Intelligent System for Prediction of Sensor Drift

V. Golovko[1], J. Savitsky[1], A. Sachenko[2], V. Kochan[2], V. Turchenko[2], T. Laopoulos[3], L. Grandinetti[4]

[1] Brest Polytechnic Institute, Belarus, cm@brpi.belpak.brest.by
[2] Ternopil Academy of National Economy, Ukraine, sachenko@cit.tane.ternopil.ua
[3] Aristotle University of Thessaloniki, Greece, laopoulos@physics.auth.gr
[4] University of Calabria, Italy, lugran@unical.it

Abstract

In this paper the features of neural networks using for improve of measurement accuracy of physical quantities by sensor drift prediction are considered. There is use a technique of data volume increasing for training of predicting neural network by using of separate approximating neural network.

1: Introduction

The new technologies require the greater quantity of the measuring information that has resulted recently in significant development of its reception means. High accuracy and significant information processing possibilities characterize the modern measuring systems. However in majority of cases is rationed (installed) the measurement error of output sensor signal instead of physical quantity. In the last decades the accuracy of sensor signal measurement has increased in dozens of times. However the analysis of [1] and [2] has shown that the sensor error has insignificantly decreased for this time. For example, at temperature measurement by Honeywell Pt100 sensors [1] and block Hydra 2625A Fluke [3] a ratio of errors of measuring channel elements more than fifty. The development of computing means has allowed considerably to increase a degree of information processing (to use complex mathematical methods of processing and to operate with knowledge of measurement object). But majority of work, which devoted to sensor signals processes [4, 5, 6, 7], consider questions that not connected with improving of measurement accuracy of physical quantity at measuring systems exploitation. In the works [8,9] using of artificial intelligence methods for improving of measurement accuracy of physical quantities are discussed.

In this work the methods predicting of sensor drift by artificial neural network system are considered.

2: General Structure

Intuitively considering intelligence in signal processing systems, the existence of a central computational processing unit is obvious. The higher the number of signals (here sensor devices), the grater the computational power needed and heavier the load of signal transmission through the system. A preferable structure is of a distributed signal processing system, were sensor device(s) information is being processed in an intermediate level and only "useful" information is transmitted to a higher hierarchical level. Considering the sensor devises as a sensors (actuators) and sensor interface circuits and the need of a human-machine interface, a general structure of a multi-sensory, multi-modal system is presented in Fig. 1. The realization of such structure is feasible under the premise that the computational power hidden in the processing levels is adequate to perform the operations, which will give to the system four basic properties: adaptability, accuracy, reliability and universality.

1. Accuracy
An accurate system must be able to compensate systematic (offset, gain, nonlinearity, cross sensitivities), systematic drift and random errors originated from sensors characteristics or system parameters. The ability of dealing with missed data due to random (transient or intermittent) faults is also desirable.

2. Universality
The universality system must provide following possibilities:

- Application of ISIS for the solving of various problems. It means the ISIS application for measurement of wide number of physical quantities. The modular structure of wide operating hardware and universal software are used for this purpose;
- The possibility of ISIS easy increasing. This ISIS property means the development and wide use of the hardware and software modular libraries. To

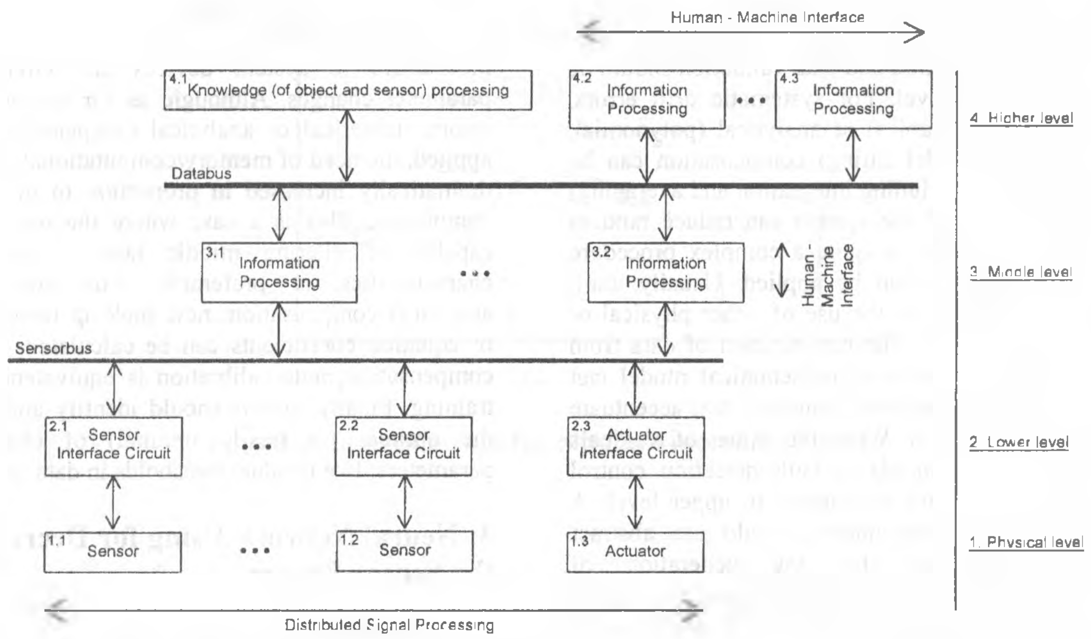

Human - Machine Interface
4.1 Knowledge (of object and sensor) processing
4.2 Information Processing
4.3 Information Processing
4. Higher level
Databus
3.1 Information Processing
3.2 Information Processing
Human - Machine Interface
3. Middle level
Sensorbus
2.1 Sensor Interface Circuit
2.2 Sensor Interface Circuit
2.3 Actuator Interface Circuit
2. Lower level
1.1 Sensor
1.2 Sensor
1.3 Actuator
1. Physical level
Distributed Signal Processing

