BACKWARDS NEURAL NETWORK OPTIMISATION

Kenneth Gock and Jayantha Katupitiya

School of Mechanical and Manufacturing Engineering
University of New South Wales
Sydney, Australia
J.Katupitiya@unsw.edu.au

Abstract: The problem of predicting multivariable process inputs for a given set of process outputs is solved by training a combination of partitioned feedforward backpropagation neural networks. Training of a single multi-layer network produces large unacceptable errors in the predictions due to the absence of monotonic process output functions. However, by configuring a system with parallel partitioned networks linked with a single output network, significant improvement in accuracy of the prediction model was achieved. Percentage errors in the prediction were found to be reduced from a maximum of 73% using a single network system to 38% using parallel networks linked with a single output network, for the worst affected process variable. The improvement in accuracy is largely dependent on the method of partitioning, the training algorithm and filtering of the training data.

Keywords: partitioned networks, multi-valued functions, neural network inversion

1. INTRODUCTION

In a production process it is often required to predict process inputs given a specific set of process outputs. Using neural networks (Hagan *et al.*, 1996) a process can be modelled to facilitate such predictions. Through the cause and effect phenomenon a forwards network can be modelled fairly accurately. In this paper, the terms *forwards network* is used to describe a network that has the process inputs - the so-called factory settings as network inputs and the process outputs – the so-called product parameters as the network outputs. A *backwards network* has the process outputs as the network inputs and the process inputs as the network outputs. Due to the inputs and outputs having non-invertible dependencies (Viharos and Monostori, 1999), the backwards network has unacceptably large output errors. The current procedure to solve this problem is as follows. First train a forwards network. Then apply an arbitrary process input vector and obtain the process output vector using the network. By comparing this output with the desired output, an error vector representing the process outputs can be obtained. These errors are then back propagated to the input later. The resulting error at the input layer is then use to correct the originally used arbitrary process input vector. This procedure is repeated until convergence is reached. ((Hoskins *et al.*, 1992), (Linden and Kindermann, 1989) & (Davis *et al.*, September 1995)). In the majority of applications it is the prediction of process inputs which is of interest to the user. The alternative of a trained backwards network to the inversion of a forwards network is discussed in this paper. The innovations in the optimisation of the backwards network to provide reliable and accurate results will be detailed. Note that the large number of process inputs (22) and outputs (8) pose a significant challenge to solving the problem.

2. SETTING UP OF THE NETWORK

Prior to optimising the backwards network, preliminary trials were conducted to determine the number of neurons, number of layers and the type of transfer functions to be used. Networks were setup with 'tansig' transfer functions for hidden layer(s) and 'purelin' for the output layer. The effects of multi-layer networks of 2, 3 and 4 layers were investigated.

[1] Partially supported by Memcor Australia Pty. Ltd

To optimise the number of neurons in each hidden layer, a programme was written to progressively increase or decrease the number of neurons in each of the hidden layers using the mean square error as the measure of accuracy. Other researchers used a similar method of increasing the number of neurons in each layer to reduce the error (Rangwala and Dornfield, (1989). Networks were optimised based on a process which had 22 process inputs and 8 process outputs. For clarity only the process output which had the most variation amongst the training data compared to the target data is shown in Fig.1.

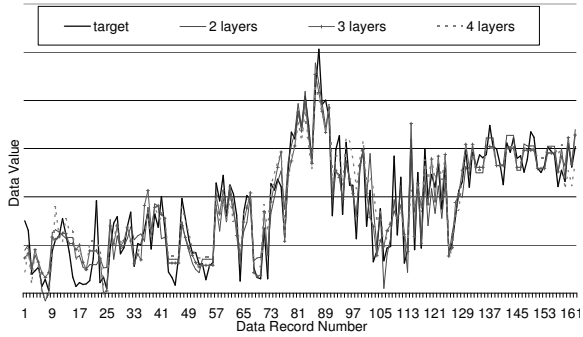

Fig.1. Effect of number of layers on accuracy.

