On the Application and Design of Artificial Neural Networks for Motor Fault Detection—Part II

Mo-Yuen Chow, *Member, IEEE* Robert N. Sharpe, *Student Member, IEEE*
and James C. Hung *Fellow, IEEE*

Abstract—**The emerging technology of artificial neural networks has been successfully used in a variety of areas such as fault detection, control, signal processing, and many others. This paper presents the general design considerations of feedforward artificial neural networks to perform motor fault detection. The paper will first discuss a few noninvasive fault detection techniques, including the parameter estimation approach, human expert approach, etc., and will then lead to the artificial neural network approach. A brief overview of *feedforward nets* and *backpropagation* training algorithm, along with its pseudocodes, will follow. Later sections will explain some of the neural network design considerations such as network performance, network implementation, size of training data set, assignment of training parameter values, and stopping criteria. Finally, a fuzzy logic approach to configure the network structure will be presented.**

I. Design and Training Considerations of the Incipient Fault Detector Artificial Neural Network (IFDANNN)

PART I of this paper [69] gives an overall description of the use of artificial neural networks in motor fault detection applications. In Part II of this paper, we will discuss how to design an artificial neural network for fault detection purposes. In order to select a *good* neural network configuration to perform motor fault detection, there are many factors to be considered. To name a few major considerations, there are: 1) *practical* considerations such as network accuracy, network robustness, and implementation feasibility; 2) *network training* considerations such as determining the input and output variables, choosing the size of the training data set, initializing network weights, choosing training parameter values (such as *learning rate* and *momentum rate*), and selecting training stopping criteria; and 3) *network design* considerations such as determining the number of input and output nodes to be used, the number of hidden layers in the network, and the number of hidden nodes used in each hidden layer. Even though choosing these parameters is still a trial-and-error process, there are some guidelines (or *rules of thumb*) that can be used in the selection of values. We will present some of these general guidelines, and then present a *fuzzy logic* approach to automatically configure a neural network that *optimizes* or *nearly optimizes* all of the network design considerations.

In order to test a network's performance, it is a common practice to choose a set of training data and a set of testing data that are statistically significant and representative of the system under consideration [52]. The training data set is used to train the neural network, while the testing data are used to test the network performance after the network is trained successfully. The input and output data are usually normalized between [0, 1]. The output normalization is necessary because the outputs of the network are bounded between [0, 1] due to the sigmoid function used. The input normalization will increase the numerical stability of the network data process [21].

A. Choice of Learning Parameters—Problem Dependent

As indicated in the pseudocode of the *backpropagation* training algorithm in Table II, Part I of this paper [69], after the network configuration and training data are determined, we need to select the training criteria and training parameters, which include the initialization of network weights, learning rate and momentum rate, and the criteria to stop network training. The choice of these values is sometimes essential to the success of network training. Unfortunately, the choice of these values is generally problem dependent, i.e., there is no generic formula that can be used to choose the parameter values. Nevertheless, some guidelines, which are briefly described below, can be followed as an initial trial. After a few trails, the network designer should have enough experience to set appropriate criteria that suit a given problem.

The initial weights of the neural network play a significant role in the training process of the network. Without *a priori* information about the final weights, it is common practice to initialize all weights *randomly* with small absolute values, e.g., between $[-0.1, 0.1]$. Also, methods to improve and optimize the *backpropagation* algorithm for faster and more efficient network training have been

Manuscript received August 31, 1992; revised November 13, 1992. This work was supported in part by the National Science Foundation under Grant ECS-8922727 and in part by the U.S. Army Corps of Engineers—Construction Engineering Research Laboratory under Grant 5-30016.

M.-Y. Chow and R. N. Sharpe are with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911.

J. C. Hung is with the Department of Electrical and Computer Engineering, University of Tennessee, Knoxville, TN 37996-2100.

IEEE Log Number 9206799.

studied and developed [21], [22]. One of the more popular methods includes the addition of a momentum term, $\alpha \Delta w_{ji}$ (*itera*) in (23) of [69], to the learning rule to avoid getting trapped in local minima. Typical values of the learning rate and momentum rate are 0.1 and 0.7, respectively. Popular criteria used to stop network training are *sufficiently small* mean-square training error and *sufficiently small* changes in network weights. How *sufficiently small* is usually up to the network designer, and is based on the desired accuracy level of the neural network. For the induction motor fault detection application, we stop network training when either the root mean-square error of the training set or the change in network weights is *sufficiently small* (less than 0.005).

B. A Priori Information about Training Data to Simplify Network Configuration

In order to successfully train a neural network to perform fault detection, input–output training patterns must be chosen correctly. In the posed problem, the steady-state current I and the rotor speed ω, of the motor measurements can be used for detecting the condition of the motor's winding insulation N_c and bearing condition B_c [3]–[5]. The variables I and ω are chosen because of their easy accessibility, reliability, and sensitivity. The values of N_c and B_c for each combination of I and ω are obtained by an expert's heuristic interpretation of the measured data. It is then logical to choose $\{I, \omega\}$ as inputs and $\{N_c, B_c\}$ as outputs of the neural network fault detector. This set of input–output patterns can then be used to train the network to perform satisfactory fault detection.