Figure. 1. General Structure of ISIS

add the new interface sensor circuit it is sufficient to have its driver. The operating program of middle level node is compiled at the high level taking into account all necessary drivers, and it is recorded to the node in remote reprogramming mode.

2.1: Distribution of operations in system levels

Having defined the system's main principles, the next step in modeling its general structure is the definition of all possibly required operations and their distribution at the various levels. Following the structure of Fig. 1 bottom-up such a description is made in the rest of the paper.

2.1.1: Physical level.

Physical level includes the sensors/actuators whose inputs are the physical quantities and outputs are signals of voltage, current or time (period or frequency in a form of pulses). In the case of actuators, excitation signals provided by the lower level is needed. The sensors interface circuit (or MM) should provide the connection of different sensors and are classified according to sensors signals, but not according to sensors types.

2.1.2: Lower level.

The sensor interface circuit must receive the sensor's signal and manipulate it to provide a digital output signal comprehensible by the middle level. Possible operations executed here are:

- Amplification.
- Filtering (DC rejection, separation of common mode from differential mode signal etc.).
- Analog to digital conversion;
- Switching.

Also, when many similar sensors (physical redundancy) are used, they can share the same interface circuit or parts of it by multiplexing. Excitation signals to actuators are supplied by the circuitry of this level. So, the wiring required between physical and lower layer is for data and actuator excitation signal transferring. Finally, as to the adaptability rule, a digital to analog conversion of the control signals provided by the higher hierarchical (middle) layer might be required. The lower level must provide digital to analog conversion for creation of sensor activaton signal (for example, set of working current RTD). Also it is sufficient to have some 8-bit DAC, its output voltage can be measured by 16-bit ADC. The universality rule requires the possibility of different types of sensor interface circuit usage in ISIS. They must provide interaction with maximum number of sensors.

2.1.3: Middle level

This layer must carry out three important tasks, control the lower level units, collect and process information from them and communicate with the upper layer. Such performance can be accomplished by the use of either a high performance microcontroller (μC) or a Digital Signal Processor (DSP).

According to adaptability and reliability rules, middle level must control sensor and sensor interface circuit modes and ranges (adjusted to improve sensitivity) as well as the multiplexing (depending on desired data rate) of the various sensor devices to the sensor bus. This leads to a bi-directional sensor bus for digital data and control signals transferring. Since either parallel or serial bus can be applied, the choice is made by compromising between data rate and wiring reduction. However latest μC and DSP include high-speed synchronous and asynchronous serial ports, making serial bus selection extremely appealing.

Error compensation and data validation should de performed in this level. For systematic drift errors, numerical (look-up tables) or analytical (polynomial, exponential etc. model fitting) compensation can be applied. Filtering (including integration and averaging) and proper design of the system can reduce random errors. Data validation is quite a complex procedure and various methods can be applied. Usually, fault detection is achieved by the use of either physical or analytical redundancy. The combination of data from similar sensors or sensor's mathematical model can generate residuals, analytical functions that accentuate in the presence of fault. When the values of residuals exceed defined thresholds, a fault detection control signal is generated and transmitted to upper level. A more advanced implementation would use abstract computing techniques for the generation of characterized fault detection signals, depending on the fault's source and duration as well as the contribution of fault to unreliability of data.

An other appealing method of error compensation and data validation would be the use of Artificial Neural Networks (ANNs). The drawback of the realization of ANNs in this layer of the system is the computational power required in the training phase. Feasible solution could be the execution of the training phase at the upper layer and the transferring of the computed neuron synapse weights down to the middle layer. Application of ANN metrologies will be further discussed on the upper level session.

Communication between middle and upper layer includes the transferring of data and status signals upwards and data and control signals downwards. Standard protocols like RS232, RS485 or IEEE1451 can be applied. The DataBus should provide the following requirements:

- The network topology is the common bus with the branching possibilities;
- The total network length should be not less than two kilometers;
- The possibility of the additional users connection to the network without its turning off;
- The cheapness cost of cable and network accessories;
- The maximum usage of already existing equipment.

2.1.4: Upper level

The main computational unit of the system collects and combines data and status signals from the various information processing devices and makes the abstract application-depended decisions. Relating to system's adaptability and reliability, self-testing and auto-calibration should be performed here.

Self-testing refers to the intelligent function of monitoring each and every sensor device and detecting of any unreasonable behavior. In case of such detection, attempt of (auto) calibration or isolation of the device while signaling for maintenance or replacement should be performed.