Fit of the graph show that a 3 layer network fits the best to the target data. It was concluded that a neural network consisting of 2 hidden and 1 output layers (3 layers), was found to provide the best model for this nonlinear process. The majority of errors were lower than for either the 2 or 4 layer systems. The same result was found to be true also by Cybenko (Cybenko, 1987) who concluded that a 3-layer network was the most flexible in modelling any arbitrary process.

3. TRAINING OF THE BACKWARDS NETWORK

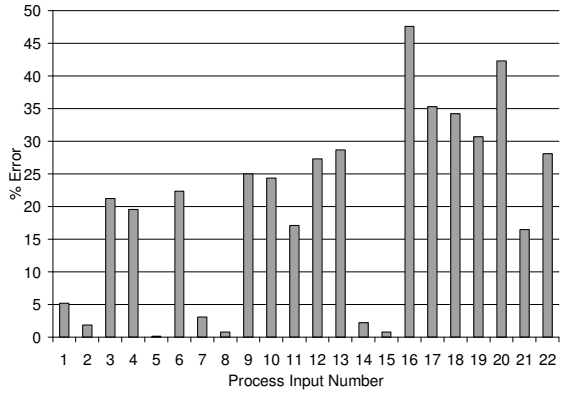

Fig.2. Output errors from a single network.

For this training, data from the same process used to investigate the number of layers for the forward network, was used to train the backwards network. As before there were 22 process inputs and 8 process outputs. A backwards network was trained using 3 layers, two 'tansig' transfer functions in the hidden layers and a 'purelin' transfer function in the output layer. With such a large number of process inputs, unacceptable errors resulted in predictions using a single backwards network. The errors resulted in predicting the 22 process inputs given the 8 process outputs, are shown in Fig.3. The prediction process used 8 process outputs which were *not* included in the training data. The percentage error was determined by comparing the output of the model with the actual process inputs.

As seen in Fig.2 the errors were quite large. The number of epochs used in training was 600. This was set based on the change in mean square error of the process input values (network outputs). The change in value reduced logarithmically with the greatest change from 1 to 200 epochs, after which the improvements were minimal. Research by Rangwala and Dornfield also showed similar results (Rangwala and Dornfield (1989)).

4. ERROR INVESTIGATION

To investigate the cause of the inaccuracies, a known arithmetic function was modelled using neural networks. For a given $f(x)$ value, the model was to predict x. The function $f(x)$ used is shown in Fig.3. Depending on the shape of the function being trained, there may not be a unique process input value for each process output value entered into the backwards network. For instance in Fig.3, for a target value of 1.0, two possible process input values of 120 and 182 are possible. For the backwards network to produce a unique output, there must be more constraints in the form of network inputs.

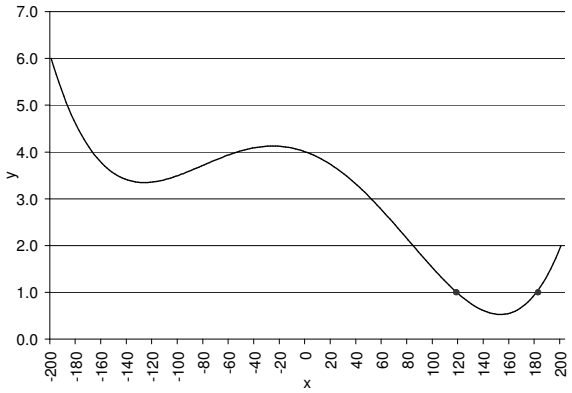

Fig.3. Multi-valued network output

For the process data there is a fixed number of inputs and outputs to the process. Hence it was not possible to produce more constraining conditions for the backwards network. The alternative was to reduce the number of network outputs (process inputs) which are outputted by the backwards network. This can be achieved by partitioning the process outputs into smaller sets and training them to certain process inputs. This study claims the partitioning of the networks as its innovation.