If *a priori* information about the relationship of the inputs and outputs is available and used correctly, then the network structure and training time can be reduced and the network accuracy can be significantly improved. For example, the plot of B_c versus the inputs $\{I, \omega\}$ (Fig. 1) shows that the boundaries which separate different bearing conditions appear to be quadratic functions of $\{I, \omega\}$. Therefore, if the input space were expanded from $\{I, \omega\}$ to high order $\{I, \omega, I^*\omega, I^2, \omega^2\}$ and used as input to the neural network (Fig. 2), then the network accuracy could be enhanced while reducing both the number of hidden nodes used and its training time. The results of using a conventional net with input $\{I, \omega\}$ and a high-order net with input $\{I, \omega, I^*\omega, I^2, \omega^2\}$ are depicted in Fig. 3. 2-*-Btrn, 2-*-Btst, 5-*-Btrn, 5-*-Btst represent the conventional net training data set, conventional net testing data set, high-order net training data set, and high-order net testing data set, respectively. The results clearly indicate that the high-order net performs better than the conventional net. This expansion of the input space from $\{I, \omega\}$ to $\{I, \omega, I^*\omega, I^2, \omega^2\}$ requires no additional motor measurements, but only a simple manipulation of the available measurements $\{I, \omega\}$. The reason for the improvement in network accuracy is that the error function $E(w)$ in (19) of [69] is a convex function in the high-order input space; thus, the training of w (basically a decent searching algorithm) is much easier and faster when high-order inputs are used. Using *a priori* information to improve network performance is both beneficial and popular in neural network applications [23], [53].

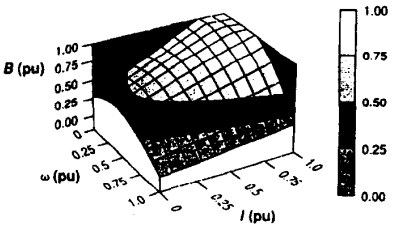

Fig. 1. 3D graph of B as a function of I and ω.

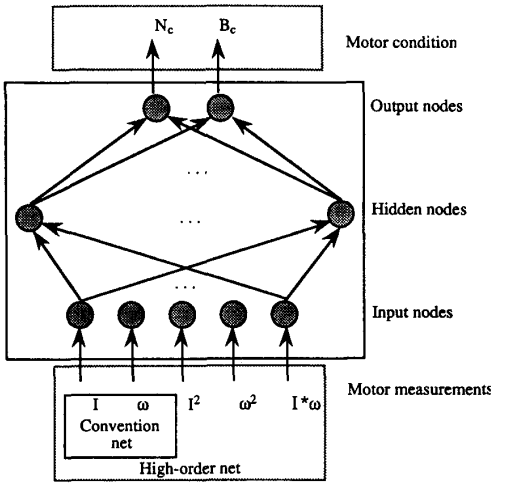

Fig. 2. Configuration of a high-order neural network for incipient fault detection.

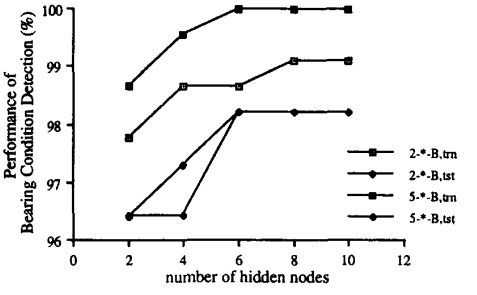

Fig. 3. Performance of the high-order neural network and conventional neural network for bearing fault detection using different number of hidden nodes.

C. Size of Network Training Data Set: Application of Learning Theory

The number of training examples used to train a neural network is sometimes critical to the success of the training process. If the number of training examples is not sufficient, then the network cannot correctly learn the actual input–output relation $\mathcal{M}$ of the system. If the number of training examples is too large, then the network training

time will be longer. References [54] and [55] describe the use of *learning theory* to estimate the number of training examples that is sufficient to train a network for a specific application. The theory is based on the *Principle of Maximum Entropy* [56] to estimate the average number of training examples needed to yield the maximum amount of information for a given network configuration. The results of using *learning theory* to estimate the average number of training patterns required to learn the prediction of the bearing condition B_c are shown in Fig. 4 with respect to different root mean-square error values E for the termination of training. *Learning theory* indicates that, on the average, using 80 training patterns uniformly distributed throughout the input space is sufficient to train the network to correctly predict the condition of the motor bearing B_c. The theoretical results of *learning theory* have been verified with experiments [55]. Thus, unnecessary training patterns, i.e., more than 80 training patterns, and unnecessary training time can be avoided. Training time is critical to some applications such as real-time adaptive neural control [8], [57]. For some applications, such as training the network to perform fault detection, the training can be performed off-line and more training data are preferred over using insufficient training data to achieve greater network accuracy. In this paper, 224 training patterns are used to train the network to perform induction motor fault detection, and 223 testing patterns are used to test the network performance. This number of training examples is sufficient to train all of the network configurations studied in this paper. For more information on the application of *learning theory*, interested readers can refer to [54], [55].

D. Network Robustness Considerations

The neural network fault detection performance is claimed satisfactory if it achieves a certain level, say above 95%, that is acceptable for the specific fault detection application. Since most fault detection schemes are for real-time applications, noise considerations and minor disturbances become important issues. Noise is known to decrease the overall performance of fault detectors [40], [57], [70]. Therefore, methods to suppress noises are needed to enhance the accuracy of fault detector neural networks operating in noisy conditions. One method to do so is to use tapped-delay lines to increase the number of inputs to the neural network [12], as shown in Fig. 5 where k is the sampling time index and n is the number of sampled measurements z used as inputs to the neural network fault detector. For the conventional net, $z = \{I, \omega\}$, while for the high-order net, $z = \{I, \omega, I^*\omega, I^2, \omega^2\}$. Since the incipient fault is a slow time-varying phenomenon compared with the measurement sampling time used, the delayed inputs should all be approximately the same value. This method creates redundant information for the neural network to perform fault detection, which helps to increase its robustness to noise and therefore improve its reliability [38]. The new neural network structure with the tapped-delay lines is called MS–IFDANN,

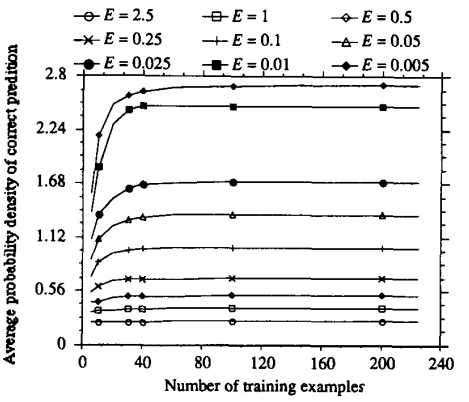

Fig. 4. Network learning curves of bearing condition, where E is the root mean-square training error used as network training stopping criterion.

