Neural Network Simulation with PVM

D.Anguita, S.Rovetta, M.Scapolla, R.Zunino
Department of Biophisical and Electronic Engineering
University of Genova
Via all'Opera Pia 11a
16145 Genova, ITALY
e-mail: ams@dibe.unige.it

INTRODUCTION

In this paper an implementation of the back propagation (BP) algorithm (Rumelhart86) on a cluster of workstations is presented. In a previous paper (Anguita94), we proposed an efficient implementation of BP on RISC architectures (Matrix Back-Propagation - MBP): now we extend that work to clusters of workstations using PVM.
Several methods have been proposed for the implementation of BP on distributed and parallel architectures. In Table 1 some of the implementations available in literature are summarized.
The speed is measured in MCUPS (Millions of Connections Updates Per Second). The three

Computer	Speed (MCUPS)	Ded.	Ref.
CNS-1 (1024)	166000	Y	Asanovic93
CNAPS (512)	2379	Y	Ienne93
SNAP (64)	302	Y	HNC94
CM-5 (32)	72	N	Adamo94
FUJITSU VP-2400/10	60	N	Sanchez94
CM-2 (64k)	40	N	Zhang90
Cray Y-MP (2)	40	N	Liu94
IBM RISC/6000	15	N	Anguita94
DEC Alpha	3.2	N	Mueller94
Transputer (16)	0.7	N	Petrowsky93
PC486	0.5	N	Mueller94

Table 1 –Distributed and parallel back propagation implementations

top computers are dedicated machines for neurocomputing applications, making use of massive parallelism to obtain high performances (the number of processors is indicated in parenthesis). This explains the large gap between general-purpose and dedicated computers.
Among the large amount of implementations available in literature, there are virtually no examples on clusters of workstations. The most interesting work in this field is (Chu92): the subject of the work is the study of the optimal mapping, with respect to the learning time, for a multi-layer feedforward network on message-passing multicomputers. The problem is obviously combinatorial and some heuristic must be introduced in order to solve it in reasonable time.
The novelty of our approach lies in developing a structured version of the back-propagation algorithm (MBP) that minimizes the learning time on a single machine and then extending it to the multiprocessor case (Distributed Matrix Back-Propagation - DMBP[1]).

A DISTRIBUTED IMPLEMENTATION ON A CLUSTER OF WORKSTATION

Our research focuses on a static mapping of the BP learning algorithm on a cluster of workstations; in other words, the load distribution is decided once at the beginning of the

[1] MBP and DMBP are available through anonymous ftp on risc6000.dibe.unige.it

computation. This allows us to model easily the available computational power, to compute in negligible time an optimal mapping and to forecast the behavior of the communication network. Further research could be done to analyze a dynamic mapping and our method can be used as a starting point for this work.

In order to model our problem, let us consider C identical workstations; each of one of them can achieve a computational speed V (measured in MFLOPS). This is quite a general assumption because, as shown in (Anguita94) it is possible to optimize the core operations of the backpropagation on a single machine in order to obtain a computational speed independent from the size of the problem. The workstations are connected by a network that can transmit data at the rate of 1/Tbit (measured in bit/sec) where Tbit is the transmission time of one bit. The speed of the network is guaranteed to be maintained constant for large enough packets thanks to PVM.

The neural network is a multi-layer feed-forward perceptron of L layers: each one composed of n_l neurons ($0 \le l \le L$). If we have n_p training patterns, we can define a matrix S_l $(n_p \times n_l)$ for each layer where the status of layer *l* for each training pattern is stored. The weights of layer *l* are stored in matrix W_l $(n_l \times n_{l-1})$.

