Mathematical literature reveals that the number of neural network structures, concepts, methods, and their applications have been well known in neural modeling literature for sometime. It started with the work of McCulloch and Pitts [93], who considered the brain a computer consisting of well-defined computing elements, the neurons. Systems theoretic approaches to brain functioning are discussed in various disciplines like cybernetics, pattern recognition, artificial intelligence, biophysics, theoretical biology, mathematical psychology, control system sciences, and others. The concept of neural networks have been adopted to problem-solving studies related to various applied sciences and to studies on computer hardware implementations for parallel distributed processing and structures of non-von Neuman design.

In 1958 Rosenblatt gave the theoretical concept of "perceptron" based on the neural functioning [105]. The adaptive linear neuron element (adaline), which is based on the perceptron theory, was developed by Widrow and Hopf for pattern recognition at the start of the sixties [131]. It is popular for its use in various applications in signal processing and communications. The inductive learning technique called group method of data handling (GMDH) and which is based on the perceptron theory, was developed by Ivakhnenko during the sixties for system identification, modeling, and predictions of complex systems. Modified versions of these algorithms are used in several modeling applications. Since then, one will find the studies and developments on perceptron-based works in the United States as well as in other parts of the world [3], [26], [82].

There is rapid development in artificial neural network modeling, mainly in the direction of connectionism among the neural units in network structures and in adaptations of "learning" mechanisms. The techniques differ according to the mechanisms adapted in the networks. They are distinguished for making successive adjustments in connection strengths until the network performs a desired computation with certain accuracy. The least mean-square (LMS) technique that is used in adaline is one of the important contributions to the development of the perceptron theory. The back propagation learning technique has become well known during this decade [107]. It became very popular through the works of the PDP group who used it in the multilayered feed-forward networks for various problem-solving.

1 SELF-ORGANIZATION MECHANISM IN THE NETWORKS

Any artificial neural network consists of processing units. They can be of three types: input, output, and hidden or associative. The associative units are the communication links between input and output units. The main task of the network is to make a set of associations of the input patterns x with the output patterns y. When a new input pattern is added to the configuration, the association must be able to identify its output pattern. The units are connected to each other through connection weights; usually negative values are called inhibitory and positive ones, excitatory.

A process is said to undergo self-organization when identification or recognition categories emerge through the system's environment. The self-organization of knowledge is mainly formed in adaptation of the learning mechanism in the network structure [5], [8]. Self-organization in the network is considered while building up the connections among the processing units in the layers to represent discrete input and output items. Adaptive processes (interactions between state variables) are considered within the units.

Linear or nonlinear threshold functions are applied on the units for an additional activation of their outputs. A standard threshold function is a linear transfer function that is used for binary categorization of feature patterns. Nonlinear transfer functions such as sigmoid functions are used to transform the unit outputs. Threshold objective functions are used in the inductive networks as a special case to measure the objectivity of the unit and to decide whether to make the unit go "on" or "off." The strategy is that the units compete with each other and win the race. In the former case the output of the unit is transformed according to the threshold function and fed forward; whereas in the latter, the output of the unit is fed forward directly if it is "on" according to the threshold objective function. A state function is used to compute the capacity of each unit. Each unit is analyzed independently of the others. The next level of interaction comes from mutual connections between the units; the collective phenomenon is considered from loops of the network. Because of such connections, each unit depends on the state of many other units. Such a network structure can be switched over to self-organizing mode by using a statistical learning law. A learning law is used to connect a specific form of acquired change through the synaptic weights—one that connects present to past behavior in an adaptive fashion so that positive or negative outcomes of events serve as signals for something else. This law could be a mathematical function, such as an energy function that dissipates energy into the network or an error function that measures the output residual error.

A learning method follows a procedure that evaluates this function to make pseudorandom changes in the weight values, retaining those changes that result in improvements to obtain the optimum output response. Several different procedures have been developed based on the minimization of the average squared error of the unit output (least squares technique is the simplest and the most popular).

$$\varepsilon = \frac{1}{2}\sum_{j,i}(\hat{y} - y)_{j,i}{}^2, \tag{7.1}$$

where $\hat{y}_j$ is the estimated output of jth unit depending on a relationship, and y_j is the desired output of the ith example. Each unit has a continuous state function of their total input and the error measure is minimized by starting with any set of weights and updating each weight w by an amount proportional to $\partial\varepsilon/\partial w$ as $\delta w_{ij} = -\alpha\,\partial\varepsilon/\partial w_{ij}$, where α is a learning rate constant.

The ultimate goal of any learning procedure is to sweep through the whole set of associations and obtain a final set of weights in the direction that reduces the error function. This is realized in different forms of the networks [29], [77], [107], [131].

The statistical mechanism built in the network enables it to adapt itself to the examples of what it should be doing and to organize information within itself and, thereby, to learn. The collective computation of the overall process of self-organization helps in obtaining the optimum output response.

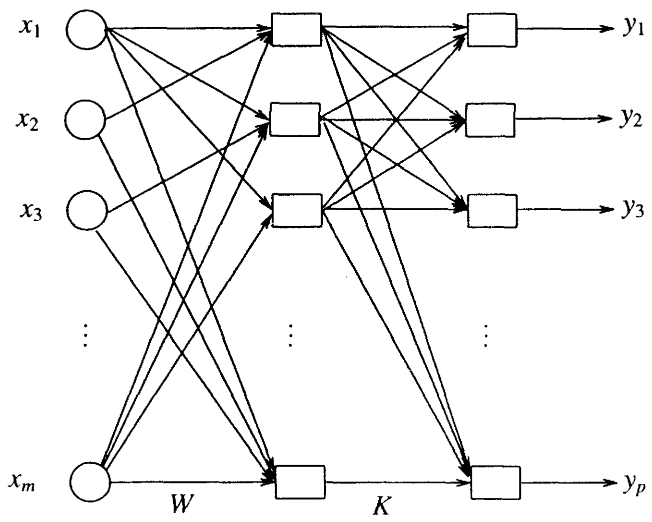

Figure 7.1. Unbounded feedforward network where X and Y are input/output vectors and W and K are weight matrices

This chapter presents differences and commonalities among inductive-based learning algorithms, deductive-based adaline, and backpropagation techniques. Multilayered inductive algorithm, adaline, backpropagation, and self-organization boolean logic techniques are considered here because of their commonality as parallel optimization algorithms in minimizing the output residual error and for their inductive and deductive approaches in dealing with the state functions. Self-organizing processes and criteria that help in obtaining the optimum output responses in the algorithms are explained through the collective computational approaches of these networks. The differences in empirical analyzing capabilities of the processing units are described. The relevance of local minima depends on various activating laws and heuristics used in the networks and knowledge embedded in the algorithms. This comparison study would be helpful in understanding the inductive learning mechanism compared with the standard neural techniques and in designing better and faster mechanisms for modeling and predictions of complex systems.

1.1 Some concepts, definitions, and tools

Let us consider a two-layered feedforward unbounded network with the matrices of connected weights of W at first layer and K at output layer (Figure 7.1). The functional algorithm is as follows:

Step 1, Initialize with random weights. Apply set of inputs and compute resulting outputs at each unit.

Step 2. Compare these outputs with the desired outputs. Find out the difference, square it, sum all of the squares. The object of training is to minimize this difference.

Step 3. Adjust each weight by a small random amount. If the adjustment helps in minimizing the differences, retain it; otherwise, return the weight to its previous value.

Step 4. Repeat from step 2 onward until the network is trained to the desired degree ofminimization.

Any statistical learning algorithm follows these four steps. In working with such self-organization networks, one has to specify and build certain features of the network such as type of "input-output" processing, state function, threshold transfer function (decision function), and adopting technique. Overall, the networks can be comprised according to the following blocks:

1. "Black box" or "input-output" processing
   - batch processing
   - iterative processing
   - deductive approach (summation functions are based on the unbounded form of the network)
   - inductive approach (summation functions are based on the bounded form of the network)
   - multi-input single output
   - multi-input multi-output
2. Considering state functions
   - linear
   - nonlinear [29], [103], [132]
   - boolean logic
   - parallel
   - sequential
3. Activating with threshold transfer functions
   - linear threshold logic unit (TLU)
   - nonlinear or sigmoid
   - objective function (competitive threshold without transformations)
4. Adapting techniques
   - minimization of mean square error function (simplest case)
   - backpropagation of the output errors
   - minimizing an objective function ("simulated annealing")
   - front propagation of the output errors.

Some of the terminology given above are meant mainly for comparing self-organization networks. The term "deductive approach" is used for the network with unbounded connections and a full form of state function by including all input variables—contrary to the inductive approach that considers the randomly selected partial forms.