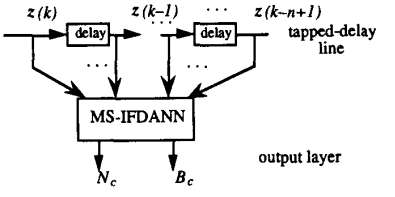

Fig. 5. Block diagram of MS–IFDANN.

or *Multisample Incipient Fault Detector Artificial Neural Network*. The performance of the neural network using tapped-delay lines for the high-order net incipient fault detector is plotted as a function of input noise level in Fig. 6. The figure indicates that the accuracy of the network improves when more data are inputted to the network, even in the presence of large noise variances. The network robustness will be incorporated as one of the network design considerations in Section II.

Another important network design issue is the selection and implementation of the network configuration. We would like to use the smallest number of nodes possible in order to reduce the manufacturing cost of the fault detector neural network, while maintaining the desired level of accuracy and robustness of the fault detector. The next section will present a *fuzzy logic* approach to design an *optimal* network configuration based on four chosen criteria.

II. A Fuzzy Logic Approach to Design an Optimal Neural Network Configuration

Choosing a neural network configuration that can meet all of the design criteria and constraints can be a formidable task. Usually, there will be certain tradeoffs between the various evaluation criteria. Thus, one of the most important steps in selecting an appropriate network configuration is to choose the priority of each design criterion and base the network configuration on these priorities. For instance, is reducing training time more important than improving network accuracy, or is network

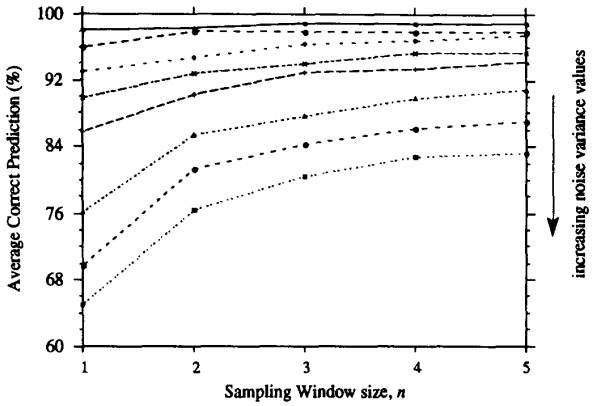

Fig. 6. Accuracy of MS-IFDANN for different noise variance values as a function of sampling window size n.

robustness less important than the size of the neural network?

In the absence of guidelines to select the *best* neural network configuration for a specific application, one approach would be to train all possible network configurations (with different combinations of input, hidden, and output nodes) and select the one that most closely matches the needs of the application. Of course, that is not a feasible method (for it is *too* time consuming), nor is this a technique that engineers should use (because it is nothing more than an exhaustive searching technique that does not utilize the engineer's knowledge and expertise).

However, the *fuzzy logic* approach presented in this paper is a technique that can be used to automatically configure an *optimal* or *near optimal* neural network structure by translating known trends and experience into network modifications, based on *heuristic reasoning* and *linguistic variables* [58], [59]. *Fuzzy logic* [60]–[63] provides a means of transferring heuristic reasoning into mathematics. It can be easily modified and customized to meet the demands of different neural network designers. While a *fuzzy logic* approach for network configuration design does not ensure *the* optimal solution, it will nevertheless select a network configuration that is reasonably close to an optimal solution, with a minimum amount of time and effort.

The motivation behind using fuzzy logic for the design of neural network configurations was realized when the following questions were asked.

1) Is the training error small enough?
2) Is the training time too long?
3) Is the network robust in the presence of input noise?
4) Are too many or too few hidden layers and hidden nodes being used?

All of these questions have "fuzzy" answers in the sense that they contain a certain amount of impression. Fuzzy logic will allow us to quantitatively evaluate the answers to these questions in a systematic manner.

For the induction motor fault detection application, a small training error, a short training time, a robust network, and a small number of hidden nodes were considered to be the first priorities of the network configuration design. Previous experience with neural network designs indicates that

1) network accuracy (reduction in error) increases as the number of hidden nodes increases,
2) network robustness increases as the number of input nodes increases,
3) network training time increases with more nodes in the network, and
4) network implementation costs increase with more nodes in the network.

These statements are based on the trends observed in the training of various network configurations. It is obvious that there are conflicts among the different neural network design factors, and that satisfying a design criterion may imply giving up one or more of the others. In general, it is desired to have a very robust system with a small error value, and one which can be quickly trained with the use of the smallest possible number of nodes. It should be noted that the measure used for the training time is based on an actual CPU time required for training, and not the number of cycles through the training set. Generally, the number of cycles through the training set is decreased with a more complicated network. However, due to the increased complexity of the network, the actual CPU time for training will increase with more sophisticated networks.

