7

A Neural Approach to Establishing Technical Diagnosis for Machine Tools

7.1 INTRODUCTION TO NEURAL NETWORK THEORY

The end of the 1980s signaled a period of research regarding the development of a new approach to data processing within computational systems. This approach earned several names: connectionism, neural network, neural processing, and parallel processing; all of these terms are synonyms for that kind of information processing that tries to simulate the thought processes of the human brain based on experience, rather than the classical method of that based on algorithms.

7.1.1 Information Neural Processing

Data neural processing means to elaborate and study networks with adaptable nodes; these networks store experiential knowledge, and are available to be utilized for the purpose for which the network was created. The word "adaptable" used for the network nodes represents that property of a node which is able to react correctly even if the stimulus received is not precisely the same as the ones already learned.

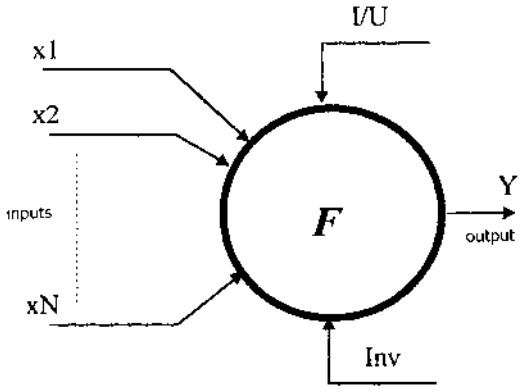

FIGURE 7.1 Adaptable node, main element of a neural network.

The adaptable node, the main element of a neural network, can be represented as an electronic device with a manifold input channel (Fig. 7.1). It consists of one output channel and two special connections: one for introducing data into the learning process, and the other as a switch of learning/utilizing nodes. The F node function is the way in which an output parameter is associated with each input dataset. For example, the function of an adaptive three-input node can look like a logic table in which the response is 1 only when the input dataset has a single 1 value (Table 7.1). This representation of an adaptive node makes obvious the association with the neuron, the adaptive node of the brain. The human brain has an average of 10^{11} neurons, organized in complex structures.

The neuron (Fig. 7.2) consists of a cellular body called the perikarion, which contains the nucleus, and two sorts of protoplasmic prolongation: the axon (cylindrical shaped, long, usually single) and the dendrites (many and short). The dendrites represent the input channel

TABLE 7.1 The Function of an Adaptive Three-Input Node Can Look Like a Logic Table

x1	0	0	0	0	1	1	1	1
x2	0	0	1	1	0	0	1	1
x3	0	1	0	1	0	1	0	1
Y	0	1	1	0	1	0	0	0

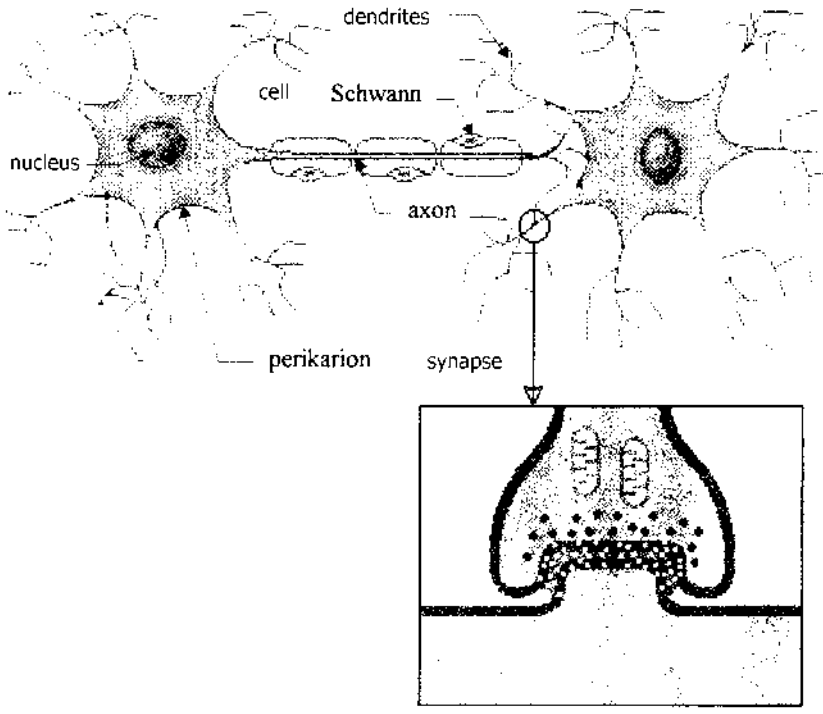

FIGURE 7.2 Neuron schema.

into the neuron, while the axon is the output channel. An electric activity characterized by short and fast impulses (about 100 impulses/second) has been noticed in the axon when the neuron "emits." Neurons are interconnected through the ends of the dendrites, called synapses. One neuron can receive 5000 to 15,000 input signals from other neurons' axons. Synapses can be excitators, if they help the neuron to emit, or inhibitors, if they discourage the neuron from emitting.

The informational model of a neuron was first proposed in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pitts. This model is still the basis of information neural processing. According to this model, the synaptic modifications are continuous, and the neuron takes into consideration all of the synaptic signals, both excitator and inhibitor; then, it sums up their effects and determines whether to emit them through the axon. Number 1 is associated with the emission state, and 0 with the repose state.

If X denotes the neuron state and the synaptic connection effect is represented by a weight W, then the effect of a synapse upon the neuron is given by the product $X \cdot W$. The weight W can have values within the interval $-1, \ldots, 1$; the negative values characterize the inhibitor synapses. Since there is a multitude of synapses, an index j has to be attached, which designates X_j and W_j as the input and the weight of the synapse j, respectively. The neural model constantly adds those effects and compares them with a threshold value T; if the sum exceeds the threshold, the neuron emits. For the McCullogh and Pitts neuron, this emission rule can be described mathematically by the relation:

$$X_1W_1 + X_2W_2 + \cdots + X_jW_j + \cdots + X_nW_n > T. \tag{7.1}$$

The electronic interpretation of this relational model is given in Figure 7.3. The basic components are: the summator amplifier, which provides an output voltage proportional to the sum of all products $X \cdot W$, and the voltage comparator, which generates a voltage equal to 1 if the output voltage of the summator exceeds the threshold voltage T. The value of the adjustable weights W is set automatically during the learning process.

On the basis of the model presented so far, two sorts of neural networks have been theorized: feedforward and feedback networks. Within these networks neurons are distributed in one or more layers that cannot be accessed directly; they are called hidden layers and have free access only at the input channel and, respectively, at the output channel of the network (Figure 7.4). The feedforward networks operate between the input channel and output terminals, learning to associate output data with the input data. In feedback networks, the information can run within a loop from the input to the output channel and vice versa, creating a so-called internal input channel. Both types of network oper-

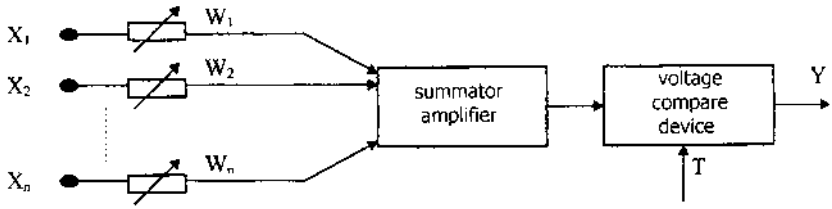

FIGURE 7.3 Electronic interpretation of Eq. (7.1) relational model.