Auto-calibration is the system's adaptation mechanism to system devices and environmental parameter changes. Although, as for systematic drift errors, numerical or analytical compensation can be applied, the need of memory/computational power will dramatically increased in proportion to the system's complexity. This is a case where the use of ANNs capable of altering middle layer's compensation characteristics is preferable. For numerical or analytical compensation, new look-up table elements or equation coefficients can be calculated. For ANN compensation, auto-calibration is equivalent to ANN training. Finally, ANNs should identify and maintain the optimal (or nearly optimal) of characteristic parameters, like residual thresholds in data validation.

3: Neural Network Using for Decreasing of the Sensor Errors

The sensor error is determined by initial spread of its conversion characteristic (at manufacturing) and by its drift in operating conditions [2]. First component is corrected relatively easy. The drift of the majority of the sensors is characterized by complex temporary functions where its parameters depend on operating conditions [10]. Thus the drift prediction after results of preliminary researches of particular type of sensors provides sufficient reliability of accuracy improving only in separate cases. The reliable improving of accuracy irrespective of operating conditions is provided by periodic testing of sensors with standard sensor or by using of special calibrator for sensor's calibration. The highest accuracy is reached at testing or calibration on operating place, but the operations that realize these methods are reasonably laborious. The decrease of costs on their realization is possible by decrease of their fulfillment frequency at the expense of high-quality prediction of sensor drift during inter-testing interval.

It is necessary to note that the functions of sensor drift usually have individual character and have significant casual component [11]. Prediction of such functions is reasonably a complex problem. For its solving it is offered to use neural networks that are optimum for problems of such kind [12]. It is defined by adaptive properties of neural networks at the expense of its self-training. It is known [13] that the quality of neural networks training in a strong degree depends on using data volume. However the aspiration to increase the inter-testing interval at the expense of high-quality prediction of sensor drift proportionally reduces numbers of data for neural network training. The method of artificial increasing of data volume is offered by using of approximating and predicting neural networks. The first network approximated the real data and permits to generate data volume which sufficient for training of predicting neural network. The necessity of two neural networks using is defined by the different requirements to their properties and their internal structure.

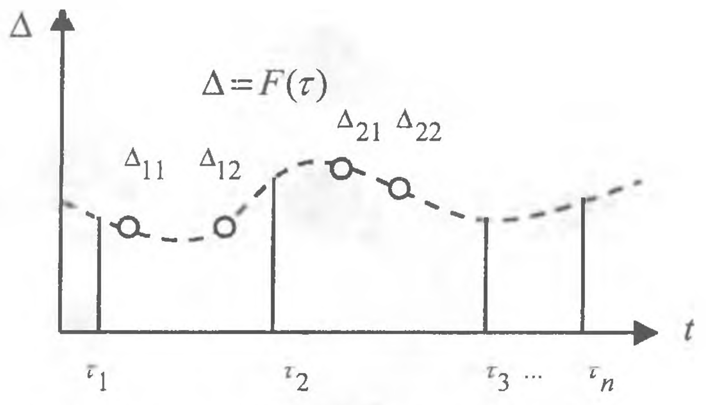
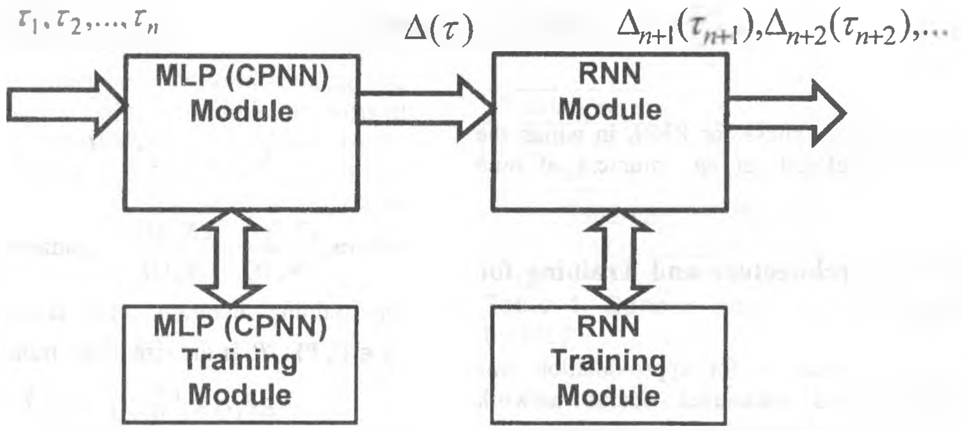


Figure 2. Sensor errors for different moment of time.



$\tau_1, \tau_2, \ldots, \tau_n$
$\Delta(\tau)$
$\Delta_{n+1}(\tau_{n+1}), \Delta_{n+2}(\tau_{n+2}), \ldots$
MLP (CPNN) Module
RNN Module
MLP (CPNN) Training Module
RNN Training Module

Figure 3. Sensor error prediction scheme.

