APPROACH TO PARALLEL TRAINING OF INTEGRATION HISTORICAL DATA NEURAL NETWORKS

V. TURCHENKO[1], C. TRIKI[2], A. SACHENKO[1]

[1]Laboratory of Automated System and Networks, Institute of Computer Information Technologies, Ternopil Academy of National Economy, 3 Peremoga Square, 46004, Ternopil, UKRAINE, Phone: +380 (352) 33-0830, Fax: +380 (352) 33-0024, E-Mail: vtu@tanet.edu.te.ua

[2] Department of Electronics, Informatics and Systems, Parallel Computing Laboratory, 87036 Rende - Cosenza, ITALY, Fax: +39 984 494713, E-mail: chefi@parcolab.unical.it

ABSTRACT

There is considered the application of neural networks for accuracy improvement of sensor signal processing by sensor drift prediction. The two methods of sensor drift prediction are described. The main characteristics and training time of integrating historical data neural network are presented on the uniprocessor computer Pentium-III-600-128 Mb RAM. The two paralleling schemes of training of integrating historical data neural network are proposed. The fulfilled experimental researches of parallel programs on the high-performance computer Origin2000 are shown high efficiency of second paralleling scheme.

KEYWORDS

Neural network, parallel training, historical data, high-performance computing.

1. INTRODUCTION

It is shown in [1, 2], that error of the modern sensor data processing systems is much less than initial sensor error in majority of cases. Besides the sensor drift is much higher than drift of the other components of measuring channel. It is possible to improve the accuracy of physical quantities measurement by sensor calibration using special calibrator or sensor's periodic testing with reference sensor on the exploitation place [3]. But the operations implementing these methods are rather laborious. The low laborious is provided by sensor drift prediction during interesting interval [1]. The most effective is using artificial intelligence methods, in particularly neural networks (NN) for this purpose [4, 5]. Thus the calibration and testing results serves as input data for training of predicting neural network (PNN) and making of the prediction. But the laborious of calibration and testing provides small quantity of the data (3-5 values) in training sample of PNN. The two methods of artificial increasing of PNN's training sample are proposed in [6]: (i) method of additional approximating neural network (ANN) and (ii) method of historical data integrating. The modern systems of sensor signals processing have multi-level structure as a rule [7-9]. Therefore execution of proposed methods fulfills on the higher computing component of such network, i.e. on the personal computer [10]. The uniprocessor personal computer (in majority of industrial solutions) cannot provide training of a several kinds of neural networks per channel in multi-channel systems [11]. The conducted experimental researches have shown, that the time of NN training can be hundreds minutes [12], that is not acceptable in real-time systems. Therefore the approach to parallel NN training on the higher level of sensor data processing systems is considered below.

2. METHOD OF ADDITIONAL NEURAL NETWORK

The main goal of additional ANN using is the increasing of training set of PNN in order to provide high-quality training. Let us consider the set of sensor calibration points $x(t)$ on the interval $t = \overline{1,n}$ (Fig. 1). The sensor drift curve, which penetrates these points, is divided on $n-1$ subintervals according to number of calibrations. The ANN task is to approximate q points into each $n-1$ subinterval. The total points for PNN training is equal $m = (n-1) \cdot q$ after approximation procedure. The PNN task is the high-quality prediction for necessary steps in future using results of ANN approximation.

3. NEURAL BASED METHOD OF HISTORICAL DATA INTEGRATION

The disadvantage of considered above ANN method is the necessity of executing of sufficient calibration/testing procedures in order to obtain the enough (for example, 3-5) sensor drift data for ANN training. But these data are not available at the beginning of sensor exploitation. Therefore it is expedient to use historical data. The term «historical» data is used for describes the results of calibration or testing of the same type sensors in the similar operating conditions.

