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

Mo-yuen Chow, Robert N. Sharpe, and James C. Hung, *Fellow, IEEE*

Abstract—**The emerging technology of artificial neural networks has been successfully used in a variety of different 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 first discusses a few noninvasive fault detection techniques, including the parameter estimation approach, human expert approach, etc., and then the artificial neural network approach. A brief overview of *feedforward nets* and the *backpropagation* training algorithm, along with its pseudo codes, will follow. Later sections 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 is presented.**

I. Introduction

RECENTLY, the emerging technology of artificial neural networks has been successfully used in many different areas such as fault detection [1]–[7], control [8]–[14], and signal processing [15]–[19], among others. The first use of artificial neural networks can be dated back to the 1940's. Several different neural network paradigms have been developed over the past few decades [20]–[25]. Some of the major paradigms are (*multilayer*) *feedforward net* [20]–[23], Kohonen net [24], Hamming net [21], [22], Hopfield net [21], [22], adaptive resonance theory [21], [22], etc. Each paradigm has its own internal network structure, properties and training algorithms. Some paradigms are more suitable to solve certain types of problems, while other paradigms are more appropriate for others. Of the different paradigms, the *feedforward net* is probably the most popular, and it constitutes more than 80% of the current artificial neural network applications. This paper discusses why and how to apply artificial neural network technology to perform motor fault detection, focusing on the use of the *feedforward net* paradigm.

The paper first discusses a few noninvasive fault detection techniques, including the parameter estimation approach and the human expert approach, and then leads to the artificial neural network approach. A brief overview of *feedforward nets* and the *backpropagation* training algorithm, along with its pseudo codes, follows. Later sections explain 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 is presented.

Manuscript received August 31, 1992; revised November 13, 1992.

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

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

IEEE Log Number 9206800.

II. Incipient Fault Detection of Induction Motors—A Brief Description

Although rotating machines are usually well constructed and robust, the possibility of incipient faults is inherent in the machines due to the stresses involved in the conversion of electrical energy to mechanical energy and vice versa. Incipient faults within a machine will affect the performance of the machine before major failures occur [1]–[6], [26]–[44]. Early fault detection allows preventive maintenance to be scheduled for machines that might not ordinarily be due for service and may also prevent an extended period of down time caused by extensive motor failure. With proper system monitoring and fault detection schemes, maintenance costs can be reduced and reliability of the machines significantly improved.

An experienced engineer can usually detect and diagnose motor faults by observing the motor's operating performance. However, experienced engineers are expensive and difficult to train. In addition, it is more desirable to automate system monitoring and fault detection schemes rather than to rely on a few experienced engineers to perform continuous on-line monitoring. Several fault detection methods have been developed, each with their own pros and cons. Some techniques require expensive diagnostic equipment and/or off-line fault analysis to determine the motor condition. For instance, the *radio frequency monitoring* scheme injects radio frequency signals to the stator winding of a machine and measures the changes of the signal waveform to determine whether the winding insulation contains faults [28]. This technique requires expensive equipment and is justified only for use with large and expensive machines. Other popular techniques, such as *particle analysis* (which requires bringing the motor oil samples to a laboratory for analysis to determine the motor condition), are more suitable for

0278-0046/93$03.00

overhaul or routine checkup rather than on-line monitoring and fault detection [36], [37].

Another fault detection technique, the *parameter estimation* approach, is a *noninvasive* fault detection scheme [34], [35], [38]. Noninvasive fault detection schemes are based on easily accessible and inexpensive measurements to predict the motor condition without disintegrating the motor structure. These schemes are suitable for on-line monitoring and fault detection purposes. Due to their economical and nondestructive features, noninvasive techniques are often preferred by many engineers. However, the *parameter estimation* approach requires an accurate mathematical model and an elaborate understanding of the system dynamics based on a set of system parameters. The parameters are usually chosen to reflect the motor conditions. For example, the motor-bearing condition will affect the *damping coefficient* of the motor's mechanical equation. As the bearing wears out, the damping coefficient increases. Thus, the parameter estimation approach can be based on the motor's mechanical equation and measurements (such as rotor speed) to estimate the value of the damping coefficient. After estimating the numerical value of the chosen parameter, e.g., damping coefficient of the motor, then a means to translate the estimated numerical value to a qualitative description, such as a good or bad bearing, is required. The major difficulty with the parameter estimation approach is that an accurate mathematical system model (e.g., motor mechanical dynamic equation) is required and is usually difficult to obtain. In addition, the interpretation of the fault conditions—which is a fuzzy concept [60]–[64]—using rigorous mathematical formulations is generally impractical and inaccurate [61], [64].

