on various nodes [9] [10]. The application had been executed in this machine through job's submission strategy in exclusive mode, i.e., only one application executes in a group of nodes at one time. All processing nodes has homogeneous characteristics with same processing power, amount of memory and cache.

The SGI Origin 2000 is a DSM (Distributed Shared Memory) machine with cc-NUMA (cache-coherent Non-Uniform Memory Access) architecture interconnecting processing nodes through a network of hubs, the Craylink, via crossbar switch [11]. The DSM characteristic allows global memory addressing, but it can not preview the access memory time due to architecture. The cache-coherent protocol ensures the system cache consistency during execution of a program. The access available for the execution of the application in this environment was shared, i.e., all programs share all nodes during execution. It could be emphasized here that time tests were made with unbalanced work on nodes.

3. Application Data Set

After the application had been tested with traditional and well-known data sets, more complex data sets were used [12]. The case reported here had been achieved interesting results. The data set had been extracted from a real-world insurance data warehouse of a private company. The target machines utilized in this study don't have the support for automatically get data from data warehouse. The extraction procedure generates a table in a text file with 80 fields as attributes and 147478 registers as lines. The attributes specify customer behavior, insurance contracts and tariff components. Some information was classified as confidential. The main purpose of this problem is to find the best model that describes the attributes relationship, given a predefined classification.

The preprocessing phase of this study generates an transformed data set with attributes and registers that properly represents data to train and test the neural network. Attributes with discrete and continuous values were evaluated for columns and registers removal, using graphical and statistical visualizers. Some algorithms for column reduction had been used too The resulted data set had 64 attributes and 130143 registers. The configuration of the neural network suitable for modeling those data has 64 neurons on input layer, 136 neurons on hidden layer and 1 neuron on output layer for classification had been obtained through a combination of heuristic approach and a sequential genetic algorithm. The parallel neural network application reads the data set and splits it into training set with 70% of data and a testing set with 30%. The training set was partitioned and distributed among processing nodes before the learning phase of the algorithm starts. At the end of training phase the control node tests the obtained model with the testing set.

The Amdahl's Law coefficient for evaluation of the parallel application performance shows that 99,3% of the serial application could be parallelized. It is due to the higher time spent by the training phase with respect to the whole time consumption of the application. Table 1 presents the elapsed time spent in application execution for limited number of iterations with different number of nodes in each machine.

Table 1 – Time consumption (seconds)

Number of nodes	ORIGIN 2000	IBM SP
4	2254,38	2407,63
6	1539,32	1318,96
16	737,97	776,36
32	340,53	—
48	271,31	—
64	215,32	—
96	179,62	—
128	138,45	—

Figure 3 shows the speed-up obtained executing the application in the ORIGIN 2000.

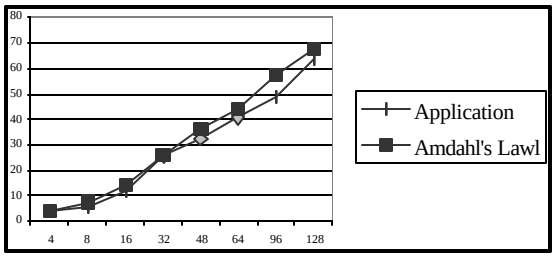

Figure 3 – ORIGIN 2000 Speed-up

Figure 4 graphically represents the number of millions of connections updated per second for the application running with different number of nodes

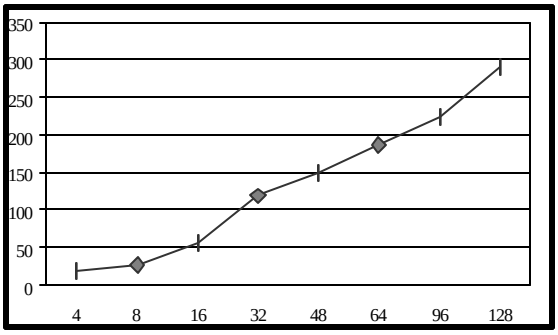

Figure 4 – ORIGIN 2000 MCUPs

4. Conclusions

The report present here shows the improvements obtained from previous work [13] on similar data but with better performance machines. The volume of data existing in real world data warehouse with complex relations can only be handled by methods implemented in high performance computers with efficient parallel methodology. The case used in this work has these characteristics and could only be handled by parallel application.

In latest year the developing of parallel data mining methods are encouraged due to announces of machines with higher performances. The use of a standard library of communication primitives allows portability of the application for several machines, bringing great vantages for the methodology. It can be point out that data preparation phase and data mining phase are both fundamental for the success of KDD process and .the use of semi-automatic and semi-automatic procedures could improve the quality of the overall process.

The neural network has the ability of produce a very precise classification model comparing with other paradigms used in classification problems. Despite the higher time computing consuming by the algorithm, the use of new approaches and some modifications of the procedures, like cross-validation, has to be considered for even better classification results.

References

[1] Fayyad, U.M. et al, *Advances in Knowledge Discovery and Data Mining*, MIT Press, California, 1996.

[2] Rumelhart, D.E., Hinton, G.E., Williams, R.J., Learning Internal Representations (Chapter 8). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol 1: Foundations*, 7ed, ed. Rumelhart, D.E., McClelland, J.L., MIT Press, Cambridge, pp.318-364, 1988.

[3] Gallant, S.I. *Neural Network Systems and Expert Systems*, MIT Press, Cambridge, 1994.

[4] Vogl, T.P.,et al, Accelerating the convergence of the Back-propagation Method, *Biological Cybernetics*, vo.5, n.3, pp.465-471, 1988.

[5] Van Ooyen, A., Nienhuis, B., Improving the Convergence of the Back-Propagation Algorithm, *Neural Networks*, v.7, n.1, pp.1-11, 1992.

[6] Fukuoka, Y., et al, A modified Back-Propagation Method to Avoid False Local Minima, *Neural Networks*, v.11, n.6, pp. 1059-1072, 1998.

[7] Haykin, S., *Fundamentals of Artificial Neural Networks*, MIT Press, USA, 1999.

[8] Sundararajan, N., Saratchandran, P., *Parallel Architectures for Artificial Neural Networks: Paradigms and Implementations*, IEEE Computer Society Press, California, 1998.

[9] IBM, *IBM Parallel Environment for AIX: Operation and Use, Vol. 1*, IBM, Denmark, 1996.

[10] Pacheco, P., *Parallel Programming with MPI*, Morgan Kaufmann Publishers, Inc., 1997.

[11] SGI, *Origin 2000 Deskside Owner's Guide*, SGI, USA, 1998.

[12] Costa, M.C.A., *Data Mining High Performance Computing using Neural Networks*, D.Sc. Thesis, COPPE/UFRJ, 1999 (in Portuguese).

[13] Lopes, M.C.S., Costa, M.C.A. & Ebecken, N.F.F., A comparison of methods for customer classification. *Proceedings of the International Conference on Data Mining*, ed. Ebecken, N.F.F., WIT Press, Great Britain, pp.333-347, 1998.

[14] Costa, M.C.A. & Ebecken, N.F.F., Data Mining High Performance Computing Using Neural Networks, *Proceedings of the High Performance Computing 2000*, ed. Brebbia, , WIT Press, USA, 2000