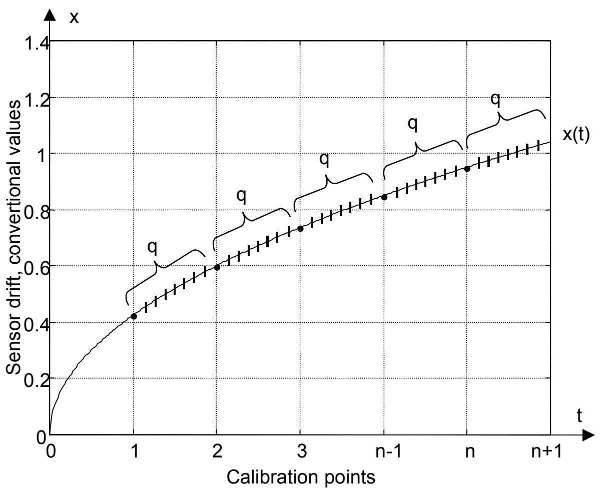

Fig. 1. Graphic Interpretation of Additional Approximation Neural Network Method

Let us consider the historical data of sensor drift as curves $x1...xn$ (Fig. 2), which are equal to xai, xbi, xci, $i=\overline{1,n}$ into calibration points a,b,c. The $i=\overline{1,n}$ is the quantity of calibrated sensors in previous moments of time. The first calibration of the new $k-$sensor allows correcting initial sensor error at the moment 0. The second calibration of the new sensor allows receiving the first real value xak of sensor drift in calibration point a. The main idea of the method is prediction of xbk point on the basis of xak and xai, $i=\overline{1,n}$, the next point xck on the basis of xbk and xbi, $i=\overline{1,n}$, etc. Thus the points xak and xai, $i=\overline{1,n}$ will be considered below as "window" of historical data. The number of available historical curves n of sensor drift determines structure of IHDNN's input layer.

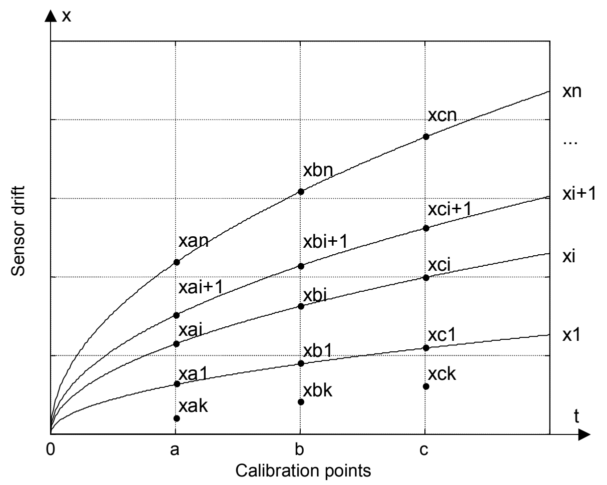

Fig. 2. "Historical" Data about Drift of Similar Sensors

It is necessary to generate the IHDNN's training set by the following algorithm:

1. One curve of sensor drift xi is considered as real data and all other curves xj, $j=\overline{1,i-1}$, $j=\overline{i+1,n}$ are considered as historical data. Thus, the real sensor drift xai is obtained in point a and xbi value is obtained in point b;
2. To calculate an absolute deviations $\Delta ij = |xai - xaj|$ of point xai from all other points xaj, where $i=\overline{1,n}$, $j=\overline{1,i-1}$, $j=\overline{i+1,n}$;
3. To sort all absolute deviations, calculated in the previous step, in decreasing order; to calculate maximum $\Delta ij^{\max} = \max \Delta ij$ and minimum $\Delta ij^{\min} = \min \Delta ij$ values of absolute deviations;
4. To generate each training vector as set of values xbi, xai, xaj, where xaj, $j=\overline{1,i-1}$, $j=\overline{i+1,n}$ values are necessary to put into training vector according to sorted (in decreasing order) values of absolute deviations xaj from value xai (Table 1);
5. To repeat steps 1-4 above for $i=\overline{1,n}$.