On the other hand, use of an *artificial neural network* for fault detection is also a noninvasive technique. But, unlike the parameter estimation technique, neural networks can perform fault detection based on measurements and training without the need of complex and rigorous mathematical models [1]–[6]. In addition, heuristic intepretation of motor conditions, which sometimes only humans are capable of doing, can be easily implemented in the neural network through supervised training. For many fault detection schemes, redundant information is available and can be used to achieve more accurate results [38]. This concept can be easily adapted in neural network implementation with its multi-input parallel processing features to enhance the robustness of the network performance. In Section III, parameter estimation, expert's diagnosis, and neural network fault detection schemes are presented to demonstrate the advantages of using the artificial neural network.

III. Incipient Fault Detection of Motors—Mathematical Description

In order to successfully perform fault detection, different sets of criteria are needed to define a motor's condition at different operating conditions. In this paper, fault detection of a split-phase squirrel-cage induction motor is used for illustration purposes. It is worthwhile to describe the fault detection problem in mathematical terms to facilitate future discussions on the subject. The following sections provide a brief overview of the mathematical complexity of the parameter estimation approach.

A. Mathematical Description of the Motor Dynamics

In this paper, the motor is assumed to be operating at steady state with a known load torque condition. Two common motor faults—*turn-to-turn insulation fault* and *bearing wear*—are studied. A turn-to-turn insulation fault in the main winding of the motor will cause a change in the corresponding number of equivalent turns N of the main winding. Bearing wear in the motor, on the other hand, is reflected in the damping coefficient B of the motor.

Let the stator flux linkages $\lambda_s = [\lambda_{as} \lambda_{bs}]^T$ and rotor flux linkages $\lambda_r = [\lambda_{ar} \lambda_{br}]^T$ of a split-phase squirrel-cage induction motor be state variables with subscripts a and b representing the a and b phases, respectively; subscript s denotes variables and parameters associated with the stator circuits and subscript r denotes variables and parameters associated with the rotor circuits. The dynamics of the split-phase squirrel-cage induction motor can be represented by state equations (1) and (2) [45]–[48].

$$\dot{\lambda}_s = \boldsymbol{R}_s \boldsymbol{i}_s - \boldsymbol{v}_s \tag{1}$$

$$\dot{\lambda}_r = \boldsymbol{R}_r \boldsymbol{i}_r - \boldsymbol{v}_r \tag{2}$$

where

$$\boldsymbol{R}_s = \begin{bmatrix} r_{as} & 0 \\ 0 & r_{bs} \end{bmatrix}$$

and

$$\boldsymbol{R}_r = \begin{bmatrix} r_{ar} & 0 \\ 0 & r_{br} \end{bmatrix}$$

where r_{as}, r_{bs}, r_{ar}, and r_{br} are the a-phase and b-phase stator winding resistance and rotor equivalent winding resistance, respectively. $\dot{\lambda}$ is the time derivative of the flux linkage, λ. The vector $\boldsymbol{i}_s = [i_{as}, i_{bs}]^T$ represents the stator winding currents, $\boldsymbol{i}_r = [i_{ar}, i_{br}]^T$ the rotor winding currents, $\boldsymbol{v}_s = [v_{as}, v_{bs}]^T$ the stator winding voltages, and $\boldsymbol{v}_r = [v_{ar}, v_{br}]^T$ the rotor winding voltages (which are zero for an induction motor). At steady state and/or small perturbation conditions, the flux linkages can be approximated by a linear relationship with respect to the currents, expressed as

$$\begin{bmatrix} \lambda_s \\ \lambda_r \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}_s(\theta) & \boldsymbol{L}_{sr}(\theta) \\ \boldsymbol{L}_{sr}(\theta)^T & \boldsymbol{L}_r(\theta) \end{bmatrix} \begin{bmatrix} \boldsymbol{i}_s \\ \boldsymbol{i}_r \end{bmatrix} \tag{3}$$

where $\boldsymbol{L}_s$ is the stator inductance, $\boldsymbol{L}_r$ is the rotor inductance, $\boldsymbol{L}_{sr}$ is the mutual inductance between stator and rotor, and θ is the rotor position.

From fundamental electromagnetic theory, the flux linkage of a winding is a function of the number of *equivalent turns* of the winding [46]–[48]. The equivalent

turns for phases a and b are expressed by the vector $N_s = [N_{as}, N_{bs}]^T$ for the stator windings and $N_r = [N_{ar}, N_{br}]^T$ for the rotor windings. The motor parameters, such as winding resistance and inductance, will change due to changing values of equivalent turns. The same motor structure with different values of equivalent turns will yield a different performance. For a squirrel-cage induction motor, the rotor is robust, and N_r is generally assumed to be constant, while N_s will change value due to deterioration in the stator winding. When N_s is a variable, R_s, L_s, and L_{sr} become functions of N_s and are expressed as $R_s(N_s)$, $L_s(N_s)$, and $L_{sr}(N_s)$, respectively. The performance of an induction motor can then be expressed as a function of N_s, its stator equivalent winding turns.

