STOCHASTIC APPROACHES TO DYNAMIC NEURAL NETWORK TRAINING. ACTUATOR FAULT DIAGNOSIS STUDY

Krzysztof Patan* Thomas Parisini**

* Institute of Control and Computation Engineering, University of Zielona Góra, ul. Podgórska 50, 65-246 Zielona Góra, Poland
  K.Patan@issi.us.zgora.pl

** Dept. of Electrical, Electronic and Computer Engineering DEEI-University of Trieste
Via Valerio 10, 34127 Trieste, Italy
parisini@univ.trieste.it

Abstract: A paper deals with application of stochastic methods for dynamic neural network training. The considered network is composed of dynamic neurons, which contain inner feedbacks. This network can be used as a part of a fault diagnosis system to generate residuals. Up-to-date training algorithms, based on the classical back propagation, suffer from entrapment in local minima of an error function. Two stochastic algorithms are tested as training algorithms to overcome these difficulties. Efficiency of the proposed learning methods is checked using data recorded at Lublin Sugar Factory, Poland.

Keywords: Actuators, dynamic modelling, fault detection, learning algorithms, neural network models.

1. INTRODUCTION

The paper deals with application of stochastic methods for training dynamic neural networks. Such networks belong to the class of locally recurrent globally feed-forward (Patan, 2000). They have an architecture that is similar to the feed-forward multi-layer perceptron and dynamic characteristics are included in their processing units. The networks under consideration are designed using the Dynamic Neuron Models (DNM). A single dynamic neuron consists of an adder module, linear dynamic system – Infinite Impulse Response (IIR) filter, and nonlinear activation module. When such neurons are connected into a multi-layer structure, a powerful approximating tool may be obtained. Taking into account that neuron by itself has dynamic characteristics, it is not required to introduce any global feedback to the network structure. In this way a simple architecture is obtained, what makes it relatively easier to elaborate a proper learning algorithm and contrary to recurrent networks to keep stability of the neural model.

Taking into consideration dynamic characteristics of the network, it is possible to apply it for modelling and identification of nonlinear systems. It is especially useful when there are no mathematical models of the modelled system available, and analytical models and parameter-identification algorithms cannot be applied (Patton et al., 2000; Frank and Köppen-Seliger, 1997). Based on the dynamic neural networks, the fault detection and isolation system for diagnosis of industrial processes can be designed (Patton et al., 2000; Chen and Patton, 1999). In recent few years this kind of neural networks was successfully applied in several fault diagnosis applications like fault detection and isolation in a two tank laboratory system or fault detection of the instrumental faults in the sugar evaporator (Patan and Korbicz, 2000; Korbicz et al., 1999). In the present study, the dy-

1 This work was supported by the European Commision within the FP5 project DAMADICS
Dynamic neural networks are applied to design fault detection systems for actuators based on data recorded at the Lublin Sugar Factory, Poland. It is worth noting that fault detection in actuators is a very difficult task, more difficult than in the case of instrumental (sensor) faults.

The fundamental training method for the considered dynamic networks is the Extended Dynamic Back-Propagation (EDBP) (Patan, 2000). This is a very simple algorithm utilizing the back-propagation error scheme. This algorithm may have both on-line and off-line forms, and therefore it can be widely used in the control theory. The identification of dynamic systems, however, is an example where training of the neural network is not a trivial problem. The error (cost) function is strongly multimodal, and during training, the EDBP often gets stuck in local minima. Even multi-starting of the EDBP cannot yield the expected results. Therefore other methods that belong to the class of global optimisation should be applied. To overcome above problems, it is proposed to use a stochastic method, which is called Adaptive Random Search (ARS) (Walter and Pronzato, 1996). Additionally, to carry out more detailed analysis and comparison, another stochastic approach Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 1992; Alessandri and Parisini, 1997) is applied.

The paper is organised as follows: first in Section 2, the brief description of the considered neural structure is presented. After that, the Sections 3, 4 and 5 present the examined learning methods in turn the EDBP, ARS and SPSA. The following Section 6 contains simulation results and comparative study. The remarks and conclusions are given in the final section.