Table 1. Internal Structure of Training Vector of IHDNN

Max. value	Interm. values	Min. value	Drift in a	Drift in b
xaj, $\Delta_{ij} = \Delta_{ij}^{\max}$	. . .	xaj, $\Delta_{ij} = \Delta_{ij}^{\min}$	xai	xbi

The multiplayer perceptron with logistic activation function of hidden neurons and one linear output neuron is used as IHDNN [6]. The output value of perceptron is

$$y_O = F_O\left(\sum_{j=1}^{M} w_{jO} \cdot F_h\left(\sum_{i=1}^{N} x_i w_{ij} - T_{hj}\right) - T_O\right), \quad (1)$$

where $F_O(x)$ and $F_h(x)$ are the activation functions of output and hidden neurons respectively; w_{jO} are the weight coefficients of output neurons; x_i are the input data of neural network; w_{ij} are the weights coefficients of the hidden layer neurons; T_{hj} are the thresholds of the hidden layer neurons; T_O is threshold of output neuron. The logistic activation function is used for neurons of the hidden layer

$$F_h(x) = \left(1 + e^{-x}\right)^{-1}. \quad (2)$$

The back propagation algorithm [13] is used for IHDNN training with calculation of adaptive learning rate on each training step [14]

$$\alpha(t) = \frac{4}{\left(1+(x_i^p)^2\right)} \times \frac{\sum_{j=1}^{M}(\gamma_j^p(t))^2 h_j^p(t)(1-h_j^p(t))}{\left(\sum_{j=1}^{M}(\gamma_j^p(t))^2 (h_j^p(t))^2 (1-h_j^p(t))^2\right)}, \quad (3)$$

where $\gamma_j^p(t) = \sum_{j=1}^{M} \gamma_0^p(t) w_{j0}(t) h_j^p(t)(1-h_j^p(t))$ is the error of j-neuron of the hidden layer with logistic activation function, $h_i^p(t)$ is the output value of j-neuron of the hidden layer in the moment t for training vector p.

After IHDNN training the vector $xai+1, xai, xa1, xak$ of data integration is formed (according to Table 1), the value xbk should be the result of integration. Then "window" of historical data is moved to the right for one calibration point, and for the next iteration we will receive xck value and so on. Therefore the number of IHDNNs equals to quantity of calibration points. After obtaining of all integration results xbk, xck and so on these data are passed to ANN input, which approximates them and forms training sample of PNN. Method of historical data integration sequentially uses three NN with different properties (IHDNN, ANN and PNN).

The experimental researches of proposed method of sensor drift prediction were executed by simulation modeling using mathematical models of sensor drift “with saturation" and “with acceleration" [15]. The analysis of computational complexity and training time of used neural networks has shown [11], that IHDNN training has the largest computational complexity. Therefore let us consider this experiment in details. It is necessary to have some sensor drift curves for fulfillment of the experiment. Let us take the input data for experiment: 10 curves of historical data and 5 calibration points. As it is shown above, one historical curve is considered as real curve during forming of the training set of IHDNN. So the IHDNN’s training sample is formed from 9 curves. Therefore quantity of input neurons of IHDNN is equal 9. The separate training sample of IHDNN is formed for each investigated curve. Therefore, the quantity of used IHDNNs is equal 10 for one calibration point. The total number of IHDNNs is equal 50.

The experimental researches were executed using uniprocessor computer P-III/600 MHz/128 Mb RAM with Linux operating system. The 98% of CPU time was free before the beginning of experiments. The IHDNN’s model of three-level perceptron, namely the 9 neurons of input layer, the 9 neurons of hidden layer with logistic activation function and one output linear neuron (described by the formulas 1-3) are used for the experiments. The researches are fulfilled for the fourth scenarios (Table 2) of iteration number (IT) and sum squared errors (SSE), where time of fifty IHDNNs is shown in column 4.

The percentage error of historical data integration did not exceed 16% for sensor drift "with saturation" and 12% for sensor drift "with acceleration" in calibration points for the fourth scenario. The percentage error of PNN prediction did not exceed 30% for both sensor drifts on increased in 12 times intercalibration interval for the fourth scenario. This prediction result allows increasing the accuracy of sensor data processing in 3 times. However such large time of training (column 4, Table 2) only for one data acquisition channel is unacceptable for real-time systems. Therefore let us consider the approaches of parallel IHDNNs training on the multiprocessor computer below.