The objective of the neural system is to predict the sensor error for any moment of time: $\tau > \tau_n$ *and* $\tau \in [\tau_1, \tau_n]$. For achievement this goal is necessary to solve the following tasks:

1. Given a finite sequence $\Delta_1(\tau_1), \Delta_2(\tau_2), \ldots, \Delta_n(\tau_n)$, find the meaning of error Δ for any moment time $\tau_{ij} \in [\tau_1, \tau_n]$:

$$\tau_{11} \to \Delta_{11}, \tau_{12} \to \Delta_{12} \ldots$$
$$\tau_{21} \to \Delta_{21}, \tau_{22} \to \Delta_{22}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\tau_{n-1,1} \to \Delta_{n-1,1}, \tau_{n-1,2} \to \Delta_{n-1,2}$$

2. Given a finite sequence for any moment of time $\tau \in [\tau_1, \tau_n]$, find the continuation the time series $\Delta_{n+1}(\tau_{n+1}), \Delta_{n+2}(\tau_{n+2}), \ldots$

Such approach permits to predict the sensor errors for any moment of time.

The common architecture of the neural system is presented on Fig. 3. It consists of two neural modules. Multilayer perceptron (MLP) or counter propagation neural network (CPNN) can use as the first module. It is meant for the approximation of the function $\Delta = f(\tau)$. In result is obtained the training set for the second module. The recurrent neural network is used as the second module. It is meant for predicting sensor errors.

The use of this neural system involves several separate phase, which have to be follows:

1. The learning of MLP or CPNN, in which suitable training set is used to train the neural networks. As far as the training algorithm is concerned, the backpropagation or counterpropagation algorithms can be applied. A more detail information about application of this algorithms will be gave below.
2. The validation phase for MLP (CPNN) [8]. In this case the neural network's generalization ability is verified by means of other data. Also the neural network accuracy is estimated.
3. The production phase for MLP (CPNN). In result can be obtained the training set for the recurrent neural network.
4. The learning of RNN. The previously obtained data are used for training of the RNN. As far as the learning algorithms concerned, it will be gave below.
5. The validation phase for RNN, which is performed same as previously.

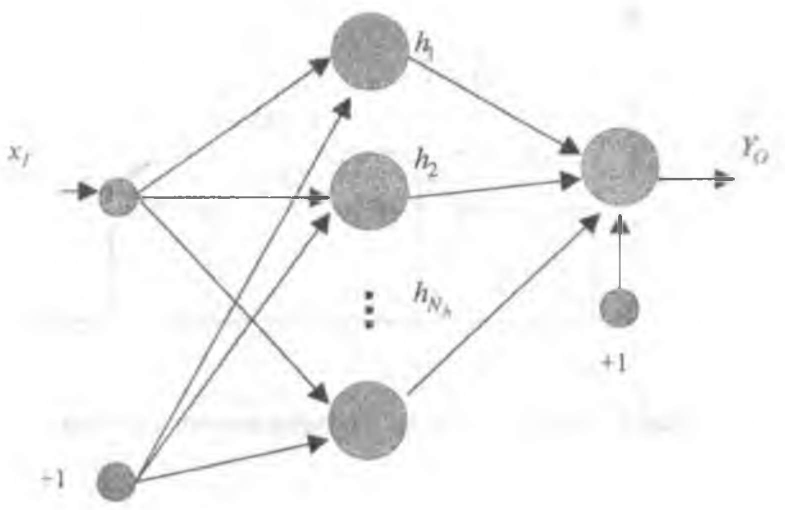


Figure 4. The MLP architecture

6. The production phase for RNN, in which the sensor errors are defined for any moment of time $t > t_n$.

4.1: The MLP Architecture and Training for Approximation

As network architecture for approximation was accepted third-layered multilevel neural network containing one hidden layer of nonlinear units and a single output linear unit, as shown on Fig. 4. The output activity of the neural network is defined by expression:

$$Y = \sum_{i=1}^{N_h} w_{i0} h_i - s_0 \tag{1}$$

where N_h - number of units of the hidden level, h_i - output activity of hidden units, s_0 - threshold for output unit, w_{i0} - weights from hidden input units i to the output unit. The output activity of the hidden units is defined as:

$$h_j = g(w_{lj} x_l + s_j) \tag{2}$$

where x_l is the input element, w_{lj} - weights from input unit to hidden units j and s_j - thresholds of the hidden units. In hidden units is used sigmoid transfer function, defined as $g(x) = \left((1+e^{-x}\right)^{-1}$.The most popular training algorithm for multilevel perceptrons is backpropagation. This algorithm is based on gradient descent method and consists of fulfilment of an iterative procedure of updating weights and thresholds for each training exemplar p of training set under following rule:

$$\Delta w_{ij}(t) = -\alpha \frac{\partial E^p(t)}{\partial w_{ij}(t)} \tag{3}$$

$$\Delta s_j(t) = -\alpha \frac{\partial E^p(t)}{\partial s_j(t)} \tag{4}$$

where $\frac{\partial E^p(t)}{\partial w_{ij}(t)}$, $\frac{\partial E^p(t)}{\partial s_j(t)}$ - gradients of error function on training iteration t for training exemplar p, $p \in \{1, P\}$, P is the size of the training set;

$$E^p(t) = \frac{1}{2}\left(Y_O^p(t) - D_O^p\right)^2 \tag{5}$$

$Y_O^p(t)$ - network output activity on training iteration t for training exemplar p, D_O^p - desirable value of a network output for training exemplar p. During training there is the reduction process of the total network error:

$$E(t) = \sum_{p=1}^{P} E^p(t) \tag{6}$$

For improvement of network training parameters and removal defects of classical back propagation algorithm, connected with empirical selection of a constant training step, use the steepest descent method for calculation of an adaptive training step, according to it:

$$\left|\begin{aligned} &\Delta w_{ij}(t) = -\alpha^p(t)\frac{\partial E^p(t)}{\partial w_{ij}(t)}, \\ &\Delta s_j(t) = -\alpha^p(t)\frac{\partial E^p(t)}{\partial s_j(t)}, \\ &\alpha^p(t) = \min\left\{E^p\left(w_{ij}(t+1), s_j(t+1)\right)\right\} \end{aligned}\right. \tag{7}$$

where $\alpha^p(t)$ - step value, adapted on each training iteration t for each external vector p.