5. BACKWARDS NETWORK PARTITIONING

To improve accuracy of the backwards network, partitioning of the training data to train smaller networks (by reducing the number of outputs per partitioned network) should avoid the issue of non-unique sets of process input values outputted by the overall backwards network. The question of how to partition will be first looked into before addressing the effects on accuracy of partitioning. There are a few areas which must be determined to partition a process; (i) number of partitions, (ii) network configuration and, (iii) partitioning criteria.

Table 1. Correlation Based Partitioning

Number of Partitions	Correlation Partitioning	Mean Square Error	Max Error (%)
1	nil	238	48
2	1: <0.7; 2: ≥0.7	174	32
3	1: <0.7; 2: ≥0.7 & <0.98 3: ≥0.98	144	36
4	1: <0.6 2: ≥0.6 & <0.8 3: ≥0.8 & <0.99 4: ≥0.99	159	42
5	1: <0.4 2: ≥0.4 & <0.5 3: ≥0.5 & <0.6 4: ≥0.6 & <0.7 5: ≥0.7	171	57

To determine the optimum number of partitions, the level of correlation between process inputs will be used as the criteria to determine the split in the networks. The partitioned networks will use an output network to link the outputs of each partitioned network. Having an output network will account for interrelationships between the partitioned network outputs. Depending on the number of partitions, the process inputs will be split according to specific ranges of correlation.

Network optimisation runs were conducted for single networks through to 5 partition splits. Table 1 shows the correlation ranges that were used to partition the process inputs for training of the backwards networks.

The networks were trained on one set of data then verification runs were conducted on a second set of data not used in the training process. By comparing the process inputs outputted by the model to the actual values in the verification data the error of the model was measured. As can be seen in Fig.4, even though the mean square error of the 3, 4 and 5-partition networks were lower than the 2-partition network, it was the 2-partition network which gave the smallest percentage peak error amongst the predicted process inputs. As it is desired to reduce the error as much as possible for the prediction of all the process inputs, the 2-partition system will be used in discussions in the remainder of this paper.

The partitioning in two different configurations (Fig.5) will be investigated in this section.

Configuration 1 (partitioned network) is to have independent backwards networks which may or may not output the same process inputs. Outputs of these independent networks are then linked by a final network which outputs all the process

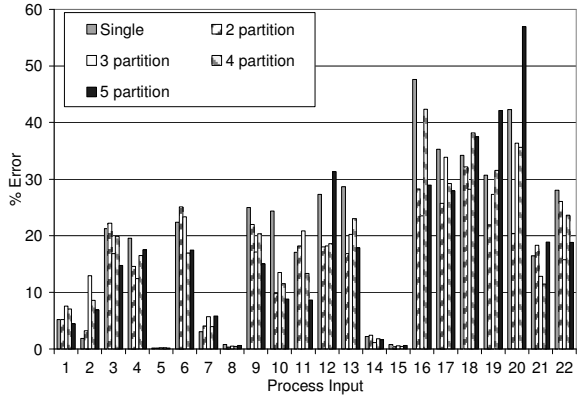

Fig.4. The effect of number of partitions on accuracy

inputs. Hashem (Hashem, 1997) conducted tests which proves that parallel networks joined together by a linear output network does indeed improve network accuracy. However, in his studies the parallel networks all had the same inputs and outputs but simply had different initial training values for weights and biases. The output network was used to provide a weighted average of the different parallel networks. In contrast to Hashem's work, the partitioned networks considered here do not have the same inputs and outputs and the output network is not to average the different results of the parallel networks but to provide nonlinear interrelationships between the different network outputs. Each of the parallel networks generate a unique set of outputs.

Configuration 2 (multiple network) is to produce a model which consists of independent backwards networks each outputting a specific set of process inputs.