The electrical torque of the motor, T_e, is a function of the motor parameters and state variables as expressed in (4).

$$T_e = i_s^T \frac{\partial}{\partial \theta} L_{sr} i_r. \tag{4}$$

Thus, the motor torque is a function of N_s and will be expressed as $T_e(N_s)$ to signify the dependence.

The equation of motion for the motor can be written as

$$T_e(N_s) = J\dot{\omega} + B\omega + T_l \tag{5}$$

where $\dot{\omega}$ is the time derivative of the rotor speed, ω; J is the inertia of the rotor and connected load, B is the damping coefficient of the motor, and T_l is the load torque, which is assumed to be known. The geometric winding placement of the motor is depicted in Fig. 1(a) and the corresponding circuit diagram is depicted in Fig. 1(b). In the figures, the *as*, *bs*, *ar*, and *br* axes are the magnetic flux axes for the *a* and *b* phases of the stator and rotor poles, respectively, while the other parameters and variables have been described previously.

For a split-phase induction motor, the motor is started with both stator windings, main (phase *a*) and start (phase *b*) windings, energized from a common single-phase source. Once the rotor has reached 60%–80% of synchronous speed, the start winding is disconnected from the source by a centrifugal type disconnect switch [45]–[48]. Therefore, at steady-state operation the start winding, that is, N_{bs}, is not included in the steady-state fault detection. Thus, for simplicity of notation, N is used to replace N_{as}, whereas N_{bs} will be ignored because only fault detection at steady-state operation is under consideration.

Let I be the rms values of i_{as}, and let ω be the average speed of the rotor. At steady state, $(dI/dt) = (d\omega/dt) = 0$. By combining and manipulating (1)–(5), with N and B as variables, the steady-state current I and rotor speed ω can be represented by a set of nonlinear algebraic equations, $f = [f_1, f_2]^T$, which are functions of the main winding equivalent turns, N, and the damping coefficient, B:

$$f(I, \omega, N, B) = 0. \tag{6}$$

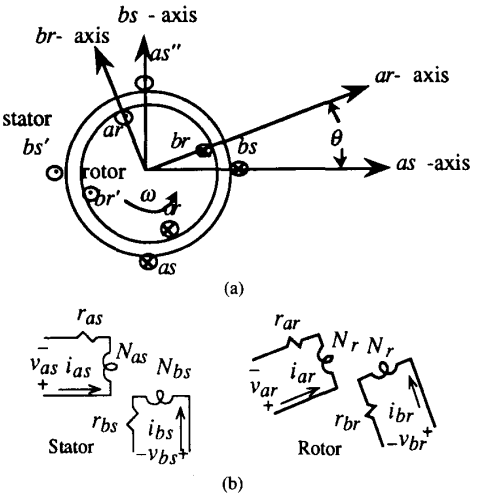


Fig. 1. (a) The geometric winding placement of a split-phase squirrel-cage induction motor. (b) The corresponding circuit diagram of a split-phase squirrel-cage induction motor.

B. Nonlinear Relation of Motor Parameters

From the induction motor dynamics analysis in (6), there exists a relationship $\mathfrak{M}_1$ between (I, ω) to (N, B) as shown in (7).

$$\mathfrak{M}_1 : (I, \omega) \rightarrow (N, B). \tag{7}$$

$\mathfrak{M}_1$ is highly nonlinear due to the nonlinearities present in the induction motor. An accurate mathematical model of $\mathfrak{M}_1$ is difficult to obtain. As stated before, conditions of the main winding and bearings of the motor are reflected in the numerical values of the main winding equivalent turns N and the damping coefficient B, respectively. For our application, the values of N and B, which quantitatively describe the motor, are quantized into three condition levels {good, fair, bad} to yield N_c and B_c, respectively, which qualitatively describe the motor's condition. This qualitative description of the motor's condition is more suitable for fault detection purposes. A second relationship $\mathfrak{M}_2$ is used to denote the relationship from the quantitative description (N, B) to qualitative description (N_c, B_c) as shown in (8).

$$\mathfrak{M}_2 : (N, B) \rightarrow (N_c, B_c). \tag{8}$$

As a result, the relationship $\mathfrak{M}$ from (I, ω) to (N_c, B_c) can be written as a composition of $\mathfrak{M}_1$ and $\mathfrak{M}_2$:

$$\mathfrak{M} = \mathfrak{M}_1 \circ \mathfrak{M}_2 : (I, \omega) \rightarrow (N_c, B_c). \tag{9}$$