A. Relationship Between Robustness and Sensitivity

In general, the network robustness is inversely related to the network sensitivity [57]. Thus, a minimization of the sensitivity will result in a maximization of the robustness. The sensitivity of a network to input perturbations can be defined as the first derivative of the output with respect to the input. The input–output sensitivity matrix $\boldsymbol{S}$ of an artificial neural network with n_j inputs and n_o outputs is given by [57]

$$\boldsymbol{S} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_{no}}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_{no}}{\partial x_2} \\ \vdots & & & \vdots \\ \frac{\partial y_1}{\partial x_{ni}} & \frac{\partial y_2}{\partial x_{ni}} & \cdots & \frac{\partial y_{no}}{\partial x_{ni}} \end{bmatrix} \tag{1a}$$

To obtain a measurement of the network's sensitivity to input noise, the measure S will be defined as the average Frobenius norm square of the sensitivity matrix $\boldsymbol{S}$ across

the training set as shown by [57]

$$S = \frac{\sum_{k=1}^{m} \|\mathbf{S}\|_2^2}{m} = \frac{\sum_{k=1}^{m} \sum_{i=1}^{n_i} \sum_{j=1}^{n_o} s_{ij}^2}{m} \quad (1b)$$

where m represents the number of training patterns and k is used to index the training patterns. The sensitivity measure S will be used as an indicator of the network's robustness to input noise in the organization of the network configuration.

B. A Fuzzy Logic Approach Using Cost Functions

The four questions posed in Section II can be associated with "costs" corresponding to the error (E), training time (T), sensitivity (S), and the number of neurons used (N). These costs will be denoted by J_E, J_T, J_S, and J_N, respectively. One method of quantitatively evaluating a network configuration is to use a cost function approach. Using the four evaluation criteria introduced earlier, a cost function of the form

$$J(n_i, n_h) = q_1 J_E(n_i, n_h) + q_2 J_T(n_i, n_h) + q_3 J_S(n_i, n_h) + q_4 J_N(n_i, n_h) \quad (2)$$

is used. By changing the values of $\boldsymbol{q} = [q_1, q_2, q_3, q_4]^T$, one can tailor the cost function of the problem at hand. For example, if one wanted to emphasize the accuracy and discount the amount of training time, then one would want to increase q_1 relative to q_2. The optimal solution would be the configuration that minimizes the cost function J.

The three-layer feedforward net configuration is used to illustrate the *fuzzy logic* technique for network configuration organization. (The methodology could be easily generalized for a multilayer feedforward net.) The network configuration problem can be posed as: given a set of input–output training patterns, how do we determine the number of inputs n_i, and hidden nodes n_h to minimize the chosen costs? (Usually, the number of outputs n_o is fixed by the training data). The flexibility in the network configuration design comes from the choice of n_i and n_h.

Figure 7(a) shows the trends in the performance measures obtained when fixing n_h and varying n_i. n_i has been normalized between its minimum and maximum admissible values (two and ten, respectively). The error E was not heavily influenced by changes in n_i. For this reason, Fig. 7(a) does not contain a plot of J_E. Similarly, Fig. 7(b) shows the trends obtained when fixing n_i and varying n_h. n_h has been normalized between its minimum and maximum admissible values (two and ten, respectively). The sensitivity S was not heavily influenced by changes in n_h. For this reason, Fig. 7(b) does not contain a plot of J_S.

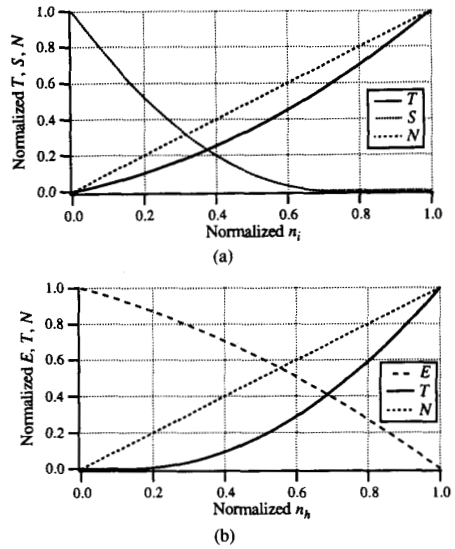

Fig. 7. (a) Normalized T, S, and N versus n_i. (b) Normalized E, T, and N versus n_h.

The observed trends in these graphs can be translated into a fuzzy rule base [64]:

If E LARGE, then LARGE INCREASE n_h.
If T LARGE, then MEDIUM DECREASE n_i.
If T LARGE, then MEDIUM DECREASE n_h.
If S LARGE, then LARGE INCREASE n_i.
If N LARGE, then MEDIUM DECREASE n_i.
If N LARGE, then MEDIUM DECREASE n_h.

These rules are also consistent with our intuition concerning the design of neural network configurations as stated in Section II. The rule base could have been developed based solely on an expert's heuristic knowledge of neural network design. Fig. (7a) and (b) have been presented for illustration and verification purposes.

Based on the rule base, the recommended change in the number of hidden nodes and number of inputs (Δn_i and Δn_h, respectively) can be obtained by the *center of area method* [65]. If the rules are weighted according to the values of q_i in (2), the defuzzification strategy is given by

$$\Delta n = \frac{\sum_{j=1}^{n} q_j \mu_j \times U_j}{\sum_{j=1}^{n} q_j \mu_j} \quad (3)$$

where f_1 is the control parameter assigned to Δn_i and f_2 is the control parameter assigned to Δn_h. Equation (3) differs from the traditional center of area method by the inclusion of the q_j term to weight the fuzzy rules according to the selected preferences. n is assigned to the number of rules contributing to Δn_i and Δn_h (n is equal to three if the rule base presented in this paper is used). μ_j represents a value from zero to one which defines E, T, S, or N's membership [66] in the fuzzy set LARGE corresponding to rule j. These membership functions μ_j

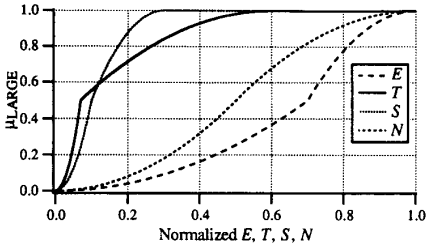

Fig. 8. Membership functions for μ_{LARGE}.

