Z. Boger and H. Guterman, Knowledge extraction from artificial neural networks models.
Invited paper, IEEE Systems, Man and Cybernetics Conference, Orlando, Oct. 1997.

Knowledge Extraction from Artificial Neural Networks Models

Zvi Boger*

Intelligent Process Control Systems
P.O. Box 15053, Be'er-Sheva 84003, Israel
e-mail: zboger @ bgumail.bgu.ac.il

Hugo Guterman

Department of Electrical and Computer Engineering
Ben-Gurion University
Be'er-Sheva 84105, Israel
e-mail: hugo@bguee.ee.bgu.ac.il

ABSTRACT

The paper describes the development and application of several techniques for knowledge extraction from trained ANN models, such as the identification of redundant inputs and hidden neurons, deriving of causal relationships between inputs and outputs, and analysis of the hidden neuron behavior in classification ANN. Example of the application of these techniques is given of the faulty LED display benchmark. References of the application of these techniques are given of diverse large scale ANN models of industrial processes.

1. INTRODUCTION

Artificial neural networks (ANN) were suggested, almost 10 years ago, for modeling the behavior of complex systems in industrial plants [1,2]. However, while widely used by financial and commercial companies to model and forecast complex economic situations, most of the published papers on industrial plant modeling are of small scale, and ANN modeling is not yet widely accepted in industry. One of the possible reasons was the slow learning or non convergence of large scale networks. Now, advances in ANN training algorithms and the availability of low-cost, high-speed computing power have made possible the ANN modeling of large-scale industrial systems. Some of the obstacles to the acceptance of artificial neural networks (ANN) in modeling are caused by concerns of using

* On a sabbatical leave, Institute for System Research, University of Maryland, College Park, MD 20742. E-mail boger@isr.umd.edu, phone 301-405-1241

"black box" models, without understanding the structure of the ANN. Other objections are more of psychological nature, as experts in their fields feel that their knowledge may be redundant if the ANN may model a system from only historical data. To gain confidence and acceptance of the use of these ANN modeling techniques, there is a need to demonstrate that the analysis of a trained ANN can capture system "knowledge" convincingly. If some of this knowledge is already known, the experts are more ready to accept, or at least to examine, new knowledge extracted by the ANN.

The types and forms of system knowledge can vary from simple relationships between the system inputs and outputs, through expert rules, sometimes based of fuzzy variables, to complex first-principle models or Expert Systems. A special form of knowledge, fault detection, is very important in real time industrial situations.

This paper describes our experience in modeling large scale systems and extracting some knowledge from the trained ANN models using relatively simple techniques. These are:

1. Training large ANN, beginning from non-random initial connection weights;
2. Identification of the relevant inputs of a system;
3. Optimizing the number of the neurons in the hidden layer;
4. Calculation of causal indices (CI), for learning some rules of the relations between each input and output.
5. Using the outputs of the hidden layer neurons for fault detection and classification;

The first three techniques have been developed by the authors already in 1990 [3-4], but the details were not published, as they were incorporated in the commercial TURBO-NEURON software package, [5]. With these algorithms we trained ANN models of industrial and other large systems with hundreds of inputs and outputs by the then available PC computers in a fraction of the time and the number of training epochs usually reported. The CI method is also not new, as it was first published in 1990, [6]. Using the hidden neurons behavior as a source of information was already reported in 1990 too [7], and system knowledge was extracted from ANNs in 1989 [8].

Industrial modeling and knowledge acquisition application results using these techniques were published in 1992 [9], and the non-random initial connection algorithm was published in 1993 [11]. The updated original paper [4] on the reduction of ANN structure is being prepared now for submission, with described applications to chemical instrumentation spectra analysis.

In this paper we describe the ANN reduction algorithms, as well as the knowledge extraction techniques used for making the ANN black-box more transparent to the users. Because of space limitations, a review of other existing ANN reduction and knowledge extraction methods is omitted, a benchmark data is used as example, and the reader is referred to some of our published industrial and other large scale applications [12-15].

