Modeling of Feedforward Neural Network in PAHRA Architecture

LIBERIOS VOKOROKOS, NORBERT ÁDÁM
Department of Computers and Informatics
Technical University of Košice
Letná 9, 042 Košice
SLOVAK REPUBLIC
liberios.vokorokos@tuke.sk http://hornad.fei.tuke.sk/kpi/person/vokorokos/dcicard.php
norbert.adam@tuke.sk http://hornad.fei.tuke.sk/kpi/person/adam/dcicard.php

Abstract: - One of the most popular neural networks are multilayered feedforward neural networks, which represent the most standard configuration of biological inspired mathematical models of simplified neural system. These networks represent massive parallel systems with a high number of simple process elements and therefore it is natural to try to implement this kind of systems on parallel computer architecture. The parallel architecture described in this article provides flexible platform for simulation of multilayered feedforward neural networks trained with back-propagation algorithm. The computation model of given architecture allows formally describe components of parallel implementation of neural network and provides mathematical tool for verification of system performance.

Key-Words: - multilayered feedforward neural network, parallel computer, PAHRA, processing element, computational model

1 Introduction

Computer systems, with the ability of emulation of human intelligence and accuracy with help of group of sensors, processors and actuators, which are working reliably and are able to predict and to react on unpredictable situations, where decisions are, in some cases, based on incomplete or inaccurate information, in the way, to solve the problem and to get accurate solution, are called intelligent systems. Those systems were in the center of interest from the beginning of computers era and found their application in the control of the systems with no human control or in the vital military and commercial systems, for example medical devices, atom piles, aeronautics or autonomous intelligent robots. Learning of those systems improves its performance.

Process of learning is not passive. It is iterative, dynamic process, oriented to the solution of specified problem. Problem is described with rules, beginning state and desired state. Solution of the problem is represented with sequence of rules, where original state is changing sequentially to the desired state. To find solution of the problem means to find appropriate sequence of rules. In cases, we do not know rules to model solution of the situation, or those rules are very complicated, or incomplete, one of usable solution is to use artificial neural networks (neural networks).

One of the most popular neural networks are multilayered feedforward neural networks (FFNN) with error back-propagation algorithm (BP algorithm), which represent the most standard configuration of biological inspired mathematical models of simplified neural system. These networks represent massive parallel systems with a high number of simple process elements and therefore it is natural to try to implement this kind of systems on parallel computer architecture [2], [3], [9].

Exploitation of inherent parallelism in neural networks [4], [8] concept of pipelined processing of patterns in network learning phase, parallel implementation of training algorithm focused on neural networks with supervised learning described in the following chapters of this article is a part of research of Parallel Hybrid Ring Architecture designed for neural network implementation [2] developed on Department of Computers and Informatics of Faculty of Electrical Engineering and Informatics at Technical University of Košice, Slovakia.

2 The Parallel Hybrid Ring Architecture

The Parallel Hybrid Ring Architecture (PAHRA) architecture is developed on DCI FEEI TU of Košice within the frame of projects [3] and [5] and is based on the conception of multiprocessor architectures. The PAHRA architecture is assigned primary to identification and analysis of security threats in architectures of distributed computer systems and dynamic networks using multilayered feedforward neural networks.
This conception of PAHRA (Fig. 1) consists of defined number of processing elements (n) with the synchronizing simplex interconnection bus (type ring), control unit and interconnection bus which provides access to the memory (storage) devices.

Each processing element has its own execution unit (or execution units, in the case of complex systems located in node of synchronizing interconnection bus), local memory and set of communication lines to be able to communicate with other processing elements. The raw computational power of each processing element is characterized by the execution time of one elementary operation (arithmetical or logical). The synchronizing interconnection bus uses point-to-point method of connection with the synchronous communication.

The use of simplex communication and the overlay of computational and communicational phases disallow all-to-all broadcasting which is characteristic for node parallelism (neuron and synapse parallelism) with vertical segmentation [6].