2. STRUCTURE OF DYNAMIC NEURAL NETWORK

Neural network presented here, is a feed-forward networks with layered dynamic neurons. Dynamics is introduced to the neuron in such a way that the neuron activation depends on its internal states. It is done by introducing a linear dynamic system – the IIR filter – to the neuron structure (Ayoubi, 1994; Patan, 2000). In this way, each neuron in the dynamic network reproduces the past signal value with the input \( u_p(k) \), for \( p = 1, 2, \ldots, P \) and the output \( y(k) \). Figure 1 shows the structure of this neuron with many inputs, which is called the Dynamic Neuron Model. In such models one can distinguish three main parts: a weight adder, a filter block, and an activation block. The behaviour of the dynamic neuron model is described by the following set of equations:

\[
\begin{align*}
x(k) &= \sum_{p=1}^{P} w_p u_p(k) \\
\bar{y}(k) &= -\sum_{i=1}^{n} a_i \bar{y}(k-i) + \sum_{i=0}^{n} b_i x(k-i) \\
y(k) &= F(g \cdot \bar{y}(k) + c)
\end{align*}
\]

where \( w_p \) denotes the input weights, \( \bar{y}(k) \) is the filter output, \( a_i, \ i = 1, \ldots, n \) and \( b_i, \ i = 0, \ldots, n \) are the feedback and, feed-forward filter parameters, respectively (\( n \) denotes filter order), \( F(\cdot) \) is a non-linear activation function that produces the neuron output \( y(k) \), and \( g \) and \( c \) are the slope parameter and threshold of the activation function, respectively.

In the dynamic neuron the slope parameter can be changed. Thus, the dynamic neuron can model the biological neuron better. Introduction of the slope parameter \( g \) to the activation operation, in addition to threshold \( c \), can be very helpful, particularly in the case of non-linear squashing activation functions, i.e.: sigmoidal or hyperbolic tangent. This another adaptable parameter can be useful in faster arrangement of the desirable shape of the activation function (Żurada, 1993).

3. EXTENDED DYNAMIC BACK-PROPAGATION ALGORITHM

Let us consider the \( M \)-layered network with dynamic neurons with differentiable activation functions \( F(\cdot) \). Let \( s_m \) denotes the number of neurons in the \( m \)-th layer, \( u^m_i(k) \) – the output of the \( i \)-th neuron of the \( m \)-th layer at discrete time \( k \). The activity of the \( j \)-th neuron in the \( m \)-th layer is defined by the formula:

\[
u^m_j(k) = F \left[ \sum_{i=0}^{s_m} g^m_j \sum_{p=1}^{m-1} w^m_{jp} u^m_p(k-i) - \sum_{i=1}^{n} a^m_{ij} y^m_i(k-i) \right]
\]

All unknown network parameters can be represented by a vector \( \theta \) composed of matrices \( w, a, b, g \) and \( c \), where \( w = [w^m_j]_{m=1,\ldots,M; j=1,\ldots,s_m; \ p=1,\ldots,s_{m-1}} \) is the weights matrix, \( a = [a^m_{ij}]_{m=1,\ldots,M; j=1,\ldots,s_m; i=1,\ldots,n} \) and \( b = [b^m_{ij}]_{m=1,\ldots,M; j=1,\ldots,s_m; i=0,\ldots,n} \) are the filter parameters matrices, \( g = [d^m_{ij}]_{m=1,\ldots,M; j=1,\ldots,s_m} \) is the slope parameters matrix, \( c = [e^m_j]_{m=1,\ldots,M; j=1,\ldots,s_m} \) is the bias parameters matrix. The main objective of learning
is to adjust elements of the vector \( \theta \) in such a way to minimise some loss (cost) function:

\[
\theta^* = \min_{\theta \in C} J(\theta)
\]