The prediction error for the two configurations are shown in Fig.6. It can be seen that configuration 2 had a greater error in predicting the process inputs. This is because with the separate networks there is no output network which provides the interrelationship between different process inputs. Also each separate network outputs different process inputs to avoid duplication and ultimately conflicting values of the same output.

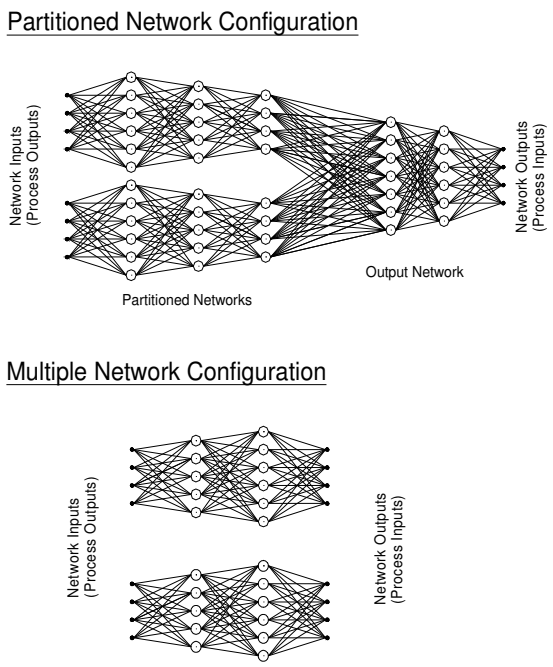

Fig.5. Partitioning configurations

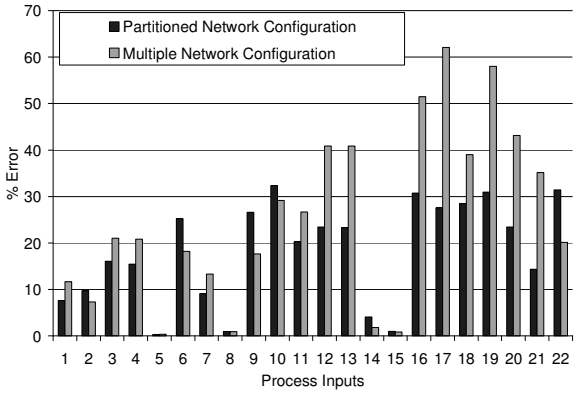

Fig.6. The effect of network configurations on accuracy

The last step is to investigate various methods of partitioning a backwards network. Below are some partitioning methods used to group the process inputs:

Correlation: The process inputs were partitioned into 2 groups with correlation <0.7 and correlation ≥0.7. By grouping according to correlation the relationship between inputs would not be lost and this would also eliminate interference from unrelated process inputs.

Multi-valued Behaviour: Each process input was varied from minimum to maximum with the other process inputs being kept constant at the average values. A neural network trained in the forwards direction was used to generate process outputs for the different data sets. Inputs which generated a curve having 2 or more possible process input values for a given process output were grouped together. The process output with the worst error was chosen as the y-axis to plot against process inputs on the x-axis. Fig.7 shows an output number plotted against a varying process input on the x-axis. This behaviour was observed for 12 out of 22 process inputs.

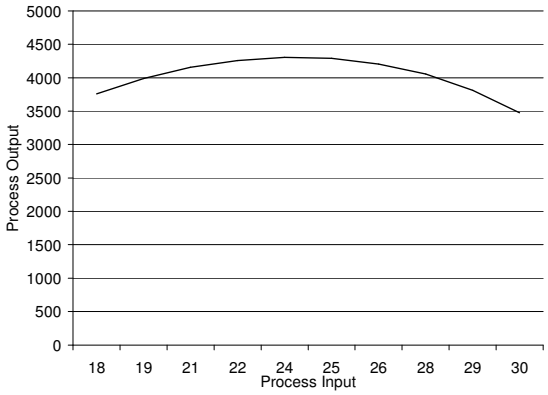

Fig.7. Multi-valued process inputs

Network per Process Input: All process outputs trained to single input - one backwards network per process input trained to all process outputs (8 off) to provide as many constraints as possible to define each process input. This method is based on the assumption of method 2 above, of a process output being multi-valued and there could be multiple process inputs with similar behaviour for a given process output. By having more process output conditions specified there is a higher possibility of having a unique set of process inputs for a given set of process outputs.