The time taken to inter-processor communication for the transmission of m words between adjacent processing elements is defined by

\[
t_{\text{com}} = t_{\text{com init}} + m \cdot t_{\text{word}}
\]

where \( t_{\text{com}} \) is the time taken to transmission of m words, \( t_{\text{com init}} \) is the transmission initialization time and \( t_{\text{word}} \) is the time taken to transmission of one word.

Structural organization of processing element is proposed as a dynamic multifunction system [7] consisting of the two pipeline pre-processing units (Fig. 2), of the execution unit and the communication unit. Pre-processing units deal with a separation of forward and backward network learning phase with BP algorithm. Execution unit deals with execution of the given instruction and the task of communication unit is to forward the result to the interconnection network.

Conventional formulation of basic neural algorithms make implementation of neural networks on existing parallel hardware more difficult, therefore the research goal was to find an optimal decomposition scheme of multilayer feedforward neural net with BP learning algorithm to individual processing elements of the PAHRA and to explore possibilities of pipeline pattern processing.

The concept of feedforward neural network and mapping of respective layer neurons is proposed in the figure 3.

### 3 Mapping of FFNN to PAHRA

The error backpropagation algorithm fall into the class of training algorithms for supervised learning. The goal of the algorithm is to find such a set of weights (network parameters) that for the given input (for the given pattern \( U^0 \)) the network will provide required output \( D \). The algorithm is divided into three phases, a forward phase, an error backpropagation phase and a phase of weight updating. A number of neurons in the network is referred to as symbol \( N \) and in individual layers as \( N^i \) (\( i = 0,1,...,L \)), while \( N^0 \) is a number of neurons in the input layer and \( N^L \) is a number of neurons in the output layer.

Characteristics of the individual phases and their computation costs (time complexity) for a multilayer feedforward neural network with BP learning algorithm is presented below:

**Forward phase** – For the given pattern \( U^0 \) from the training set, the activation value of neurons from hidden layer and output layer is calculated using this formula:

\[
o^l_k = \varphi \left( \sum_{j=1}^{N^l} w_{kj} o^{l-1}_j + \theta^l_k \right), \quad k = 1,2,...,N^l, \quad l = 1,2,...,L-1
\]

Where \( \varphi(.) \) is a bipolar sigmoidal function. Let \( t_{\text{add}}, t_{\text{mul}} \) and \( t_{\text{af}} \) to be parameters determining a time required for the the operations of addition, multiplication and calculation of activating value. By introduction of the following substitution relations:

\[
M_z = t_{\text{mul}} + t_{\text{add}}, \quad t_{\text{mul}} = \alpha M_z, \quad t_{\text{af}} = \beta M_z
\]

the computation cost of the forward phase (\( c_1 \)) is equal to

\[
c_1 = M_z \left[ N^0 N^1 + \sum_{i=2}^{L-1} (N^{i-1}N^i) + \beta \sum_{i=2}^{L-1} N^i \right]
\]

**Error backpropagation phase** – In this phase using an error function a generalized error signal \( \delta^l_k \) (\( l = 1, 2,...,L-1 \)) is calculated for the kth neuron of the given layer. For the k-th neuron in the output layer a generalized error signal \( \delta^L_k \) is calculated using the formula

\[
\delta^L_k = (D_k - a_k) o^L_k
\]

\[
\delta^l_k = \alpha \varphi'(o^l_k) \sum_{j=1}^{N^l} w_{kj} \delta^l_j
\]
\[ \delta^l_i = \left( o^l_i - d^l_i \right) \frac{\partial \varphi}{\partial \left( \sum_{j=1}^{N^{l-1}} (w^l_{ij} \cdot o^{l-1}_j) + \theta^l_i \right)} \]  

(5)