2. ANN REDUCTION TECHNIQUE

When training large-scale ANN, the network size and structure are major concerns. The training effort increases with the number of connections in the ANN, and it is known that there is an optimal number of neurons in the hidden layer. The generalization capacity decreases with the ratio of the number of connections to the number of training examples. Thus ANN pruning is employed to reduce the ANN size. Many techniques of ANN pruning are available, but as seen in recent papers, the search for good pruning method continues. In principle, most of the methods start by training an ANN with large number of neurons in the hidden layer(s), which will be referred to as hidden neurons. After, or during, the training they eliminate either redundant neurons or the least important connections in the ANN.

In this paper an alternative method is described. We take advantage of the ability of a non-random initial connection weights algorithm to train easily large ANNs. Thus all the available inputs are initially used. A quasi-optimal number of neurons in the (one) hidden layer is calculated from the training data. After the ANN is trained, the redundant inputs and hidden neurons are identified and a smaller ANN is trained with a reduced set of inputs and hidden neurons. This procedure can be repeated until no improvement is gained by further reduction of the number of inputs. A feed-forward network architecture with three layers (input-hidden-output) is assumed. The sigmoid activation function id used, although radial base activation function has been also used. The training method used here is the improved conjugate gradient (CG) optimization method [16], modified to include a prior estimation of the number of hidden layer nodes and selection of initial weights by principal component analysis (PCA) of the training set, [3]. More information on PCA analysis can be found in [17].

These modifications of the basic back-propagation algorithm significantly reduce the length of the training cycle. Improvements derive from three factors:
1. The architecture of the hidden layer is established prior to learning based on the intrinsic dimensionality of the training data, eliminating the need for trial and error in the number of hidden nodes;
2. A non-random initial condition, usually within the range of attraction of the global minimum of the error, is calculated based on the PCA analysis, avoiding repeated training runs with different random initial conditions;
3 The learning rate and momentum factor are automatically adjusted during the training by conjugate gradient formula for optimal learning progress.

Once the initial (unreduced) ANN is obtained, there is still the need to reduce the model complexity. One important motivation is to construct a parsimonious model with an optimal trade-off between fitting accuracy and the number of adjustable parameters in the model. By removing unnecessary structures in the network, over-fitting a limited set of process data with a model containing too many degrees of freedom is avoided. The reduced network model also highlights the dominant underlying process relationships, represented by the remaining inputs, nodes and connections in the reduced network.

When the PCA-CG algorithm is used, the number of nodes in the hidden layer is selected in advance to represent the intrinsic dimensionality of the data set. Inputs that are linearly correlated do not contribute independent information and thus the information content of the correlated inputs can be captured in lower dimensions. The amount of information lost in the projection of the input data into a lower-dimensional space is quantified by the fraction of the variance of the original data not represented in the reduced space. Typically, we specify as many hidden nodes as dimensions needed to capture 70-90% of the variance of the input data set. This immediately places limits on the number of hidden nodes in the initial network, and subsequently it remains only to remove nodes in the hidden layer that represent dimensions of the hidden layer that do not specifically contribute to the output of interest. In our experience, the suggested number of hidden neuron is already very close to the optimal size before the network reduction procedures are applied. Thus, most of the effort was directed at reducing the number of network inputs, (the number of outputs is not a factor since they are fixed by the requirements of the model). Currently, the statistical network reduction method does not address removal of connections between nodes retained in the reduced model. Nonetheless, many connections are eliminated when the non-relevant inputs and hidden layer nodes are removed.

A procedure that can avoid the time consuming stages of ANN pruning by the other published techniques is shown in the Appendix. This method is based on elementary statistical considerations involving the variation of the inputs to the hidden layer nodes in the trained network. If the variance of the total (summed) input to a hidden node, calculated over the training set as a whole, is small, then the node contributes little more than a constant bias to the subsequent layer. The node can be removed and its contribution can be included in the biases of the output nodes. Likewise, a hidden node can be subsumed by output layer biases if variation of the total input is contained within the flat asymptotes at either extreme of the sigmoid function, as in such case the variance is "compressed" to a small value. Nodes of the hidden layer that show the least activity in terms of output variation in response to the different training examples are prime candidates for pruning. Since the outputs of these nodes are directly recaptured as biases in the output layer, retraining after each node is removed is not required, making the procedure practical for large networks.