State functions

Unbounded structure considers the summation function with all input variables at each node:

$$s_j = \sum_{i=1}^{n} w_{ji}x_i + w_{j0}, \tag{7.2}$$

where n is the total number of input variables; s_j is the output of the node; x_i are the input terms; w_{j0} is the biased term, and w_{ji} are the connection weights.

Bounded structure considers the summation function with a partial list (r) of input variables:

$$s_j = \sum_{i=n1}^{n2} w_{ji}x_i + w_{j0}, \tag{7.3}$$

where $(n2 - n1) = r$ and $r+1$ is the number of the partial list of variables. A network with an unbounded/bounded structure with threshold logic function is called *deductive* because of its *apriori* fixedness. A network with a bounded structure and a threshold objective function is *inductive* because of its competitiveness among the units with randomly connected partial sets of inputs.

Parallel function is defined as the state function with the inputs from the previous layer or iteration $'j'$; whereas, the *sequential form* depends on the terms from the previous iteration and the past ones of the same iteration:

$$s_j = \sum_{k=1}^{r} w_{j,j-k}\, s_{j-k} + \sum_{i=n1}^{n2} w_{ji}\, x_i + w_{j0}. \tag{7.4}$$

The computationally sequential one takes more time and can be replaced by a parallel one if we appropriately choose input terms from the previous layer.

Transfer functions

These are used in the TLUs for activating the units. Various forms of transfer functions are used by scientists in various applications. The analytical characteristics of linear type TLUs are extensively studied by the group of Fokas [19]. Here is a brief listing of linear and nonlinear TLUs for an interested reader.

Linear type TLUs or discrete-event transformations. The following are widely used threshold logic functions in perceptron and other structures.

(i) *Majority rule:*

$$F(u) = \begin{array}{ll} 1 & \text{if } u > 0 \\ 0 & \text{if } u \leq 0; \end{array}$$

(ii) *Signum function:*

$$F(u) = \begin{array}{ll} 1 & \text{if } u > 0 \\ -1 & \text{if } u \leq 0; \end{array}$$

(iii) *Piecewise linear function:*

$$F(u) = \begin{array}{ll} u & \text{if } u > 0 \\ 0 & \text{if } u \leq 0; \end{array}$$

(iv) *Signum-0-function*:

$$F(u) - \begin{array}{ll} 1 & \text{if } u > 0 \\ 0 & \text{if } u = 0 \\ -1 & \text{if } u < 0; \text{ and} \end{array}$$

(v) *Parity rule:*

$$F(u) = 1 \quad \text{if } u \text{ is even}$$
$$0 \quad \text{if } u \text{ is zero or odd} . \tag{7.5}$$

This is used in cellular automata and soliton automata [19]. In all the cases u is unit output.

Nonlinear or discrete analogue transformations

(i) Here are some forms of sigmoid function ($F(u)$ = tanh u) often used in various applications. They provide continuous monotonic mapping of the input; some map into the range of —1 and 1, and some into the range of 0 and 1:

$$\begin{aligned}
F(u) &= (1+e^{-u'})^{-1};\\
F(u) &= (e^{u'}-e^{-u'}) * (e^{u'}+e^{-u'})^{-1}, \quad F'(u) = (1+F) * (1-F);\\
F(u) &= (1-e^{-2u}) * (1+e^{-2u})^{-1};\\
F(u) &= 2 * (1+e^{-2u})^{-1} - 1;\\
F(u) &= \frac{1}{2}(1+\tanh\, u'); \quad \text{and}\\
F(u) &= (e^{u}-1) * (e^{u}+1)^{-1},
\end{aligned} \tag{7.6}$$

where $u' = u * g$, in which g is the gain width. In all the nonlinear cases the curve has a characteristic sigmoidal shape that is symmetrical around the origin. For example, take the last one. When u is positive, the exponential exceeds unity and the function is positive, implying preference for growth. When u is negative, the exponential is less than unity and the function is negative, reflecting a tendency to retract. When u is zero, the function is zero, corresponding to a 50-50 chance of growth or retraction. For large positive values of u, the exponentials dominate each term and the expression approaches unity, corresponding to certain growth. For large negative values of u, the exponentials vanish and the expression approaches -1, corresponding to certain retraction. Here are some other types of transformations:

(ii) *Sine function:*

$$F(u) = \sin(u').$$

The use of this function leads to a generalized Fourier analysis.

(iii) *Parametric exponential function:*

$$F(u) = a + be^{-u'},$$

where a and b are the parameters;

(iv) *Gaussian function:*

$$F(u) = e^{\sum_i -\frac{1}{2}\frac{(u_i-\mu_i)^2}{\sigma_i^2}},$$

where μ is the mean value and a is the covariance term; and

(v) *Green function:*

$$F(u) = \sum_{\alpha=1}^{n} c_\alpha G(u; t_\alpha), \tag{7.7}$$

where c_α are coefficients which are unknown, and t_α are parameters which are called centers in the radial case [101].

Threshold objective functions. There are various forms of threshold objective functions such as regularity, minimum-bias, balance-of-variables, and prediction criterion, used mainly in inductive networks. These are built up based on objectives like regularization, forecasting, finding physical law, obtained minimum biased model or the combination of two or three objectives which might vary from problem to problem.

2 NETWORK TECHNIQUES

The focus here is on the presentation of emperical analyzing capabilities of the networks; i.e., multilayered inductive technique, adaline, backpropagation, and self-organization boolean logic technique, to represent the input-output behavior of a system. The aspects considered are: basic functioning at unit-level based on these approaches connectivity of units for recognition and prediction type of problems.

2.1 Inductive technique

Suppose we have a sample of N observations, a set of input-output pairs (I_1, o_1), $(I_2, o_2), \cdots$, $(I_N, o_N) \epsilon N$, where N is a domain of certain data observations, and we have to train the network using these input-output pairs to solve an identification problem. For the given input $I_j (1 \leq j \leq N)$ of variables x corrupted by some noise is expected to reproduce the output o_j and to identify the physical laws, if any, embedded in the system. The prediction problem concerns the given input I_{N+1} that is expected to predict exactly the output o_{N+1} from a model of the domain that it has learned during the training.

In the inductive approaches, a general form of summation function is considered Kolmogorov-Gabor polynomial which is a discrete form of Volterra functional series [21]:

$$\begin{aligned}\hat{y} &= a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \cdots \\ &= a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_{11} x_1^2 + a_{12} x_1 x_2 + \cdots + a_{111} x_1^3 + a_{112} x_1^2 x_2 + \cdots \\ &= a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_{11} x_{m+1} + a_{12} x_{m+2} + \cdots + a_{mm} x_{m1},\end{aligned} \tag{7.8}$$

where the estimated output is designated by $\hat{y}$, the external input vector x by $(x_1, x_2, \cdots, x_{m1})$, and a are the weights or coefficients. This is linear in parameters a and nonlinear in x. The nonlinear type functions were first introduced by the school of Widrow [132]. The input variables x could be independent variables or functional terms or finite difference terms; i.e., the function is either an algebraic equation, a finite difference equation, or an equation with mixed terms. The partial form of this function as a state functional is developed at each simulated unit and activated in parallel to build up the complexity.

Let us see the function at the unit level. Assume that unit n receives input variables; for instance, $(x_2, x_5) \subset x$—i.e., the state function of the unit is a partial function in a finite form of (7.8):

$$s_n = w_{n0} + w_{n1} x_2 + w_{n2}, x_5, \tag{7.9}$$

where w are the connection weights to the unit n. If there are ml input variables and two of them are randomly fed at each unit, the network needs $C_{m1}^2 (= m1(m1 - 1)/2)$ units at first layer to generate such partial forms. If we denote y^p as the actual value and s_n^p as the estimated value of the output for the function being considered for pth observation, the output error is given by

$$e_n^p = s_n^p - y^p \qquad (p \epsilon N). \tag{7.10}$$

The total squared error at unit n is:

$$\varepsilon = \sum_{p \epsilon N} (e_n^p)^2. \tag{7.11}$$

This corresponds to the minimization of the averaged error ε in estimating the weights w. This is the least squares technique. The weights are computed using a specific training set at all units that are represented with different input arguments of $m1$. This is realized at each unit of the layered network structure.

Multilayered structure is a parallel bounded structure built up based on the connectionistic approach; information flows forward only. One of the important functions built into the structure is the ability to solve implicitly defined relational functionals. The units are determined as independent elements of the partial functionals; all values in the domain of the variables which satisfy the conditions expressed as equations are comprised of possible solutions [15], [29]. Each layer contains a group of units that are interconnected to the units in the next layer. The weights of the state functions generated at the units are estimated using a training set A which is a part of N. A threshold objective function is used to activate the units "on" or "off" in comparison with a testing set B which is another part of N. The unit outputs are fed forward as inputs to the next layer; i.e., the output of nth unit if it is in the domain of local threshold measure would become input to some other units in the next level. The process continues layer after layer. The estimated weights of the connected units are memorized in the local memory. A global minimum of the objective function would be achieved in a particular layer; this is guaranteed because of steepest descent in the output error with respect to the connection weights in the solution space, in which it is searched according to a specific objective by cross-validating the weights.