According to expression (7) the formulas for calculation of adaptive step for sigmoid and linear functions of activation were obtained.

For linear transfer function the adaptive training step is defined by expression:

$$\alpha^{p}(t)=\frac{1}{\sum_{i=1}^{N_{h}}(h_{i}^{p}(t))^{2}+1} \tag{8}$$

where $h_i^p(t)$ - elements of input activity of the linear unit at the time t for the training vector p.

For sigmoid transfer function adaptive training step is computed as:

$$\alpha^{p}(t)=\frac{4}{1+(x_{j}^{p})^{2}}\times\frac{\sum_{j=1}^{N_{h}}(w_{j0})^{2}h_{j}^{p}(t)(1-h_{j}^{p}(t))}{\left(\sum_{j=1}^{N_{h}}(w_{j0})^{2}(h_{j}^{p}(t))^{2}(1-h_{j}^{p}(t))^{2}\right)} \tag{9}$$

where $w_{i0}(t)$ - weights from hidden units to output unit, adapted on training iteration t.

At training the network is actual problem of initialization of neural network units weights. The speed, accuracy and stability of training depends on it in many respects, especially if is used training sets with analogue values. Usually initialization consists of appropriation to weights and thresholds of units of the random evenly distributed values from some range: $w_{ij}=R(c,d), s_j=R(c,d)$. Upper and low bounds of this range is defined empirically. In our paper the following procedure of delimitation of a range for sigmoid function is offered:

$$w_{Ic}=\frac{\bar{x}_{I}}{(x_{I})^{2}+1}\ln\left(\frac{c}{1-c}\right) \tag{10}$$

$$w_{Id}=\frac{\bar{x}_{I}}{(x_{I})^{2}+1}\ln\left(\frac{d}{1-d}\right) \tag{11}$$

where $\bar{x}_I$ is expectation of the external activity on the MLP input, $\bar{D}_0$ is expectation of the external activity of desired output of the network, w_{Ic}, w_{Id} are bounds of the weights range, c, d are upper desired value and low desired value of the output activity for nonlinear units.

For linear output unit we proposed to calculate parameters w_{ic}, w_{id} as:

$$w_{ic}=\frac{c}{N_{h}c^{2}+1}\bar{D}_{0} \tag{12}$$

$$w_{id}=\frac{d}{N_{h}d^{2}+1}\bar{D}_{0} \tag{13}$$

The weights and thresholds of the units are initialized as:

$$\begin{aligned} w_{ij}&=R(w_{ic},w_{id})\\ s_{j}&=R(w_{ic},w_{id}) \end{aligned} \tag{14}$$

For stabilization of the training procedure is used the following algorithm of *level-by-level* training:

1) Calculate upper and lower bounds of the weights range, using expressions (10), (11) for hidden units and expressions (12), (13) for output unit. Update weights and thresholds according to expressions (14).

2) For the training vector p calculate output activity $Y_O{}^p(t)$ of the neural network.

3) Calculate an error of an output unit.

4) Update weights and thresholds *only* for the output unit according to expressions (7), using adaptive step (8).

5) Calculate the error of units of the hidden level for the network with the *updated weights for an output level.*

6) Update weights and thresholds of units of the hidden level, using adaptive step (9) for sigmoid transfer function.

The application of this algorithm has allowed to stabilize learning process of the recurrent neural network with varied functions of activation and considerably to reduce time of training.

For simulation is used the time series of sensor errors, described by function $f(\tau)=\sqrt{\tau}+\sin(3\tau)$. Computing experiments for sensor error approximation by MLP shown in the Table 4.1.

Training set size	80
Number of hidden units	6
Parameters c, d (for hidden units)	0.1, 0.9
Total mean square training error	$1.92*10^{-5}$
Number of approximation steps	195
Approximation error in percentage	1.89%

Table 1. Sensor error approximation results by MLP

One can see on the table 1, that MLP approximate the function with small error.

Let's examine the CPNN for approximation.