Table 2. Training Time of 50 IHDNNs on the Uniprocessor Computer

Scenario	IT	SSE	Time, min
1	40000	0,00001	39,8
2	80000	0,00001	76,2
3	200000	0,000001	138,9
4	500000	0,0000001	206,2

3. THE WAYS OF PARALLEL TRAINING OF INTEGRATING HISTORICAL DATA NEURAL NETWORK

The preliminary analysis of routine structure, which implements historical data integration has shown, that two schemes of paralleling are possible on the multiprocessor computer:

A. Parallel Calculation of Weighed Sum and Multiplication Cycles

The back propagation algorithm for IHDNN training contains about 24 operators of calculation of the weighed sum and multiplication cycles:

$$y = \sum_{i=1}^{N} w_i x_i - T , \quad (4)$$

where w_i are the weight coefficient of neuron’s inputs, x_i are the input data, T is the neuron’s threshold. The relations between x_i and w_i in all cases are absent between corresponding pairs. The quantity of executed operations of the weighted sum depends on quantity of neurons in each neural network layer. For example, for three-level perceptron with 9 neurons of input layer and 9 neurons of the hidden layer the quantity of executed operations is equal 663. Therefore, it is expedient to execute M multiplications on M available processors of the multi-processor computer (Fig. 3). The quantity of neurons in neural network layers must be large enough and also the calculation time must be larger than communication time between processors with the purpose of reduction of temporary losses at parallel calculation.

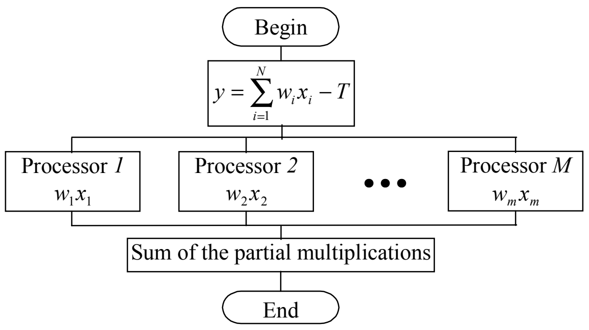

Fig. 3. General algorithm of parallel weighted sum calculation

B. Parallel training of each separate IHDNN

Let is present *N* IHDNN implementations and *M* processors of the multiprocessor computer. It is obviously, that the second way of paralleling can be presented as parallel calculation of all IHDNNs on *M* processors. Thus the parallel program is necessary to implement as two-level cycle structure (Fig. 4). All implementations of IHDNNs do not depend from each other. They will use identical algorithm for formation of training set using another calibration points.

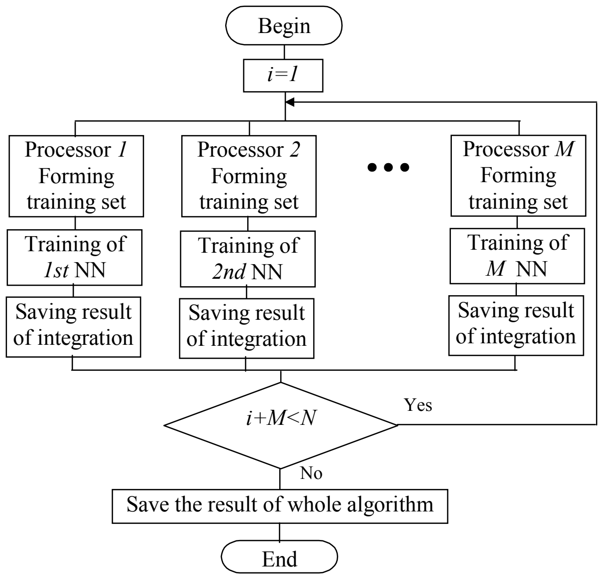

Fig. 4. General algorithm of parallel training of IHDNNs

4. EXPERIMENTAL RESEARCHES

The parallel computer Origin2000 is used for the experiments, which placed in Parallel Computing Laboratory, University of Calabria, Italy (origin.parcolab.unical.it). Computer Origin2000 contains 8 RISC-processors MIPS R10000 with clock rate 250MHz and 512 MB of the RAM. Each processor has 4 MB cache memory. Origin2000 has operating system UNIX (IRIX). All parallel routines are developed using MPI technology.