where \( \theta^* \) is the optimal network parameter vector, \( J : \mathbb{R}^l \rightarrow \mathbb{R}^1 \) represents some loss function to be minimised, \( l \) is the dimension of the vector \( \theta \), and \( C \subseteq \mathbb{R}^l \) is the constraint set defining the allowable values for the parameters \( \theta \). The way of derivation of the Extended Dynamic Back Propagation is the same as the standard Back Propagation. The loss function

\[
J(k; \theta) = \| y(k) - \hat{y}(k; \theta) \|\]

where \( y(k) \) is the desired output of the network, \( \hat{y}(k; \theta) \) is the actual response of the network on the given input pattern \( u(k) \), should be minimised based on a given set of input-output patterns. The adjustment of the parameters of the \( j \)-th neuron in the \( m \)-th layer according to the off-line EDBP has the form (Korbicz et al., 2001):

\[
\theta_j^m(k+1) = \theta_j^m(k) + \eta \sum_{i=1}^{N} \delta_j^m(i) S_{ij}^m(i)
\]

where \( N \) is the dimension of the training set, \( \eta \) represents the learning rate, and \( \delta_j^m \) is the error for the \( j \)-th neuron in \( m \)-th layer and \( S_{ij}^m \) is the sensitivity function for elements of unknown generalised parameter \( \theta \) for the \( j \)-th neuron in \( m \)-th layer (Korbicz et al., 2001; Korbicz et al., 1999). The EDBP algorithm belongs to the class of classical optimisation methods, which are based on an idea of “hard selection”. During optimisation procedure, new base solutions over the best points obtained so far are calculated. Therefore, capabilities of crossing saddles between hills of a multimodal function are very small, and effectiveness of these methods in global optimisation problems is limited.

4. ADAPTIVE RANDOM SEARCH

Let us assume that the sequence of solutions \( \theta_0, \theta_1, \ldots, \theta_k \) is already appointed, a way of achieving next point \( \theta_{k+1} \) is formulated as follows (Walter and Pronzato, 1996):

\[
\theta_{k+1} = \theta_k + r_k
\]

where \( \theta_k \) is the estimate of the \( \theta^* \) at the \( k \)-th iteration, and \( r_k \) is the perturbation vector generated randomly according to the normal distribution \( \mathcal{N}(0, \sigma) \). New solution \( \theta_{k+1} \) is accepted when the cost function \( J(\theta_{k+1}) \) is less than \( J(\theta_k) \) otherwise \( \theta_{k+1} = \theta_k \). To start the optimization procedure, it is necessary to determine the initial point \( \theta_0 \) and the variance \( \sigma \). Let \( \theta^* \) be a global minimum to be located. When \( \theta_k \) is far from \( \theta^* \), \( r_k \) should have a large variance to allow large displacements, which are necessary to escape the local minima.

On the other hand, when \( \theta_k \) is close \( \theta^* \), \( r_k \) should have a small variance to allow exact exploration of parameter space. The idea of the ARS is to alternate two phases: variance-selection and variance-exploitation (Walter and Pronzato, 1996). During the variance-selection phase, several successive values of \( \sigma \) are tried for a given number of iteration of the basic algorithm. The competing \( \sigma_i \) is rated by their performance in the basic algorithm in terms of cost reduction starting from the same initial point. Each \( \sigma_i \) is computed according to the formula:

\[
\sigma_i = 10^{-i} \sigma_0, \quad \text{for} \quad i = 1, \ldots, 4
\]

and it is allowed for 100/i iterations to give more trials to larger variances. \( \sigma_0 \) is the initial variance and can be determined, e.g. as a spread of the parameters domain:

\[
\sigma_0 = \theta_{\text{max}} - \theta_{\text{min}}
\]

where \( \theta_{\text{max}} \) and \( \theta_{\text{min}} \) are the largest and lowest possible values of parameters, respectively. The best \( \sigma_i \) in terms the lowest value of the cost function is selected for the variance-exploitation phase.

The best parameter set \( \theta_k \) and the variance \( \sigma_i \) are used in the variance-exploitation phase, whilst the algorithm (6) is run typically for one hundred iterations.