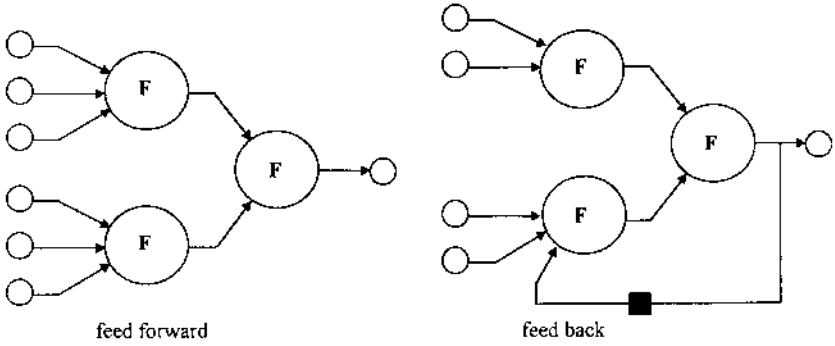

FIGURE 7.4 Neurons distributed in one or more layers that cannot be accessed directly.

ate under emission rules that determine whether the neuron will emit for each dataset. These rules are associated with each connection in the network, and they determine the weight of each of those connections. As a consequence of the application of the emission rules, the network becomes able to generalize, that is, to provide an adequate response for input datasets that were not introduced during the learning process.

The learning algorithms, also called learning laws, are based on the following condition: when the weights are recalculated for a new input dataset, no discontinuity with what was done for previous sets should occur. The biological basis of these laws derives from the hypothesis that, if a neuron is part of a network in which a synaptic input emits continuously at the same time as the neuron, then the weight of that synaptic connection will increase. The disadvantage of this theory is that it leaves the responsibility of the neuron's emission to the action (eventually learned) of some synapses. Although the technique proposed by Hebb was useful for the investigation of the neural network, another rule proved more suitable: the delta rule, which can be described through the following steps:

1. Select an input dataset.
2. When an error is detected in the network's response, calculate the deviation from the desired output parameter.
3. Adjust the active weights (i.e., those which are emitting) and the threshold value, in order to partially correct the error.
4. Return to step 1 until no input dataset causes errors.

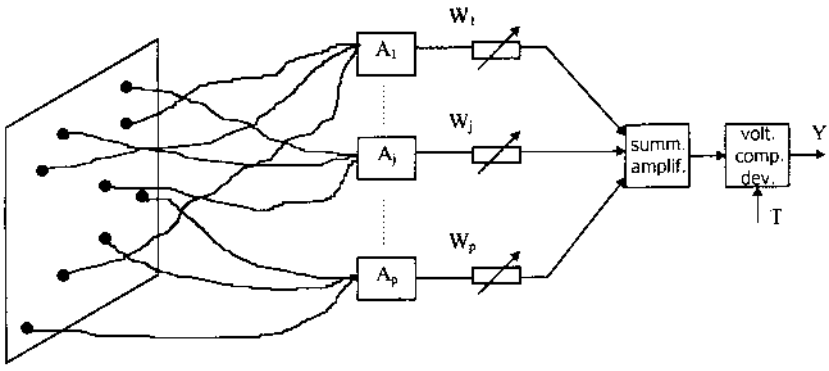

FIGURE 7.5 Perceptron electronic model: A_j are the preprocessing units, called associative units.

The McCulloch and Pitts model was improved by Frank Rosenblatt by adding fixed preprocessing units; their duty is to extract specific features from the input signals. Thus the perceptron was born, which is defined as a model recognition device. Figure 7.5 shows the perceptron electronic model; A_j are the preprocessing units, called associative units. In terms of these units, the perceptron order is defined; the preceptron order is equal to the number of inputs of the associative unit with the largest number of inputs.

7.1.2 The Learning Process in Neural Networks

The information neural processing paradigm principle states that neural networks can be trained merely through examples from the exterior. In other words, any learning algorithm utilized in multilayer networks is based on output error estimation. This is the so-called hard learning problem, and it is one of the most important problems in neural networks.

A major step in overcoming this problem was made in 1982 by John Hopfield, an American biologist and chemist; he presented neural networks as content addressable memories (associative memories). Through his work, he made two major contributions. He developed a type of network analysis that uses the concept of energy, concluding that a network reaches, while operating, its energetic minimum, after

which the output signal set does not change (i.e., stability occurs). In addition, he showed that learning rules, such as the delta rule, can be utilized to adjust the network parameters purposely to create the energetic minimum. The Hopfield neuron model has the characteristic parameters V_I, the output signal ($V = 0$ if the neuron does not emit; $V = 1$ if the neuron emits); T_{ij}, the weight of the connection of the neuron i with the neuron j ($T = 0$ if the neuron i is not connected to the neuron j); and U_I, the threshold value for which the neuron emits. In short:

$$\begin{aligned} V_i \text{ becomes 1 if } \sum_{j \neq i} T_{ij} V_j > U_i \\ V_i \text{ becomes 0 if } \sum_{j \neq i} T_{ij} V_j < U_i \end{aligned} \tag{7.2}$$

The energy of a neuron can be calculated with

$$E_i = -V_i \left(\sum T_{ij} V_j - U_i \right) \tag{7.3}$$

The amount between brackets is called the activation energy and is denoted by A_i. In order to have a stable state of a neural network, none of the nodes should be activated in such a way that the emission conditions change. So, if a neuron is emitting ($V = 1$), its activation will be positive, so that the emission is not further stopped; the same is true if the node is not emitting ($V \neq 1$); its activation will be negative, so that it does not further emit. Consequently, when the neural network changes its state, either it keeps the same energetic level, or it goes down to a lower energetic level. When lower energy levels are no longer accessible, the network remains stable at its most recent state.

From an energetic analysis perspective, it becomes clear that a learning procedure application, such as the delta rule, is simply a way to decrease the state energy to a minimum. The Hopfield model has its disadvantages: the most important is that the neural network "gets stuck" in false energetic minimums. The way that a neural system runs out of false minimums was discovered by Geoffrey Hinton; it consists of "noise" utilization (i.e., applying an uncertainty degree to the state energy). Intuitively, this method can be illustrated by representing the network state as a ball on a waved surface (Fig. 7.6). If the ball has an internal property which makes it "jump," it is probable that it will spend the longest time in the deepest valley it can reach.

FIGURE 7.6 Representing the network state as a ball on a waved surface.

At the crux of any neural network activation remains the phenomenon of increasing the aleatory motion of gaseous molecules while temperature increases, discovered by Ludwig Boltzmann at the end of the nineteenth century. Analogously with Boltzmann's research, the uncertainty degree, introduced to assess a neural network state, was named temperature. Thus, at zero temperature the network will behave according to the Hopfield model; at higher temperatures, an uncertainty degree proportional to the temperature has to be introduced in the neuron activation function. This procedure has the advantage of helping the network to run out of its false minimums but there is another side of the coin: the network no longer remains stable. Hinton proposed a "thermal regime" be utilized to enable this trouble to be outrun: starting the network at a high temperature and then cooling it gradually until it reaches the "thermal equilibrium." In this way, the network has the greatest chance to end in a state associated with the lowest minimum to be reached for specific input data. This manner of approach to neural networks has been called "simulated tempering."