A similar analysis of variance can determine the importance of network inputs. Inputs that make relatively small contributions to the variance at the input of the hidden layer serve only as biasing constants and can be replaced by fixed biases at the hidden layer nodes. When analyzing a given input, its collective effect on all hidden layer nodes is of interest, rather than the effects on individual hidden nodes, since removing the input de-couples it simultaneously from all hidden nodes.

Two methods of retraining, if desired to re-tune the network, can be carried out after groups of hidden nodes and input nodes are removed. One is to repeat the

use of the PCA method of ANN design with only the significant group of inputs, changing the value of information content criterion so that an optimal number of hidden nodes are re-calculated. The second is to include the output values of the removed input and hidden nodes in the biases of the next layer of nodes and continue training from that starting point. Only the first method is usually used, as the PCA-CG algorithm accelerates the training process so much that not much time is lost by abandoning the already developed larger ANN. In reducing the number of relevant wave-lengths to infra-red and x-ray fluorescence spectra identification, several such repeats were possible, reducing in one case the number of inputs from 600 to the significant 9 wave-length in 7 re-trainings [18,19].

In training ANN, there is always the possibility of getting into local minima. The usual way to avoid it is re-starting the training from different random initial connection weights. As the PCA-CG always starts from the same initial conditions (for a given data set and ANN configuration), we developed a proprietary technique for getting out of local minima. Recently an algorithm for improving this ability was developed by imposing gradually relaxed constraints on the connection weight changes during the training.

3. KNOWLEDGE EXTRACTION METHODS

Once a reduced ANN is trained, some knowledge may be readily extracted from the resulting ANN structure and behavior. One obvious source is the identity of the inputs with high and low relevance value. If a reliable system model can be trained without the discarded inputs, maybe those inputs need not be closely monitored. In fault detection and analysis models, the inputs with large relevance values are more likely to be related to the fault causes or outcomes.

The causal index method is another easily, somewhat qualitatively, method for rule extraction. The CI is calculated as the sum of the product of all "pathways" between each input to each output,

$$CI = \sum_{j=1}^{h} W_{kj} * W_{ji} \qquad (1)$$

where there are h hidden neurons, W_{kj} are the connection weights from hidden neuron j to output k, W_{ji} are the connection weights between input i to hidden neuron j.

Plotting the CI for each output as a function of the inputs' number reveals the direction (positive or negative) and the relative magnitude of the relationship of the inputs on the particular output. Although somewhat heuristic, it is more reliable than the local sensitivity checks, as it is based on the whole ANN trained on all the available states. It has to be remembered, however, that the interactions between inputs are not readily visible.

The neurons in the hidden layer are supposed to learn concepts, which then should be orthogonal to each other, as in the PCA dimensionality reduction. This led to examining the possibility that a unique binary pattern of the hidden neurons could indicate the correctness of the ANN classification. A similar suggestion is given in [20], calculating

the maximum information content of the hidden neuron outputs using entropy concepts. Entropy is calculated as the sum over all hidden neurons of p * ln(p), when p are the hidden neurons outputs. Experience with well-trained classifying ANNs showed this orthogonality by plotting the output of a hidden neuron against the output of another hidden neuron. In most cases the plotted points were on the plot corners and axes. Thus p = [0, 1] decreases the entropy. In a recent paper [21] it was shown that the incorrect binary pattern of the hidden neurons could identify false classifications of chemical substances by an "artificial nose" sensor, when the trained ANN was presented with a highly noisy and drifting input spectra.

.

Plotting the output of one hidden neuron against that of another one, for all examples, reveals those examples that are not conforming to this pattern, suggesting outliers or faulty inputs. Even in non-classifying ANN, the orthogonal nature of the PCA selected hidden neurons' outputs will give a plot in which most of the points are on the axes, thus identifying possible non regular examples.