The algorithm can be terminated when the maximum number of algorithm iteration \( n_{\text{max}} \) is reached or when assumed accuracy \( J_{\text{min}} \) is obtained. Taking into account local minima, algorithm can be stopped when \( \sigma_4 \) has been selected a given number of times. It means that algorithm get stuck in local minimum and cannot escape its basin of attraction. Apart from its simplicity, the algorithm possesses the property of global convergence. Moreover, adaptive parameters of the algorithm, cause that a chance to get stuck in local minima is decreased.

5. SIMULTANEOUS PERTURBATION

STOCHASTIC APPROXIMATION

In recent years, a growing interest in stochastic optimisation algorithms that do not depend on gradient information or measurements has been observed. This class of algorithms is based on an approximation of the gradient of the loss function. The general form of Stochastic Approximation (SA) recursive procedure is as follows (Spall, 1992):

\[
\theta_{k+1} = \theta_k - \alpha_k g_k(\theta_k)
\]

where \( g_k(\theta_k) \) is the estimate of the gradient \( \partial J / \partial \theta \) based on the measurements of the loss function \( L(\cdot) \). The essential part of (9) is the gradient approximation. The SPSA has all the
elements of $\theta$ randomly perturbed to obtain two measurements $L$, but each component $g_k(\hat{\theta})$ is formed from a ratio involving the individual components in the perturbation vector and the difference in the two corresponding measurements. For two-sided simultaneous perturbation, estimation of gradient is obtained according to the formula (Spall, 1999):

$$g_k(\hat{\theta}) = \frac{L(\hat{\theta} + c_k \Delta_k) - L(\hat{\theta} - c_k \Delta_k)}{2c_k \Delta_k}$$

(10)

where the distribution of the user-specified $p$-dimensional random perturbation vector, $\Delta_k = (\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kp})^T$ is independent and symmetrically distributed about 0 with finite inverse moments $E(|\Delta_{ki}|^{-1})$ for all $k, i$. One of the possible distributions that satisfies these conditions is the symmetric Bernoulli ±1. Two commonly used distributions that do not satisfy these conditions are the uniform and normal ones. The rich bibliography presents sufficient conditions for convergence of the SPSA ($\hat{\theta} \rightarrow \theta^*$ in the stochastical almost sure sense). However, the efficiency of the SPSA depends on the shape of the $J(\theta)$, the values of gain sequences $\{a_k\}$ and $\{c_k\}$ and distribution of the $\{\Delta_{ki}\}$. The choice of the gain sequences is critical to the performance of the algorithm. In the SPSA, the gain sequences are calculated as follows (Spall, 1992):

$$a_k = \frac{a}{(A + k)^{\alpha}} \quad c_k = \frac{c}{K^{\gamma}}$$

(11)

where $a$, $c$, $A$, $\alpha$ and $\gamma$ are non-negative coefficients. To apply the SPSA to global optimisation, it is needed to use a stepwise (slowly decaying) sequence $\{c_k\}$ (Spall, 1999).

6. SIMULATION STUDY

All the considered training methods have been implemented in Borland C++ Builder Enterprise Suite Ver. 5.0. The simulations have been performed using the PC Computer with Athlon K7 550 processor and 128 MB RAM. To check efficiency of the training methods, modelling of the actuator behaviour in the first section of the sugar evaporation station at the Lublin Sugar Factory is studied.

6.1 Sugar Factory actuator

In Lublin Sugar Factory, Poland, a sucrose juice is concentrated in a multiple-stage evaporator to produce a syrup. The liquor goes through a series of five stages of vapourisers, and in each passage its sucrose concentration increases. The sugar evaporation control should be performed in such a way that the energy used is minimised to achieve the required quality of the final product.

The actuator to be diagnosed is the valve at the inlet of the first section of the evaporation station.