The introduction of temperature in the neuron activation function is done through a probabilistic function, called the emission probability Boltzmann function:

$$p(1) = \frac{1}{1 + \exp(-A/T)} \tag{7.4}$$

The plot of this function is represented in Figure 7.7 for "temperatures" of 0.5 and 0.25 (these are arbitrary units, not related either to Celsius or Kelvin degrees or to other measure units of temperature as a physical parameter). The result of the introduction of temperature as a moderator in the neuron activation function can be pursued in changes of the way in which the electron's emission is interpreted. According to the Hopfield model, if the A activation were negative, the neuron would not emit $[p(1) = 0]$; if the activation were positive, the neuron would

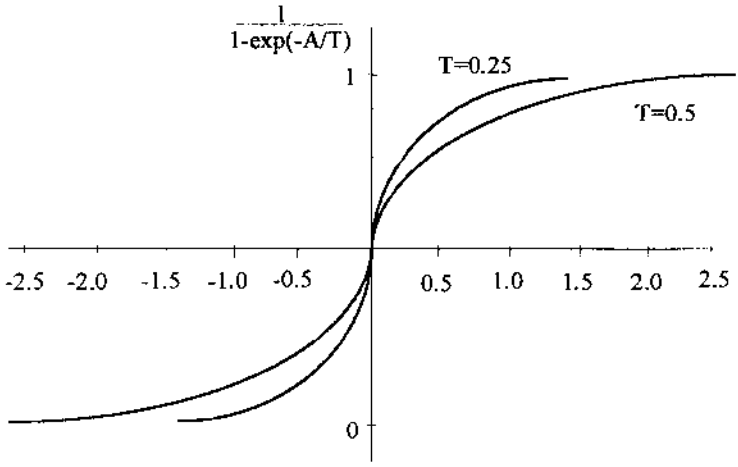

FIGURE 7.7 Plot of emission probability Boltzmann function.

have emitted $[p(1) = 1]$; this is shown in Figure 7.8a. The new model, named the Boltzmann machine, suggests that, for negative values of the A activation, there is a probability $p(1) = 1$ that the neuron emits; this probability goes to zero as A becomes negatively smaller and smaller. Similarly, there is a probability $p(0)$ that the neuron does not emit, even if the activation is positive; this probability gets lower when the activation gets higher (Fig. 7.8b).

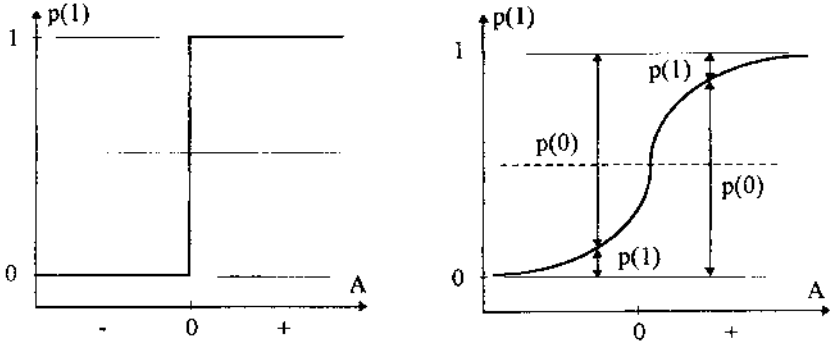

FIGURE 7.8 (a) Hopfield model; (b) new model, named the Boltzmann machine.

The Boltzmann machine showed that the probability that the network comes to an end in a certain state depends nearly exclusively on that state energy. That means that, through the learning process, the energies corresponding to the minimums of the system have to be well controlled.

Starting from the assumption that not all units of a network are defined by the learning dataset, Hinton showed the necessity of hidden units, essential for solving the hard learning problem. The learning procedure, based merely on information as to whether the visible units behave correctly, has to ensure that hidden units develop their own correct weights and thresholds, too. If we assume a neural network has ν visible units, it can have 2^{ν} possible states. If S_1, $S_2, \ldots,$ S_r are input datasets for training the network through visible units, state probabilities can be calculated: $P^+(S_1)$, $P^+(S_2), \ldots,$ $P^+(S_r)$, where the "+" sign is used to indicate that these are desired probabilities. On the other hand, the "−" sign is used for the same probabilities, but results from the free (untrained) run of the network, that is, $P^-(S_1)$, $P^-(S_2), \ldots, P^-(S_r)$. The function introduced to measure "the distance" between these two probability sets is:

$$G = \sum_a P^+(S_a) \ln \frac{P^+(S_a)}{P^-(S_a)} \tag{7.5}$$

The first term of the previous relation, $P^+(S_a)$, makes the states with the largest occurrence probability have the largest effect upon the sum; the natural logarithm becomes zero when the two probability sets are identical $[P^+(S_a) = P^-(S_a)]$.

It can be shown that the "distance" changing rate between the two probability sets depends on the trained, respectively, untrained, visible units' temperature and average state probabilities, after the relation:

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T}(p_{ij}^+ - p_{ij}^-) \tag{7.6}$$

The meaning of this relation consists of the following: in order to lower the rate G by changing the weight w_{ij}, all that needs to be known is the local information $(p_{ji}^+ - p_{ji}^-)$. If this term is positive, the weight w_{ij} has to be increased; in the opposite case, it has to be decreased. Obviously, the process is finished when $G = 0$, so the network learned to reproduce the state probabilities, and it is considered completely trained.

As shown before, there are two types of neural networks: feedforward and feedback. For feedback networks it is simple to introduce into the loop the signal associated with the output error, and this is done on purpose to diminish this error while training. As far as feedforward networks are concerned, they are the most utilized ones, and they cannot work freely because of the lack of reaction to inform; therefore, a training method for the hidden units is necessary, through a process of propagation of the measured error on a backward direction, from the output channel.

The hidden layers can be considered the location in the network in which input parameters are partially processed and labeled before the final result is reached in the output layer. In these layers, representations are formed, which are not provided during the training process. A generalization of the delta rule forms the basis of the hidden units' behavior through the converse error propagation process. The goal of this method is to minimize the overall output error εp, defined as the half-sum of the squares of all neurons' output errors:

$$\varepsilon_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \tag{7.7}$$

where t_{pj} is the target output of the neuron j for the input dataset p; o_{pj} is the actual output of the neuron j for that input dataset.

The training process through converse error propagation has two steps. The first is the forward step, during which the input data are applied and let run to the output channel. The output parameters are calculated and compared to the target parameters (which have to be known). During the backward step, the errors (those resulting as a consequence of the comparison in step one) are propagated backwards to the input layer. The purpose is to recalculate the neural connection weights. Another forward step follows, then another backward step, until the error minimizes within a preset limit. For this manner of feedforward neural network training, it has been noted that the most suitable neuron activation function is the same type as the function presented in relation (7.4) and Figure 7.7, also called the sigmoid function. A feedforward type neural network with three layers (input, hidden, and output), having a sigmoid activation function, and being trained through the converse error propagation method, can be represented schematically as in Figure 7.9. It should be noted that a larger number of neurons in the hidden layer can guarantee a better result in network training and usage.

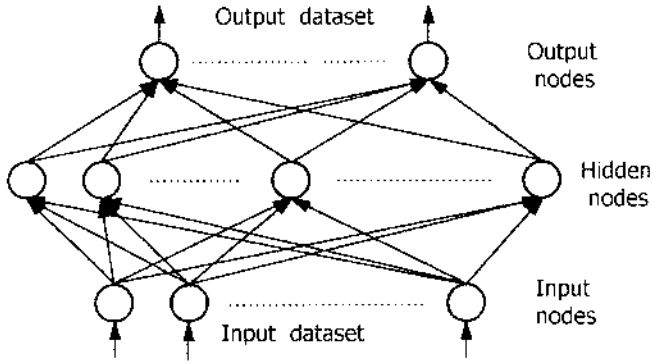

FIGURE 7.9 Feedforward type neural network with three layers represented schematically.

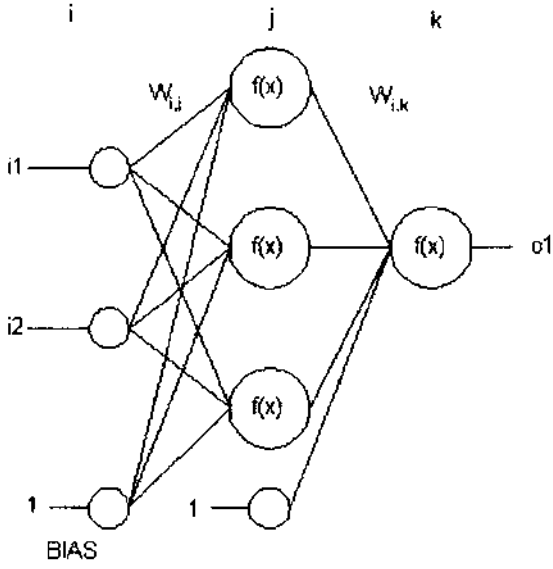

FIGURE 7.10 Structure of Windows Neural Network.