2.2 Adaline

Adaline is a single element structure with the threshold logic unit and variable connection strengths. It computes a weighted sum of activities of the inputs times the synaptic weights, including a bias element. It takes +1 or —1 as inputs. If the sum of the state function is greater than zero, output becomes +1, and if it is equal to or less than zero, output is —1; this is the threshold linear function. Recent literature reveals the use of sigmoid functions in these networks [98]. The complexity of the network is increased by adding the number of adalines, called "madaline," in parallel. For simplicity, the functions of the adaline are described here.

Function at Single Element

Let us consider adaline with m input units, whose output is designated by y and with external inputs $x_k(k - 1, \cdots, m)$. Denote the corresponding weights in the interconnections by w_k. Output is given by a general formula in the form of a summation function:

$$s = w_0 + \sum_k w_k x_k, \tag{7.12}$$

where w_0 is a bias term and the activation level of the unit output is

$$\mathcal{S} = f(s). \tag{7.13}$$

Given a specific input pattern x^p and the corresponding desired value of the output y^p, the output error is given by

$$e^p = s^p - y^p \qquad (p \epsilon N), \tag{7.14}$$

where N indicates the sample size. The total squared error on the sample is

$$\varepsilon = \sum_{p \epsilon N} (e^p)^2. \tag{7.15}$$

The problem corresponds to minimizing the averaged error ε for obtaining the optimum weights. This is computed for a specific sample of training set. This is realized in the iterative least mean-square (LMS) algorithm.

LMS algorithm or Widrow-Hopf delta rule

At each iteration the weight vector is updated as

$$w^{p+1} = w^p + \frac{\alpha}{|x^p|^2} e^p x^p, \tag{7.16}$$

where w^{p+1} is the next value of the weight vector; w^p is the present value of the weight vector; x^p is present pattern vector; e^p is the present error according to Equation (7.14) and $|x^p|^2$ equals the number of weights.

pth iteration:

$$\begin{aligned} e^p &= y^p - x^{p^T} w^p \\ \delta e^p &= \delta(y^p - x^{p^T} w^p) = -x^{p^T} \delta w^p, \end{aligned} \tag{7.17}$$

where T indicates transpose. From Equation (7.16) we can write

$$\delta w^p = w^{p+1} - w^p = \frac{\alpha}{|x^p|^2} e^p x^p. \tag{7.18}$$

This can be substituted in Equation (7.17) to deduce the following:

$$\begin{aligned} \delta e^p &= -x^{p^T} \frac{\alpha}{|x^p|^2} e^p x^p \\ &= -x^{p^T} x^p \frac{\alpha}{|x^p|^2} e^p \\ &= -\alpha e^p. \end{aligned} \tag{7.19}$$

The error is reduced by a factor of a as the weights are changed while holding the input pattern fixed. Adding a new input pattern starts the next adapt cycle. The next error is reduced by a factor a, and the process continues. The choice of a controls stability and speed of convergence. Stability requires that $2 < \alpha < 0$. A practical range for a is given as $1.0 > \alpha > 0.1$.

2.3 Back Propogation

Suppose we want to store a set of pattern vectors $x^p, p = 1, 2, \cdots, N$ by choosing the weights w in such a way that when we present the network with a new pattern vector x^i it will respond by producing one of the stored patterns which it resembles most closely. The general nature of the task of the feed-forward network is to make a set of associations of the input patterns x_k^p with the output patterns y_l^p. When the input layer units are put in the configuration x_k^p the output units should produce the corresponding y_l^p. S_i are denoted as activations of output units based on the threshold sigmoid function and z_j^p are those of the intermediate or hidden layer units.

(i) For a *2-layer net,* unit output is given by:

$$\mathcal{S}_i^p = f(\sum_k w_{ik} x_k^p); \tag{7.20}$$

(ii) For a *3-layer net:*

$$\mathcal{S}_i^p = f(\sum_j w_{ij} z_j^p) = f(\sum_j w_{ij} f(\sum_k w_{jk} f_{j.k}^p)). \tag{7.21}$$

In either case the connection weights w's are chosen so that $\mathcal{S}_i^p = y_i^p$. This corresponds to the gradient minimization of the average of ε (7.22) for estimating the weights. The computational power of such a network depends on how many layers of units it has. If it has only two, it is quite limited; the reason is that it must discriminate solely on the basis of the linear combination of its inputs [95].

Learning by Evaluating Delta Rule

A way to iteratively compute the weights is based on gradually changing them so that the total squared-error decreases at each step:

$$\varepsilon = \frac{1}{2} \sum_{i,p} (\mathcal{S}_i^p - y_i^p)^2. \tag{7.22}$$

This can be guaranteed by making the change in w proportional to the negative gradient ε with respect to w (sliding down hill in w space on the error surface ε).

$$\delta w_{ij} = -\alpha \frac{\partial \varepsilon}{\partial w_{ij}}, \tag{7.23}$$

where α is a learning rate constant of proportionality. This implies a gradient descent of the total error ε for the entire set p. This can be computed from Equations (7.20) or (7.21).

For a *2-layer net:*

$$\begin{aligned} \delta w_{ik} &= -\alpha \frac{\partial \varepsilon}{\partial w_{ik}} = -\alpha (\sum_{i,p} \frac{\partial \varepsilon}{\partial \mathcal{S}_i} \frac{d\mathcal{S}_i}{dx_i} \frac{\partial x_i}{\partial w_{ik}}) \\ &= \alpha \sum_{i,p} [y_i^p - f(s_i^p)] f'(s_i^p) x_k^p \equiv \alpha \sum_{i,p} \delta_i^p x_k^p, \end{aligned} \tag{7.24}$$

where $s_i^p = \sum_k w_{ik} x_k^p$ is the state function and $f'()$ is the derivative of the activation function $f()$ at the output unit i. This is called a generalized delta rule.

For a *3-layer net:* input patterns are replaced by z_j^p of the intermediate units.

$$\delta w_{ij} = \alpha \sum_p \delta_i^p z_j^p. \tag{7.25}$$

By using the chain rule the derivative of (7.21) is evaluated:

$$\delta w_{jk} = \alpha \sum_{i,p} \delta_i^p w_{ij} f'(s_j^p) x_k^p \equiv \sum_p \delta_j^p x_k^p. \tag{7.26}$$

This can be generalized to more layers. All the changes are simply expressed in terms of the auxiliaryquantities $\delta_i^p, \delta_j^p, \cdots$ and the δ's for one layer are computed by simple recursions from those of the subsequent layer. This provides a training algorithm where the responses are fed forward and the errors are propagated back to compute the weight changes of layers from the output to the previous layers.

2.4 Self-organization boolean logic

In the context of principle of self-organization, it is interesting to look at a network of boolean operators (gates) which performs a task via learning by example scheme based on the work of Patarnello and Carnevali [99].

The *general problem of modeling* the boolean operator network is formulated as below. The system is considered for a boolean function like addition between two binary operands, each of L bits, which gives a result of the same length. It is provided with a number of examples of input values and the actual results. The system organizes its connections in order to minimize the mean-squared error on these examples between the actual and network results. Global optimization is achieved using simulated annealing based on the methods of statistical mechanics.

The *overall system* is formalized as follows. The network is configured by N_G gates and connections, where each gate has two inputs, an arbitrary number of outputs, and realizes one of the 16 possible boolean functions of two variables. The array Λ_i $(i = 1, 2, \cdots, N_G)$ with integer values between 1 and 16 indicates the operation implemented by ith gate. The experiments performed are chosen to organize the network in such a way that a gate can take input either from the input bits or from one of the preceding gates (the feedback is not allowed in the circuit). This means that $X_{i,j}^{(l,r)} = 0$ when $i > j$. The incidence matrices $X_{i,j}^{(l)}$ and $X_{i,j}^{(r)}$ represent the connections whose elements are zero except when gate j takes its left input from output gate i; then $X_{i,j}^{(l)} = 1$ and $X_{i,j}^{(r)} = 1$ is for right input. The output bits are connected randomly to any gate in the network.