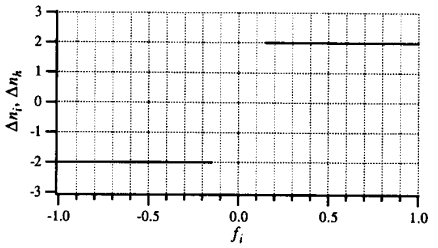

Fig. 9. Quantizer for translating f_i into Δn_i and Δn_h.

map the value of E, T, S, or N into the fuzzy set LARGE. This may be a linear relationship, an S-function [67], or many other possibilities [68]. The S-functions used for $\mu_{\text{LARGE}}(E)$, $\mu_{\text{LARGE}}(T)$, $\mu_{\text{LARGE}}(S)$, and $\mu_{\text{LARGE}}(N)$ are shown in Fig. 8. (E, T, S, and N in Fig. 8 have been normalized between their smallest and largest values for all combinations of the number of inputs $n_i \in \{2, 4, 6, 8, 10\}$ and the number of hidden nodes $n_h \in \{2, 4, 6, 8, 10\}$.) It should be noted that the specific shape of the membership functions is problem dependent. The membership functions shown in Fig. 8 are the result of many refinements to make the convergence of the fuzzy procedure as quick and accurate as possible. However, acceptable membership functions can be obtained by one's own interpretation of what constitutes a large error, a large training time, a large sensitivity, and a large number of neurons for a given application. U_j is the fuzzy control applied to rule j which represents either a LARGE INCREASE or MEDIUM DECREASE in the rule base. Equation (3) evaluates to a real number and will have to be quantized into an integer change in the number of inputs Δn_i or the number of hidden nodes Δn_h. The transition from real to integer values is achieved by using a quantizer as shown in Fig. 9 where a resolution of ± 2 neurons has been used. In the region $-0.15 < f_i < +0.15$, Δn is not specified. This region corresponds to a fuzzy recommendation with a small amount of supporting evidence. If both f_1 and f_2 lie in this region, then the value of f_i with the largest absolute value will be assigned a value of $+2$ or -2 according to the sign of the larger f_i.

Through iterative use of the *fuzzy* rules and (3), the network configuration will be changed to one corresponding to the optimal or near optimal configuration. While the optimal solution is not guaranteed, the *fuzzy* configuration should be acceptably close to the optimal. The flowchart of the fuzzy logic neural network configuration approach is depicted in Fig. 10. The "Exit" condition will test true when the fuzzy procedure tries to drive both n_i and n_h outside their domain (changes in one parameter are allowed if operating on a boundary) or when the rules begin to toggle between two different configurations. After exiting the fuzzy procedure, the network configuration that was trained and had the smallest value of J will be selected as the fuzzy solution.

As an example, a neural network was developed to learn the mapping form $\{I, \omega\}$ to B_c for the induction

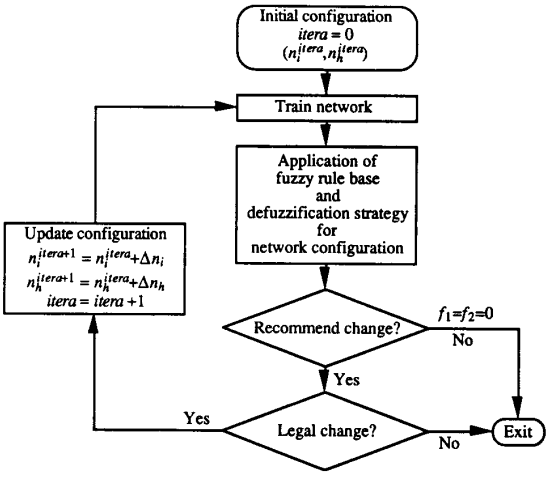

Fig. 10. Flowchart of fuzzy logic neural network configuration process.

motor fault detection application. The network was trained with inputs of $\{I, \omega\}$ as opposed to the higher order net consisting of $\{I, \omega, I^*\omega, I^2, \omega^2\}$ for simplicity of illustration. To determine the optimal solution for various combinations of $\boldsymbol{q} = [q_1, q_2, q_3, q_4]^T$ and to test the accuracy of the *fuzzy logic* approach, the network was trained for all combinations of the number of inputs $n_i \in \{2, 4, 6, 8, 10\}$, number of hidden nodes $n_h \in \{2, 4, 6, 8, 10\}$, and number of outputs $n_o = \{1\}$. The value of n_i is always a multiple of two since we are training with $\{I, \omega\}$. The training of these 25 different network configurations was terminated when there was no change in mean-square training error.

The optimal solution (configuration which minimizes J) for different values of $\boldsymbol{q}$ was found. To compare the fuzzy solution to the one determined by the exhaustive search technique, the procedure was applied to a network with $\boldsymbol{q} = [0.3, 1, 0.7, 0.4]^T$. This choice of $\boldsymbol{q}$ represents a network with an emphasis on the training time and the sensitivity discounting the number of neurons used. The results are presented in Table I, with the fuzzy logic correctly driving the initial $\mathcal{N}_{2,2,1}$ configuration to the optimal $\mathcal{N}_{6,2,1}$ configuration.