7.2 UTILIZATION OF NEURAL NETWORKS IN MACHINE TOOLS DIAGNOSTICS

7.2.1 Multicriteria Application for "Good/Defective" Classification

The problem of identifying defective elements in the kinematic structure of a machine tool can be solved by utilizing feedforward type neural networks. The following deals with elaboration and training of a neural network for bearing diagnosis. The data used for network training come from the experimental research presented in Chapter 5. Windows Neural Network is a user interface, written in Visual Basic, for building and utilizing feedforward neural networks, fully connected and trained through the converse error propagation algorithm. The structure of the neural networks with which this application works (Fig. 7.10) contains the input layer, one or more hidden layers, and the output layer. The output channel of each neuron is connected to all neural input channels of the next layer, which can introduce BIAS units (tendency or prediction units), which facilitate the network's training. There is no activation function in the input layer; this layer can only distribute the input data to the first hidden layer. The hidden layers and the output layer have an activation function, which can be:

A linear function:

$$f(x, T) = xT \tag{7.8}$$

A hyperbolic tangent function:

$$f(x, T) = \text{th}(xT) \tag{7.9}$$

A sigmoid function:

$$f(x, T) = 1[1 + \exp(-xT)] \tag{7.10}$$

From a mathematical point of view, the network runs the following algorithm.

For the hidden layer output:

$$\begin{aligned} h(j) &= \sum_i w(i, j)xi(i) \\ s(i) &= f(h(j)) \end{aligned} \tag{7.11}$$

For the output layer output:

$$h'(k) = \sum_j w(j,k)xs(j)$$
$$o(k) = f(h'(k)) \tag{7.12}$$

where $i(i)$ represents the network input channels; $o(k)$, the network output channels; $w(i,j)$, the weight of the connection of the neuron I with the neuron j in the next layer; and f, the activation function.

The objective function has to minimize the final error:

$$\varepsilon_{RMS} = \sum_{p=\mathrm{I}}^{p_{\max}} \sum_{k=\mathrm{I}}^{k_{\max}} [t(p,k) - o(p,k)]^2 \tag{7.13}$$

where $t(p,k)$ is the output target value for the output dataset p, and $o(p,k)$ is the actual output value for the same dataset.

The DIAGNO neural network for identifying defective bearings has the structure shown in Figure 7.11. The network's dimension is given by the number of layers and the number of neurons in each layer; consequently, the built network is a 3 × 4 × 1 type, with BIAS units in layers 2 and 3. The neuron activation function is the sigmoid function (Fig. 7.10).

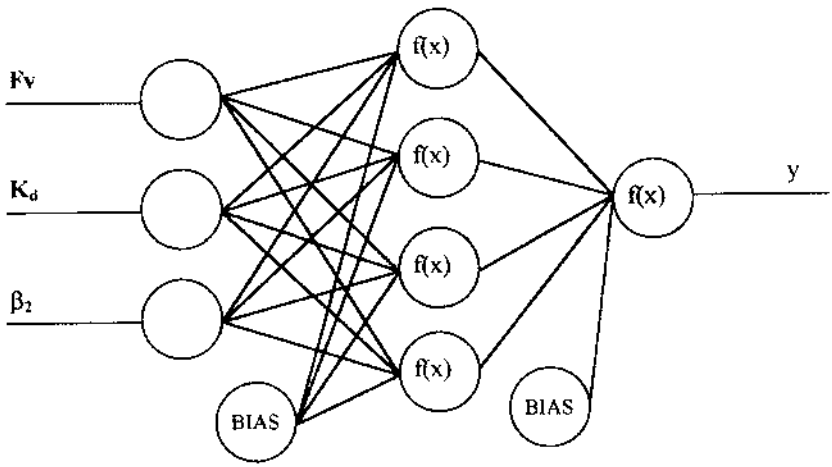

FIGURE 7.11 DIAGNO neural network for identifying defective bearings.

The selection criteria for data needed to train the network are the peak factor criterion, the diagnosis index, and the Kurtosis index. Data corresponding to the evaluation through these criteria come from the database of vibration signals stored and processed in the time field (see Chapter 5); they are provided to the network in three ways. The training datasets, containing input data and the corresponding output target parameter, are shown in Table 7.2. These data cannot be introduced in the network as they are, since they could not be processed by the activation functions. For instance, the sigmoid function can only take values between -2 and 2 at the input channel; other values would saturate the neuron and lead to an output with value 0 or 1 all the time. To avoid this inconvenience, the input data should be logarithmically or linearly normalized, as the variation field of those data is larger or narrower. This operation is done automatically, on demand, from the main menu of the application; the normalization can be global (to all of the network's nodes) or individual (node by node).

The next step is represented by the network's converse propagation algorithm adjustment, through the two parameters that connect the new weights with the derivatives of the old weights, after the relations:

$$\begin{aligned} dW(i,j,t+1) &= \eta dW(i,j,t) + \alpha dW(i,j,t-1) \\ W(i,j,t+1) &= W(i,j,t) + dW(i,j,t) \end{aligned} \tag{7.14}$$

where η is the learning parameter and α the moment parameter. The best results were obtained for $\eta = 0.2$ and $\alpha = 0.5$, as the front panel of this application shows (Fig. 7.12). It should be mentioned that during the network run there was no need to introduce "noise" and "temperature" to help the network avoid getting stuck in local minimums. Since temperature is a multiplicator of the activation function argument, not to take it into consideration means $T = 1$.

The structural network training process runs in epochs. An epoch represents the number of input datasets in terms of which weights are calculated. Thus, if the length of an epoch is 1, a continuous training is done (weights are recalculated after each dataset); if the length of an epoch is equal to the number of datasets, a simultaneous training is done (weights are recalculated after each pass over all datasets). Simultaneous training is useful when there are few input datasets, as in the present case. At the same time, this sort of training is faster than continuous training. Training this neural network needed 555 iterations, that is, 555 passes over the input datasets. Training ended when the target was

TABLE 7.2 Training Datasets, Containing Input Data and Corresponding Output Target Parameter

Set number	0	1	2	3	4	5	6	7	8	9
Peak factor	10.0	7.5	20.0	27.0	6.0	25.5	26.0	29.0	14.0	24.5
Diagnosis index	.500	.650	.170	.013	.800	.018	.015	.010	.400	.019
Kurtosis index	4.1	3.7	7.2	9.2	3.4	9	9.3	12.0	5.8	8.8
Target output	1	1	0	0	1	0	0	0	1	0

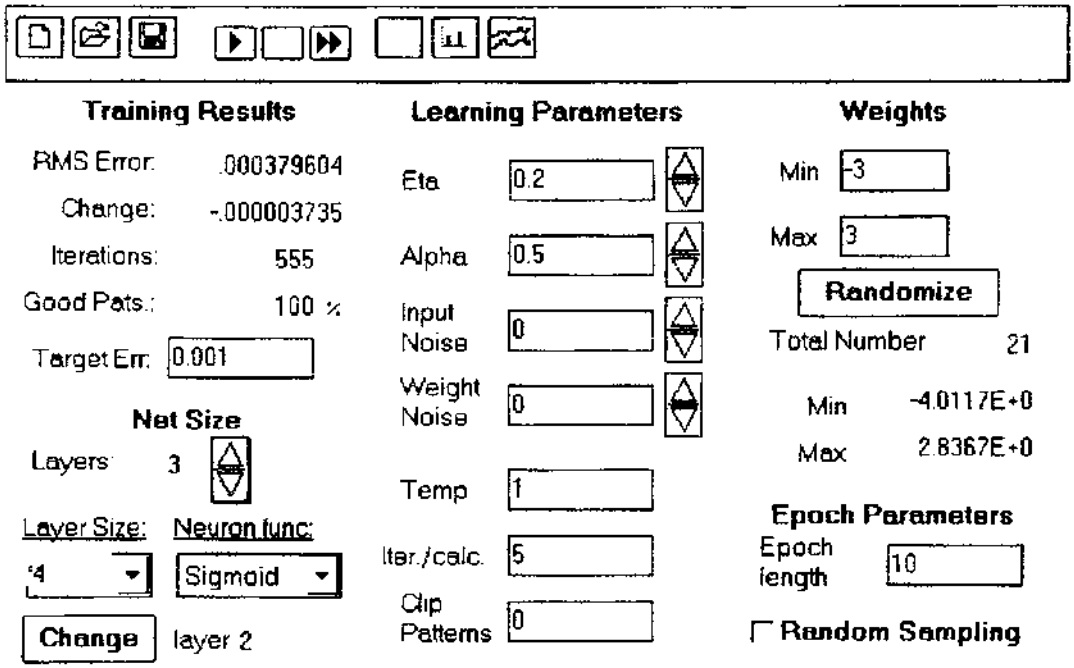

FIGURE 7.12 Front panel caption.