4.2: The CPNN Architecture and Training for Approximation

The architecture used of CPNN is represented in the Fig 5. It consists of three layers. The hidden layer consists of Cohonen neurons (not shaded circles) and nonlinear neurons (shaded circles), which amount is equaled among themselves. Such neurons will derivate pairs, and each Cohonen neuron in a pair has horizontal connection with a nonlinear neuron, appropriate to it, as shown Fig. 4. Besides all neurons of a hidden layer are connected to output neuron, which has linear function of activation. Let's use sequential numbering of pairs of neural elements. Let's designate through w_{Ij} and u_{Ij} accordingly weight factors of Cohonen neurons and nonlinear neurons of the hidden layer. Then the output value h_j of j'th nonlinear hidden neuron can be defined as

$$h_{j}=F\left(y_{j}u_{Ij}x_{I}\right) \tag{15}$$

where y_j - output of a Cohonen neuron in j-th pair of neurons, F - hyperbolic tangent. The output value of j-th Cohonen neuron is equaled to one, if this neuron is a winner, and zero otherwise. For definition of a neuron - winner the Euclidean distance is used:

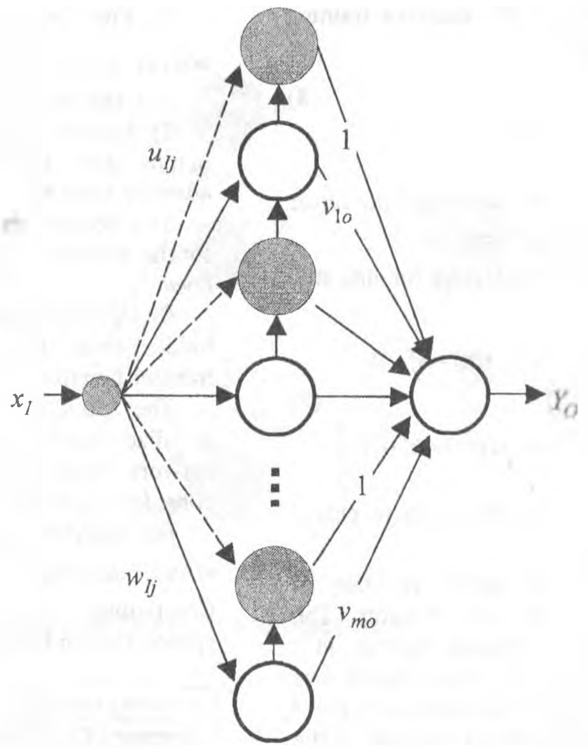


Figure 5. The counterpropagation neural network architecture

$$D_j = |X^p - W_j| \qquad (16)$$

In the correspondence with it:

$$y_k = \begin{cases} 1, \text{if } D_k = \min|X^p - W_i| \\ 0, \textit{otherwise} \end{cases} \qquad (17)$$

The weights from nonlinear neurons of the hidden layer to output neuron are equaled to one.

If a neuron - winner in the hidden layer has number k, the output of the neural network is equal:

$$Y_O = v_k + h_k \qquad (18)$$

The amount m of neurons of the hidden layer is selected equal:

$$m = P - 1 \qquad (19)$$

where P -size of the training set.

The training of CPNN is made on the following algorithm.

1. The set-up a Cohonen weights by the empirical rule is made:

$$w_{Ij} = x_i + \frac{x_{i+1} - x_i}{3} \qquad (20)$$

where w_{Ij} - weight of the i -th of the Cohonen neuron, x_i - is i -th component of the training set ordered on increase, $i = \{1,...,m\}$.

2. Further in a cycle the set-up a weights of remaining neural elements is executed. On an input of the neural network values of the training set sequentially move and the following operations are made for each value:

- The value of an output of the neural network Y_O^p and number k of a neuron - winner of a Cohonen layer is calculated.
- The set-up of an appropriate neuron of the output layer is made:

$$\Delta v_k(t) = -\alpha(Y_O^p - D_O^p) \qquad (21)$$

where D_O^p - target value of CPNN normalized in a range [-1, 1], $\alpha = 0.01$ - training step.

- The appropriate neuron of the hidden layer is set up:

$$\Delta u_{Ik}(t) = -\alpha(Y_O^p - D_O^p) \cdot (1 - (h_k^p)^2) x_I^p \qquad (22)$$

3. The training total root-mean-square error is calculated:

$$E = \frac{1}{2} \sum_{p=1}^{P} \left(Y_O^p - D_O^p\right)^2 \qquad (23)$$

The steps 2 and 3 are repeated before stabilization of training error.

Computing experiments for sensor error approximation for various sizes if training set by CPNN shown in the table 2.

Training set size	80
Total mean square training error	$1.587*10^{-5}$
Number of approximation steps	195
Approximation error in percentage	2.58%

Table 2. Sensor error approximation results by CPNN

4.3: The RNN Architecture and Training for Time Series Prediction

As basic network architecture for prediction of the sensor error was accepted fully connected third-layered recurrent neural network containing one hidden layer of nonlinear units and a single output linear unit, as shown in the Fig. 6. The output activity of the neural network is defined by expression:

$$Y^r = \sum_{i=1}^{N_h} w_{i0} h_i^r - s_0 \qquad (24)$$

where N_h - number of units of the hidden level, h_i^r - output activity of hidden units at the moment r, s_0 - threshold for output unit, w_{i0} - weights from hidden input units i to the output unit.

The output activity of the hidden units on the current moment r for training exemplar p is defined as:

$$h_j^r = g\Big(\sum_{i=1}^{N_I} w_{ij} x_i^r + \sum_{k=1}^{N_h} w_{kj} h_k^{r-1} + w_{0j} Y^{r-1} + s_j\Big) \qquad (25)$$

where x_i^r is a i'th element of the input vector x^r, P is size of the training set, N_I - size of an input vector, w_{ij} - weights from external units i to hidden units j, w_{kj} - weights from hidden units k to hidden units j, $h_k^{r-1}(t)$ - output activity of a hidden unit k for the previous moment of time $r-1$, w_{0j} - weight to the hidden units from an output unit, $Y^{r-1}(t)$ - network output activity for the previous moment of time $r-1$ and s_j - thresholds of the hidden units.