With an initial configuration as far away from the optimal as possible, the fuzzy procedure was still able to drive the network configuration to an optimal or near optimal solution. As an example, the results are shown in

TABLE I
CONFIGURING THE MS-IFDANN WITH $\boldsymbol{q} = [0.3, 1, 0.7, 0.4]^T$, $\mathcal{N}_{2,2,1}$ INITIALLY, AND $\mathcal{N}_{6,2,1}$ OPTIMALLY

	k	Configuration	J	Δn_i	Δn_h	Next Configuration
	0	$\mathcal{N}_{2,2,1}$	0.7347	+2	+2	$\mathcal{N}_{4,4,1}$
	1	$\mathcal{N}_{4,4,1}$	0.4826	+2	0	$\mathcal{N}_{6,4,1}$
	2	$\mathcal{N}_{6,4,1}$	0.4240	0	−2	$\mathcal{N}_{6,2,1}$
Optimal	3	$\mathcal{N}_{6,2,1}$	0.3807	+2	+2	$\mathcal{N}_{8,4,1}$
	4	$\mathcal{N}_{8,4,1}$	0.4375	−2	−2	Exit

TABLE II
CONFIGURING THE MS-IFDANN WITH $\boldsymbol{q} = [1, 0.1, 0.7, 0.15]^T$, $\mathcal{N}_{2,2,1}$ INITIALLY, AND $\mathcal{N}_{10,8,1}$ OPTIMALLY

	k	Configuration	J	Δn_i	Δn_h	Next Configuration
	0	$\mathcal{N}_{2,2,1}$	1.4374	+2	+2	$\mathcal{N}_{4,4,1}$
	1	$\mathcal{N}_{4,4,1}$	0.9647	+2	+2	$\mathcal{N}_{6,6,1}$
	2	$\mathcal{N}_{6,6,1}$	0.5916	+2	+2	$\mathcal{N}_{8,8,1}$
	3	$\mathcal{N}_{8,8,1}$	0.5997	+2	0	$\mathcal{N}_{10,8,1}$
Optimal	4	$\mathcal{N}_{10,8,1}$	0.2626	−2	−2	$\mathcal{N}_{8,6,1}$
	5	$\mathcal{N}_{8,6,1}$	0.5976	+2	+2	Exit

Table II for $\boldsymbol{q} = [1, 0.1, 0.7, 0.15]^T$. This represents a network design focusing on the error and the sensitivity discounting the training time and the number of neurons used. For this value of $\boldsymbol{q}$, the optimal network is the $\mathcal{N}_{10,8,1}$ configuration.

The *fuzzy logic* procedure provides a method of systematically changing the network configuration to minimize a given cost function which is based on different design criteria. The major benefit from using a *fuzzy logic* approach is the reduction in time required to obtain a satisfactory network configuration since the time-consuming exhaustive searching can be avoided.

CONCLUSIONS

Part I of this paper has provided an overview of *feedforward nets* and the *backpropagation* training algorithm, along with their respective pseudocodes, and a general methodology for the design of feedforward artificial neural networks to perform motor fault detection. Part II of this paper has discussed some neural network design considerations such as network performance, network implementation, size of training data set, assignment of training parameter values, and stopping criteria. Finally, a fuzzy logic approach to configure the network structure has been presented to automate the network design. Successful results have been obtained from using artificial neural networks on motor fault detection and fuzzy logic in the network configuration design. The emerging technologies are promising for future widespread industrial usage.

REFERENCES

[1] L. Villalobos and S. Gruber, "A system for neural network-based inspection of machined surfaces," *J. Neural Network Computing*, vol. 2, no. 2, pp. 18–30, 1990.

[2] M.-Y. Chow, P. M. Mangum, and S. O. Yee, "A neural network approach to real-time condition monitoring of induction motors," *IEEE Trans. Ind. Electron.*, vol. 38, pp. 448–453, Dec. 1991.

[3] M.-Y. Chow and S. O. Yee, "Application of neural networks to incipient fault detection in induction motors," *J. Neural Network Computing*, vol. 2, no. 3, pp. 26–32, 1990.

[4] ——, "Methodology for on-line incipient fault detection in single-phase squirrel-cage induction motors using artificial neural networks," *IEEE Trans. Energy Conversion*, vol. 6, pp. 536–545, Sept. 1991.

[5] I. E. Alguindigue and R. E. Uhrig, "Vibration monitoring with artifical neural networks," *Proc. SMORN VI Symp. Nucl. Reactor Surveillance and Diagnostics*, Gatlinburg, TN, May 1991.

[6] W. E. Dietz, E. L. Kiech, and M. Ali, "'Jet and rocket engine fault diagnosis in real time," *J. Neural Network Computing*, vol. 1, no. 1, Summer 1989.

[7] M.-Y. Chow, S. O. Yee, and L. S. Taylor, "Recognizing animal-caused faults in power distribution systems using artificial neural networks," presented at the PES SM, 1992, Seattle, WA; also, *IEEE Trans. Power Delivery*, to be published.

[8] W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds. *Neural Networks for Control*. Cambridge, MA: M.I.T. Press, 1990

[9] S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, Mar. 1990.

[10] M.-Y. Chow and R. J. Thomas, "Neural network synchronous machine modeling," presented at the 1989 Int. Symp. Circuits Syst., Portland, OR, May 1989.

[11] M.-Y. Chow and S. O. Yee, "An adaptive backpropagation through time training algorithm for a neural controller," in *Proc. 1991 IEEE Int. Symp. Intell. Contr.*, Arlington, VA, Aug. 1992, pp. 170–175.