Auto-associative ANN (AANN), in which the ANN predicts as output it's own inputs, has been suggested as a way to identify and correct faulty or abnormal inputs, useful in a continuous sensor monitoring in industrial plant. Usually a five-layer AANN is designed, in which the second and fourth layers are calculated by PCA techniques [22]. However, the regular three-layer ANNs trained with the PCA-CG algorithm are also capable of identifying abnormal situations [23].

4. A FAULTY LED DISPLAY EXAMPLE

As an example of the ability of the relevance finding algorithms the training of a faulty Light Emitting Diode (LED) display example will be described. A data base containing 2048 examples of 24 binary inputs is given in [24]. The first seven of these inputs represent the on/off binary state of a seven-segment LED numeral display (Fig. 1). The other 17 inputs are random zeros or ones.

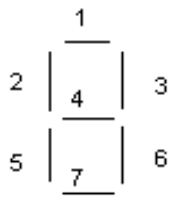

Figure 1. 7 segment numeral LED display

The output of the data base is the correct numeral. However, there is a 10% chance that the state of the inputs is faulty, 1 instead of 0, or 0 instead of 1. It means that in some case there might be even multiple faults in an example. A 70:30 random partition of the database was made into training and testing data. Preprocessing was done by converting the inputs to the [-1 1] range, and the outputs to the [0.1 - 0.9] range, to utilize better the sigmoid function characteristics. The A 25-6-10 ANN was trained by the PCA-CG algorithm to classify the examples into their correct 10 classes. A constrained connection weight change algorithm, and a proprietary algorithm, improved the ability to avoid or exit local minima. The ANN converged after 79 training epochs, to give 74.6% correct classification (77.6%

and 72.0% for the training and testing sets, respectively). The analytical Baysian solution is 74% correct classification, but it does not take into consideration the other random inputs [25]. The inputs were sorted by their relevance, and inputs with the largest relevance values cumulatively summed to 90% are retained for re-training. As seen in Fig. 2, inputs 1 - 7 have the highest relevance values, and thus identified correctly.

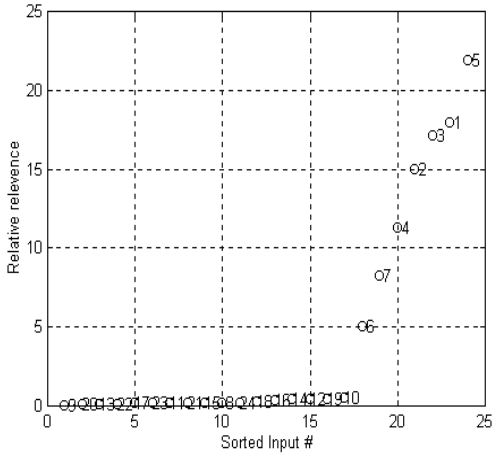

Figure 2. Identification of the significant inputs to the faulty LED model

The causal index clearly identifies the inputs that, when present, indicates which numerals are allowed and which are not. For instance, to be numeral 6, segment 5 has to be on, and if segment 3 is on, it cannot be numeral 6. (Fig 3). The CI relevance for the classification is correct, as in the four instances that segment 5 is present, only in numeral 6 segment 3 is not present. Also in the two instances that segment 3 is not present, only in numeral 6 segment 5 is present.

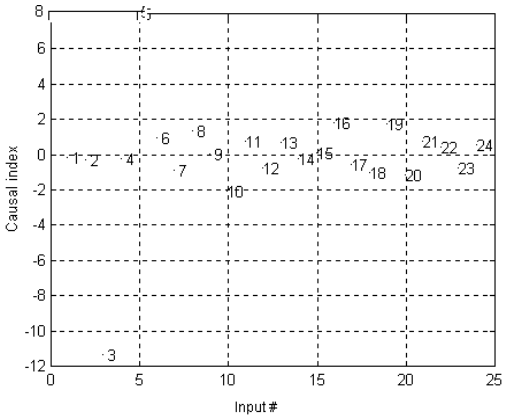

Figure 3. Causal indices of numeral 6.