Maximum Magnitude: Similar to Multi-valued Behaviour, but inputs generating similar magnitudes of output error were grouped together. This was to determine if the inaccurate process inputs were affecting the more accurate ones.

Input Magnitude: Networks split according to the magnitudes of inputs from min to max values. Group 1 for magnitudes 0 to 100, Group 2 for 100+. To investigate possible inaccuracies introduced in the normalising process. Accuracy of smaller magnitude inputs may be less since normalisation is done across the minimum and maximum range of all inputs for the network.

Input Range: As networks are trained on normalised values between -1 and 1, inputs with similar input ranges were grouped together to provide constraints on non-singular outputs across the whole range of inputs. Input groupings were for input ranges <30 and ≥30.

All trials were conducted on the same training data set. To verify the accuracy of the networks, data not used in training were applied to the networks. Fig.8 shows the results of the different partitioning methods. Included for comparison are the errors of a single backwards network (no partitioning). The most significant reduction in prediction errors of the backwards network were seen when the process inputs were partitioned according to correlation. All other techniques listed above failed to provide any

significant improvement to the accuracy of the predicted results. This is interesting as Hashem's (Hashem, 1997) studies specifically mention that high correlation between network outputs is a good warning to collinearity of outputs which lead to inversion round-off errors and sensitivity to small variations in data.

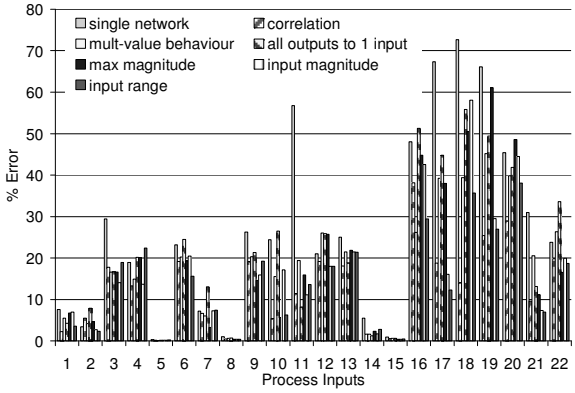

Fig.8. Accuracies of partitioning methods

6. DATA FILTERING & MANIPULATION

To further improve the performance of the backwards network, different methods of filtering or manipulating the data were trialled. In the methods where magnitudes of the training data were grouped together according to similar ranges or were re-scaled to have similar ranges, the approach was to minimise the possibility of round-off errors during the data normalising stage before training. During normalising all the values in the input and output matrices are normalised between -1 and 1 based on the minimum and maximum values of each of the matrices. By keeping magnitudes similar then the level of inaccuracies generated from the smaller values will be reduced. Below are some of the methods tried:

Method 1: The process inputs and outputs were grouped according to their magnitude. The inputs and outputs were partitioned into 2 groups such that values with higher magnitude values were in one group and the lower magnitude ones in the other. To account for the different combinations of high and low magnitude inputs and outputs, 4 networks were produced which were then linked together by a single output network. Network arrangement was similar to Configuration 1 of Fig.5 except there were 4 partitioned networks instead of 2.

Method 2: The process inputs were grouped with correlation <0.7 and with correlation ≥0.7 with training data scaled to largest I/O range and offset to give same minimum and maximum value for all inputs and outputs. Outputs from the trained networks were then re-scaled to represent actual values.