[12] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," *Computer Mag.*, Mar. 1988.

[13] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, pp. 1550–1560, Oct. 1990.

[14] W. T. Miller, R. P. Hewes, F. H. Glanz, and L. G. Kraft, "Real-time dynamic control of an industrial manipulator using a neural-network based learning controller,"' *IEEE Trans. Robotics Automation*, vol. 6, pp. 1–9, Feb 1990.

[15] B. P. Yuhas, M. H. Goldstein, Jr., T. J. Sejnowski, and R. E. Jenkins, "Neural network models of sensory integration for improved vowel recognition," *Proc. IEEE*, vol. 78, pp. 1658–1668, Oct. 1990.

[16] S. Z. Lerner and J. R. Deller, Jr., "Speech recognition by a self-organizing feature finder," *Int. J. Neural Syst.* vol. 2, no. 1 & 2, pp. 55–78, 1991.

[17] H. Szu and X. Hartley, "Fast simulated annealing," *Proc. IEEE*, vol. 75, p. 1538, 1987.

[18] H. Szu and K. Scheff, "Simulated annealing feature extraction from occluded and cluttered objects," in *Proc. Int. Joint Conf. Neural Networks*, Washington, DC, Jan. 1990.

[19] G. L. Bilbro, W. E. Snyder, S. J. Garnier, and J. W. Gault, "Mean field annealing: A formalism for constructing GNC-like algorithms," *IEEE Trans. Neural Networks*, vol. 3, pp. 131–18, Jan. 1992.

[20] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral science," Ph.D. dissertation, Harvard Univ., Cambridge, MA, Aug. 1974.

[21] D. E. Rummelhart and J. L. McClelland, *Parallel Distributed Processing*. Cambridge, MA: M.I.T. Press, 1986.

[22] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, Apr. 1987.

[23] Y.-H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley, 1989.

[24] T. Kohonen, "'The self-organizing map," *Proc. IEEE*, vol. 78, pp. 1464–1480, Sept. 1990.

[25] K. Fukushima, "Cognition: A self-organizing multilayered neural network," *Biol. Cybern.*, vol. 20, pp. 121–136, 1975.

[26] J. Reason, "'Boost availability with up-to-date maintenance techniques," *Elec. World*, Aug. 1988.

[27] J. Douglas, J. Edmonds, and J. C. White, "Early warning for hydro generator failure," *EPRI J.*, pp. 31–35, June 1988.

[28] J. E. Timperley, "Incipient fault identification through neutral RF monitoring of large rotating machines," *IEEE Trans. Power App. Syst.*, vol. PAS-102, Mar. 1983.

[29] P. J. Tavner and J. Penman, *Condition Monitoring of Electrical Machines*. New York: Research Studies Press Ltd., Wiley, 1989.

[30] P. F. Albrecht, J. C. Appiarius, R. M. McCoy, E. L. Owen, and D. K. Sharma, "Assessment of the reliability of motors in utility applications—Updated," *IEEE Trans. Energy Conversion*, vol. EC-1, Mar. 1986.