In this work we use two types for hidden units transfer function. At one case is used non-standard logarithmic function $g(x) = \ln\left((x + \sqrt{x^2 + a})/\sqrt{a}\right)$, $(a > 0)$. The choice by this transfer function is stipulated by that it is unlimited on all define area. It allows better to simulate and predict complex non-stationary processes. The parameter a defines declination of the activation function (see Fig 7). At other case in hidden units is used sigmoid transfer function, defined as $g(x) = \left(1 + e^{-x}\right)^{-1}$

For logarithmic activation function the estimate of an adaptive training step can be received by the following expression:

$$\alpha^p(t) = \frac{1}{\left(1 + \sum_{i=1}^{N_I}(x_i^p)^2 + \sum_{k=1}^{N_h}(h_k^{p-1}(t))^2 + (Y^{p-1}(t))^2\right)} \times \frac{\sqrt{a}\sum_{j=1}^{N_h}(w_{j0})^2\left(\sqrt{(B_j^p(t))^2 + a}\right)^{-1}}{\left(\sum_{j=1}^{N_h}(w_{j0})^2\left((B_j^p(t))^2 + a\right)^{-1}\right)} \qquad (26)$$

where

$$B_j^p(t) = \sum_{i=1}^{N_I} w_{ij}(t) x_i^p + \sum_{k=1}^{N_h} w_{kj}(t) h_k^{p-1}(t) + w_{0j}(t) Y^{p-1}(t) + s_j$$

is weighed sum of inputs of the hidden unit j.

For standard sigmoid transfer function adaptive training step is computed as:

$$\alpha^p(t) = \frac{4}{\left(1 + \sum_{i=1}^{N_I}(x_i^p)^2 + \sum_{k=1}^{N_h}(h_k^{p-1}(t))^2 + (Y^{p-1}(t))^2\right)} \times \frac{\sum_{j=1}^{N_h}(w_{j0})^2 h_j^p(t)(1 - h_j^p(t))}{\left(\sum_{j=1}^{N_h}(w_{j0})^2 (h_j^p(t))^2 (1 - h_j^p(t))^2\right)} \qquad (27)$$

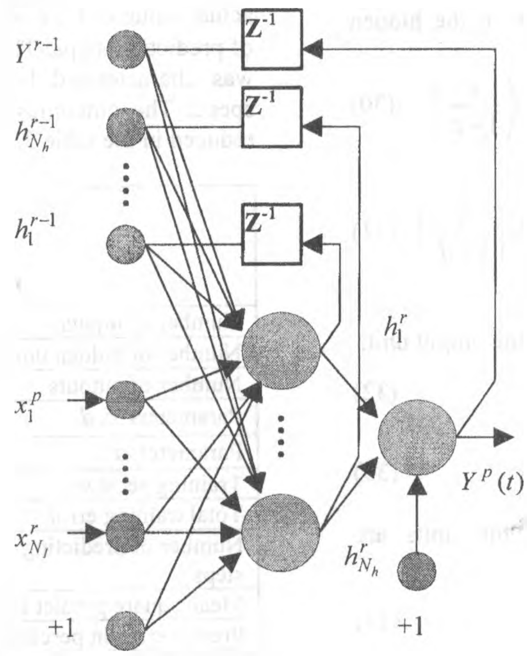


Figure 6. The recurrent neural network architecture

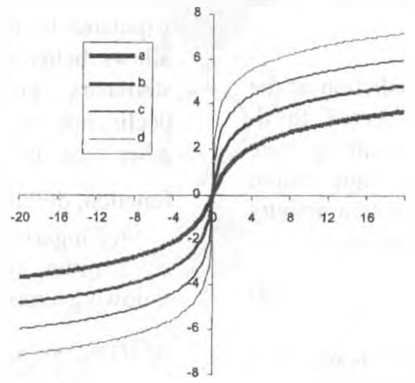


Figure 7. The logarithmic activation function for various parameters a: a) $a = 1.0$, b) $a = 0.1$, c) $a = 0.01$, d) $a = 0.001$

For adaptive initialization of weights and thresholds in RNN before training is used the next expressions:

1. For logarithmic activation function in the hidden units:

$$w_{ic} = \frac{\bar{x}_i}{\sum_{k=1}^{N_i}(\bar{x}_k)^2 + N_h c^2 + \bar{D}_0^2 + 1} \frac{\sqrt{a}e^c - \sqrt{a}e^{-c}}{2} \tag{28}$$

$$w_{id} = \frac{\bar{x}_i}{\sum_{k=1}^{N_i}(\bar{x}_k)^2 + N_h d^2 + \bar{D}_0^2 + 1} \frac{\sqrt{a}e^d - \sqrt{a}e^{-d}}{2} \tag{29}$$

where $\bar{x}_k$ is expectation of the external activity on the k'th inputs, $\bar{D}_0$ is expectation of the external activity of desired output of the network, w_{ic}, w_{id} are bounds of the weights range, c, d are upper desired value and low desired value of the output activity for nonlinear units.