The generalized error signal \( \delta^l_i \) (\( i = 1, 2, \ldots, L-1 \)) for the \( i \)-th neuron in the hidden layer is determined by the formula

\[ \delta^l_i = \left( \frac{\partial \varphi}{\partial \left( \sum_{j=1}^{N^{l-1}} (w^l_{ij} \cdot o^{l-1}_j) + \theta^l_i \right)} \right) \sum_{j=1}^{N^{l-1}} w^l_{ij} \delta^l_j, \quad i = 1, 2, \ldots, N^{l-1} \]  

(6)

The cost of the error backpropagation phase is

\[ c_2 = M_a \left[ N^l + \sum_{i=2}^{L-1} (N^{l-1}N^l) + \alpha \sum_{i=1}^{L} N^l \right] \]  

(7)

**Weight updating phase** – In this phase an update of weights is performed based on \( \delta^l_i \), a \( \delta^{l-1}_i \).

\[ W^l_{ij} = W^l_{ij} + \eta \delta^l_i \delta^{l-1}_j, \quad l = 1, 2, \ldots, L \]  

(8)

where \( \eta \) is a learning speed. Calculation cost of this phase is expressed as:

\[ c_3 = M_a \left[ N^0N^l + \sum_{i=2}^{L-1} (N^{l-1}N^l) \right] \]  

(9)

Using (4), (7) and (9) the total cost of neural network training for one pattern from the training set in case of calculation on one computing element is given as:

\[ C_{net} = c_1 + c_2 + c_3 = M_a \left[ 2N^0N^l + 3 \sum_{i=2}^{L-1} (N^{l-1}N^l) + (\alpha + \beta) \sum_{i=1}^{L} (N^l) + N^l(1 + \beta) \right] \]  

(10)

### 3.1 Algorithmic mapping

The proposed mapping scheme uses parallelism on the node level as well as the level of synaptic weights. In case of neural level parallelism all of the neuron input synapses, in regard of processing element to which a neuron was assigned, are allocated on the given processing element. In case of synaptic interconnection parallelism only the output synapses are allocated on the given processor element to which the corresponding neuron was assigned. The applied decomposition scheme assumes FFNN and utilizes neuron level parallelism in case of a hidden layer. The weights are calculated in parallel using synaptic level parallelism. According to the figure 3, the hidden layers \( l = 1, \ldots, L-2 \) are mapped one to one on coordinating processors and neurons in the last hidden layer are classified into \( p = P - (L-2) \) classes – in every class there are \( N^l / p \) neurons – where \( P \) is a number of system processing elements referred to as \( PE_i, i = 0, 1, \ldots, P-1 \).

**Forward phase** – In this phase an output of neuron mapped on the given \( PE_i \) is calculated. For the given pattern from the training set the output of neurons on hidden layers is calculated using (2). For the output calculation of neurons in the output layer there are outputs and weight values calculated on other \( CP_j \) required. Therefore the calculation cost of this phase will consist of two parts. The first part is represented by calculation of activation value and consequently the calculation of output of individual neurons and the second phase is determined by communication time (transfer of required data) among coordinating processors \( CP_k \) (\( k = L-2, L-1, \ldots, P-1 \)):

\[ c^{l=2}_1 = M_a \left[ N^0 + N^0 + \beta \right] + T_{com} \]  

(11)

**Error backpropagation phase** – In this phase every node calculates generalized error signal \( \delta^l_i \) (\( i = 1, 2, \ldots, L \)) regarding the local neuron. The time complexity of error backpropagation phase for \( L = 2 \) and \( L > 2 \)
Weight update phase – In this phase an update of local neurons weights associated to PEi is performed. For the weight update between ith input neuron and jth hidden neuron, there is required to know the output of ith input neuron and the value of error signal \( \delta_j \) for the jth hidden neuron. As the input and output synapse is represented on every coordinating processor, required values are stored locally in system coordinating processor and therefore \( c_5 \) does not contain a communication component:

\[
c_5^{l-2} = M_a \left[ \frac{N^l}{P} \left( (N^2 + \alpha) + N^2 \right) \right] ,
\]