Using the notations defined above, the *parameter estimation* approach can be summarized as follows. First, it estimates the values of N and B based on $\mathfrak{M}_1$; then it determines the motor's conditions based on the model $\mathfrak{M}_2$ and on the estimated values of N and B, which are denoted as $\hat{N}$ and $\hat{B}$, respectively. The qualitative motor condition $(\hat{N}_c, \hat{B}_c)$ is then predicted based on $(\hat{N}, \hat{B})$ and $\mathfrak{M}_2$. The overall scheme is depicted in Fig. 2. The performance of the parameter estimation technique heavily

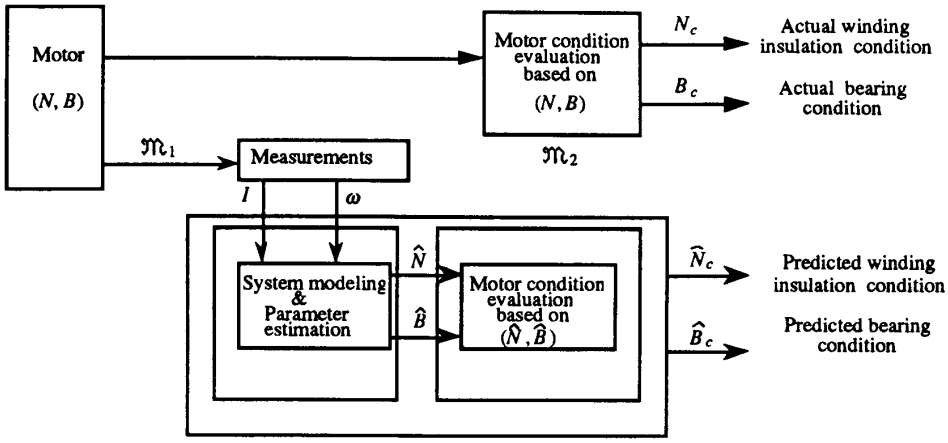


Fig. 2. Parameter estimation motor fault detection/diagnosis process.

depends on the accuracy of $\mathfrak{M}_1$ and $\mathfrak{M}_2$ to reflect the actual situation. $\mathfrak{M}$ is very complex due to the high degree of nonlinearity of the motor dynamics ($\mathfrak{M}_1$) and the discretization ($\mathfrak{M}_2$); thus, obtaining an accurate analytical expression ($\mathfrak{M}$) for a given induction motor is rather difficult.

IV. Artificial Neural Networks for Fault Detection

A. Expert Approach for Fault Detection

As stated previously, the interpretation of a motor's condition based on numerical value is usually a difficult task because fault interpretation is a fuzzy concept and usually requires experience. Therefore, in many cases, a *heuristic interpretation* of the results—which only humans are capable of doing—becomes necessary. An experienced engineer can diagnose the motor's conditions based on its operating conditions and measurements, as depicted in Fig. 3, without knowing the exact mathematical model of the motor. The approach is simple and reliable, and the complicated relation $\mathfrak{M}$ in (9) is implicitly embedded in the engineer's knowledge about the motor. However, an experienced engineer may not be able to give detailed explanations regarding his or her reasoning and logic used to make the decisions simply because experience belongs to the fuzzy logic realm and is difficult to describe accurately in exact mathematical terms.

As it turns out, this *human expertise* approach has many advantages over the *parameter estimation* approach. However, the major drawback of the human expertise approach is that experience is difficult to be transferred and automated. Both researchers and engineers usually transfer experience and knowledge through languages and mathematics (similar to the parameter estimation technique), which is sometimes time consuming and inaccurate. In actuality, the experience and knowledge used by expert engineers to perform motor fault detection/diagnosis are also implicitly *embedded* in the historical fault detection data performed by the experts.

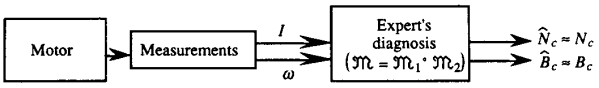


Fig. 3. Experts' motor fault detection/diagnosis process.

B. Learning Skills of Artificial Neural Networks

With the emerging technology of artificial neural networks, the human expertise approach can be mimicked and automated. Artificial neural networks can be trained to perform motor fault detection by learning experts' knowledge using a representative set of motor data. Such a scheme is depicted in Fig. 4. At the beginning of a neural network's learning (or training) session, the detection and diagnosis of the motor's condition—$\hat{N}_c$ and $\hat{B}_c$ for our specific application—made by the neural network fault detector will not be correct. Based on the difference between the correct decision $y = [N_c, B_c]^T$ made by the expert and $\hat{y} = [\hat{N}_c, \hat{B}_c]^T$ made by the neural network, an error quantity $e = (N_c - \hat{N}_c)^2 + (B_c - \hat{B}_c)^2$ is generated and used to adjust the neural network's internal parameters (called *network weights*) to give a *better* $\hat{y}$ that is closer to the correct decision. By training a neural network to *learn* the fault detection (which is to learn the desired relation $\mathfrak{M}$) based solely on input-output examples without the need of mathematical models, the complexity of the *parameter estimation* approach can be avoided. Once the neural network is trained appropriately, the network weights will contain the knowledge needed to perform fault detection (Fig. 5).