A. Parallel Calculation of Weighed Sum and Multiplication Cycles

The researches of this paralleling scheme were conducted using the test parallel routine, which executes $5 \cdot 10^6$ operations of weighed sum (4). It is used floating point operands of multiplying operation in (4). The execution time of the routine increases with increasing of quantity of used processors (Fig. 5). Therefore this paralleling scheme is not effective for this task. Because the maximum quantity of weighed sum operations in the IHDNN training routine equals 663 (see section 3A), that it is much less than in the test routine. This paralleling scheme is possible to use for the tasks, which have much greater number of multiplying operations. In another case it is possible to use a smaller quantity of processors, for example 2 or 4. In any case, it is necessary to research this task in future on the lower level of the Origin2000 parallelisation.

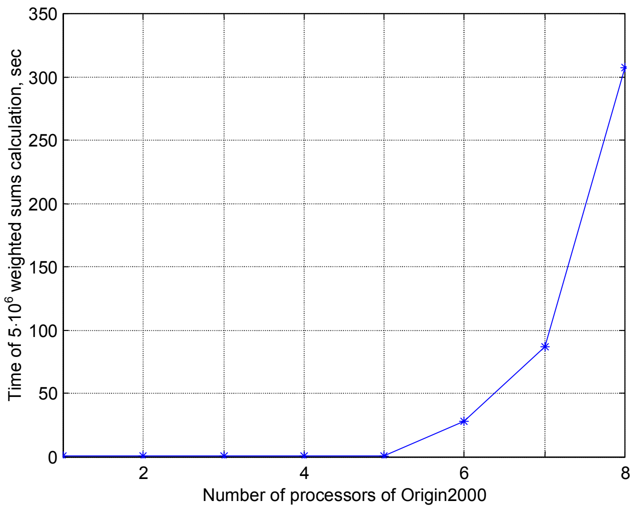

Fig. 5. Time of parallel calculation of weighted sum on Origin2000

B. Parallel training of each separate IHDNN

The same model of three-level perceptron (9-9-1) is used for the experiments on parallel training of 50 IHDNNs. The time of parallel training of all 50 IHDNNs has considerably decreased with increasing of quantity of used processors (Fig. 6). The time of training has decreased from 83 to 40 times for first - fourth scenarios at usage of eight Origin2000 processors (Fig. 7) in comparison with uniprocessor computer P-III/600 MHz/128 Mb RAM. The obtained results confirm the efficiency of the second paralleling scheme. The speed of IHDNNs training increases in 2-3 times with increasing of quantity of used processors.

5. CONCLUSION

The proposed method of sensors drift prediction allows providing error reduction of physical quantity measurement in intelligent systems by self-adaptation. The self-adaptation is provided by interaction of three neural networks with various properties. The proposed method allows successfully to predict various kinds of sensor drift and to reduce sensor errors in 3-5 times on early stage of sensor exploitation. This method is used into the prototype of Intelligent Sensing Instrumentation Structure, where training of all neural networks (IHDNN, ANN and PNN) is performed on a higher system level, i.e. on the personal computer. Usage high-performance computer Origin2000 on the higher level (even with access through the Internet) allows to increase the speed of neural networks training in 40-83 times.