TABLE 7.3 Weights w_{ij} Calculated for Each Neural Connection

Total calculated weights: 21				
Between input layer and second hidden layer:				
2.567564	−0.552004	−0.623598	0.690139	—
1.273501	0.300688	1.931408	−0.233710	—
−3.382554	1.183681	0.871934	1.333970	—
2.154515	−0.137544	−0.532012	−1.897794	—
Between second hidden layer and third output layer:				
−2.092981	−2.371565	4.373098	−1.461245	−0.007646

reached with an error smaller than the preset one, in this case. It can be noticed in the window with the training results (Fig. 7.12) that the actual root mean square error is $\varepsilon_{\text{RMS}} = 0.000379604$.

As a consequence of training the network with experimental data, weights w_{ij} have been calculated for each neural connection (Table 7.3), through the converse error propagation algorithm. Weight distribution in the network is shown in Figure 7.13 in the shape of a 10-interval histogram. The number of weights in each interval is represented in terms

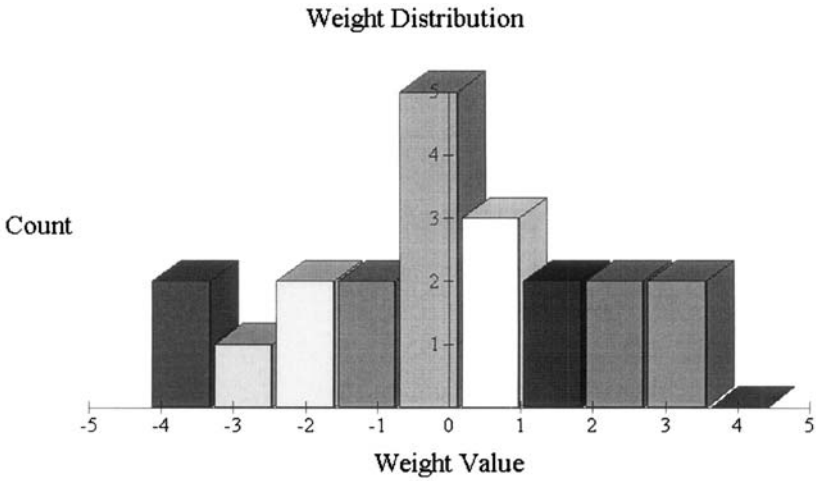

FIGURE 7.13 Weight distribution in the network.

of the weights' value, between −4 and +4. This representation provides information about the quality of the result obtained in the calculus of weights: normally, for a network with the same activation function in all layers, the weight distribution should be as close as possible to a Gauss distribution. Figure 7.13 shows the weight distribution for a DIAGNO neural network, correlated with data in Table 7.3. The shape of this histogram gets closer to a Gauss distribution, so the weights are correct. In these conditions, a graphical representation of the actual network outputs can be plotted at the same time as the target outputs. In Figure 7.14 the network's target outputs, as they were set (Table 7.2), are drawn with a solid line, and the trained network's actual outputs with a dashed line.

The error between the target output and the actual output can also be plotted, as in Figure 7.15. It can be clearly seen that the neural network observes some trouble in interpreting datasets 2 and 8 (see Table 7.2). Indeed, during the training process, the two input datasets had been introduced artificially by the author within categories "defective" and "good."

At this time, the DIAGNO neural network training process can be considered finished and the network can be utilized in the classification. Tests of the built neural network have been carried out on a group of four datasets known from experimental research; these datasets are presented in Table 7.4. It should be noted that this time the input target value for each input dataset was not indicated; the neural network will have

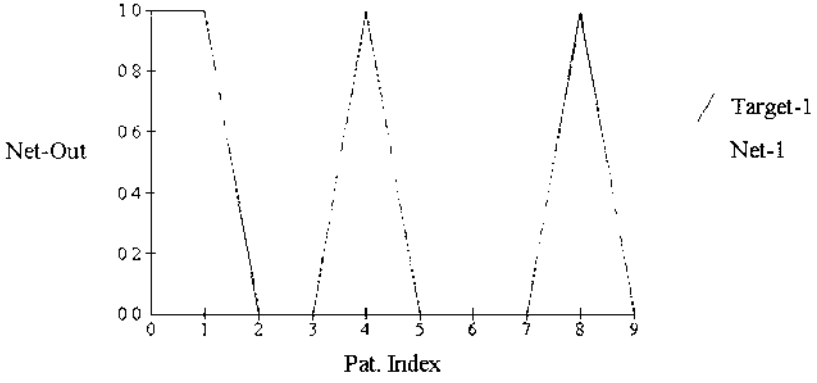

FIGURE 7.14 Network's target outputs, as they were set (Table 7.2) (solid line) and trained network's actual outputs (dashed line).

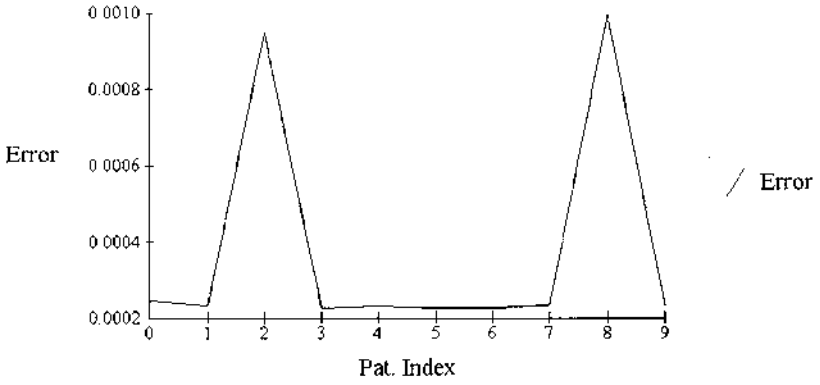

FIGURE 7.15 Error between target and actual output.

to perform the set classification, in the manner it learned during the training process. The weights of neural connections calculated during the training process were used as they were during the test. After the normalizing operation, the test data were plugged into the network. The training parameters were kept the same ($\eta = 0.2$ si $\alpha = 0.5$); it was not necessary to use noise or temperature to avoid the network getting stuck in false minimums. Data for running the classification test for the four bearing types from which input data were taken are shown in Table 7.5. The root mean square error in the test was smaller than the one that resulted in training the network, as was expected.

Figure 7.16 illustrates the result of the neural network test: indeed, the first two test datasets belonged to two radial ball bearings 6209 which were working perfectly; the last two datasets were taken from the same bearings, having pitting on the outer race track due to an intensive wear on the stand described in Chapter 5.