V. Brief Overview of Feedforward Net Paradigm and Backpropagation Training Algorithm

To technically describe how neural networks learn from experts' knowledge, it is desirable to explain a neural network's structure and training in mathematical terms. The following sections provide an overview of the mathematical description of feedforward nets and the backprop-

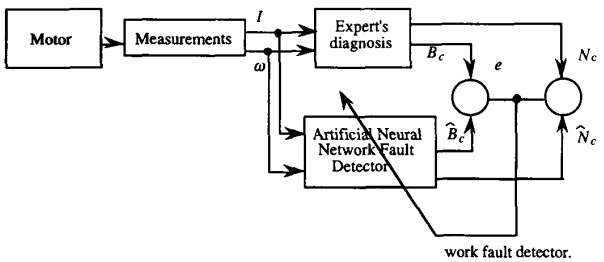


Fig. 4. Schematic diagram of the training of artificial neural network fault detector.

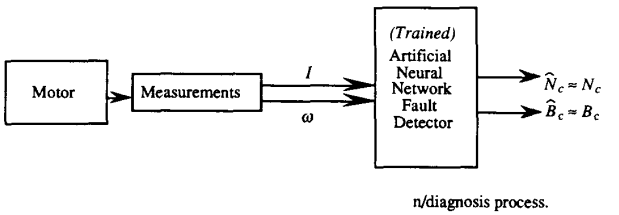


Fig. 5. Artificial neural network motor fault detection/diagnosis process.

agation training algorithm, including pseudo codes to ease the discussion. For more detailed descriptions on these topics, please refer to the literature, [20]–[23].

A. Brief Description of Multilayer Feedforward Artificial Neural Networks

The basic multilayer feedforward net contains three layers: input, hidden, and output, as shown in Fig. 6. This type of neural network has one input layer, one output layer, and any number of hidden layers in between [21], [22]. Each network layer contains processing units called *nodes* or *neurons*. Each node in a network layer will send its output to all the nodes of the next layer. In the input layer, the nodes receive signals from the *outside world*, such as motor current and speed measurements in our application example (using current transducers and tachometers, respectively). The input layer of the neural network serves as an interface that takes information from the outside world and transmits it to the internal processing units of the network, analogous to a human's interface parts such as our eyes' retina and our fingers' sensing cells. Similarly, the output layer of the neural network serves as an interface that sends information from the neural network's internal processing units to the external world. The nodes in the hidden layers are the neural network's processing units.

Usually in feedforward neural networks, all the nodes in the hidden and output layers have the same structure shown in Fig. 7. Each node in the hidden and output layer receives signals $\mathbf{v} = [v_1, v_2, \cdots, v_k]^T$ from the nodes of the previous layer, scaled by the weights $\mathbf{w}_j = [w_{j1}, w_{j2}, \cdots, w_{jk}]^T$. The jth node in layer l, for $j = 1, \cdots, k$,

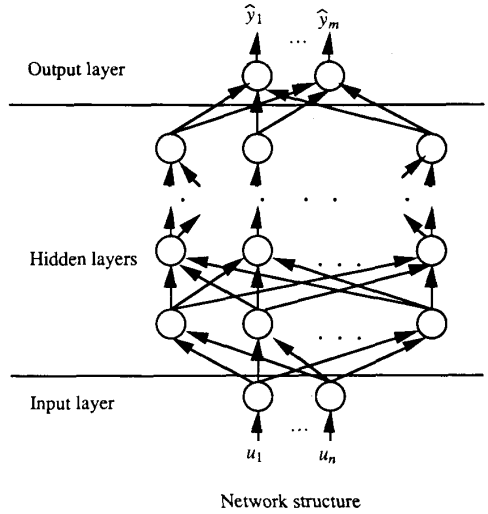


Fig. 6. Basic structure of a multilayer feedforward artificial neural network.