We explored two techniques to split matrices S_l and W_l into sub-matrices and distribute them on the cluster. The first technique is called *by pattern*: each S_l is divided into *P* sub-matrices S_{lp} $(1 \le p \le P)$ of size $n'_p \times n_l$ where $n'_p = n_p / P$. The second technique is called *by net*: the *l*-th layer is divided into *N* groups of neurons so that matrix W_l resulting in sub-matrices W_{ln} $(1 \le n \le N)$ of size $d_l \times n_{l-1}$ where $d_l = n_l / N$. In the same way we obtain a sub-vector b_{ln} of size $d_l \times 1$ from the bias vector b_l of size $n_l \times 1$. If we assign to each workstation a fixed number of sub-matrices, our problem is to find N and P such that the total time for one complete epoch (computation+communication) is minimum. Obviously $N \cdot P \le C$. In Table 2 the steps of DMBP are reported: we assume that k_1 and k_2 are the number of operations needed to compute respectively the activation function and its derivative. Due to lack of space it is not possible to explain in detail the table.

The computation time T_{OP} is the sum of the number of operations (column II of the table) divided by V. The communication time T_C is the following:

$$T_C = \frac{D}{T_{bit}}\left[2(P-1)\sum_{l=1}^{L} n_l(n_{l-1}+1) + n_P(N-1)\left(2\sum_{l=2}^{L-1} n_l + n_l + n_L\right)\right]$$

where D is the size of the data to be transmitted (usually 32 or 64 bits).

Step	# of transm.	# of operations
Feed Forward: computation of S		$2n'_p(n_{l-1}+1)d_l + k_1 n'_p d_l$
Reconstruction of S	$n'_p d_l (N-1)$	
Error Back Propagation: computation of Δ		$(k_2+2)d_L n'_p$
Reconstruction of Δ	$n'_p d_l (N-1)$	$(k_2+1+2n_{l+1})d_l n'_p$
Partial Weight variation: computation of ΔW *and* Δb		$2d_l n'_p n_{l-1} + d_l n'_p$
Transmission of ΔW	$d_l(n_{l-1}+1)$ if p≠1	
Total Weight variation: sum of partial variations		$(P-1)d_l(n_{l-1}+1)$
Transmission of ΔW *and* Δb	$(P-1)d_l(n_{l-1}+1)$ if p=1	
Weight Updating		$4d_l(n_{l-1}+1)$

Table 2 – Calculation of execution time

We can circumvent the problem of a non homogeneous cluster, composed by workstations of different computational power, if we model each machine as a sub-cluster of identical *virtual* workstation whose computational power is the greatest common divisor of all the workstations in the cluster. Obviously, the communication time between virtual workstations that compose a single real workstation must be considered null.
DMBP was implemented on a cluster composed by an IBM RISC6000/550, an HP 9000/750 and an HP 9000/720. The measured sustained performance of the Ethernet with PVM software was about 3.5 Mbit/s.

EXPERIMENTAL RESULTS

To test the asyntotical behavior of our model we implemented a large parity-problem (500x500x1): this is a classic toy problem used in literature to test the learning performance of a network. In Fig. 1 the comparison of the distributed and serial version of the program is presented. The speed of the learning is measured in MCUPS (Millions of Connections Updates per Second).

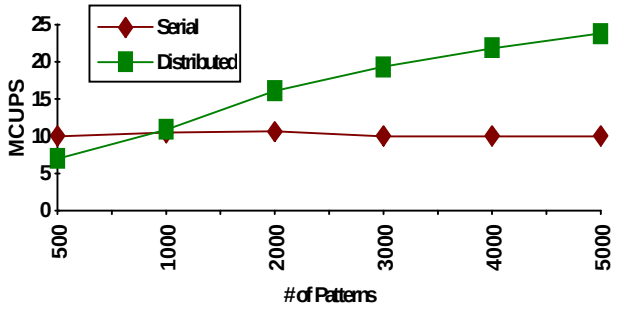

Fig. 1 – Learning speed of DMBP

Fig. 2 shows the difference between our mathematical model and the actual speed of DMBP. The model is pessimistic about the performance of DMBP if compared to the actual observed data: this derives from the assumption that the computation and communication phases cannot be overlapped. This assumption is, in our case, not true.