[31] J. T. LaForte, R. M. McCoy, and D. K. Sharma, "Impulse voltage withstand capability of rotating machine insulation as determined from model specimens," *IEEE Trans. Energy Conversion*, vol. 3, Mar. 1988.
[32] O. M. Nassar, "The use of partial discharge and impulse voltage testing in the evaluation of interturn insulation failure of large motors," *IEEE Trans. Energy Conversion*, vol. EC-2, Dec. 1987.
[33] A. K. Sood, A. Amin Fahs, N. A. Henein, "Engine fault analysis, Part I: Statistical methods," *IEEE Trans. Ind. Electron.*, vol. IE-32, Nov. 1985.
[34] ——, "Engine fault analysis, Part II: Parameter estimation approach," *IEEE Trans. Ind. Electron.*, vol. IE-32, Nov. 1985.
[35] A. Keyhani and S. M. Miri, "Observers for tracking of synchronous machine parameters and detection of incipient faults," *IEEE Trans. Energy Conversion*, vol. EC-1, June 1986.
[36] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—A survey and some new results," *Automatica*, vol. 26, no. 3, pp. 459–474, 1990.
[37] M.-Y. Chow and R. J. Thomas, "Detection of damper winding currents and the damping coefficient of a synchronous machine using a predictor corrector estimator," presented at the 1988 Int. Symp. Circuits Syst., Espoo, Finland, June 1988.
[38] J. C. Hung and B. J. Doran, "High-reliability strapdown platforms using two-degree-of-freedom gyros," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-9, pp. 253–259, Mar. 1973.
[39] S. Liu and J. C. Hung, "Pulsating parameter method for fault diagnosis for a hydraulic pump," in *Proc. IECON'91*, pp. 2145–2150.
[40] M.-Y. Chow and S. O. Yee, "Real time application of artificial neural networks for incipient fault detection of induction machines," in *Proc. 3rd Int. Conf. Ind. Eng. Appl. Artificial Intell. Expert Syst.*, Charleston, S.C., July 1990.
[41] D. R. Boothman and E. C. Elgar, "Thermal tracking—A rational approach to motor protection," *IEEE Trans. Power Eng. Syst.*, Sept./Oct. 1974.
[42] D. E. Schump, "Reliability testing of electric motor," *IEEE Trans. Ind. Appl.*, vol. 25, May/June 1989.
[43] S. Cambrias and S. A. Rittenhouse, *Generic Guidelines for the Life Extension of Plant Electrical Equipment*, Electric Power Research Institute EL-5885, pp. 3-3-1–3-3-14, July 1988.
[44] R. W. Smeaton, *Motor Application and Maintenance Handbook*, 2nd ed. New York: McGraw-Hill, 1987.
[45] P. C. Krause, "Simulation of unsymmetrical 2-phase induction machines," *IEEE Trans. Power App. Syst.*, vol. PAS-84, Nov. 1965.
[46] ——, *Analysis of Electric Machinery*. New York: McGraw-Hill.
[47] A. E. Fitzgerald, C. Kingsley, Jr., and S. D. Umans, *Electric Machinery*, 4th ed. New York: McGraw-Hill.
[48] G. R. Slemon and A. Straughen, *Electric Machines*. Reading, MA: Addison-Wesley, 1980.
[49] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Contr., Signals, Syst.*, vol. 2, pp. 303–314, 1989.
[50] K. Hornik, M. Stinchcombe, and H. White, "Multi-layer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.
[51] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: The Johns Hopkins Univ. Press, 1983.
[52] R. E. Walpole and R. H. Myers, *Probability and Statistics for Engineers and Scientists*, 2nd ed. New York: Macmillan, 1978.
[53] C. L. Giles and T. Maxwell, "Learning, invariance, and generalization in higher-order neural networks," *Appl. Opt.*, vol. 26, pp. 4972–2978.
[54] N. Tishby, E. Levin, and S. A. Solla, "Consistent inference of probabilities in layered networks: Predictions and generalization," *Proc. IJCNN*, Washington, DC, June 1989.
[55] M.-Y. Chow, G. Bilbro, and S. O. Yee, "Application of learning theory to a single phase induction motor incipient fault detection artificial neural network," *Int. J. Neural Syst.*, vol. 2, no. 1 & 2, pp. 91–100, 1991.
[56] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Univ. Illinois Press, 1963.
[57] M.-Y. Chow and S. O. Yee, "Robustness test of an incipient fault detector artificial neural network," in *Proc. IJCNN'91*, Seattle, WA, July 1991, pp. 173–178.
[58] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning, Parts 1 and 2," *Inform. Sci.*, pp. 199–249, 301–357, 1975.
[59] M. Sugenon, "An introductory survey of fuzzy control," *Inform. Sci.*, vol. 36, pp. 59–79, 1985.
[60] G. J. Klir and T. A. Folger, *Fuzzy Sets, Uncertainty, and Information*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[61] H.-J. Zimmerman, *Fuzzy Set Theory—and Its Applications*. Kluwer Academic.
[62] L. A. Zadeh, "A theory of approximate reasoning," in J. Hayes, D. Michie, and L. I. Mikulich, Eds., *Machine Intelligence, Vol. 9*. New York: Halstead, 1979, pp. 149–194.
[63] M. de Glas, "A mathematical theory of fuzzy systems," in M. M. Gupta and E. Sanchez, Eds., *Fuzzy Information and Decision Processes*. New York: North-Holland, 1982, pp. 401–410.
[64] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, Jan. 1973.
[65] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller, Part II," *IEEE Trans. Sys., Man, Cybern.*, vol. 20, Mar./Apr. 1990.
[66] L. A. Zadeh, "Fuzzy sets," *Information and Control, Vol. 8*. New York: Academic, 1965, pp. 338–353.
[67] ——, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets Syst.*, vol. 1, pp. 3–28, 1978.
[68] R. K. Ragade and M. M. Gupta, "Fuzzy set theory: Introduction," in M. M. Gupta, Ed., *Fuzzy Automata and Decision Processes*. New York: Elsevier North-Holland, 1977, pp. 105–131.
[69] M. Y. Chow, R. N. Sharpe, and J. C. Hung, "On the application and design of artificial neural networks for motor fault detection—Part I," *IEEE Trans. Ind. Electron.*, Apr. 1993.
[70] R. G. Brown and P. T. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 2nd ed. New York: Wiley, 1992.

Mo-yuen Chow received the B.S. degree from the University of Wisconsin, Madison, in 1982 and the M.Eng. and Ph.D degrees from Cornell University, Ithaca, NY, in 1983 and 1987, respectively, all in electrical engineering.

After graduation, he joined the faculty of North Carolina State University, Raleigh. Currently, he is an Assistant Professor in the Electrical and Computer Engineering Department at North Carolina State University. He is also the Principal Investigator of various research projects. His research interests include system monitoring and fault detection, artificial neural networks, fuzzy logic, rotating machine analysis, and system and control theory.

Robert N. Sharpe was born in North Carolina on August 31, 1966. He received the B.S. and M.S. degrees in mechanical engineering from North Carolina State University, Raleigh, in 1988 and 1990, respectively.

Currently, he is a Research Assistant in the Electrical and Computer Engineering Department at North Carolina State University. His research interests include system monitoring and fault detection, artificial neural networks, fuzzy logic, and control theory.

Mr. Sharpe is a member of Tau Beta Pi, Pi Tau Sigma, and Gamma Beta Phi.

James C. Hung (S'55–M'60–SM'62–F'84) received the B.S. degree in electrical engineering from the National Taiwan University. Taiwan China, in 1953, and the M.E.E. and Sc.D. degrees from New York University in 1956 and 1961, respectively.

From 1954 to 1961 he was associated with the Department of Electrical Engineering at New York University. He joined the University of Tennessee, Knoxville, as an Assistant Professor in 1961, and became Associate Professor and Professor in 1962 and 1965, respectively. He was named Distinguished Professor of the University of Tennessee, Knoxville, in 1984, in the Department of Electrical and Computer Engineering. He is a specialist in system analysis, system design, and data processing, with applications to navigation, guidance, and control.

Dr. Hung is a member of Sigma Xi, Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi. He is a Registered Professional Engineer in the State of Tennessee.