The two most significant of such inhibiting segments, as derived from the causal index, are presented in table 1.

Numeral	0	1	2	3	4	5	6	7	8	9
Input #	4	1	6	2	1	3	3	7	(11)	5
CI	-9	-14	-9	-11	-12	-13	-11	-9	-1	-8

Numeral	0	1	2	3	4	5	6	7	8	9
Input #	6	2	2	5	5	5	(10)	4	(18)	(21)
CI	(-2)	-11	-8	-8	-6	-9	(-2)	-8	(-1)	(-1)

Table 1. Inhibiting rules derived from causal indices of the reduced ANN model of the faulty LED.

For example, the inhibiting rules for numeral 3 are the segments 2 and 5. In parentheses are the cases when the numeral does not have a second inhibiting segment,(numerals 6,9,0) - the causal index is close to zero. When there is no inhibiting segment (numeral 8), the inhibiting segment is not even of a significant input.

5. USING THE HIDDEN NEURONS FOR FAULT DETECTION

To demonstrate this on the faulty LED example, a reduced 7-6-10 ANN was retrained on the same sets, with the same

algorithms, in 62 epochs. The apparent correct classification percentage is 77.7% for the training set, 71.3% for the testing set. These figures are actually better, as some of the inverted bits errors result in the correct input pattern for a different numeral.

Fig. 4 shows the plot of the first and second hidden layer neurons' outputs, of the test examples. Most of the outputs values are on the corners or axis of the plot. The ones distant from the axis are suspect as mis-classified. The actual mis-classified points are marked with an 'x', and it may be seen that only a few are not marked so. Similar results are possible by plotting other hidden layer neurons' outputs. Although not conclusive, this method may help identify possible mis-classified outputs for further examination.

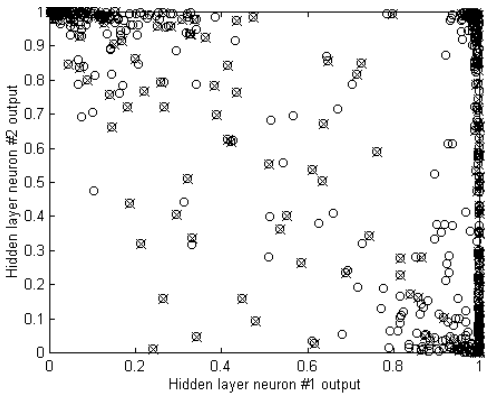

Figure 4. Identification of suspect examples of the faulty LED

To demonstrate this on the LED example, a reduced 7-5-7 ANN was retrained on the same sets, in 62 epochs.
The classification error was 70.0% and 69.9% of the training and testing sets, respectively. However, additional analysis of the "predicted" output segments showed that the correct classification rate was could be increased to 90.4% and 93.0% respectively. Some of the faulty inputs gave a pattern that was correct for a different numeral. Some of the predicted outputs was different by only one segment from the correct numeral, and thus easily corrected. Additional correction may be deduced from the examination of the hidden layer neurons' output pattern.

Out of the 182 examples with "corrected" inputs, 46 examples had the correct hidden neurons' output pattern, thus confirming the correction. In 37 cases the hidden layer neurons' outputs corresponded to the predicted outputs, even if the nominal numeral was not identified In 41 cases, the corrected inputs were not confirmed. For the rest of the cases, 76, no help was provided by this analysis, but this fact made the classification suspect. Thus of the 8.9% "corrected" cases, about half could be considered more likely to be true.

6. CONCLUSIONS

As the rather simple faulty LED example results show, the PCA-CG non-random initial connection weight algorithm enabled the easy training of the classifying ANN with the theoretical expected correct classification. The ANN reduction algorithm identified the relevant inputs. The rules extracted from the trained ANN are the right ones for this example. The analysis of the hidden neurons' outputs helps in identifying, and even correcting, faulty inputs and classifications.

As the referenced applications results show, these techniques have been applied to large scale ANN modeling of industrial and other interesting systems.