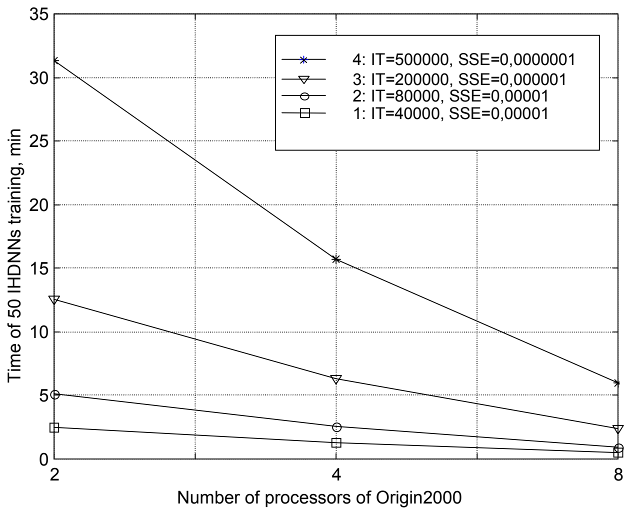

Fig. 6. Time of parallel calculation of each separate IHDNN on Origin2000

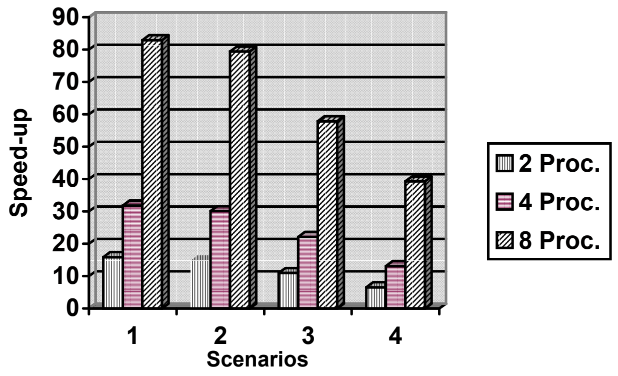

Fig. 7. Speed-ups in comparison with Pentium-III/600/128RAM

ACKNOWLEDGEMENT

The authors would like to thank European organization INTAS, grant reference number INTAS-OPEN-97-0606. The authors also would like to thank Dr. M. Mancini for the help during fulfillment of the experimental researches.

REFERENCES

[1] A.Sachenko, V.Kochan, V.Turchenko, Intelligent Distributed Sensor Network, *Proceedings of 15th IEEE Instrumentation and Measurement Technology Conference IMTC/98,* St. Paul, USA, 1998, 60-66.

[2] A. Sachenko, V. Kochan, R. Kochan, V. Turchenko, K. Tsahouridis, Th. Laopoulos, Error Compensation in an Intelligent Sensing Instrumentation System, *18th IEEE Instrument. and Meas. Tech. Conference IMTC/2001*, Budapest, Hungary, 2001, 869-874.

[3] Brignell J., Digital compensation of sensors, *Scientific Instruments*, 20(9), 1987, 1097-1102

[4] C.Alippi, A.Ferrero, V.Piuri, Artificial Intelligence for Instruments & Applications, *IEEE I&M Magazine*, Jun'98, 9-17.

[5] P. Daponte, D. Grimaldi, Artificial Neural Networks in Measurements, *Measurement*, 23, 1998, 93-115.

[6] A.Sachenko, V.Kochan, V.Turchenko, V.Golovko, Y.Savitsky, A.Dunets, T. Laopoulos, Sensor Errors Prediction Using Neural Networks, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks IJCNN'2000*, Como, Italy, 2000, vol. IV, 441-446.

[7] http://www.fluke.com/products/home.asp?SID=7& AGID=6&PID=5308

[8] K. B. Lee, IEEE 1451: A Standard in Support of Smart Transducer Networking, *Proceedings 17th IEEE Instrument. and Measur. Tech. Conference IMTC/2000,* Baltimore, USA, 2000, 525-528.

[9] http://content.honeywell.com/sensing/prodinfo/sds/

[10] A.Sachenko, V.Kochan, V.Turchenko, T.Laopoulos, V.Golovko, L.Grandinetti, Features of Intelligent Distributed Sensor Network Higher Level Development, *Proceedings of the 17th IEEE Instrum. and Measurement Technology Conference IMTC/2000*, Baltimore, USA, 2000, 335-340.

[11] V. Turchenko, V. Kochan, A. Sachenko, Estimation of Computational Complexity of Sensor Accuracy Improvement Algorithm Based on NN, *Lecture Notes in Computing Science*, 2130, 2001, 743-748.

[12] A.Sachenko, V.Kochan, V.Turchenko, Sensor Drift Prediction Using Neural Networks, *Proceedings of the Intern. Workshop on Virt. and Intellig. Measur. Syst. VIMS'2000*, Annapolis, USA, 2000, 88-92.

[13] Rumelhart D., Hinton G., Williams R, Learning Represent. by Back propagation Errors", *Nature*, 1986, 323, 533-536.

[14] V.Golovko, Y.Savitsky, T.Laopoulos, A.Sachenko, L.Grandinetti, Technique of Learning Rate Estimation for Efficient Training of MLP, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks IJCNN'2000*, Como, Italy, 2000, vol. I, 323-328.

[15] V.Turchenko, V.Kochan, A.Sachenko, Neural-Based Data Processing in Intelligent Distributed Sensor Network, *Proceedings of the Intern. Conf. on Neural Networks and Artificial Intelligence ICNNAI'2001*, Minsk, Belarus, 2001, 193-198.