Proceedings of the 17th International Conference on Force, Mass, Torque and Pressure Measurements,
IMEKO TC3, 17-21 Sept. 2001, Istanbul, Turkey

A Neural System for an Automatic Weighing

Mustafa Danacı and Şeref Sağıroğlu
Erciyes University, Eng. Faculty, Computer Eng. Dept., Kayseri/TURKEY

Abstract

Multilayered perceptron neural networks are employed to model a weighing system to estimate the applied mass accurately while the system is still in the transient mode. This is achieved through the networks trained with the backpropagation and the extended delta-bar-delta algorithms to compare the performance of the networks. The results obtained from neural models show that the both neural models are found successful especially if the data sets are noisy.

1. Introduction

A number of works have been done to achieve the masses with the weighing systems. [1-6]. Transient effects were shortened with the adaptive filtering but the effects were still exist [1-3]. Non-Linear Regression (NLR) was used to model the output waveform of the platform in short time. This was faster than the previous work [4].

Further work on this topic was undertaken using simpler sub-model where they have indicated that the number of unknown parameters can be reduced and therefore a good initial guess could be made for fast weighing [5].

Model based techniques required accurate models for dynamic weighing systems. The difficulty might occur in real-time operation [2-6]. Artificial neural networks (ANNs) were applied to model the weighing system to achieve the mass accurately and fast. A multilayered perceptron network (MPN) trained with the standard backpropagation was applied but it has large number of weights [6].

In this work, the structures of MPNs are simplified with the use of two different learning algorithms.

2. Weighing System

Figure 1 illustrates a typical weighing system. The weighing platform response is governed by the solution to the second order differential equation,

$$(m(t) + ms)\frac{\partial^2}{\partial t^2} y(t) + C\frac{\partial}{\partial t} y(t) + K\, y(t) = g\, m(t) \qquad (1)$$

where *m*(*t*), *ms*, *K*, and *C* represent the applied mass, the platform mass, the spring constant and the damping coefficient. respectively. *y*(*t*) is the resulting away from its initial position of the platform, and *g* is the gravitational constant.

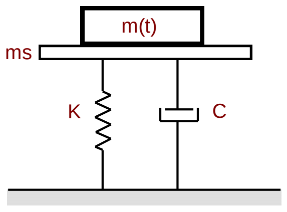

Figure 1. Weighing platform.

The weighing system considered in this study is subject to a sudden disturbance and consequently the step-input function is assumed as the input signal.

3. Multilayered Perceptron Networks

Multilayered perceptron networks (MPNs) are the simplest and therefore most commonly used ANN architectures. In this paper, they have been adapted for the estimation of the masses applied to the weighing platform. MPNs are trained with the two supervised learning algorithms, the extended delta-bar-delta (EDBD) and the backpropagation (BP), in this work. As shown in Figure 2, an MPN consists of three layers: an input layer, an output layer and an intermediate or hidden layer.

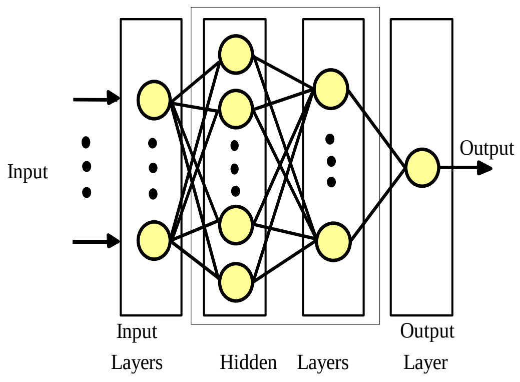

Figure 2. The MPN model.

Neurons, shown with circles in Figure 2, in the input layer only act as buffers for distributing the input signals x_i to neurons in the hidden layer. Each neuron *j* in the hidden layer sums up its input signals x_i after weighting them with the strengths of the respective connections w_{ji} from the input layer and computes its output y_j as a function *f* of the sum:

$$y_j = f(\sum w_{ji} x_i) \tag{2}$$

f can be a simple threshold function, a sigmoidal or hyperbolic tangent function. The output of neurons in the output layer is computed similarly.

Training a network consists of adjusting weights of the network using two learning algorithms, the BP and the EDBD [8]. A learning algorithm gives the change Δw_{ji}(k) in the weight of a connection between neurons *i* and *j*. In the following section, the two learning algorithms used in this study have been explained briefly.

3.1. Back-Propagation (BP):

The algorithm [8-10] is the most commonly adopted MPN training algorithm. It is a gradient descent algorithm and gives the change $\Delta w_{ji}(k)$ in the weight of a connection between neuron *i* and *j* as follows,

$$\Delta w_{ji}(k) = \eta \delta_j x_i + \alpha \Delta w_{ji}(k-1) \qquad (3)$$

where x_i is the input, η is a parameter called the learning coefficient, α is the momentum coefficient, and δ_j is a factor depending on whether neuron *j* is an output neuron or a hidden neuron.