Method 3: The process inputs were grouped with correlation <0.7 and ≥0.7. Training records with outlying results were deleted based on process experience of the operator.

Method 4: The process inputs were grouped with correlation <0.7 and ≥0.7. Process variables which had errors greater than 25% when compared to the target were omitted in the next network training. That is, process input columns 6, 16, 17, 18 & 22 of Fig.9 not included in training of networks.

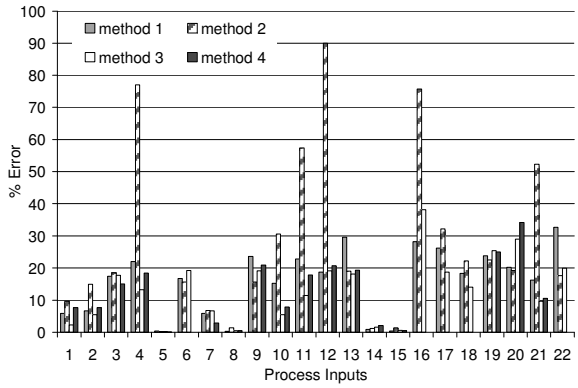

Fig.9. Data filtering/manipulation on accuracy

7. TRAINING ALGORITHMS ON ACCURACY

On establishing the network configuration to provide the most accurate model, the last step will be to determine the training algorithm to improve the model accuracy. Three algorithms will be look at; (i) Gradient decent algorithm with momentum (traingdm), (ii) One step secant algorithm (trainoss) and, (iii) BFGS quasi-Newton back propagation algorithm (trainbfg). The same training data set, consisting of only 153 records, was used to train three backwards networks, one network for each of the training algorithms. All results of the prior section used the gradient descent algorithm with momentum due to it's speed of execution. The time for training using the 'trainoss' algorithm was generally 50% more than for 'traingdm'. However, 'trainbfg' algorithm over 3000% longer to complete due to the extra storage and computations performed per epoch. As shown in Fig.10, the error in the resulting networks for each of the training algorithms shows significant improvement in 'trainbfg' over the 'traingdm' algorithm when network verification was conducted using the training data. However, as shown in Fig.11, when data not used in training was applied to the networks 'traingdm' returned the more accurate results due to convergence to the global minimum (Rangwala and Dornfield (1989)). This can be attributed to the 'trainoss' and 'trainbfg' algorithms being over-trained to the training data. Over-training of networks were found by Hoo et al (Hoo *et al.,* 2002) to provide a model with the inability to generalise when presented with a data set different to the one used in training. Hence any variation to the training data would return large errors. Based on the

fact that the networks are used to predict process inputs not conducted experimentally (untrained data), the 'traingdm' algorithm is the best choice providing significantly improved network accuracy over 'trainoss' and 'trainbfg'. It has the additional advantage of requiring less computations and memory during training than the other two algorithms.