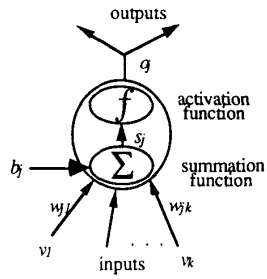


Fig. 7. Basic structure of the jth neural network node (or neuron).

computes the following quantity:

$$s_j = \sum_{i=1}^{k} w_{ji} v_i + b_j = \mathbf{w}_j^T \mathbf{v} + b_j \tag{10}$$

where b_j is the bias of the jth neuron. Quantity s_j will be processed by an *activation function* (usually nonlinear) to give the output o_j of the jth neuron:

$$o_j = f(s_j). \tag{11}$$

The activation function can be any function that is monotonically increasing and differentiable. The *sigmoid* function is perhaps the most popular activation function because it resembles the behavior of many biological neurons. The sigmoid activation function can be expressed as

$$o_j = \frac{1}{1 + e^{-s_j}} \tag{12}$$

where s_j is defined in (10) and o_j is the output of the jth node. A typical neuron output with respect to input as described in (12) is shown in Fig. 8 for a sigmoid activation function. Unlike the hidden and output layer nodes, the input layer nodes conventionally use a linear activation function instead of a sigmoid function, and the inputs

to the input neurons are $v = u$, representing the signal measured from outside world. The input–output data are conventionally normalized between [0, 1] for numerical stability reason.

The pseudo code for a multilayer perceptron is given in Table I. L represents the number of layers in the network (the inputs are counted as layer zero, i.e., $o_i^0 = u_i$).

It has been proven that three-layer feedforward nets (one input layer, one hidden layer, and one output layer) are able to learn any complicated continuous nonlinear functions [49], [50]. However, there is no theory or training algorithm to determine the exact number of nodes required to solve a given problem. At the present time, the design of network configurations to solve a particular engineering problem is still a trial and error process. However, the implications of the proof are encouraging and provide theoretical support of the feedforward net's capabilities.

Different authors prefer to use different network configurations to implement their applications, such as [9] and [12]. Nevertheless, the three-layer network configuration is the most popular and is used in this paper. A three-layer network can be denoted as $\mathfrak{N}_{n_i, n_h, n_o}$, where n_i is the number of input nodes, n_h the number of hidden nodes, and n_o the number of output nodes.

B. Mathematical Formulation of Neural Network Training (Learning)

There exist many different ways to view an artificial neural network. From the system engineering point of view, it may be easier to visualize a neural network as the learning of the input–output mapping of a system. This learning capability can be used to perform fault detection, for instance. This section presents the mathematical description of training an artificial neural network to learn the input–output relation of a given system, then extends the technique to perform fault detection in an induction motor. Such a technique can be extended to other engineering applications with appropriate modifications.

Let a system (Fig. 9), such as a motor, have inputs $\boldsymbol{u} = [u_1, u_2, \cdots, u_n]^T$ and corresponding outputs $\boldsymbol{y}(\boldsymbol{u}) = [y_1, y_2, \cdots, y_m]^T$. It is convenient to represent the system as the relationship $\mathfrak{M}$ from the input space $\mathfrak{U}$: $\{\boldsymbol{u} \in \mathfrak{U} | \boldsymbol{u}$ is the input to the system$\}$ to the output space $\mathfrak{Y}$: $\{\boldsymbol{y}(\boldsymbol{u}) \in \mathfrak{Y} | \boldsymbol{y}$ is the output of the system with input $\boldsymbol{u}\}$, as expressed in (13).

$$\mathfrak{M} : \mathfrak{U} \rightarrow \mathfrak{Y}. \tag{13}$$

Without causing confusion, the same notation $\mathfrak{M}$ is used in (13) and in (9) to denote input–output mapping. Now, let an artificial neural network learn such relationship $\mathfrak{M}$. The actual output of the artificial neural network is denoted as $\hat{\boldsymbol{y}}(\boldsymbol{u}|\boldsymbol{w})$, which belongs to output space $\mathfrak{Y}$, parametrized with weight $\boldsymbol{w} \in \mathfrak{W}$, the weight space, and given input $\boldsymbol{u}$.

The network training process can then be thought of as training the network to represent $\mathfrak{M}$ through the input–output relation as close as possible by adjusting the

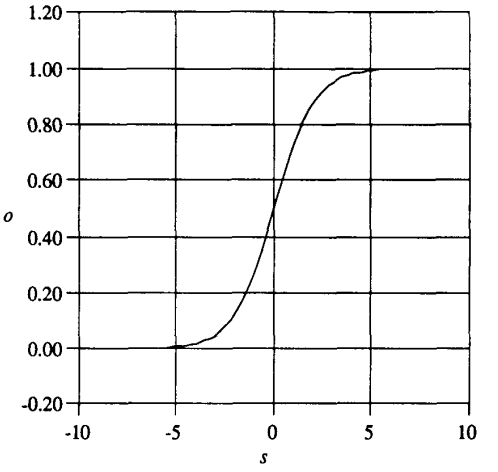


Fig. 8. A typical sigmoid activation function.