TABLE 7.4 Test of the Built Neural Network Carried Out on a Group of Four Datasets Known from Experimental Research

Set number	0	1	2	3
Peak factor	6.50	11.00	26.70	25.40
Diagnosis index	0.780	0.480	0.013	0.017
Kurtosis factor	3.50	4.50	9.70	9.10

TABLE 7.5 Data for Running the Classification Test for Four Bearing Types from Which Input Data Were Taken

RMS error: 0.000103 Good pats: 100.0%			
Set number	Output actual value	Output adopted value	Error
0001	0.994650	1.000000	0.000029 +
0002	0.993171	1.000000	0.000047 +
0003	0.012820	0.000000	0.000164 +
0004	0.013081	0.000000	0.000171 +

In conclusion, the DIAGNO neural network is able, after a preliminary learning process, to distinguish perfectly working bearings and defective bearings with remarkable accuracy. Other tests carried out with the same neural network, over other experimental datasets, gave the same results every time.

7.2.2 Application for Neural Diagnosis of the Working State

The goal of a diagnostic operation is to detect "ante factum" a defect, in order to prevent a machine's being out of operation due to damage.

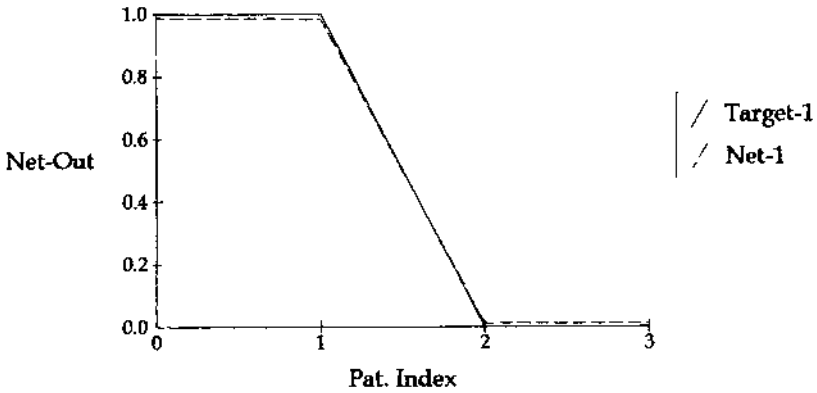

FIGURE 7.16 Result of neural network test.

Starting from this consideration, a neural network utilized for monitoring and diagnosing should be able to evaluate the real working state of the supervised element. Therefore, a neural network named DIAGNOZA has been built, having the structure shown in Figure 7.17; this network has been trained to classify supervised bearings into four categories:

Bearings in a perfect working state (target $y = 1$)
Bearings that have conditions for defects to occur (target $y = 2$)
Bearings in a limited working state (target $y = 3$)
Defective bearings (target $y = 4$)

The network is of the multilayer feedforward type, size $2 \times 5 \times 1$, with BIAS type prediction units, able to be trained through the converse error propagation algorithm. The neuron activation function is a sigmoid one [relation (7.10)], the same for neurons in all layers. The network is fully connected, meaning each neural output channel from a layer is connected to all neural input channels in the next layer.

The two network input channels were given data from the experimental research and interpolations of those data (Table 7.6); the classification criteria considered this time were the peak factor criterion (Fv)

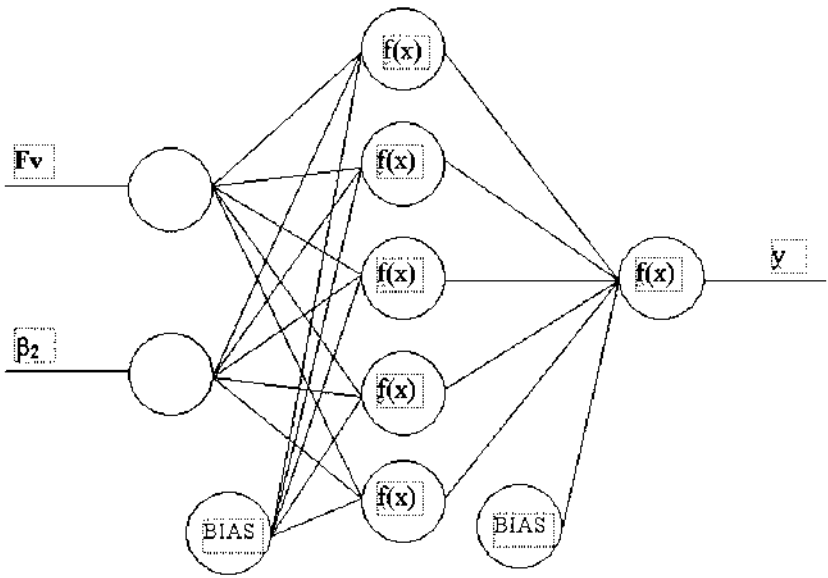

FIGURE 7.17 Structure of DIAGNOZA neural network.

TABLE 7.6 Data from Experimental Research and Interpolations of Those Data

Set no.	1	2	3	4	5	6	7	8	9
Peak factor	25.25	14.50	7.36	26.64	24.21	6.64	26.35	22.89	17.12
Kurtosis index	9.79	5.12	3.61	11.23	8.73	3.47	10.68	8.13	6.25
Target	4	2	1	4	3	1	4	3	2
Set no.	10	11	12	13	14	15	16	17	18
Peak factor	19.07	23.64	27.01	8.71	5.72	8.25	20.85	25.98	12.14
Kurtosis index	6.70	8.46	11.88	4.09	3.06	3.88	7.45	10.19	4.88
Target	2	3	4	1	1	1	3	4	2
Set no.	19	20	21	22	23	24	25	26	—
Peak factor	15.63	26.83	6.12	6.00	14.07	20.03	25.55	21.79	—
Kurtosis index	5.73	11.57	3.24	3.14	4.81	7.22	9.59	7.89	—
Target	2	4	1	1	2	3	4	3	—

and Kurtosis index criterion (β_2). Data were linearly normalized in order to be accepted by the neuron activation function. The best results were obtained for the values of the learning parameters $\eta = 0.2$ and $\alpha = 0.5$, without using noise or temperature as aiding factors in the network. On the front panel of this application (see Fig. 7.18), the results of the network training can be read, after using the converse error propagation method. Thus, after a relatively large number of iterations (6140), the working root mean square error decreased under the preset value $\varepsilon = 0.001$ for all of the input datasets.

As a consequence of the training process, 21 weights of the neural connections were evaluated: 15 connections between layers 1 and 2, and 6 connections between layers 2 and 3, also taking into consideration the connections of the BIAS prediction units in layers 2 and 3. These weights are presented synthetically in Table 7.7. The histogram of this weight distribution is shown in Figure 7.19. The envelope of these histograms is close to a Gauss distribution, which confirms the weight calculus was correct.

The neural network running on the input data was performed using the simultaneous training method (the length of an epoch is equal to the number of datasets), so the weights were recalculated after each pass over all 26 datasets, and this allowed the training time to be shortened. Table 7.8 presents the results of the run: the output actual values, the target output values, and the interpreting error for each training dataset. A graphical plot of the actual and target outputs is shown in Figure 7.20; Figure 7.21 shows the output error histogram of the trained network. It should be noted that the neural network had classification trouble with sets 5 and 25, where the concordance between the evaluation criteria was not too good.

Training of the DIAGNOZA neural network can be considered successfully finished; DIAGNOZA may now be utilized in the problems of diagnostic classification for which it was built. The test for the DIAGNOZA neural network was performed on a group of six experimental datasets (Table 7.9), coming up after monitoring the forced wear of the same radial ball bearings type 6209 (see Chapter 5).