Thus some of the “black box” perception of the ANN may be reduced, leading to an increased safe use by industry of ANNs in modeling, and eventually in control.

7. REFERENCES

[1] J.C. Hoskins and D. M. Himmelblau, "Artificial neural networks models of knowledge representation in chemical engineering," *Comput. Chem. Engng.*, Vol. 12, 1988, pp. 881-887.

[2] V. Bhat and T.J. Mc Avoy, “Use of neural nets for dynamic modeling and control of chemical systems.” *Comput. Chem. Engng.*, Vol. 14, 1990, pp. 573-583.

[3] H. Guterman, “Application of principal component analysis to the design of neural networks,” Laboratory for Intelligent Systems in Process Engineering, MIT, Unpublished work, 1990.

[4] H. Guterman, Z. Boger and M.A. Kramer, "Neural network reduction: Application of a statistical relevance approach," Laboratory for Intelligent Systems in Process Engineering, MIT, Unpublished work, 1990. In preparation for submission to *Chemometrics and Intelligent Laboratory Systems*.

[5] TURBO - NEURON is a product of NEXSYS - Expert Neural Networks Ltd., TEMED Science Based Industrial Park, DN Arava, Israel 86800.

[6] Baba,. I. Enbutu, and M. Yoda, "Explicit representation of knowledge acquired from plant historical data using neural network," *Proc. Int. Joint Conf. on Neural Networks.* San Diego, Vol. 3, 1990, pp. 155-160.

[7] L. Bochereau and P. Bourgine, “Extraction of semantic features logical rules from a multi-layer neural network.” *Proc. Int. Joint Conf. on Neural Networks.* Washington, Vol. 2, 1990, pp. 579-583.

[8] Y. Niida, T. Tani, T. Hirobe, and I. Kohijima, "Application of neural network to rule extraction from operation data" *Proc. AIChE Annual Meeting*, San Francisco, 1989.

[9] Z. Boger, "Application of neural networks to water and wastewater treatment plant operation," *Trans. of the Instrument Society of America*, Vol. 31, No. 1, 1992, pp. 25-33.

[10] Z. Boger, Y. Argaman, D. Farchile, M. Goldstein, B. Kravitz, E. Simovici, O. Gucci, “Knowledge acquisition in wastewater treatment plants.” *Proc. of the Israeli-German Seminar on Joint Water and Wastewater Research Program*, Karlsruhe, Germany, Paper no. 4, 1992.

[11] H. Guterman, “Application of principal component analysis to the design of neural networks.” *Neural, Parallel & Scientific Computing*, Vol. 2, 1994, pp. 43-54.

[12] Z. Boger and M. Ben-Haim: “Application of neural networks techniques in solvent extraction: Modeling, knowledge acquisition and dynamic prediction.” in D. H. Logsdail and M.J. Slater (eds.): *Solvent Extraction in the Process Industry*, Elsevier Applied Science, Vol. 2, 1993, pp. 1198-1205.

[13] Z. Boger and Z. Karpas, “Application of neural networks for interpretation of ion mobility and x-ray fluorescence spectra”. *Analytica Chimica Acta*, Vol. 292, 1994, pp. 243-251.

[14] Z. Boger, H. Guterman and T. Segal, “Application of large scale artificial neural networks for modeling the response of a naphtha stabilizer distillation train“, *Proc. AIChE Annual*

Meeting, Chicago, 1996. (Available on the Internet http://www.che.wisc.edu/aiche/1996/10b-abstracts/24.html)

[15] Z. Boger, "Experience in developing and analyzing models of industrial plants by large scale artificial neural networks," *Proc. of Int. Conf. Artificial Neural Networks, ICANN'95* Paris, Session A6, 1995.

[16] J. Leonard, and M.A. Kramer, "Improvement of the back-propagation algorithm for training neural networks." *Comput. Chem. Engng.*, Vol. 14**,** 1990, pp. 337-341.

[17] A. Fukunaga and W. Koontz, "Application of Karhunen - Loewe expansion to feature selection and ordering." *IEEE Trans. Comput. C -19* , 1970, pp. 311-320.