For this valve two measurements are available: $LC51\text{.}03\text{.}CV$ denotes the control value to the valve on the juice inlet to the evaporation section (the actuator input), and $F51\text{.}01$ is the juice flow on the inlet to the evaporation station (the actuation). With these two signals the neural model of the actuator can be defined as:

$$F51\text{.}01 = F_N(LC51\text{.}03\text{.}CV)$$

(12)

where $F_N$ denotes the nonlinear function.

Experiment. During an experiment, a neural model of a structure $N^2_{5,1}$ (two processing layers, one input neuron, five neurons in hidden layer and one output), was trained using in turn the EDBP, ARS and SPSA methods. Each neuron has the first filter order and hyperbolic tangent activation function. This particular network structure was chosen experimentally using a trial-and-error method. The model of the valve is identified using real process data from the sugar factory recorded during the sugar campaign in October 2000. In the sugar factory control system, the sampling time is equal to 10 s. Thus, during one work shift (6 hours) approximately 2160 training samples per one monitored process variable are collected. For many industrial processes, measurement noise is of a high frequency (Lissane Elhaq et al., 1999). Therefore, to eliminate this noise, a low pass filter of the Butterworth type of the second order was used. Moreover, the input samples were normalized to zero mean and unit standard deviation. In turn, the output data should be transformed taking into consideration the response range of the output neurons. For the hyperbolic tangent activation function, this range is [-1.1]. To perform such kind of transformation, the simple linear scaling can be used. Additionally, to avoid saturation of the activation functions, the output was transformed into the range [-0.8,0.8]. It is necessary to notice that, if the network is used with other data sets, it will be required to memorise maximum and minimum values of the training sequence. To carry out experiments, two data sets were used. The first set, containing 500 samples, was used for training and another one, containing 1000 samples, was used to check the generalisation ability of the networks.

EDBP algorithm. The algorithm was run over 20 times with different initial points. The learning process was carried out off-line performing 5000 steps. To speed up the convergence of learning, the adaptive learning rate is used. The initial value of the learning rate $\eta$ is 0.005. The obtained accuracy is 0.098. To check quality of the modelling, the neural model is tested using another data set of 1000 samples. Figure 2 shows the validation phase, with the output of the actuator (black) and the output of the neural model (grey). As it can be seen, the generalization abilities of the dynamic
network are quite good.

**ARS algorithm.** Many experiments were performed to find the best value of initial variance $\sigma_0$. Eventually, this value was found to be $\sigma_0 = 0.05$. With this initial variance, the algorithm was carried out for 200 iterations. The modelling results for the testing set are presented in Fig. 3. The characteristics of the algorithm are included in Table 1. The ARS is time consuming, but for the training set it can find a better solution than the EDBP. The influence of the initial network parameters are examined also. The most frequently used range of the parameters values is $[-1; 1]$. The simulations show that more narrow intervals e.g. $[-0.7; 0.7]$ or $[-0.5; 0.5]$ assure faster convergence.

**SPSA algorithm.** This algorithm is a simple and very fast procedure. However, choice of the proper parameters is not a trivial problem. There are 5 parameters which have a crucial influence on convergence of the SPSA. In spite of speed of the algorithm, the user should spend a lot of time to select proper values. Sometimes it is very difficult to find good values and algorithm fails. The experiment was carried out for 7500 iterations using the following parameters $\alpha = 0.001$, $A = 100$, $c = 0.5$, $\alpha = 0.25$ and $\gamma = 0.05$. The modelling results are shown in Fig. 4 for the testing set. The parameter $\gamma$ controls the decreasing ratio of sequence $\{c_k\}$ and is set to a small value to enable a property of global optimisation. Parameter $\alpha$ is set to a very small value to assure convergence of the algorithm. The dynamic neural network is very sensitive to large changes in parameters values (dynamic filters) and large values of $\alpha$ like 0.4 can cause that learning process will be divergent.

Taking into account that the first value of the sequence $\alpha_k$ is small, the parameter $\alpha$ is set to 0.25 (the optimal value is 1, Spall (1999) proposes to use 0.602). In spite of difficulties in selection of the network parameters, the modelling results are quite good. Moreover, generalisation ability for this case is better than for both the EDBP and the ARS (see table 1).