After data were linearly normalized, they ran in a trained neural network. The output root mean square error was 0.000307, therefore under the limit of 0.001 imposed for all sets. The network identified accurately the working state of the tested bearings; furthermore, during the test a false target was indicated for one of the input datasets, but the classification performed by the network did not change. This validated

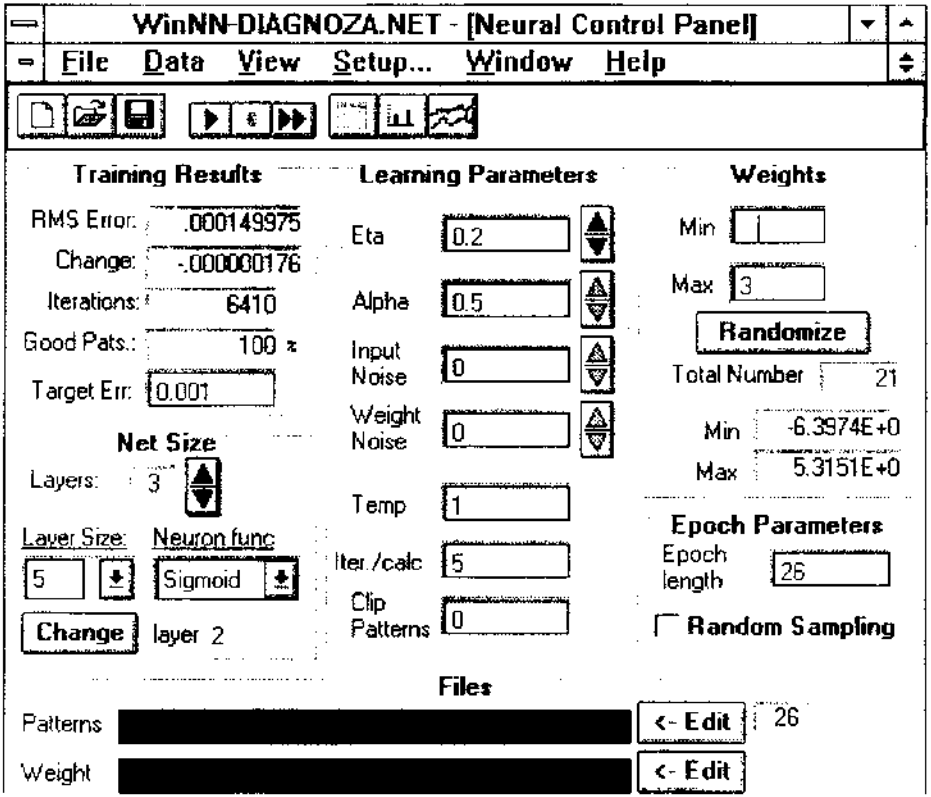

FIGURE 7.18 Results of network training can be read after using converse error propagation method.

TABLE 7.7 Twenty-One Weights of the Neural Connections Evaluated

Total evaluated weights: 21					
Between layer 1 and layer 2					
1.754083	−1.154256	−0.962436	—	—	—
4.748973	5.315055	−2.021535	—	—	—
3.361797	−1.410271	3.961644	—	—	—
1.547162	−6.397412	4.818911	—	—	—
−3.090095	0.540312	−5.167025	—	—	—
Between layer 2 and layer 3					
−2.278813	1.963403	5.296241	−4.803792	−3.292156	0.073231

that a good training of the network was achieved. Figure 7.22 represents the actual values (the white bar) and the adopted values (the black bar) of the network output, in the test case.

In conclusion, training of DIAGNO and DIAGNOZA neural networks with data obtained from experimental research led to excellent performance of recognition and classification of elements in the machine tool structure. The tests performed—the most representative ones have been presented here—confirm the capabilities of this type of information processing.

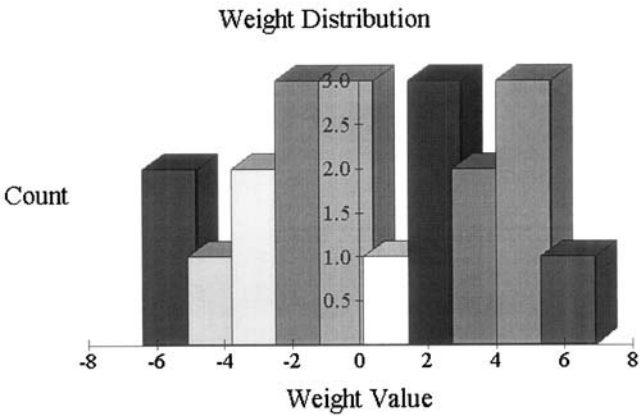

FIGURE 7.19 Histogram of weight distribution presented in Table 7.7.

TABLE 7.8 Output Actual Values, Target Output Values, and Interpreting Error for Each Training Dataset

RMS error: 0.000150
Good pats: 100.0%

Set	Actual value	Target value	Error	Set	Actual value	Target value	Error
0001)	3.966598	4.000000	0.000124+	0002)	2.027430	2.000000	0.000084+
0003)	1.008047	1.000000	0.000007+	0004)	3.988856	4.000000	0.000014+
0005)	3.087048	3.000000	0.000842+	0006)	1.003739	1.000000	0.000002+
0007)	3.987121	4.000000	0.000018+	0008)	2.956078	3.000000	0.000214+
0009)	1.933849	2.000000	0.000486+	0010)	2.038465	2.000000	0.000164+
0011)	2.988344	3.000000	0.000015+	0012)	3.990406	4.000000	0.000010+
0013)	1.042064	1.000000	0.000197+	0014)	1.002214	1.000000	0.000001+
0015)	1.024447	1.000000	0.000066+	0016)	3.014108	3.000000	0.000022+
0017)	3.982717	4.000000	0.000033+	0018)	1.977742	2.000000	0.000055+
0019)	2.026888	2.000000	0.000080+	0020)	3.989686	4.000000	0.000012+
0021)	1.002707	1.000000	0.000001+	0022)	1.002615	1.000000	0.000001+
0023)	2.003102	2.000000	0.000001+	0024)	2.939797	3.000000	0.000403+
0025)	3.905148	4.000000	0.001000+	0026)	3.020487	3.000000	0.000047+

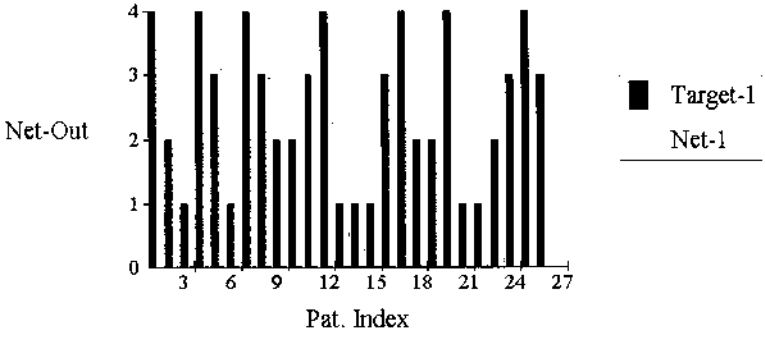

FIGURE 7.20 Graphical plot of actual and target outputs.

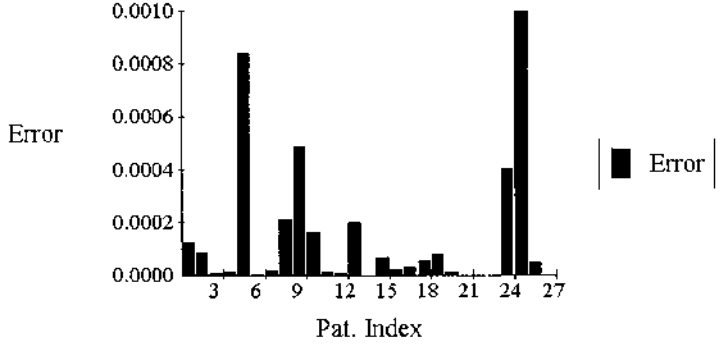

FIGURE 7.21 Output error histogram of trained network.