The *training* is performed by identifying and correcting, for each example, a small subset of network connections which are considered responsible for the error. The problem is treated as a global optimization problem, without assigning adhoc rules to back propagate corrections on some nodes. The optimization is performed as a Monte Carlo procedure toward zero temperature (simulated annealing), where the energy or "cost" function ε of the system is the difference between the actual result and the calculated circuit output, averaged over the number of examples N_A fed to the system (chosen randomly at the beginning and kept fixed during the annealing).

$$\varepsilon(\Lambda, X) = \sum_{l=1}^{L} \varepsilon_l = \sum_{l=1}^{L} \frac{1}{N_A} \sum_{k=1}^{N_A} (q_{lk} - \hat{q}_{lk})^2, \tag{7.27}$$

where q_{lk} is the actual result of the lth bit in the kth example, $\hat{q}_{lk}$ $(= f(\Lambda, X))$ is the estimated output of the circuit. Thus, ε is the average number of wrong bits for the examples used in the training for a random network of $\varepsilon_l \sim 1/2$.

The search for the optimal circuit is done over the possible choice for X by choosing A randomly at the beginning and keeping it fixed during the annealing procedure and performing the average. The optimization procedure proceeds to change the input connection of a gate according to the resulting energy change $\Delta\varepsilon$. If $\Delta\varepsilon < 0$, the change is accepted; otherwise, it is accepted with the probability $\exp(-\Delta\varepsilon/T)$, where T is the temperature—a control parameter which is slowly decreased to zero according to some suitable "annealing schedule." The "partition" function for the problem is considered as

$$Z = \sum_{(X)} \exp(-\varepsilon/T). \tag{7.28}$$

The *testing part* of the system is straight forward; given the optimal circuit obtained after the training procedure, its correctness is tested by evaluating the average error over the exhaustive set of the operations, in the specific case all possible additions of $2L$-bit integers, of which there are $N_B = 2^L.2^L$ $(= 2^{2L})$.

$$\Delta(B) = \sum_{l=1}^{L} \Delta_l(B) = \sum_{l=1}^{L} \frac{1}{N_B} \sum_{k=1}^{N_B} (q_{lk} - \hat{q}_{lk})^2, \tag{7.29}$$

where the quantities q_{lk} and $\hat{q}_{lk}$ are the same as those in the above formula.

The performance of the boolean network is understood from the quantities ε and $\Delta(B)$; the low values of the ε mean that the system is trained very well and the small values of $\Delta(B)$ mean that the system is able to generalize properly. So, usually one expects the existence of two regimes (discrimination and generalization) between which possibly a state of "confusion" takes place.

Experiments are shown [100] for different values N_G and N_A with $L = 8$. It is found that a typical learning procedure requires an annealing schedule with approximately 3.10^6 Monte Carlo steps per temperature, with temperature ranging from $T \sim O(1)$ down to $T \sim O(10^{-6})$ (roughly 70 temperatures for a total of $\sim$ 200 million steps). The schedule was slow enough to obtain correct results when N_G is large, and is redundantly long when N_G is small. The system achieved zero errors ($\Delta(B) = 0$ as well as $\varepsilon = 0$; i.e., it finds a rule for the addition) in some cases considered ($N_G = 160, N_A = 224$ or 480). In these cases, as not all possible two-input operators process information, one can consider the number of "effective" circuits, which turn out to be approximately 40.

According to the annealing schedule, reaching $T \sim 0$ implies that learning takes place as an ordering phenomenon. The studies conducted on small systems are promising. Knowing Z exactly, the thermodynamics of these systems are analyzed using the "specific heat," which is defined as

$$C_v = \frac{\partial \varepsilon}{\partial T}. \tag{7.30}$$

The "specific heat" C_v is a response function of the system and a differential quantity that indicates the amount of heat a system releases when the temperature is infinitesimally lowered. The interesting features of these studies are given below:

- for each problem there is a characteristic temperature such that C_v has a maximum value;
- the harder problem, the lower its characteristic temperature; and
- the sharpness of the maximum indicates the difficulty of the problem, and in very hard problems, the peak remains one of the singularities in large critical systems.

In these networks, the complexity of a given problem for generalization is architecture-dependent and can be measured by how many networks solve that problem from the trained circuits with a reasonably high probability. The occurrence of generalization and learning of a problem is an entropic effect and is directly related to the implementation of many different networks.

3 GENERALIZATION

Studies have shown that any unbounded network could be replaced by a bounded network according to the capacities and energy dissipations in their architectures [18]. Here two types of bounded network structures are considered.

One of the important functions built into the feedforward structure is the ability to solve implicitly defined relational functionals—the units of which are determined as independent elements of the partial functionals. All values in the domain of the variables that satisfy the conditions, expressed as equations are comprised of possible solutions.

3.1 Bounded with transformations

Let us assume that unit k receives variables. For instance, $(x_k, x_{k+1}) \subset X$; that is, the state function of the unit is a partial function in a finite form of (7.8):

$$s_k = w_{k0} + w_{k1}x_k + w_{k2}x_{k+1} = f(x_k, x_{k+1}), \tag{7.31}$$

where w are the connection weights to the unit k. There are n input variables and two of them are consecutively fed at each unit. There are n units at each layer. If we denote y^p as the actual value and s_k^p as the estimated value of the output for the function being considered for the pth observation, the output error is given by

$$e_k^p = s_k^p - y^p \qquad (p \epsilon O). \tag{7.32}$$

The total squared-error at unit k is:

$$\varepsilon^2 = \sum_{p \epsilon O} (e_k^p)^2. \tag{7.33}$$

This corresponds to the minimization of the averaged error ε in estimating the weights w. The output s_k is activated by a transfer function such as a sigmoid function $F(\ )$:

$$x_k' = F(s_k), \tag{7.34}$$

where x_k' is the activated output fed forward as an input to the next layer.

The schematic functional flow of the structure can be given as follows. Let us assume that there are n input variables of x including nonlinear terms fed in pairs at each unit of the first layer (Figure 7.2). There are n units at each layer. The state functions at the first layer are:

$$\begin{aligned} s_j &= w_{j0} + w_{j1}x_j + w_{j2}x_{j+1} && 1 \le j < n \\ &= w_{j0} + w_{j1}x_j + w_{j2}x_1 && j = n. \end{aligned} \tag{7.35}$$

These are formed in a fixed order of cyclic rotation. The outputs $s_j, (j = 1, 2, \cdots, n)$ are activated by a sigmoid function and fed forward to the second layer:

$$\begin{aligned} s_j' &= w_{j0}' + w_{j1}'x_j' + w_{j2}'x_{j+1}' && 1 \le j < n \\ &= w_{j0}' + w_{j1}'x_j' + w_{j2}'x_1' && j = n, \end{aligned} \tag{7.36}$$

where $x_j' = F(s_j), (j = 1, 2, \cdots, n)$ are the activated outputs of first layer and s_j' are the outputs of the second layer. The process is repeated at the third layer:

$$\begin{aligned} s_j'' &= w_{j0}'' + w_{j1}''x_j'' + w_{j2}''x_{j+1}'' && 1 \le j < n \\ &= w_{j0}'' + w_{j1}''x_j'' + w_{j2}''x_1'' && j = n, \end{aligned} \tag{7.37}$$

where $x_j'' = F(s_j'), (j = 1, 2, \cdots, n)$ are the activated outputs of the second layer fed forward to the third layer; s_j'' are the outputs; and x_j''' are the activated outputs of the third layer. The process goes on repetitively as the complexity of the state function increases as given

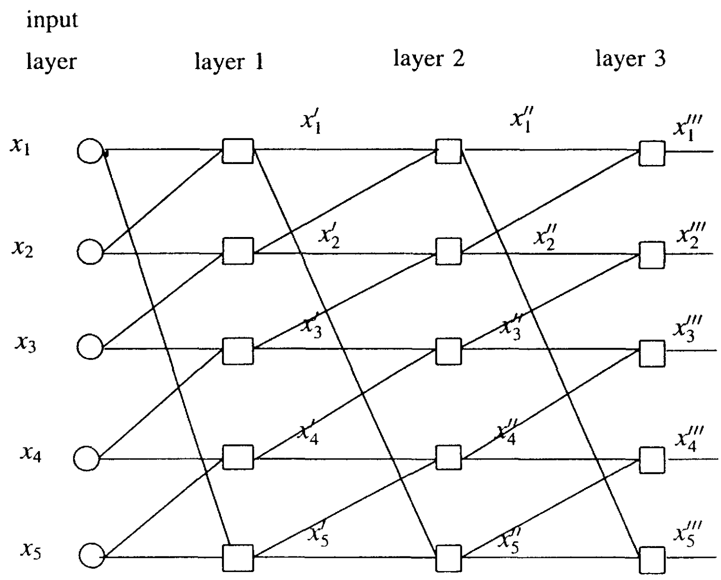

Figure 7.2. Bounded network structure with five input terms using a sigmoid function

below. For example, the state function at the unit k of the third layer with the activated output of x'''_k is described as:

$$\begin{aligned}
x'''_k &= F(w''_{k0} + w''_{k1}x''_k + w''_{k2}x''_{k+1}) = F(f(x''_k, x''_{k+1})) \\
&= F(s''_k) \\
&= F(F(s'_k), F(s'_{k+1})) \\
&= F(F(F(s_k), F(s_{k+1})), F(F(s_{k+1}), F(s_{k+2}))) \qquad (7.38) \\
&= F(F(F(f(x_k, x_{k+1})), F(f(x_{k+1}, x_{k+2}))), F(F(f(x_{k+1}, x_{k+2})), F(f(x_{k+2}, x_{k+3})))),
\end{aligned}$$

where s_k, s_{k+1}, s_{k+2} are the unit outputs at the first layer evaluated from the input variables of $(x_k, x_{k+1}, x_{k+2}, x_{k+3}) \subset X$. The optimal response according to the transformations is obtained through the connecting weights and is measured by using the standard average residual sum of squared error. This converges because of the gradient descent of the error by least-squares minimization and reduction in the energy dissipations of the network that is achieved by nonlinear mapping of the unit outputs through the threshold function, such as the sigmoid function.