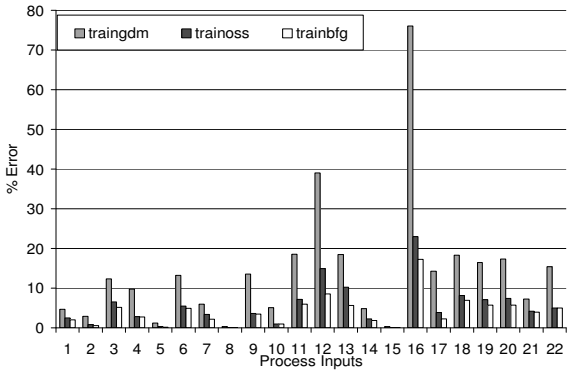

Fig.10. Accuracy using training data

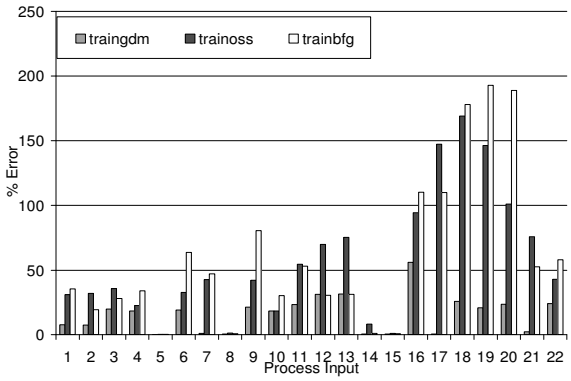

Fig.11. Accuracy using non-training data

8. CONCLUSIONS

The study indicated that a number of steps are necessary to improve the accuracy of a neural network that represents a complex process which is intended to predict the process inputs for a desired set of process outputs. The multi-values nature of the functions relating the process inputs to outputs cause considerable difficulties in the prediction process. As a first step towards improvement, the outliers of the training data must be identified and eliminated. Secondly, it was found that it is best to use 'tansig' transfer functions for hidden layers and 'purelin' for the output layers. Next, the rate of reduction of mean square error can be used to determine the most suitable number of neurons for a hidden layer. The gradient descent algorithm with momentum is computationally very efficient and does not lead to overtraining. The networks trained using this method showed robustness. Using two split networks based on network input correlation gives best results. A third network is necessary to link the two split networks to model the interrelationships between

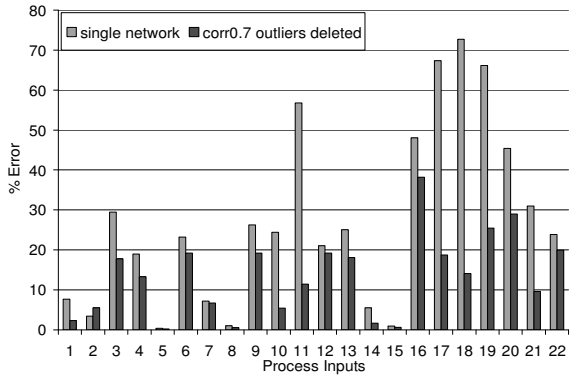

Fig.12. Comparison of partitioned and single network accuracies

networks. The comparisons shown in Fig.12 prove the validity of the above factors.

9. REFERENCES

Cybenko G. (1987). Approximation by superpositions of a sigmoidal function. *Mathematics Of Control Signal Systems* **2**, 303-314.

Davis D., Z. Chen, J. Hwang, L. Tsang and E. Njoku (September 1995). Solving inverse problems by bayesian iterative inversion of a forward model with applications to parameter mapping using smmr remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing* **33, No.5**, 1182-1193.

Hagan, M., H. Demuth and M. Beale (1996). *Neural Network Design*. PWS Publishing Company, Boston.

Hashem S. (1997). Optimal linear combinations of neural networks. *Neural Networks* **10, No.4**, 599-614.

Hoo K., E. Sinzinger, M. Piovoso (2002). Improvements in the predictive capability of neural networks. *Journal of Process Contro* **12**, 193-202.

Hoskins D., J. Hwang and J. Vagners (1992). Iterative inversion of neural networks and its application to adaptive control. *IEEE Transactions on Neural Networks* **3**, 292-301.

Linden A. and J. Kindermann (1989). Inversion of multilayer nets. In: *Proc. Int. Joint Conf. Neural Networks* **2**, 425-430.

Rangwala S. and D. Dornfield (1989). Learning and optimization of machining operations using computing abilities of neural networks. *IEEE Transactions on Systems, Man and Cybernetics* **19**, **No.2**, 299-314.

Viharos Z. and L. Monostori (1999). Automatic input-output configuration and generation of ANN-based process models and their application in machining. *Lecture Notes of Artificial Intelligence – Multiple Approaches to Intelligent Systems, Cairo, Egypt.* Springer Computer Science Book, Springer-Verlag Heidelberg. 659-668.