**Comparison of the methods.** All examined methods need different numbers of floating-point operations per one iteration as well as different numbers of network evaluations, in order to calculate values of the loss function. The characteristics of the learning methods are shown in Table 1. The best accuracy for the training set was obtained using the ARS. This accuracy was reached at the lowest number of iteration but during one algorithm step it is required much more floating-point operations than the others algorithms. It is caused by a large number of network evaluations. Therefore, the learning time for this algorithm is the greatest. A slightly worse result was achieved using the SPSA and the worst quality was obtained using the EDBP. The simplest algorithm here, taking into account the number of floating-point operations per one iteration, is the SPSA. However, the SPSA approximates the gradient and it is needed to perform more steps to obtain a similar accuracy as for the gradient based one. In this example, the actuator is described by nonlinear relation and the algorithms belonging to the global optimisation techniques performed their task with better quality than gradient based one. Simultaneously, the SPSA is much faster than the ARS. Concerning the similar accuracy of the training, the generalisation ability of the neural model trained by the SPSA is better than

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>EDBP</th>
<th>ARS</th>
<th>SPSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning time</td>
<td>12.79 min</td>
<td>33.9 min</td>
<td>16.1 min</td>
</tr>
<tr>
<td>Number of iterations</td>
<td>5 000</td>
<td>200</td>
<td>7 500</td>
</tr>
<tr>
<td>SSE – training</td>
<td>0.098</td>
<td>0.07</td>
<td>0.0754</td>
</tr>
<tr>
<td>SSE – testing</td>
<td>0.564</td>
<td>0.647</td>
<td>0.377</td>
</tr>
<tr>
<td>Floating operations</td>
<td>$3.1 \cdot 10^3$</td>
<td>$1.6 \cdot 10^3$</td>
<td>$2.5 \cdot 10^3$</td>
</tr>
<tr>
<td>Network evaluations</td>
<td>5 000</td>
<td>61 600</td>
<td>15 000</td>
</tr>
<tr>
<td>Float. oper./iteration</td>
<td>$6.2 \cdot 10^5$</td>
<td>$8 \cdot 10^7$</td>
<td>$3.3 \cdot 10^5$</td>
</tr>
<tr>
<td>Net. eval./iteration</td>
<td>1</td>
<td>308</td>
<td>2</td>
</tr>
</tbody>
</table>
the neural model trained by the ARS. Thus, the best neural model was obtained using the SPSA algorithm.

**Fault detection.** One of the most known fault detection and isolation method is technique based on a residual generation. First, a model of the diagnosed process should be designed. After that a residual can be calculated by comparison system and model outputs. When a process is healthy, a residual should be near zero. In the case when a process is faulty, a residual should generate large deviation from zero (e.g. exceeding out a fixed threshold).

The residual generation using neural model trained with the SPSA for the normal operation conditions is shown in Fig. 5. As it can be seen, the quality of the modelling has a crucial importance for the fault detection sensitivity. If the process will be poor identified it can result in a large number of false alarms in the normal operation conditions.

![Residual signal generated using neural model trained with the SPSA algorithm.](image)

Fig. 5. Residual signal generated using neural model trained with the SPSA algorithm.

7. CONCLUDING REMARKS

The main objective of this work was application of the stochastic approaches to train the dynamic neural network and performing comparative studies of the proposed methods and the off-line version of the gradient based algorithm, using real process data. The performed simulations show that stochastic approaches can be an effective alternative to gradient based methods. The ARS is a very simple algorithm and it can be very useful in practice for engineers, because the user should only determine one parameter to start the optimisation procedure. In contrary, the SPSA is much faster than the ARS, but to start the optimisation process, five parameters should be determined. To define these values, the user should possesses a quite large knowledge about this method, to use it properly. Taking into account the property of global optimisation, both the stochastic approaches can be effectively used for modelling of nonlinear processes and fault diagnosis of industrial processes.

8. REFERENCES