3.2 Bounded with objective functions

Let us assume that unit j at the first layer receives variables. For instance, $(x_2, x_5) \subset X$; i.e., the state function of the unit is a partial function in a finite form of (7.8):

$$s_j = w_{j0} + w_{j1}x_2 + w_{j2}x_5 = f(x_2, x_5), \qquad (7.39)$$

where w are the connection weights to the unit j. If there are ml input variables and two of them are randomly fed at each unit, the network needs $C^2_{m1}(= m1(m1-1)/2)$ units at the first layer to generate such partial forms. If we denote y^p as the actual value and s^p_j as the estimated value of the output for the function being considered for pth observation, the output error is given by (7.28). The total squared error at unit j is computed as in (7.29).

This corresponds to the minimization of the averaged error ε in estimating the weights w. Each layer contains a group of units, which are interconnected to the units in the next layer. The weights of the state functions generated at the units are estimated using a training set A which is a part of N. An objective function as a threshold is used to activate the units "on" or "off" in comparison with a testing set B which is another part of N. The unit outputs are fed forward as inputs to the next layer; i.e., the output of jth unit—in the domain of local threshold measure—would become input to some other units in the next level. The process continues layer after layer. The estimated weights of the connected units are memorized in the local memory. A global minimum of the objective function would be achieved in a particular layer; this is guaranteed because of steepest descent in the output error with respect to the connection weights in the solution space, in which it is searched according to a specific objective by cross-validating the weights.

The schematic functional flow of the structure can be described as follows. Let us assume that there are $m1$ input variables of x, including nonlinear terms fed in pairs randomly at each unit of the first layer. There are C^2_{m1} units in this layer that use the state functions of the form (7.35):

$$\begin{aligned} x'_n &= f(x_i, x_j) \\ &= w'_{n0} + w'_{n1}x_i + w'_{n2}x_j, \end{aligned} \tag{7.40}$$

where x'_n is the estimated output of unit n, $n = 1, 2, \cdots, C^2_{m1}$; $i, j = 1, 2, \cdots, m1$; $i \neq j$; and w' are the connecting weights. Outputs of $m2\,(\leq C^2_{m1})$ units are made "on" by the threshold function to pass on to the second layer as inputs. There are C^2_{m2} units in the second layer and state functions of the form (7.35) are considered:

$$\begin{aligned} x''_n &= f(x'_i, x'_j) \\ &= w''_{n0} + w''_{n1}x'_i + w''_{n2}x'_j, \end{aligned} \tag{7.41}$$

where x''_n is the estimated output, $n = 1, 2, \cdots, C^2_{m2}$; $i, j = 1, 2, \cdots, m2$; $i \neq j$; and w'' are the connecting weights. Outputs of $m3\,(\leq C^2_{m2})$ units are passed on to the third layer according to the threshold function. In the third layer C^2_{m3} units are used with the state functions of the form (7.35):

$$\begin{aligned} x'''_n &= f(x''_i, x''_j) \\ &= w'''_{n0} + w'''_{n1}x''_i + w'''_{n2}x''_j, \end{aligned} \tag{7.42}$$

where x'''_n is the estimated output, $n = 1, 2, \cdots, C^2_{m3}$; $i, j = 1, 2, \cdots, m3$; $i \neq j$; and w''' *are* the connecting weights. This provides an inductive learning algorithm which continues layer after layer and is stopped when one of the units achieves a global minimum on the objective measure. The state function of a unit in the third layer might be equivalent to the function of some original input variables of x:

$$\begin{aligned} x'''_n &= f(x''_i, x''_j) \\ &\equiv f(f(x'_g, x'_h), f(x'_k, x'_l)) \\ &\equiv f(f(f(x_p, x_q), f(x_p, x_r)), f(f(x_q, x_r), f(x_u, x_v))) \\ &\equiv f(x_p, x_q, x_r, x_u, x_v), \end{aligned} \tag{7.43}$$

where $(x''_i, x''_j) \subset X''$ and $(x'_g, x'_h, x'_k, x'_l) \subset X'$ are the estimated outputs from the second and first layers, respectively, and $(x_p, x_q, x_r, x_u, x_v) \subset X$ are from the input layer (Figure 7.3). A typical threshold objective function such as regularization is measured for its total squared

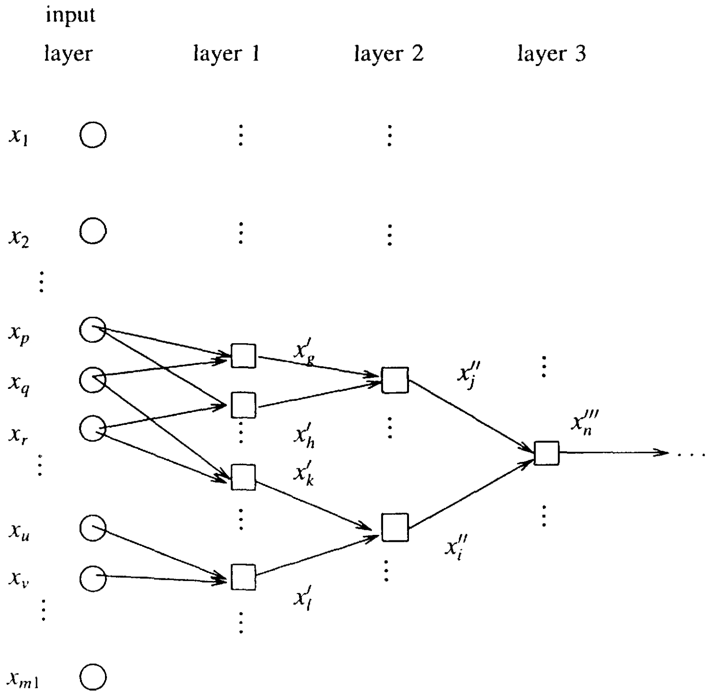

Figure 7.3. Functional flow to unit n of third layer in a multilayered inductive structure

error on testing set B as:

$$\Delta^2(B) = \sum_{s \in B} (x'''_n - y)^2_s, \tag{7.44}$$

where y is the actual output value and x'''_n is the estimated output of unit n of the third layer. The optimal response according to the objective function is obtained through the connecting weights w, which are memorized at the units in the preceding layers [90]. Figure 7.4 illustrates the multilayered feedforward network structure with five input variables and with the selections of five at each layer.

4 COMPARISON AND SIMULATION RESULTS

The major difference among the networks is that the inductive technique uses a bounded network structure with all combinations of input pairs as it is trained and tested by scanning the measure of threshold objective function through the optimal connection weights. This type of structure is directly useful for modeling multi-input single-output (MISO) systems, whereas adaline and backpropagation use an unbounded network structure to represent a model of the system as it is trained and tested through the unit transformations for its optimal connection weights. This type of structure is used for modeling multi-input multi-output (MIMO) systems.