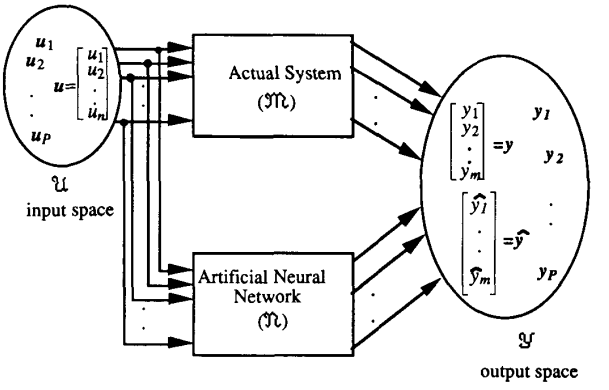


Fig. 9. Schematic diagram of the input–output relation of an actual system and an artificial neural network.

TABLE I
MULTILAYERED FEEDFORWARD NET PSEUDO CODE

```
/* Step through the layers from the input to the output */
    for (l = 1; l ≤ L; l = l + 1){
 /* Step through the nodes on layer l, where N_l is the number of
                    nodes in layer l */
 for (j = 1; j ≤ N^l; j = j + 1){
                    /* Sum over the nodes in the previous
                    layer */
                    s_j^l = 0
                    for (i = 1; i ≤ N^(l-1); i = i + 1){
                        s_j^l = s_j^l + w_ji^l × o_i^(l-1)
                    }
                    /* Add bias term */
                    s_j^l = s_j^l + b_j^l
                    /* If a sigmoid activation function is
                    used, */
                    o_j^l = 1 / (1 + e^(-s_j^l))
        }
}
```

network internal parameters (or weights) w, which can be represented mathematically as

$$\min_{w \in \mathfrak{W}} \|\hat{y}(u,w) - y(u)\|_2, \qquad \text{for all } u \in \mathfrak{U} \quad (14)$$

with the constraint imposed on the training so that the difference between the network output $\hat{y}(u,w)$ and the actual output $y(u)$ with given the same input u, i.e., $\|\hat{y}(u,w) - y(u)\| \leq \epsilon_a$, where $\|\cdot\|$ is some appropriate norm [51] and ϵ_a is a preset error tolerance of the difference between $\hat{y}(u,w)$ and $y(u)$. If ϵ_a, is *very* small, then $\hat{y}$ closely represents y.

The exact continuous relation of the system $\mathfrak{M}(u)$ is usually not precisely known. (If $\mathfrak{M}(u)$ was precisely known, then there is no need to train a neural network to learn the relation!) However, the discrete sets of measurements $\{(u_1, y_1), (u_2, y_2), \cdots, (u_P, y_P)\}$ of u and $y(u) = \mathfrak{M}(u)$ can usually be obtained accurately. The network training process consists of training the network to learn $\mathfrak{M}$ based on a set of discrete input–output measurements. In terms of artificial neural network terminology, u_p is the pth input pattern, $y_p = y(u_p)$ is the corresponding output pattern, and (u_p, y_p) is the pth training pattern. For the induction motor fault detection application, $u = [I, \omega]^T$ = [stator current, rotor speed]^T and $y = [N_c, B_c]^T$ = [winding insulation condition, motor bearing condition]^T.

Suppose there are P input–output training patterns available to train the neural network. The training process can be represented mathematically as

$$\min_{w \in \mathfrak{W}} \|\hat{y}(u_p,w) - y(u_p)\|_2 = \min_{w \in \mathfrak{W}} \sqrt{\frac{1}{P} \sum_{p=1}^{P} \left(\hat{y}(u_p,w) - y(u_p)\right)^2}, \qquad p = 1,2,\cdots,P \quad (17)$$

such that $\|\cdot\|_2 \leq \epsilon$, where ϵ is a preset error tolerance. If $\|\cdot\|_2 \leq \epsilon$, then the network is said to have learned the actual system mapping satisfactory, i.e., $\|\hat{y}(u,w) - y(u)\| \leq \epsilon_a$ in (15), provided that the training data are representative of the system behavior. The two-norm measure, which is one of the most popular measures used in neural network training, is used in (17). However, other appropriate norm measures can also be used to suit some specific applications.

Let us define e_p as the training error of training pattern p, i.e., the difference between the output of the network and the output of the system with input u_p and current network weight w:

$$e_p = e(u_p,w) = \|\hat{y}(u_p,w) - y(u_p)\|. \quad (18)$$

During network training, we would like to minimize the error for all training patterns (rather than minimizing the error for a single training pattern). Therefore we define the training error of the network as a quantity E as

$$E(w) = \frac{1}{P} \sum_{p=1}^{P} e(u_p,w). \quad (19)$$

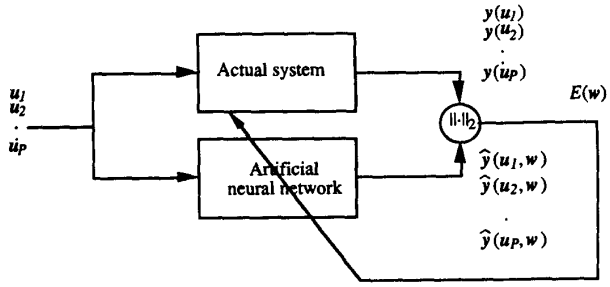


Fig. 10. Supervised training of artificial neural networks.