Training an MPN by the BP to compute *y* involves presenting it sequentially with different training tuples. Differences between the target output and the actual output of the MPN are back-propagated through the network to adapt its weights using eqn.(3). The adaptation is carried out after the presentation of each tuple.

Each training epoch is completed after all tuples (patterns) in the training set have been applied to the networks.

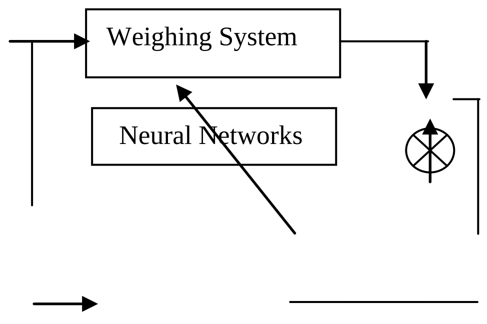

Figure 3. Neural modelling for weighing system.

3.2. Extended Delta-Bar-Delta (EDBD):

As its name implies, this algorithm is an extension of DBD (EDBD) [8]. Aimed at speeding up the training of MPNs, EDBD is based on the hypothesis that a learning coefficient and a momentum coefficient suitable for one weight may not be appropriate for all weights. By assigning a learning coefficient and a momentum coefficient to each weight and permitting it to change over time, more freedom is introduced to facilitate convergence towards a min. value of $E(w_{ji})$. The changes in weights are calculated as:

$$\Delta w_{ji}(k) = -\alpha_{ji}(k)\frac{\partial E(k)}{\partial w_{ji}(k)} + \mu_{ji}(k)\Delta w_{ji}(k-1) \qquad (4)$$

where $\alpha_{ji}(k)$ and $\mu_{ji}(k)$ are learning and momentum coefficients, respectively.

The use of momentum in EDBD is one of the differences between it and DBD. The learning coefficient change is given as:

$$\Delta\alpha_{ji}(k) = \begin{cases} A_\alpha \exp\left(-\gamma_\alpha \left|D_{ji}(k)\right|\right) & \text{if } D_{ji}(k-1)\dfrac{\partial E}{\partial w_{ji}(k)} > 0 \\ -\varphi_\alpha \alpha_{ji}(k) & \text{if } D_{ji}(k-1)\dfrac{\partial E}{\partial w_{ji}(k)} < 0 \\ 0 & \textit{otherwise} \end{cases} \qquad (5)$$

where A_α, γ_α and φ_α are the positive constants. $D_{ji}(k-1)$ represents a weighted average of

$\dfrac{\partial E}{\partial w_{ji}(k-1)}$ and $\dfrac{\partial E}{\partial w_{ji}(k-2)}$ given by

$$D_{ji}(k-1) = (1-\theta)\frac{\partial E}{\partial w_{ji}(k-1)} + \theta\,\frac{\partial E}{\partial w_{ji}(k-2)}$$

The momentum coefficient change is obtained as:

$$\Delta\mu_{ji}(k) = \begin{cases} A_{\mu}\exp\left(-\gamma_{\mu}\left|D_{ji}(k)\right|\right) & \text{if} \quad D_{ji}(k-1)\dfrac{\partial E}{\partial w_{ji}(k)} > 0 \\ -\varphi_{\mu}\mu_{ji}(k) & \text{if} \quad D_{ji}(k-1)\dfrac{\partial E}{\partial w_{ji}(k)} < 0 \\ 0 & \textit{otherwise} \end{cases} \qquad (6)$$

where A_μ, γ_μ and φ_μ are the positive constants. Note that increments in the learning coefficient are not constant, but vary as an exponentially decreasing function of the magnitude of the weighted average gradient component $D_{ji}(k)$. To prevent oscillations in the values of the weights, $\alpha_{ji}(k)$ and $\mu_{ji}(k)$ are kept below preset upper bounds α_{max} and μ_{max}.

4. MPN Results

An input-output model of MPN describes the dynamic system based on input and output data sets. The model assumes that the new system output can be predicted by the past inputs of the neural system. If the system is further supposed to be deterministic, time invariant, single-input-single-output, the input-output model becomes:

$$y(n) = f(x(n), x(n-1), \ldots, x(n-100)) \qquad (7)$$

where $x(n)$ and $y(n)$ represent the input-output pair of the system at time t, positive integer N is the number of past input samples and f is the static non-linear function which maps the past inputs to a new output. The system in equation (7) can be represented as in Figure 3.

Two types of training and test data patterns were generated from the computer simulation to train neural models. Each input pattern consists of 100 samples, $y(n)$, $y(n\text{-}1)$,...,$y(n\text{-}100)$. This is the half of the previous work. The parameters in all simulations are selected as: K=1000 N/mm, C=50N/(mm/sec), ms=0kg, g=10mm/sec^2, the sampling interval ts=0.02msec., the masses applied covering the range $m(t)$=5,10,...,100kg. Initial conditions are zero (the platform parameters and the sampling intervals are the same as in [5,6].

The first type of data patterns consists of the noise-free (NF) patterns and the second type of data patterns also consists of noisy-data (ND) with the 3% for the MPN training and recall, respectively. The both data types were constructed from 20 and 8 data sets for the MPN training and test purposes.