2. For sigmoid activation function in the hidden units:

$$w_{ic} = \frac{\bar{x}_i}{\sum_{i}^{N_i}(\bar{x}_i)^2 + N_h c^2 + \bar{D}_0^2 + 1} \ln\left(\frac{c}{1-c}\right) \tag{30}$$

$$w_{id} = \frac{\bar{x}_i}{\sum_{k=1}^{N_i}(\bar{x}_k)^2 + N_h c^2 + \bar{D}_0^2 + 1} \ln\left(\frac{d}{1-d}\right) \tag{31}$$

3. For linear activation function in the output unit:

$$w_{ic} = \frac{c}{N_h c^2 + 1} \bar{D}_0 \tag{32}$$

$$w_{id} = \frac{d}{N_h d^2 + 1} \bar{D}_0 \tag{33}$$

The weights and thresholds of the units are initialized as:

$$w_{ij} = R(w_{ic}, w_{id})$$
$$s_j = R(w_{ic}, w_{id}) \tag{34}$$

For training of the RNN is used described above level-by-level training algorithm. For simulation were used the time series of sensor errors, taken from MPL (CPNN) approximation results. It size is 195 units. For training were used 105 units of this number. The training was carried out the method of the sliding window. For simulation two types of neural networks were used. One of them contained the sigmoid activation function of the hidden units. In other network the logarithmic function of activation of the hidden units with the parameter $a = 0.01$ was used. Both networks consist of 10 input units, 5 hidden units, 1 output unit. The prediction was carried out on 90 steps forwards. For an estimation of the prediction results is used the mean square predict error computed as:

$$E_{pr}(L) = \frac{1}{L}\sum_{l=1}^{L}\left(Y(l) - x(l)\right)^2 \tag{35}$$

where $Y(l)$ - predict value for the step l, $x(l)$ - actual value of time series in the moment l, L - total of prediction steps. The training both neural networks was characterized by high accuracy, stability and speed. The outcomes of training and prediction are reduced in the table 3.

	Sigmoid network architect.	Logarithmic network architect.
Number of inputs	10	10
Number of hidden units	5	5
Number of outputs	1	1
Parameters c, d	0.9, 0.1	4.2, -4.2
Parameter a	-	0.01
Training set size	105	105
Total training error	3E-5	4E-5
Number of predicting steps	90	90
Mean square predict error	3.57E-5	1.19E-5
Predict error in percentage	4.34%	3.21%

Table 3. Sensor error prediction experiments by RNNs

5: Conclusion

As it is visible, some operating performances of measuring engineering hinder direct use of neural networks for increase of an exactitude of measuring channels. However use of features of neural networks and organization of their correct interaction with the measuring system allow to overcome originating difficulties and to supply increase of an exactitude of a measurement for want of minor working costs. In this work the unique technology of forecasting of errors of gauges with use of a complicated neural system is circumscribed. The effective methods of neural networks training of different architectures are indicated. The computing experiments with hypothetical datas of errors of sensors demonstrate potential possibilities of application of this intellectual neural system in actual problems

Acknowledgement

Authors would like to thank INTAS, grant reference number INTAS-OPEN-97-0606.

6: References

[1] http://www.honeywell.com/sensing/prodinfo/temperature/catalog/c15_93.pdf

[2] Samsonov G.V., Kits A.I., Kiuzdeni O.A., "Sensors for temperature measurement in industry", *Naukova dumka*, Kiev, 1972

[3] www.fluke.com/products/data_acquisition/hydra/home.asp?SID=7&AGID=0&PID=5308

[4] Iyengar S.S., "Distributed Sensor Network - Introduction to the Special Section", *Transaction on Systems, Man and Cybernetics*, vol. 21, No. 5, 1991, pp 1027-1035

[5] Brignell J., "Digital compensation of sensors", *Scientific Instruments*, vol. 20, No 9, 1987, pp.1097-1102

[6] Sydenham P.H., "Sensing Science and Engineering Group", *Measurement*, vol 14, No 1, 1994, pp.81-87

[7] Lee K., "The Emerging IEEE-P1451 Standards for Smart Sensors and Actuators", *Instrumentation and Measurement Society Newsletter*, 1997, Issue No 136

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

[9] C.Alippi, A.Ferrero, V.Piuri, "Artificial Intelligence for Instruments & Applications", *IEEE I&M Magazine*, June 1998, pp.9-17.

[10] Rogelberg I.L., Nushnov A.G., Pokrovskaya G.N. et al., "Stability of the thermo-E.M.F. force of K-type thermocouples at heating in the air at 1200°Ñ temperature", *Hypromettsvetobrabotka*, vol. 24, 1967, pp.54-65

[11] Sachenko A., Milchenko V., Kochan V., Chyrka M., Karachka A., "Experimental researches of the calibrated characteristics of non-stability of K-type thermocouples", *Izmeritelnaja technika*, No 10, 1985, pp.28-29.

[12] Jain A.K., Mao J., Mohiuddin K.M., "Artificial neural networks: a tutorial", *Computer*, March 1996, pp.31-44.

[13] Kroese B. An introduction to Neural Networks.– Amsterdam: Univercity of Amsterdam – 1996.–120p.