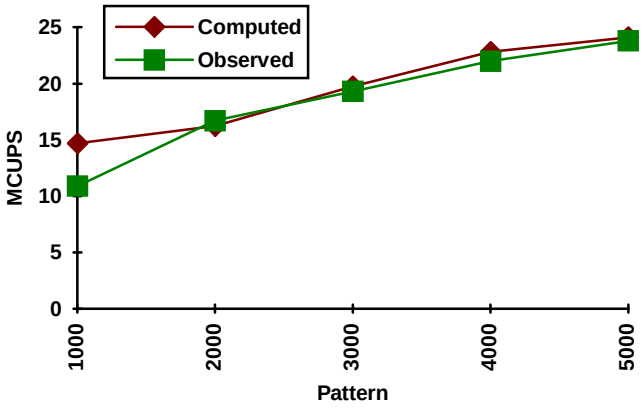

Fig. 2 – Fitness of the model

DMBP was used mainly for large learning experiments in the field of image compression with neural networks (Passaggio94). The network is an autoassociative MLP (256-16-256) with 2048 patterns. We measured 8.1 MCUPS compared to 3.7 MCUPS for the serial version. Furthermore, this application was unable to run in serial mode on the smallest machine due to main memory shortage, while it executed flawlessly on the cluster.

REFERENCES

J.M.Adamo, D.Anguita. *Object Oriented Design of a BP Neural Network Simulator and Implementation on the Connection Machine (CM-5)*. ICSI Tech. Rep. TR-94-046, Berkeley, CA, September 1994.

D.Anguita, G.Parodi, R.Zunino. *An Efficient Implementation of Back-Propagation on RISC-based workstations.* Neurocomputing 6, 1994.

K.Asanovic, J.Beck, T.Callahan, J.Feldman, B.Irissou, B.Kingsbury, P.Kohn, J.Lazzaro, N.Morgan, D.Stoutamire, J.Wawrzynek. *CNS-1 Architecture Specification*. ICSI Tech. Rep. TR-93-021, Berkeley, CA, April 1993.

L.C.Chu, B.W.Wah. *Optimal mapping of neural networks learning on message passing multicomputers*. Journal of Parallel and Distributed Processing, No. 14, 1992.

HNC. *SNAP-SIMD Numerical Array Processor*. HNC, 5930 Cornerstone Court West, S.Diego, CA, 1994.

P.Ienne. *Architectures of Neurocomputers: Review and Performance Evaluation*. Tech. Rep. 93/21, Swiss Federal Inst. Of Technology, Lausanne, January 1993.

X.Liu, G.L.Wilcox. *Benchmarking of the CM-5 and the Cray Machines with a Very Large Backpropagation Neural Network*. Proc. of the IEEE Int. Conf. On NN, June 28-July 2, 1994, Orlando, FL.

U.A.Mueller, M.Kocheisen, A.Gunzinger. *High-Performance Neural Net Simulation on a Multiprocessor System with Intelligent Communication*. Advances in Neural Information Processing Systems 6, J.D.Cowan, G.Tesauro, J.Alspector (Eds.), 1994.

F.Passaggio, R.Zunino, D.Anguita. *Human Visual System Neural Approach to Image Compression*. Proc. of WCNN, June 5-9, 1994, S.Diego, CA.

A.Petrowsky, G.Dreyfus. *Performance Analysis of a Pipelined Backpropagation Parallel Algorithm*. IEEE Trans. on Neural Network, Vol.4, No.6, Nov. 1993.

E.Sanchez, S.Barro, C.V.Regueiro. *Artificial Neural Networks Implementation on Vectorial Supercomputers*. Proc. of IEEE Int. Conf. On NN, June 28-July 2, 1994, Orlando, FL.

X.Zhang, M.Mckenna, J.P.Mesirov, D.L.Waltz. *An Efficient Implementation of the Back-Propagation Algorithm on the Connection Machine CM-2*. Advances in Neural Information Processing Systems 2, D.S.Touretzky (Ed.), 1990.