Mechanisms shown in the generalized bounded network structures are easily worked out for any type of systems—MISO or MIMO. In adaline and backpropagation, input and

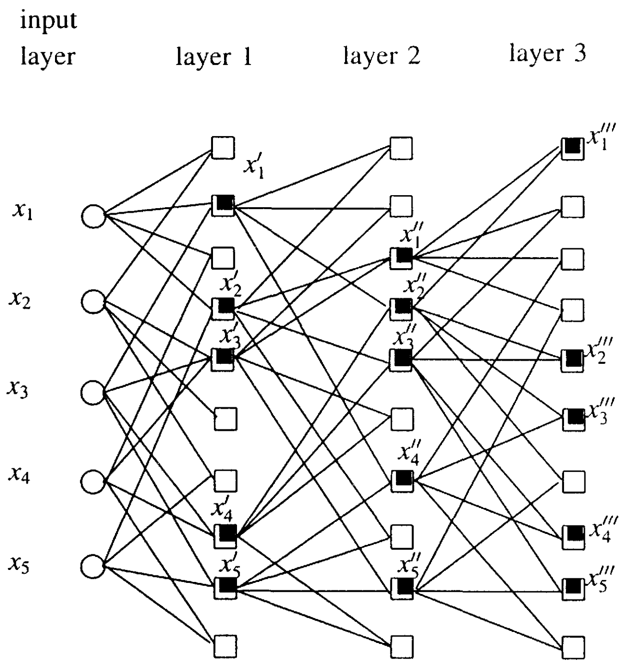

Figure 7.4. Feedforward multilayered inductive structure with $ml = 5, m2 = 5$, and $m3 = 5$ using threshold objective function

output data are considered either $\{-1, +1\}$ or $\{0, 1\}$. In the inductive approach, input and output data are in discrete analogue form, but one can normalize data between $\{-1, +1\}$ or $\{0, 1\}$. The relevance of local minima depends on the complexity of the task on which the system is trained. The learning adaptations considered in the generalized networks differ in two ways: the way they activate and forward the unit outputs. In backpropagation the unit outputs are transformed and fed forward. The errors at the output layer are propagated back to compute the weight changes in the layers and in the inductive algorithm the outputs are fed forward based on a decision from the threshold function. The backpropagation handles the problem that gradient descent requires infinitesimally small steps to evaluate the output error and manages with one or two hidden layers. The adaline uses the LMS algorithm with its sample size in minimizing the error measure, whereas in the inductive algorithm it is done by using the least squares technique. The parameters within each unit of inductive network are estimated to minimize, on a training set of observations, the sum of squared errors of the fit of the unit to the final desired output.

The batchwise procedure of least squares technique sweeps through all the points of the measured data accumulating $\partial\varepsilon/\partial w$ before changing the weights. It is guaranteed to move in the direction of steepest descent. The online procedure updates the weights for each measured data point separately [131]. Sometimes this increases the total error ε, but by making the weight changes sufficiently small the total change in the weights after a complete sweep through all the measured points can be made to closely and arbitrarily approximate the steepest descent. The use of batchwise procedure in the unbounded networks requires more computer memory, whereas in the bounded networks such as multilayered inductive networks, this problem does not arise.

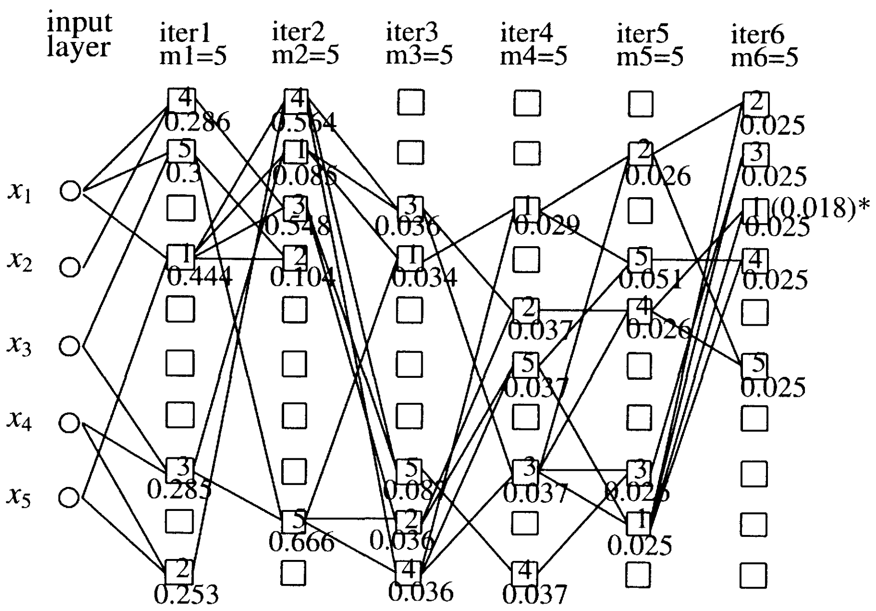

Figure 7.5. Bounded inductive network structure with linear inputs using threshold objective function (only activated links are shown)

Simulation experiments are conducted to compare the performances of inductive versus deductive networks by evaluating the output error as a learning law [91], [92]. Here the above general types of bounded network structures with inputs fed in pairs are considered. One is deductive network with sigmoid transfer function $\tanh(y * u_0)$, where u_0 is the gain factor and another is inductive network with threshold objective function which is a combined criterion ($c2$) of regularity and minimum-bias. As a special case, sinusoidal transformations are used for deductive network in one of the studies. In both the structures, the complexity of state function is increased layer by layer. The batchwise procedure of least squares technique is used in estimating the weights. Various randomly generated data and actual emperical data in the discrete analogue form in the range $\{-1, +1\}$ are used in these experiments. The network structures are unique in that they obtain optimal weights in their performances. Two examples for linear and nonlinear cases and another example on deductive network without any activations are discussed below:

(i) In linear case, the output data is generated from the equation:

$$y = 0.433 - 0.195x_1 + 0.243x_2 + 0.015x_3 - 0.18x_4 + \epsilon, \tag{7.45}$$

where $x_1, \cdots, x_4$ are randomly generated input variables, y is the output variable, and ϵ is the noise added to the data.

(a) Five input variables $(x_1, x_2, \cdots, x_5)$ are fed to the inductive network through the input layer. The global measure is obtained at a unit in the sixth layer ($c2$ = 0.0247). The mean-square error of the unit is computed as 0.0183. Figure 7.5 shows the iterations of the self-organization network (not all links are shown for clarity). The values of $c2$ are given at each node.

(b) The same input and output data are used for the deductive network; unit outputs are activated by sigmoid function. It converges to global minimum at a unit in the third layer. The residual mean-square error (MSE) of the unit is 0.101.

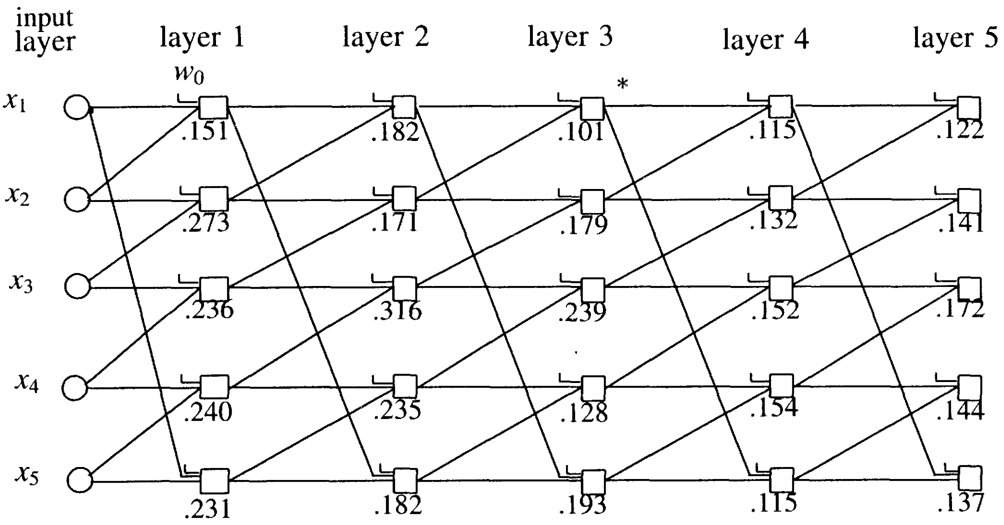

Figure 7.6. Bounded network structure with linear inputs and sigmoid output activations; w_0 is the biased term at each node

Figure 7.6 gives the evolutions of the generation of nodes by the network during the search process and residual MSE at each node is also given. "$*$" indicates the node which achieved the optimum value in all the networks given.

(ii) In a nonlinear case, the output data is generated from the equation:

$$y = 0.433 - 0.095x_1 + 0.243x_2 + 0.35x_1^2 - 0.18x_1x_2 + \epsilon, \tag{7.46}$$

where x_1, x_2 are randomly generated input variables, y is the output variable, and ϵ is the noise added to the data.

(a) x_1, x_2, x_1^2, x_2^2, x_1x_2 are fed as input variables. In the inductive case the global measure is obtained at a unit in the third layer ($c2 = 0.0453$). The residual MSE of the unit is computed as 0.0406. Figure 7.7 gives the combined measure of all units and residual MSE at the optimum node. Table 7.1 gives the connecting weight values (w_0, w_j, and w_i), the value of the combined criterion, and the residual MSE at each node.