The schematic diagram of the training process is depicted in Fig. 10.

How to train multilayer feedforward nets, i.e., minimize E in (19), was previously a major roadblock for the successful application of neural networks. In the 1970's, a few breakthroughs in multilayer feedforward net training algorithms were developed [20], [21]. Presently, there are a few well-known training algorithms, among which are the *backpropagation* algorithm [20]–[22] and *simulated annealing* [17]–[19], available to train neural networks. These methods train the network to minimize E by iteratively adjusting the network weights. The *backpropagation* algorithm is by far the most popular training algorithm for feedforward nets and is briefly presented in the next section.

The backpropagation training algorithm is basically a steepest descent method that searches for an optimal w to minimize the error E in (19). The weights are updated iteratively as

$$w(itera + 1) = w(itera) - \eta \left.\frac{\partial E(w)}{\partial w}\right|_{w = w(itera)}, \quad (20)$$

where *itera* denotes the current training iteration number and η is the learning rate, usually a small positive number, e.g., 0.1.

For the output layer, the quantity $\partial E(w)/\partial w$ in (20) can be easily calculated by taking the derivative of E with respect to the weights w of output layer, while $\partial E(w)/\partial w$ for the weights in the hidden layers requires the use of the *chain rule* to backpropagate the error signal obtained in the output layer to the hidden layers. The *delta rule* used to adapt the network weights has been derived in detail in [21], [22] and is listed in (21)–(24).

$$\delta_j = o_j(1 - o_j)(y_j - o_j) \quad \text{for output layer node } j \quad (21)$$

$$\delta_j = o_j(1 - o_j) \sum_k \delta_k w_{kj} \quad \text{for hidden layer node } j \quad (22)$$

$$\Delta w_{ji}(itera + 1) = \eta \delta_j o_j + \alpha \Delta w_{ji}(itera) \quad (23)$$

$$w_{ji}(itera + 1) = w_{ji}^n + \Delta w_{ji}(itera + 1) \quad (24)$$

TABLE II
BACKPROPAGATION TRAINING ALGORITHM PSEUDO CODE

```
/* Step through the layers from the output to the first hidden layer */
        for (l = L; l ≥ 1; l = l − 1){
        /* Step through the nodes on layer l */
        for (j = 1; j ≤ N^l; j = j + 1){
    δ_j^l = 0
    /* If we are operating on the output layer, */
    if (l = L)
        δ_j^L = o_j^L(1 − o_j^L)(y_j − o_j^L)
    /* otherwise, */
    else{
        for (k = 1; k ≤ N^{l+1}; k = k + 1){
            δ_j^l = δ_j^l + δ_k^{l+1} × w_jk^{l+1}
        }
        δ_j^l = o_j^l(1 − o_j^l)δ_j^l
    }
    /* update weights */
    /* step through weights connected to node j, layer l */
    for (i = 1; i ≤ N^{l−1}; i = i + 1){
        Δw_ji^l(itera + 1) = η δ_j^l o_j^l + α Δw_ji^l(itera)
        w_ji^l(itera + 1) = w_ji^l(itera) + Δw_ji^l(itera + 1)
    }
        }
}
```

where $\delta_j o_j = -(\partial E(w)/\partial w)$ is the descent slope of the iteration process in (20). In (23), η is called the learning rate (equivalent to step-size in steepest descent algorithm) and α is called the momentum rate. The extra term $\alpha\,\Delta w_{ji}(itera)$ added in (23) compared with (20) is called the momentum term, included in the weight update equation to try to avoid local minimum [21], [22]. The choices of η and α are presented later. Detailed discussion of the algorithm can be found in [20]–[23], and the pseudo code of the backpropagation training algorithm is listed in Table II.

ACKNOWLEDGMENT

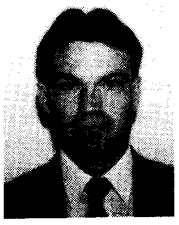
The authors acknowledge the support of the National Science Foundation, for Grant ECS-8922727, and the U.S. Army Corps of Engineers—Construction Engineering Research Laboratory, for Grant 5-30016.

REFERENCES

Please see Reference section of "On the Application and Design of Artificial Neural Networks for Motor Fault Detection—Part II," *IEEE Transaction on Industrial Electronics*, April, 1993.

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.