TABLE 7.9 Test for DIAGNOZA Neural Network Performed on 6 Experimental Datasets

Set No.	1	2	3	4	5	6
Peak factor	8.53	23.14	16.05	25.73	18.56	23.91
Kurtosis index	3.97	8.29	5.48	9.81	6.50	8.60

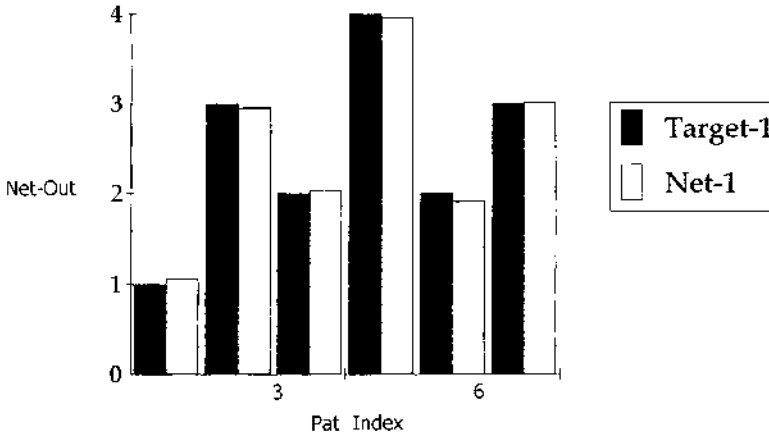

FIGURE 7.22 Actual values (white bar) and adopted values (black bar) of network output in the bearings test.

7.3 FINAL REMARKS AND PROSPECTS FOR UTILIZING NEURAL NETWORKS FOR MACHINE TOOL DIAGNOSIS

As noted, utilization of neural networks allowed the exploitation at a higher level of the data library coming from experimental work and running. The success rate of the neural diagnosis depends mainly on the following factors.

1. Neural network architecture: The number of the network's hidden layers can be neither too high nor too low. A network without hidden layers becomes a simple linear separator and does not reach its goal; on the other hand, a too large number of hidden layers makes the learning process inefficient. Usually, the number of hidden layers is 1 or 2. During the DIAGNOZA neural network elaboration, the same input datasets and the same initial aleatory weights were used for two networks: type $2 \times 5 \times 1$ and type $2 \times 4 \times 4 \times 1$. It was noticed that—although both networks had a 100% success rate—the learning process took longer in the case of the network with two hidden layers (the rapidity of convergence cannot be a disadvantage, since the learning process is offline). The classification errors were nearly one size order larger for more than half of the training datasets.

2. The training dataset size: It is well known and experimentally proven that neural network performance increases when the training dataset is larger. It is recommended that, when evaluating the necessary number of datasets for training, the complexity of the problem to be learned by the neural network be taken into consideration.

Information processing by means of neural networks proved to be a viable alternative for classical monitoring and diagnostic techniques. A comparison between the success rates of these techniques and those based on neural processing (Table 7.10) proves that the latter are better.

As noted, the neural networks elaborated were trained with data that had already undergone a previous process within criterial estimations (the peak factor criterion, the diagnosis index criterion, or Kurtosis criterion). But these networks are also capable of learning from unprocessed data, provided the target is indicated correctly. Such an approach could be done by building a neural network that can be trained by means of the captured signal; this network can be represented by a power spectrum. Figure 7.23 illustrates this spectacular change from neural networks fed with structured data (Fig. 7.23a) to a neural network able to recognize defects through a global representation of vibration signals (Fig. 7.23b).

In previous chapters, an analysis of characteristic frequencies was described for some elements of a kinematic chain structure. It was shown that, within the power spectrum obtained by applying the fast Fourier transform to the signal captured in the time field, the increase of amplitude for these frequencies is the result of certain typical defects' nucle-

TABLE 7.10 Comparison Between the Success Rates of Classical Monitoring and Diagnosing Techniques and Those Based on Neural Processing

No.	Monitoring/diagnosing technique	Success rate (%)
1.	Peak factor method	50–70
2.	Diagnosis index method	62–70
3.	Kurtosis method	67–75
4.	Spectrum comparison method	70–85
5.	Envelope method	75–85
6.	Diagnosis neural networks	95–100
7.	Classification neural networks	98–100

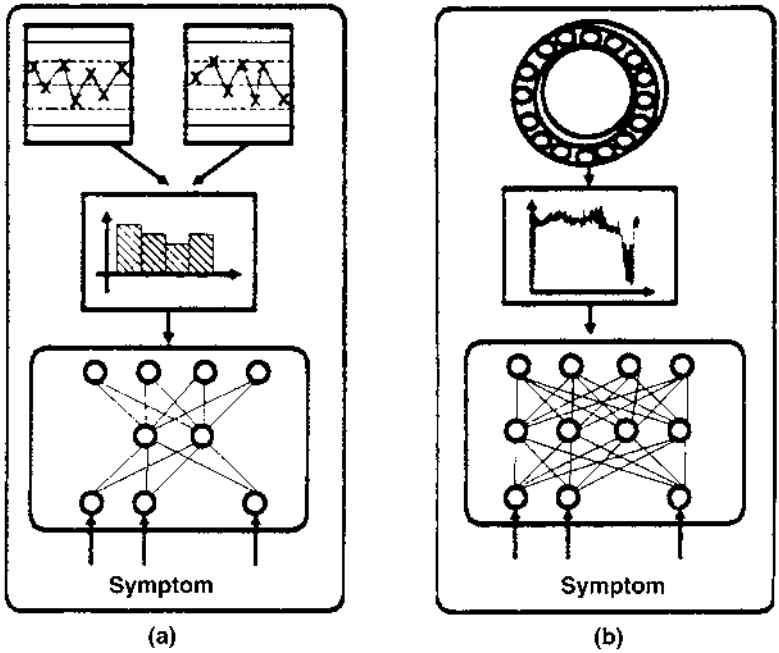

FIGURE 7.23 (a) Neural networks fed with structured data; (b) neural network able to recognize defects through global representation of vibration signals.

ation. The nature and proportion of these defects can be identified by supervising the characteristic frequencies.

It is possible to build a diagnostic and classification system for defects on the basis of power spectrum recognition, by means of a principle schema similar to the one in Figure 7.24. In the signal captured and then processed in the frequency field, characteristic frequencies and eventually other characteristic parameters should be monitored; these parameters should take the shape of a state vector. A neural network trained with this sort of dataset provides at the output the interpretation of those data from a diagnostic viewpoint. Using a simple algorithm, these output parameters are related to different types of defects for the monitored/diagnosed element; the probability of the occurrence of these defects is indicated.

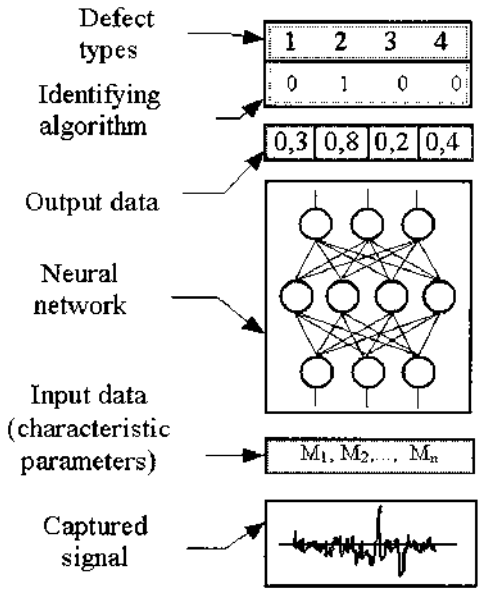

FIGURE 7.24 Principle schema of diagnostic and classification system for defects on basis of power spectrum recognition.

Elaboration of these neural networks represents a major step in developing diagnostic expert systems. Since the algorithmic method of learning is generally accepted for expert systems over the method of learning from experience, two components are necessary to build neural networks: a memorized list of rules and a set of procedures that allows conclusion interpolation by utilizing those rules and experimental data. Under these circumstances, the information neural processing would be the most advantageous way to assimilate the experience.