(b) The same input/output data is used for the deductive network; sigmoid function is used for activating the outputs. It is converged to global minimum at a unit in the second layer. The average residual error of the unit is computed as 0.0223 for an optimum adjustment of $g = 1.8$. Figure 7.8 gives the residual MSE at each node. Table 7.2 gives the connecting weight values (w_0, w_j, and w_i), and the residual MSE at each node.

(c) In another case, the deductive network with the same input/output data is activated by the transfer function $F(u) = \sin(u * g)$, where u is the unit output and g is the gain factor. The global minimum is tested for different gain factors of g ($= 1 \pm \tau$), where τ varies from 0.0 to 1.0. As it varies, optimal units are shifted to earlier layers with a slight change of increase in the minimum. For example, at $\tau = 0.5$ the unit in the third layer achieves the minimum of 0.0188 and at $\tau = 0.8$ the unit in the second layer has the minimum of 0.0199. The global minimum of 0.0163 is achieved at the second unit of the sixth layer for $\tau = 0.0$ (Figure 7.9).

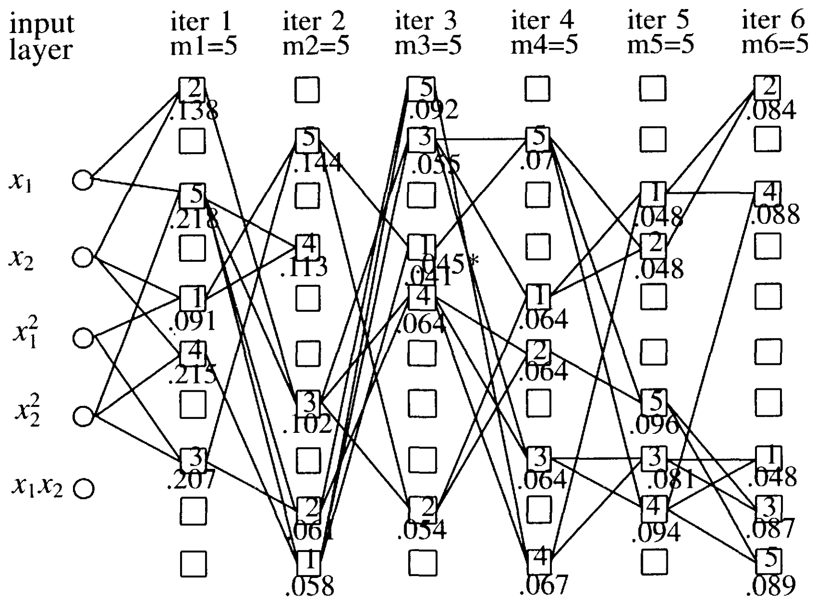

Figure 7.7. Bounded inductive network structure with nonlinear inputs using threshold objective function (only activated links are shown)

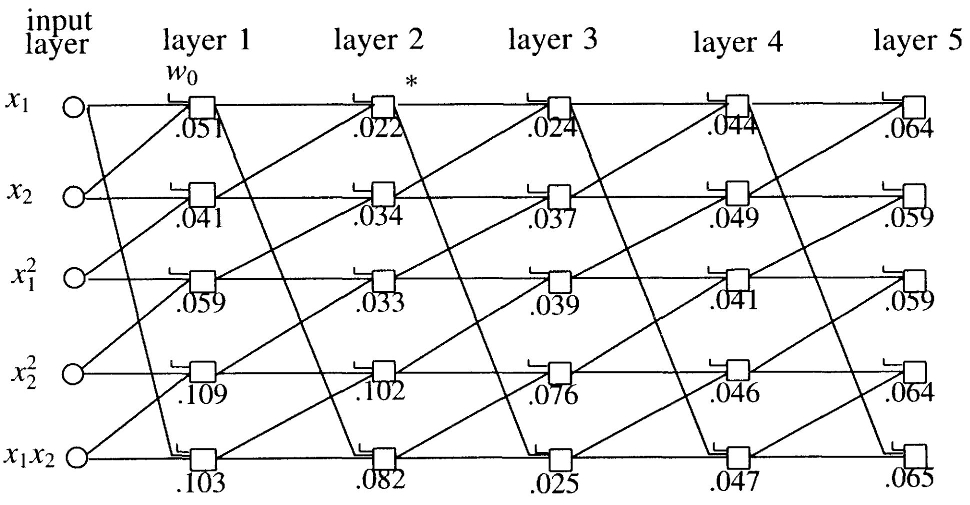

Figure 7.8. Bounded network structure with nonlinear inputs and sigmoid output activations; w_0 is the biased term at each node

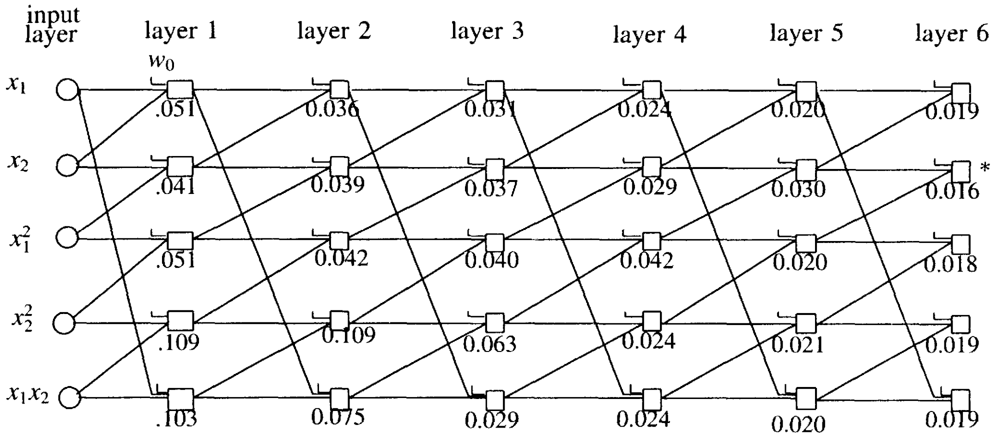

Figure 7.9. Bounded network structure with nonlinear inputs and sinusoidal output transformations; w_0 is the biased term at each node

(iii) Further, the network structures are tested for their performances without any threshold activations at the units; i.e., the unit outputs are directly fed forward to the next layer. Global minimum is not achieved; the residual error is reduced layer-by-layer as it proceeds—ultimately, the network becomes unstable. This shows the importance of the threshold functions in the convergence of these networks.

The resulting robustness in computations of self-organization modeling is one of the features that has made these networks attractive. It is clear that network models have a strong affinity with statistical mechanics. The main purpose of modeling is to obtain a better input-output transfer relationship between the patterns by minimizing the effect of noise in the input variables. This is possible only by providing more knowledge into the network structures; that is, improving the network performance and achieving better computing abilities in problem solving. In the inductive learning approach the threshold objective function plays an important role in providing more informative models for identifying and predicting complex systems. In the deductive case the unit output transformation through the sigmoid function plays an important role when the functional relationship is sigmoid rather than linear. Over all, one can see that the performance of the neural modeling can be improved by adding one's experience and knowledge into the network structure as a self-organization mechanism. It is an integration of various concepts from conventional computing and artificial intelligence techniques.