The MPN architecture has 100 inputs in the input layer, 20 neurons in the hidden layer and one neuron in the output layer. The parameters of the both learning algorithms are: *for BP*, the learning coefficients were set to 0.3 for the hidden layer, and 0.15 for the output layer, and the momentum coefficient was also set to 0.4; *for EDBD*, κ_α=0.095, κ_μ=0.01, γ_μ=0.0, γ_α=0.0, φ_μ=0.01, φ_α=0.1, θ=0.7, λ=0.2, α_{max}=μ_{max}=2.0. The results obtained from this work are presented in Tables 1 and 2.

Table 1. MPN test results for various masses.

Applied Mass (Kg)	MPN Result Mass (Kg)			
	EDBD		BP	
	NF	ND (%3)	NF	ND (%3)
8	8.719	8.270	7.520	7.567
17	16.893	17.005	17.019	17.056
24	23.497	23.558	23.937	23.932
39	39.436	39.395	38.994	39.020

52	51.841	51.877	51.959	51.941
76	76.259	76.225	75.982	76.037
89	89.678	89.658	88.982	88.909
91	91.499	91.492	91.048	90.982

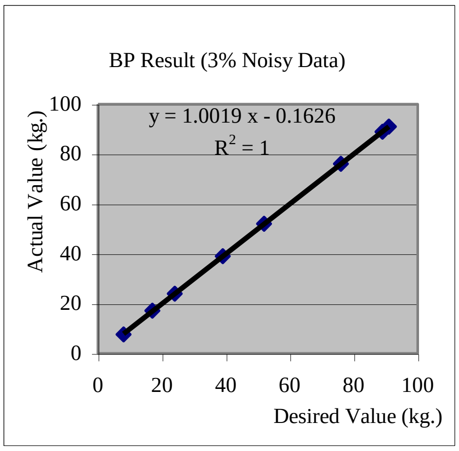

Figure 4. BP result with 3% noisy data.

Table 2. Results of neural models.

Data type	Training algorithm	RMS error in training	RMS error in test
NF	EDBD	0.394	0.471
NF	BP	0.065	0.187
ND	EDBD	0.488	0.378
ND	BP	0.085	0.162

5. Conclusion

MPNs were employed to model the weighing system to estimate the applied mass accurately while the system is still in the transient mode. This is achieved through training the networks from the platform waveforms resulting from applied masses. Two learning algorithms were used to train the networks to compare their performances in obtaining the masses precisely. The results obtained from neural models showed that the both neural models were successful. MPN trained with the BP algorithm was found the most successful training algorithm.

When this work is compared with the previous work [6], the model is simpler and enables more accurate results (see Fig.4). In the previous work, the model had 20101 weights for 200 numbers of samples for each pattern (2% noisy data and $r^2 = 0.9998$). In this work, the model had 2021 weights for 100 number of samples for each pattern, added 3% noisy data and correlation coefficient is $r^2=1$ using BP (Fig.4). The number of weights is reduced 994.6%. This reduction will certainly reduce the cost of the online usage and the process time.

References

[1]. Brendal, A., *High Speed Check Weighing: Part 1-4*, Meas. and Cont. Vol.12, 1979.

[2]. Shu, W-Q., *Dynamic Weighing Under Non-zero Initial Condition*, IEEE Trans.on Inst and Meas., Vol.42, No.4, 1993, p.806-811.

[3]. Shi, J., White, N.M. and Brignell, J.E., *Adaptive Filters Cell Response Correction*, Sensors and Actuators, Vol.37-38, 1993, p.280-285.

[4]. Danaci, M. and Horrocks, D.H., *A Non-linear Regression Technique for Improved Dynamic Weighing*, Proc. Euro.

Conf. on Circuit Theory and Design, Istanbul, Turkey, 1995.

[5]. Horrocks, D.H. and Danaci, M., *High Speed Modelling for Dynamic Weighing by Non-linear Auto Regression*, Non-linear Electromagnetic Systems, (Edited by Moses, A.J. and Basak, A.), IOS Press, UK, 1996, p.548-551.

[6]. Danaci, M., Ozcalik H.R. and Aksoy, M.S., *Artificial Neural Network Techniques for Dynamic Weighing Systems*, 2nd Int. Manufacturing Systems, **1**, Sakarya University, 1998, p.333-341,

[7]. Haykın, S., *Neural Networks: A Comprehensive Foundation*. ISBN 0-02-*Cognition*, **1**, (Cambridge, MA:MIT Press), 1988.

352761-7, Macmillan College Publishing Company, NY, USA, 1994.

[8]. Minai A.A. and R.D. Williams, *Back-propagation Heuristics: A Study of the*

Contact Person for Paper:

Dr**.** Mustafa DANACI

Erciyes University, Computer Eng. Dept.

38039-Kayseri/TURKEY

Tel: +90 352 437 49 01 Ext: 32550

E-mail:danaci@erciyes.edu.tr