[18] Z. Boger, "Artificial neural networks for quantitative stationary spectroscopic measurements," *Proc. of the 10th Israeli Conf. on Artificial Intelligence, Computer Vision and Neural Networks, AICVNN-93,* Ramat-Gan, Israel, 1993, pp. 185-194.

[19] Z. Boger and Z. Karpas, "Use of neural networks for quantitative ion mobility spectrometric measurements," *J. Chem. Info. Comp. Sci.*, Vol. 34, 1994, pp. 576-580,

[20] R. Kamimura and S. Nakanishi, "Hidden information maximization for feature detection and rule discovery," *Network: Computation in Neural Systems,* Vol. 6, 1995, pp. 577-602.

[21] Z. Boger, L. Ratton, T.A. Kunt, T.J. Mc Avoy, R.E. Cavicchi and S. Semancik, "Robust classification of "Artificial Nose" sensor data by artificial neural networks," *Proc. of the IFAC ADCHEM '97 Conference,* Banff, Canada, 1997.

[22] M.A. Kramer, "Auto-associative neural networks," *Comput. Chem. Engng.* Vol. 16, 1992, pp. 313-328.

[23] Z. Boger, "Large scale neural networks - possible applications in nuclear power plants," *Proc. of the ANS Topical Meeting on Nuclear Power Instrumentation, Control, and Man-machine Interface Technologies,* Oak-Ridge, Tennessee, 1993, pp. 624-631.

[24] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees.* Wadsworth International Group, Belmont, California. pp. 43-49, 1984. Available as anonymous ftp from ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

[25] S. Geva, University of Queensland, Brisbane, Australia, personal communication.

APPENDIX: NETWORK REDUCTION ALGORITHM (Guterman *et. al.*, [3])

Basic statistics [A1], provide that the variance of a linear combination of random variables **x** is given by:

$$y = \sum_{i=1}^{n} a_i x_i = \mathbf{AX} \qquad \text{(A1)}$$

$$\mathrm{var}(y) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j r_{ij} = \mathbf{ARA^T} \qquad \text{(A2)}$$

where **R** is an n x n covariance matrix **R** = cov(**x**). If the variable set **x** is un-correlated, the calculation can be further simplified to:

$$\mathrm{var}(y) = \sum_{i=1}^{n} a_i^2 r_{ii} \qquad \text{(A3)}$$

However, in normal industrial plant data the variables are usually correlated in some way, so Eq. (A2) is used in the present procedure.

Defining the matrix $\mathbf{W_H}$ to be the input-hidden layer weight set of the original (trained but unreduced) NN, and taking $\mathbf{x}_p$ to represent the network inputs for the p^{th} training example, then the input to the j^{th} hidden layer node for the p^{th} example is:

$$y_{jp} = \sum_{i=1}^{n} (\mathbf{W_H})_{ji}\, x_{ip} \qquad \text{(A4)}$$

where n is the number of network inputs. Following Eq. (A2), the variance $(V_H)_j$ of the input to hidden layer node j over the full set of training examples is:

$$(V_H)_j = \mathrm{var}(y_j) = (\mathbf{W_H})_j \mathbf{R} (\mathbf{W_H})_j^T \qquad \text{(A5)}$$

where $(\mathbf{W_H})_j$ represents the j^{th} row of $\mathbf{W_H}$ and $\mathbf{R}$ is the covariance matrix for the network inputs $\mathbf{x}$, estimated from the training set. From Eq. (A5), the relative variance of the input to hidden node j can be calculated as:

$$(V_H)_j^{rel} = \frac{(V_H)_j}{\sum_{k=1}^{n_h} (V_H)_k} \qquad \text{(A6)}$$

where n_h is the number of hidden nodes. Eqs. (A5 - A6) are the essential equations for determining the statistical relevance of a hidden node.