Table 7.1. Network structure with threshold objective function

```
LAYER=      1         (m1= 5)
            J= 1      I= 2
  .411        .186      .147;   c2=   .138E+00,   MSE=   .513E-01
            J= 1      I= 3
  .454        .145      .134;   c2=   .416E+00,   MSE=   .122E+00
            J= 1      I= 4
  .425        .213      .120;   c2=   .218E+00,   MSE=   .657E-01
            J= 1      I= 5
  .455        .069      .268;   c2=   .279E+00,   MSE=   .103E+00
            J= 2      I= 3
  .434        .155      .179;   c2=   .907E-01,   MSE=   .406E-01
```

```
             J= 2       I= 4
   .405        .629       -.376;  c2=  .215E+00,  MSE=  .137E+00
             J= 2       I= 5
   .458        .052        .284;  c2=  .226E+00,  MSE=  .997E-01
             J= 3       I= 4
   .452        .203        .133;  c2=  .207E+00,  MSE=  .589E-01
             J= 3       I= 5
   .465        .073        .260;  c2=  .266E+00,  MSE=  .102E+00
             J= 4       I= 5
   .466        .008        .329;  c2=  .257E+00,  MSE=  .109E+00
           ---------------------
 LAYER=      2         (m2= 5)
             J= 1       I= 2
   .024       1.097       -.151;  c2=  .154E+00,  MSE=  .523E-01
             J= 1       I= 3
   .033       2.313      -1.363;  c2=  .144E+00,  MSE=  .609E-01
             J= 1       I= 4
  -.033        .208        .822;  c2=  .295E+00,  MSE=  .527E-01
             J= 1       I= 5
   .004       -.451       1.423;  c2=  .113E+00,  MSE=  .412E-01
             J= 2       I= 3
  -.079        .186        .933;  c2=  .224E+00,  MSE=  .523E-01
             J= 2       I= 4
  -.054        .076        .989;  c2=  .165E+00,  MSE=  .536E-01
             J= 2       I= 5
   .020       -.099       1.045;  c2=  .102E+00,  MSE=  .443E-01
             J= 3       I= 4
  -.019       -.665       1.664;  c2=  .263E+00,  MSE=  .598E-01
             J= 3       I= 5
   .020       -.437       1.388;  c2=  .613E-01,  MSE=  .381E-01
             J= 4       I= 5
   .023       -.794       1.747;  c2=  .581E-01,  MSE=  .417E-01
           ---------------------
 LAYER=      3         (m3= 5)
             J= 1       I= 2
   .008       1.439       -.472;  c2=  .919E-01,  MSE=  .390E-01
             J= 1       I= 3
   .001        .098        .886;  c2=  .548E-01,  MSE=  .374E-01
             J= 1       I= 4
  -.008        .399        .596;  c2=  .119E+00,  MSE=  .399E-01
             J= 1       I= 5
   .008       4.123      -3.144;  c2=  .453E-01,  MSE=  .406E-01*
             J= 2       I= 3
   .000        .047        .939;  c2=  .642E-01,  MSE=  .374E-01
             J= 2       I= 4
  -.013        .146        .858;  c2=  .111E+00,  MSE=  .404E-01
             J= 2       I= 5
   .003       -.456       1.430;  c2=  .128E+00,  MSE=  .411E-01
             J= 3       I= 4
   .004       1.154       -.174;  c2=  .969E-01,  MSE=  .372E-01
             J= 3       I= 5
   .001        .929        .055;  c2=  .537E-01,  MSE=  .373E-01
             J= 4       I= 5
  -.009        .715        .281;  c2=  .105E+00,  MSE=  .406E-01
           ---------------------
 LAYER=      4         (m4= 5)
             J= 1       I= 2
   .004       -.390       1.372;  c2=  .896E-01,  MSE=  .353E-01
             J= 1       I= 3
   .004       -.400       1.385;  c2=  .699E-01,  MSE=  .353E-01
```

```
        J= 1      I= 4
 -.007    .713     .283;  c2=  .918E-01,  MSE=  .363E-01
        J= 1      I= 5
  .002   -.172    1.156;  c2=  .121E+00,  MSE=  .351E-01
        J= 2      I= 3
  .001    .001     .986;  c2=  .636E-01,  MSE=  .350E-01
        J= 2      I= 4
  .000    .867     .121;  c2=  .636E-01,  MSE=  .350E-01
        J= 2      I= 5
  .002   2.012   -1.025;  c2=  .819E-01,  MSE=  .351E-01
        J= 3      I= 4
  .001    .992    -.005;  c2=  .636E-01,  MSE=  .350E-01
        J= 3      I= 5
  .001   1.118    -.130;  c2=  .716E-01,  MSE=  .350E-01
        J= 4      I= 5
 -.002    .253     .738;  c2=  .669E-01,  MSE=  .351E-01
      ---------------------
LAYER=  5        (m5= 5)
        J= 1      I= 2
  .004   1.419    -.436;  c2=  .971E-01,  MSE=  .352E-01
        J= 1      I= 3
  .003   3.864   -2.879;  c2=  .105E+00,  MSE=  .354E-01
        J= 1      I= 4
  .001    .337     .649;  c2=  .484E-01,  MSE=  .350E-01
        J= 1      I= 5
  .001   -.137    1.123;  c2=  .484E-01,  MSE=  .350E-01
        J= 2      I= 3
  .004   -.585    1.567;  c2=  .113E+00,  MSE=  .351E-01
        J= 2      I= 4
  .004   -.438    1.421;  c2=  .983E-01,  MSE=  .352E-01
        J= 2      I= 5
  .004   -.446    1.429;  c2=  .964E-01,  MSE=  .352E-01
        J= 3      I= 4
  .003  -2.476    3.461;  c2=  .814E-01,  MSE=  .353E-01
        J= 3      I= 5
  .003  -2.602    3.587;  c2=  .935E-01,  MSE=  .353E-01
        J= 4      I= 5
  .001   -.172    1.158;  c2=  .340E+01,  MSE=  .350E-01
      ---------------------
LAYER=  6        (m6= 5)
        J= 1      I= 2
 -.004   -.141    1.132;  c2=  .836E-01,  MSE=  .364E-01
        J= 1      I= 3
  .004   -.555    1.539;  c2=  .899E-01,  MSE=  .353E-01
        J= 1      I= 4
  .004   -.557    1.542;  c2=  .883E-01,  MSE=  .353E-01
        J= 1      I= 5
  .003  -7.773    8.758;  c2=  .983E-01,  MSE=  .352E-01
        J= 2      I= 3
  .004   -.456    1.439;  c2=  .972E-01,  MSE=  .352E-01
        J= 2      I= 4
  .004   -.445    1.428;  c2=  .982E-01,  MSE=  .352E-01
        J= 2      I= 5
 -.004    .666     .323;  c2=  .895E-01,  MSE=  .363E-01
        J= 3      I= 4
  .001    .492     .494;  c2=  .483E-01,  MSE=  .350E-01
        J= 3      I= 5
  .004   1.659    -.675;  c2=  .870E-01,  MSE=  .353E-01
        J= 4      I= 5
  .004   1.677    -.693;  c2=  .888E-01,  MSE=  .354E-01
```

```
                  ---------------------------------------
```

Table 7.2. Network structure with sigmoid function

```
 LAYER=    1
           J= 1      I= 2
   .411      .186      .147;     MSE=  .513E-01
           J= 2      I= 3
   .434      .155      .179;     MSE=  .406E-01
           J= 3      I= 4
   .452      .203      .133;     MSE=  .589E-01
           J= 4      I= 5
   .466      .008      .329;     MSE=  .109E+00
           J= 5      I= 1
   .455      .268      .069;     MSE=  .103E+00
        ---------------------
 LAYER=    2
           J= 1      I= 2
  -.500    -1.100     2.489;     MSE=  .223E-01*
           J= 2      I= 3
  -.477     1.803     -.436;     MSE=  .336E-01
           J= 3      I= 4
  -.489     1.989     -.613;     MSE=  .328E-01
           J= 4      I= 5
  -.856      .115     1.709;     MSE=  .102E+00
           J= 5      I= 1
  -.757     1.304      .402;     MSE=  .824E-01
        ---------------------
 LAYER=    3
           J= 1      I= 2
  -.484     1.052      .329;     MSE=  .242E-01
           J= 2      I= 3
  -.456     1.464     -.117;     MSE=  .368E-01
           J= 3      I= 4
  -.614      .960      .577;     MSE=  .393E-01
           J= 4      I= 5
  -.722     1.497      .169;     MSE=  .764E-01
           J= 5      I= 1
  -.488      .158     1.229;     MSE=  .249E-01
        ---------------------
 LAYER=    4
           J= 1      I= 2
  -.458      .905      .454;     MSE=  .438E-01
           J= 2      I= 3
  -.441     1.641     -.304;     MSE=  .492E-01
           J= 3      I= 4
  -.502     1.757     -.349;     MSE=  .410E-01
           J= 4      I= 5
  -.467      .290     1.080;     MSE=  .456E-01
           J= 5      I= 1
  -.436    -4.405     5.736;     MSE=  .465E-01
        ---------------------
 LAYER=    5
           J= 1      I= 2
  -.437     1.088      .253;     MSE=  .643E-01
           J= 2      I= 3
  -.455      .954      .405;     MSE=  .591E-01
           J= 3      I= 4
  -.476      .560      .830;     MSE=  .586E-01
```

```
          J= 4      I= 5
  -.422   19.103  -17.783;    MSE=  .642E-01
          J= 5      I= 1
  -.426    -.459    1.786;    MSE=  .651E-01
        ---------------------
LAYER=    6
          J= 1      I= 2
  -.427    1.159     .175;    MSE=  .778E-01
          J= 2      I= 3
  -.428    -.111    1.444;    MSE=  .741E-01
          J= 3      I= 4
  -.441    1.755    -.406;    MSE=  .729E-01
          J= 4      I= 5
  -.413     .121    1.195;    MSE=  .796E-01
          J= 5      I= 1
  -.417    -.037    1.358;    MSE=  .791E-01
      ---------------------------------------
```