\[
c_5^{l-2} = M_a \left[ \frac{N^l}{P} \left( (N^2 + \alpha) \right) \right] + T_{com}^F
\]

\[
C_{par}^{l-2} = c_5^{l-2} + c_5^{l-2} + c_5^{l-2} =
\]

\[
= M_a \left[ \frac{N^l}{P} \left( (2N_0 + 3N^2 + \alpha + \beta) \right) + N^2 (1 + \beta) \right] + T_{com}^F
\]

\[
C_{par}^{l-2} = c_5^{l-2} + c_5^{l-2} + c_5^{l-2} =
\]

\[
= M_a \left[ \frac{N^l-1}{P} \left[ 3 \left( N^{l-2} + N^l \right) + (\alpha + \beta) \right] \right] + T_{com}^F
\]

In the proposed mapping scheme all values associated with the given local processing element are packed and forwarded to interconnection network as the one message. Let \( c_{5,5} \) be a transfer cost of the one scalar parameter (e.g, one \( o_i \)) to \( r \) receivers on parallel computer. Let \( c_{5,5} \) be a transfer cost of one vector (acquired by packing of \( N^l / m \) \( o_i \) outputs, where \( m \) is a number of coordinating processors controlling the calculation in the given layer) to \( r \) receivers. Let us assume that arbitrarily many messages can be sent in parallel, but one processor can send and receive only one message in a time. Sending vector of the length \( v \) on common parallel computer is more efficient then sending of scalars, therefore we can assume that \( c_{5,5} \ll v c_{5,5} \).

Then a communication time for message forwarding of D size is determined by formula

\[
t_{com} = m \left( T_{mi} + T_c \mu(D) \right)
\]

where \( T_{mi} \) is a time required for communication initialisation, \( T_c \) is a time of one vector sending and \( \mu(D) \) is a scaling/packing of scalar parameters into one message. Then the communication time can be expressed by formula

\[
t_{com} = m \left[ T_{mi} + T_c \mu\left( \frac{N^l}{m} \right) \right]
\]

Let \( T_{mi} = \gamma M_a \), \( T_c = \delta M_a \mu(\frac{N^l}{m}) = \varepsilon M_a \) then

\[
t_{com} = m R_{com} M_a
\]

By substitution of (17) to (14a) we obtain:

\[
C_{par}^{l-2} = c_5^{l-2} + c_5^{l-2} + c_5^{l-2} =
\]

\[
= M_a \left[ \frac{N^l}{P} \left( 2N^0 + 3N^2 + \alpha + \beta \right) + N^2 (1 + \beta) \right] + T_{com}^F
\]

By substitution of (17) to (14b), while for the forward phase \( m = P \) is applied and for the backward phase \( m = P - p \), we obtain:

\[
C_{par}^{l-2} = c_5^{l-2} + c_5^{l-2} + c_5^{l-2} =
\]

\[
= M_a \left[ \frac{N^l-1}{P} \left[ 3 \left( N^{l-2} + N^l \right) + (\alpha + \beta) \right] \right]
\]

Above mentioned formulas were derived for non-pipeline pattern processing. By using of pre-processing unit for forward (characterized by \( c_1 \) parameter) and pre-processing unit for backward learning phase (parameters \( c_2 \) and \( c_3 \)) from the derived formulas for parallel execution of network learning the \( c_1 \) parameter is eliminated, because pre-processing units work in parallel. The calculation cost of pipeline processing for FFNN with number of layers \( L \geq 2 \):
From (20) there is clear that a calculation cost decreases and a communication cost increases by addition of coordinating processors into a system. According to this it is evident that there is an optimal number of processing elements using which a total cost will be minimal. The optimal number of PEs will be determined by solving of equation

$$\frac{\partial C_{\text{par}}}{\partial P} = 0$$

(21)

According to this for the optimal number of PEs will be applicable that:

$$P_{\text{opt}}^{l-2} = \left( \frac{N^0 \left( N^0 + 2N^1 + \alpha + \beta \right)}{A_{\text{com}}} \right)^{\frac{1}{l-2}}$$

$$P_{\text{opt}}^{l-2} = \left( \frac{N^L \left( 2 N^{l-2} + N^l \right) + (\alpha + \beta)}{A_{\text{com}}} \right)^{\frac{1}{l-2}} + (L - 2)$$

(22)

In the proposed mapping scheme the maximum number of PEs is determined by number of neurons in the last hidden layer.