Moving to the input relevance analysis, $\mathbf{R}_i$, the i^{th} row of $\mathbf{R}$, is associated with the i^{th} input variable. From Eq. (A5), the contribution of input i to the variance $(V_H)_j$ is:

$$(V_I)_{ji} = (\mathbf{W_H})_{ji} \mathbf{R}_i (\mathbf{W_H})_j^T \qquad \text{(A7)}$$

and the contribution of input i to the total variance of the hidden layer inputs is:

$$(V_I)_i = \sum_{j=1}^{n_h} (\mathbf{W_H})_{ji} \mathbf{R}_i (\mathbf{W_H})_j^T \qquad \text{(A8)}$$

An equivalent, more concise form of Eq. (A8) is:

$$(V_I)_i = (\mathbf{W_H})_i^T \mathbf{W_H} \mathbf{R}_i^T \qquad \text{(A9)}$$

where $(\mathbf{W_H}^T)_i$ is the i^{th} row of the transpose of $\mathbf{W_H}$, i.e. the i^{th} column of $\mathbf{W_H}$ expressed as a row vector. The relative contribution of input i to the variance of the hidden layer inputs can now be calculated:

$$(V_I)_i^{rel} = \frac{\left((V_I)_i\right)}{\sum_{k=1}^{n} (V_I)_k} \qquad \text{(A10)}$$

Eqs. (A9 - A10) are the equations used to determine the statistical relevance of network inputs.

The V_H^{rel} values serve as a check on the significance of particular hidden layer nodes. Nodes with low V_H^{rel} values are good candidates for elimination, and if trial removal yields an improvement in AIC and/or FPE, these nodes can be permanently removed. Calculation of the adjustment of the bias for an output node k compensating for the removal of the

hidden node j is based on the expected value of oj, **E**(oj):

$$\phi_k^{'} = \phi_k + (\mathbf{W_o})_{kj}\,\mathbf{E}(o_j) \qquad \text{(A11)}$$

where $\mathbf{W_O}$ represents the hidden-output layer weight set. A reasonable approximation of Eq. (A11) is given by:

$$\phi_k^{'} = \phi_k + (\mathbf{W_o})_{kj}\, f\Big(\sum_{i=1}^{n} (\mathbf{W_o})_{ji}\mathbf{E}(x_i) + \phi_j\Big) \qquad \text{(A12)}$$

Since Eq. (A12) is approximate, additional training by PCA-CG is used to re-tune the network after hidden nodes are removed. Input variables with low VI^{rel} values do not contribute much information to the network and can be eliminated. Analogous to Eq. (A11) the bias adjustment for hidden layer node j to compensate for removal of input i is:

$$\phi_j^{'} = \phi_j + (\mathbf{W_H})_{ji}\,\mathbf{E}(x_i) \qquad \text{(A13)}$$

Procedurally, one begins by training the NN with the whole data set, with a reasonable value of the information content used for estimating of the number of the hidden nodes, for example, 70%. After training the ANN with the PCA-CG algorithm, the best candidates for removal according to Eqs. (A6) and (A10) are identified. The group of top candidates is removed, and the network is retrained using PCA-CG. On the retraining step, a different information content can be used in selection of the hidden layer architecture, according to the relative variance values of the hidden nodes.

Our experience shows that optimum results are obtained when there is only one hidden node with a relative variance smaller than 10%, so a higher information content is used when the least significant hidden node has relative variance higher than 10%, and a smaller information content is used when more than one hidden node have a relative variance smaller than 10%. After retraining, various criteria can be checked, such as the as the Akaike information theoretic content (AIC) or the final prediction error (FPE), [A2]. Evaluation of the criteria requires only a single forward pass through the training set to determine the prediction error. As long as the FPE and AIC decline, more inputs and/or hidden nodes should be removed (if the AIC has a negative value, it means that the model is over-fitted). When the network is in the range of minimum of the FPE and AIC (the minima many occur at different points), the same analysis of variance procedure may be performed on the reduced net as a further check that the optimal net was obtained.

References:

[A1] A. Vandaele, *"Applied Time Series and Box - Jenkins Models."* Academic Press, 1983.

[A2] L. Ljung, *System Identification - Theory for the User*. Prentice - Hall, 1987.