Then the acceleration of learning process in PAHRA (number of PE is P) in relation to sequential execution of BP algorithm (P = 1) is determined as:

$$S_{\text{seq/par pipe}} = \frac{C_{\text{seq}}}{C_{\text{par/pipe}}}$$

(23)

In case that $\sum_{i=1}^{l-1} N^i \not< P$ then S will converge to P.

Because, the PAHRA is being developed for the verification of proposed computational model was used a software tool, which emulates the PAHRA processing elements. The model of simulation environment does not consider a time aspect of processing elements communication to other system components.

Two experiments were performed during simulation with simulation parameters listed in tables 1 and 2. The results of simulation are presented in the figures 4, 5 and 6, 7.

As a number of neurons increases in the hidden layer, the calculation time extends (Fig. 4 and 5). The growth rate for sequential algorithm (Seq a {Seq+Pipe}) is significantly higher then for the parallel algorithm (Par and {Par+Pipe}). The results of experiment 2 (Fig. 6 and 7) proofs that by incrementation of processing element number the execution time is decreasing in case of parallel algorithm (Par a {Par+Pipe}). Acceleration of parallel calculation compared with simulation no. 2 for FFNN with one hidden layer is presented in the figure 6 and acceleration for FFNN with three layers is presented on Fig. 7.

### Table 1 Simulation parameters

<table>
<thead>
<tr>
<th>Neural Network Topology</th>
<th>No. of processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_0 \times N_1 \times N_2 \times N_3 \times N_4$</td>
<td>(P)</td>
</tr>
<tr>
<td>160</td>
<td>160</td>
</tr>
<tr>
<td>160</td>
<td>172</td>
</tr>
<tr>
<td>160</td>
<td>184</td>
</tr>
<tr>
<td>160</td>
<td>196</td>
</tr>
<tr>
<td>160</td>
<td>200</td>
</tr>
</tbody>
</table>

### Table 2 Simulation parameters

<table>
<thead>
<tr>
<th>Neural Network Topology</th>
<th>No. of processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_0 \times N_1 \times N_2 \times N_3 \times N_4$</td>
<td>(P)</td>
</tr>
<tr>
<td>64</td>
<td>50</td>
</tr>
<tr>
<td>64</td>
<td>50</td>
</tr>
<tr>
<td>64</td>
<td>50</td>
</tr>
<tr>
<td>64</td>
<td>50</td>
</tr>
</tbody>
</table>

5 Conclusion

Conventional algorithms of neural network learning are formulated for sequential execution model that makes an implementation of neural networks to the massively parallel system environment more difficult. A basic model of artificial neural network has several attributes, which require massively parallel implementation. These attributes includes high-parallel operations, elementary execution units (neuron), a local memory for neuron with a small capacity (shared memory) and an error tolerance of neuron interconnection. By analysis of the common formulation of backpropagation algorithm there was discovered that a common formulation of algorithm provides maximal effectiveness for parallel implementation of neural network on PAHRA with a number of processing elements $P_{\text{opt}}$. Efficiency of the new formulation is expressed by the calculation price for parallel implementation of algorithm comparing to...
sequential algorithm. Performed simulations showed that pipeline processing of patterns on PAHRA provides maximizing of computation performance with minimal hardware costs.

Acknowledgement
This work was supported by the Slovak Research and Development Agency under the contract No. APVV-0073-07 and by Agency of the Ministry of Education of the Slovak Republic for the Structural Funds of the EU under the project Centre of Information and Communication Technologies for Knowledge Systems (project number: 26220120020).

References: