Complex Systems **5** (1991) 239–258

Comparison of Inductive Versus Deductive Learning Networks

H. Madala
Department of Mathematics and Computer Science, Clarkson University, Potsdam, NY 13699, USA

Abstract. This paper studies differences and similarities among inductive GMDH, deductive adaline, and back propagation techniques. All these are considered as parallel optimization algorithms because each one minimizes the output residual error in its own way. Self-organizing processes and criteria that help obtain 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, which depend on various activating laws and heuristics, is studied by explaining the functionalities of these algorithms. This study is helpful in understanding the inductive learning mechanism in comparison with the standard neural techniques, and in designing better and faster mechanisms for modeling and predictions of complex systems.

1. Introduction

The theory of neural networks began in 1943 with the work of McCulloch and Pitts [13], who considered the brain as a computer consisting of well-defined computing elements, the neurons. In 1958 Rosenblatt introduced the theoretical concept of the "perceptron" based on neural functioning [16]. There exist system theoretic approaches to brain functioning discussed in various disciplines like cybernetics, pattern recognition, artificial intelligence, biophysics, theoretical biology, mathematical psychology, and control system sciences. Neural networks have been adopted in problem solving studies related to various applied sciences and studies on the progress of computer hardware implementations for parallel distributed processing and structures of non-von Neuman design. One can find studies and developments on perceptron-based works as early as the 1960s in cybernetics, systems engineering, and other fields. A number of neural network structures, concepts, methods, and their applications have been well known in neural modeling literature for some time [1, 8, 11].

There has been 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 in them. The techniques differ as to the mechanisms adopted in the networks, and are distinguished for making successive adjustments in connection strengths until the network performs a desired computation with certain accuracy. The adaptive linear neuron element (adaline) was introduced by Widrow and Hopf in the early 1960s [19]. The least mean square (LMS) technique 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 the past decade [18]. It was widely used by the PDP group for solving various problems in multilayered feed-forward networks. Elsewhere, an induction mechanism called the group method of data handling (GMDH) was developed using randomly connected inputs, and connection strengths were established by minimizing the mean square error [3, 6, 9]. Ivakhnenko collected concepts from the perceptron theory and cybernetics [21], mixed these concepts with traditional system modeling techniques, and developed the GMDH technique for complex systems modeling during the 1960s. GMDH uses the principle of induction by generating different partial functionals, the principle of evolution by forwarding the processing flow from layer to layer, and the principle of natural selection by using the threshold objective function as a deciding function.

GMDH, adaline, and back propagation techniques are considered here because of their similarities as parallel optimization algorithms in minimizing the output residual error, and for their inductive and deductive approaches in dealing with the state functions. There is no study existing in the literature comparing inductive and deductive approaches, where the former is activated with threshold objective functions and the latter is with threshold linear or nonlinear functions. This paper considers generalization of learning laws and functionalities used in these approaches. "Generalization" is normally a helpful phenomenon; it allows us to deal effectively with the statistical mechanisms embedded in various existing situations. The term "generalization" is used here with the aim of studying the differences and the similarities between the approaches, and of studying their performances.

2. Neural approach and self organization

Rosenblatt described the perceptron as a probabilistic model. He pointed out that single layered networks could not solve the problem of pattern recognition [17], and that at least two stages are required: $X \to H$ transformation and $H \to Y$ transformation. He insisted that $X \to H$ transformation is realized by random links, but $H \to Y$ transformation is more deterministically realized by learned links. This corresponds to the a priori and conditional probabilistic links in the Bayes formula:

$$p(y_j) = \sum_{1}^{N} \left[p_0 \prod_{i=1}^{n} p(y_j/x_i) \right] \qquad j = 1, 2, \ldots, m \tag{2.1}$$

where p_0 is the a priori link corresponding to the $X \to H$ transformation, $p(y_j/x_i)$ are conditional links corresponding to the $H \to Y$ transformation, N is the sample size, and n and m are the number of vector components in X and Y, respectively. Consequently perceptron structures are of two types: probabilistic and algebraic or nonparametric and parametric. Here our concern is with parametric network structures. Connection weights among the $H \to Y$ links are established using adaptive techniques. The main emphasis is on an optimum adjustment of the weights in the links for achieving desired output. Neural nets have gradually become multilayered feed-forward network structures of information processing used to solve various problems.

We understand that information is passed on to the layered network through the input layer, and the result of the network's computation is read out at the output layer. The 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 put in the configuration, the association must be able to identify its output pattern. 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 [2, 4]. 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. An important characteristic of any neural network like adaline or back propagation is that output from each unit passes through a threshold logic unit (TLU). A standard TLU is a threshold linear function that is used for binary categorization of feature patterns. Nonlinear transfer functions such as sigmoid functions are used as a special case for the continuous output. When the output of a neuron is activated through the TLU, it mimics a biological neuron as "on" or "off." In networks like GMDH, the TLU uses a measure of an objective function to make the unit "on" or "off"; that is why this is called as threshold objective function. A state function is used to compute the capacity of the 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 states 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 that minimize the average squared error of the unit output

$$E = \frac{1}{N}\sum_i (\hat{y}_i - y_i)^2 \tag{2.2}$$

where $\hat{y}_i$ is the estimated output depending upon a relationship, y_i is the desired output, and N is the sample size. 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 [9, 10, 18, 19].

The statistical mechanism helps evaluate the units until the network performs a desired computation to obtain certain accuracy in response to the input signals. It enables the network to adapt itself to the examples of what it should be doing and to organize information within itself and thereby learn. The collective computation of the overall process of the self organization helps in obtaining the optimum output response.

3. Network algorithms

The focus here is on presentation of empirical analyzing capabilities of the networks GMDH, adaline, and back propagation in representing the input-output behavior of a system. Aspects considered are the basic functioning at the unit level based on these approaches, and the connectivity of units for recognition and prediction.

3.1 GMDH

Suppose we have a sample of N observations, a set of input-output pairs (I_1, o_1), $(I_2, o_2), \ldots$, $(I_N, o_N) \in O$, where O 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 $(1 \le j \le N)$ it is expected to reproduce the output o_j and to identify the physical laws, if any, embedded in the system. The prediction problem is that a given input I_{N+1} is expected to predict exactly the output o_{N+1} from a model of the domain it has learned during the training.

In GMDH, a general form of a summation function is considered as a Kolmogorov-Gabor polynomial that is in the discrete form of the Volterra functional series:

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

$$= a_0 + a_1x_1 + a_2x_2 + \cdots + a_{11}x_{m+1} + a_{12}x_{m+2} + \cdots + a_{mm}x_{m1} \tag{3.1}$$

where the estimated output is designated by $\hat{y}$, the external input vector x is designated by $(x_1, x_2, \ldots, x_{m1})$, and the a's are the weights or coefficients. This function is linear in parameters a and nonlinear in x. The nonlinear functions were first introduced by Widrow [20]. The input variables x could be independent variables, functional terms, or finite difference terms; that is, 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 is activated in parallel to build up the complexity.

3.2 Function at unit level

Let us assume that unit n receives input variables, for instance $(x_2, x_5) \subset x$; that is, the state function of the unit is a partial function in a finite form of (3.1):

$$s_n = w_{n0} + w_{n1}x_2 + w_{n2}x_5 \tag{3.2}$$

where the ws are the connection weights to the unit n. If there are $m1$ 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 let y^p be the actual value and s^p_n be the estimated value of the output for the function being considered for the pth observation, the output error is given by

$$e^p_n = s^p_n - y^p \qquad (p \in O) \tag{3.3}$$

The total squared error at unit n is

$$E = \sum_{p \in O} (e^p_n)^2 \tag{3.4}$$

This corresponds to the minimization of the averaged error E in estimating the weights w, which 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 GMDH structure.

GMDH *multilayer structure* is a parallel bounded structure built up based on the connectionist approach, and information flows forward only. One of the important functions built into the 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 comprise the possible solutions [5, 9]. 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 O_A, which is a part of O. A threshold objective function is used to activate the units "on" or "off" in comparison with a testing set O_B, which is another part of O. The unit outputs are fed forward as inputs to the next layer. The output of the nth unit in the domain of a local threshold measure would become an input to 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 per a specific objective by cross-validating the weights.

3.3 Learning by induction

The schematic functional flow of the multilayer structure can be described as follows. Let us assume that there are $m1$ input variables of x, including nonlinear terms fed in pairs at each unit of the first layer. There are C^2_{m1} units in this layer, which uses state functions of the form (3.2):

$$\begin{aligned} x'_n &= f(x_i, x_j) \\ &= w'_{n0} + w'_{n1}x_i + w'_{n2}x_j \end{aligned} \tag{3.5}$$

where x'_n is the estimated output of unit n for $n = 1, 2, \ldots, C^2_{m1}$; $i, j = 1, 2, \ldots, m1$; $i \neq j$; and w' are the connecting weights. Outputs of $m2$ ($\leq C^2_{m1}$) units are turned "on" by the threshold function and pass to the second layer as inputs. There are C^2_{m2} units in the second layer, and state functions of the form (3.2) are considered:

$$\begin{aligned} x''_n &= f(x'_i, x'_j) \\ &= w''_{n0} + w''_{n1}x'_i + w''_{n2}x'_j \end{aligned} \tag{3.6}$$

where x''_n is the estimated output, $n = 1, 2, \ldots, C^2_{m2}$; $i, j = 1, 2, \ldots, m2$; $i \neq j$; and w'' are the connecting weights. Outputs of $m3$ ($\leq C^2_{m2}$) units are passed to the third layer per the threshold function. In the third layer C^2_{m3} units are used with the state functions of the form (3.2):

$$\begin{aligned} x'''_n &= f(x''_i, x''_j) \\ &= w'''_{n0} + w'''_{n1}x''_i + w'''_{n2}x''_j \end{aligned} \tag{3.7}$$

where x'''_n is the estimated output, $n = 1, 2, \ldots, C^2_{m3}$; $i, j = 1, 2, \ldots, m3$; $i \neq j$; and w''' are the connecting weights. This provides an inductive learning algorithm that 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:

$$\begin{aligned} x'''_n &= f(x''_i, x''_j) \\ &\equiv f(f(x'_g, x'_h), f(x'_k, x'_l)) \\ &\equiv f(f(f(x_p, x_q), f(x_p, x_r)), f(f(x_q, x_r), f(x_u, x_v))) \\ &\equiv f(x_p, x_q, x_r, x_u, x_v) \end{aligned} \tag{3.8}$$

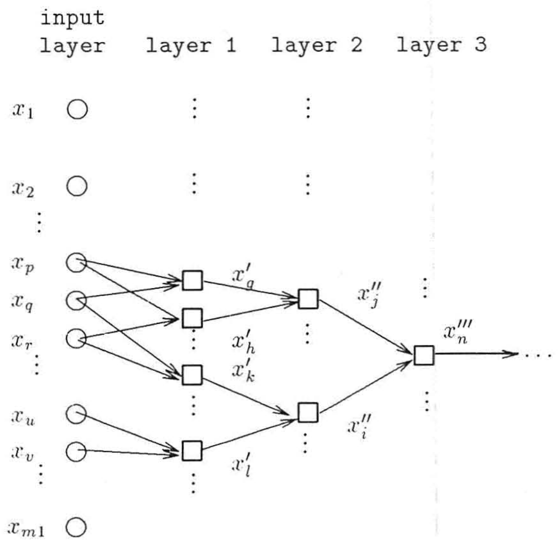

Figure 1: Functional flow to unit n of layer 3 in GMDH.

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 1). A typical threshold objective function such as regularization is measured for its total squared error on testing set O_B as

$$\Delta = \sum_{k \in O_B} (x'''^k_n - y^k)^2 \tag{3.9}$$

where y^k is the actual output value and x'''^k_n is the estimated output of unit n of the third layer for the kth observation. The optimal response per the objective function is obtained through the connecting weights w, which are memorized at the units in the preceding layers [12]. There are various forms of threshold objective functions in these networks based on the objectives like regularization, forecasting, finding physical laws, obtaining minimum biased models, or a combination of these, which might vary from problem to problem.

3.4 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 usage of sigmoid functions in these networks [15]. The complexity of the network is increased by adding the number of adalines in parallel, which is called as "madaline." For simplicity, the functionality of the adaline is described here.

3.5 Function at a single element

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

$$s = w_0 + \sum_k w_k \, x_k \tag{3.10}$$

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

$$\mathcal{S} = f(s) \tag{3.11}$$

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

$$e^p = s^p - y^p \qquad (p \in N) \tag{3.12}$$

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

$$E = \sum_{p \in N} (e^p)^2 \tag{3.13}$$

The problem corresponds to minimizing the average error E in obtaining the optimum weights. This is computed for a specific training set, and is realized in the iterative least mean square (LMS) algorithm.

3.6 LMS algorithm or Widrow-Hopf delta rule

At each iteration the weight vector w is updated as

$$w^{p+1} = w^p + \frac{\alpha}{|x^p|^2} e^p x^p \tag{3.14}$$

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 the present pattern vector; e^p is the present error per equation (3.12); and $|x^p|^2$ equals the number of weights.

The pth iteration is

$$\begin{aligned} e^p &= y^p - x^{p^T} w^p \\ \Delta e^p &= \Delta(y^p - x^{p^T} w^p) = -x^{p^T} \Delta w^p \end{aligned} \tag{3.15}$$

where T indicates transpose. From equation (3.14) we can write

$$\Delta w^p = w^{p+1} - w^p = \frac{\alpha}{|x^p|^2} e^p x^p \tag{3.16}$$

This can be substituted in equation (3.15) to deduce the following:

$$\begin{aligned}\Delta e^p &= -x^{p^T} \frac{\alpha}{|x^p|^2} e^p x^p \\ &= -x^{p^T} x^p \frac{\alpha}{|x^p|^2} e^p \\ &= -\alpha e^p \end{aligned} \tag{3.17}$$

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

3.7 Back propogation

Suppose we want to store a set of pattern vectors x^p, $p = 1, 2, \ldots, N$ 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 that it resembles most closely. The general nature of the task to be performed by 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. $\mathcal{S}_i$ denote activations of output units based on the threshold sigmoid function, and z_j^p are those of the intermediate or hidden layer units.

1. For a *2-layer net*, unit output is given by

$$\mathcal{S}_i^p = f(\sum_k w_{ik}\, x_k^p) \tag{3.18}$$

2. For a *3-layer net*,

$$\mathcal{S}_i^p = f(\sum_j w_{ij}\, z_j^p) = f(\sum_j w_{ij}\, f(\sum_k w_{jk}\, f_{j.k}^p)) \tag{3.19}$$

In either case the connection weights w are chosen so that $\mathcal{S}_i^p = y_i^p$. This corresponds to the gradient minimization of the average of E (see equation (3.20) below) for estimating the weights. The computational power of such a network depends on how many layers 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 [14].

3.8 Learning by evaluating the delta rule

One way to iteratively compute the weights is to change them little by little so the total squared error decreases at each step:

$$E = \frac{1}{2} \sum_{i,p} (\mathcal{S}_i^p - y_i^p)^2 \tag{3.20}$$

This can be guaranteed by taking the change in w proportional to the negative gradient E with respect to w (sliding downhill in w space on the surface E).

$$\delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{3.21}$$

where η is a learning rate constant of proportionality. This implies a gradient descent of the total error E for the entire set p. This can be computed from equation (3.18) or (3.19).

For a *2-layer net*,

$$\begin{aligned} \delta w_{ik} &= -\eta \frac{\partial E}{\partial w_{ik}} = -\eta \left(\sum_{i,p} \frac{\partial E}{\partial S_i} \frac{dS_i}{dx_i} \frac{\partial x_i}{\partial w_{ik}} \right) \\ &= \eta \sum_{i,p} [y_i^p - f(s_i^p)] \, f'(s_i^p) \, x_k^p \equiv \eta \sum_{i,p} \delta_i^p \, x_k^p \end{aligned} \tag{3.22}$$

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 the generalized delta rule.

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

$$\delta w_{ij} = \eta \sum_p \delta_i^p \, z_j^p \tag{3.23}$$

By using the chain rule the derivative of (3.19) is evaluated:

$$\delta w_{jk} = \eta \sum_{i,p} \delta_i^p \, w_{ij} \, f'(s_j^p) \, x_k^p \equiv \sum_p \delta_j^p \, x_k^p \tag{3.24}$$

This can be generalized to more layers. All the changes are simply expressed in terms of the auxiliary quantities $\delta_i^p, \delta_j^p, \ldots$, and the δs 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 of the previous layers.

4. Discussion

The major difference between the networks is that GMDH uses a bounded network structure with all combinations of input pairs as it is trained and tested by scanning the measure of the threshold objective function through the optimal connection weights. This type of structure is useful for modeling multi-input–single-output (MISO) systems. In contrast, adaline and back propagation 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. Studies have shown that any unbounded network can be replaced by a bounded network per the capacities and energy dissipations in their architectures [7]. Mechanisms in both cases are easily worked out for any type of system, MISO or MIMO. In adaline and back propagation, input and output data are considered either $\{-1,+1\}$ or $\{0,1\}$. In GMDH, input and output data are in discrete analog form, but one can normalize data between $\{-1,+1\}$ or $\{0,1\}$. Orthogonalization of the input data could be done in both cases as initial processing to reduce the noise. The relevance of local minima depends on the complexity of the task for which the system is trained.

The learning adaptations considered in these networks differ in two ways: how they activate and how they forward the unit outputs. In back propagation the unit outputs are transformed and fed forward, and the errors at the output layer are propagated back to compute the weight changes in the layers. In GMDH the outputs are fed forward based on a decision from the threshold function. Back propagation handles the problem that gradient descent requires infinitesimally small steps to evaluate the output error, and manages with one or two hidden layers. Adaline uses the LMS algorithm with its sample size to minimize the error measure, whereas GMDH uses the least-squares technique. The parameters within each unit of GMDH are estimated so that, on a training set of observations, the sum of the squared errors of the fit of the unit to the final desired output is minimized.

Simulation experiments are conducted to compare the performances of inductive versus deductive networks by evaluating the output error as a learning law. Here two general types of bounded network structures with inputs fed in pairs are considered: one is a deductive network (figure 2) with the sigmoid transfer function $\tanh(y * u_0)$ where u_0 is the gain factor, and another is an inductive network (figure 3) with the threshold objective function, which is a combined measure of regularization (3.9) and minimum-bias error. In both structures the complexity of the state function is increased layer-by-layer. The least-squares technique is used in estimating the weights. Various randomly generated data and actual empirical data in the discrete analog form in the range $\{-1,+1\}$ are used in these experiments. The network structures are unique in their performances in obtaining the optimal weights. Here examples for linear and nonlinear cases and for a deductive network without any activations are discussed.

1. In the linear case, the output data is generated from the equation

$$y = 0.433 - 0.195x_1 + 0.243x_2 + 0.015x_3 - 0.18x_4 + \epsilon \tag{4.1}$$

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

The inductive network with the threshold objective function is fed with five input variables $(x_1, x_2, \ldots, x_5)$. The global measure is obtained at a unit in the sixth layer (combined $= 0.0247$). The average residual error of the unit is computed as 0.0183.

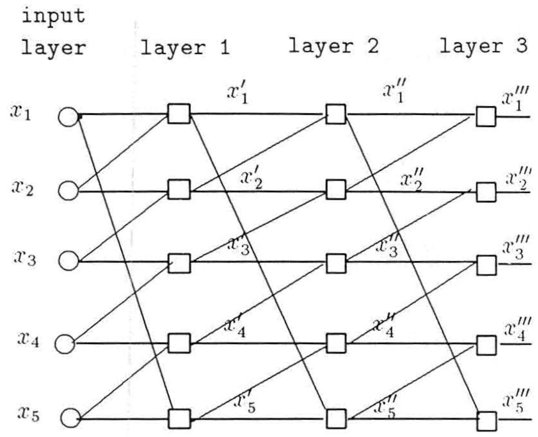

Figure 2: Bounded network structure with 5 input terms using a sigmoid function.

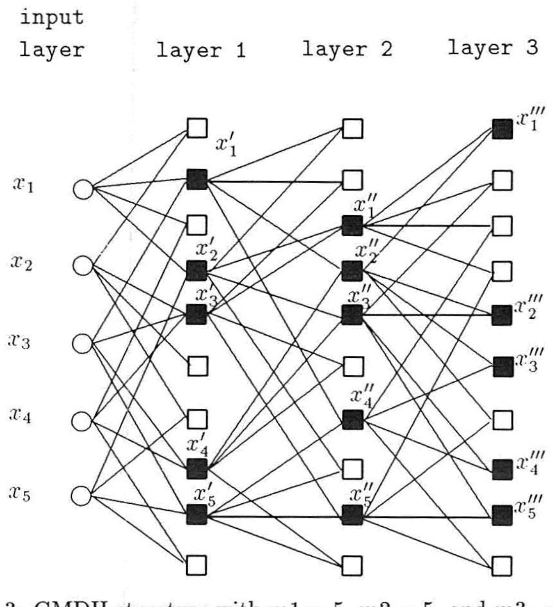

Figure 3: GMDH structure with $m1 = 5$, $m2 = 5$, and $m3 = 5$ using threshold objective function.

The deductive network with the sigmoid transfer function, which uses the same input and output data, converges to a global minimum at a unit in the third layer. The average residual error of the unit is 0.101.

2. In the nonlinear case, the output data is generated from the equation

$$y = 0.433 - 0.095x_1 + 0.243x_2 + 0.35x_1^2 - 0.18x_1x_2 + \epsilon \qquad (4.2)$$

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

In the inductive network, x_1, x_2, x_1^2, x_2^2, and x_1x_2 are fed as input variables. The global measure is obtained at a unit in the third layer (combined = 0.0453). The average residual error of the unit is computed as 0.0406. Table 1 gives the connecting weight values (w_0, w_j, and w_i), combined measure, and residual error at each node.

The deductive network, which uses the same input and output data, converges to a global minimum at a unit in the second layer. The average residual error of the unit is computed as 0.0223. Table 2 gives the connecting weight values (w_0, w_j, and w_i) and residual error at each node.

3. Further, the deductive network structure is tested for its performance without any threshold activations at the units; that is, the unit outputs are directly fed forward to the next layer. A global minimum is not achieved; the residual error is reduced layer-by-layer as it proceeds, and 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 the modeling is to obtain a better input-output transfer relation between the patterns by minimizing the effect of noise in the input variables. This is possible only by providing more knowledge to the network structures, thereby 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. Overall, one can see that performance of the neural modeling can be improved by providing one's experience and knowledge to the network structure as a self-organization mechanism. It is an integration of various concepts from conventional computing and artificial intelligence techniques.

ACKNOWLEDGMENTS

The author is very much thankful to Dr. A. S. Fokas for useful discussions and comments.

```
LAYER=   1        (m1= 5)
J= 1     I= 2
.411     .186      .147;  COMBINED=  .138E+00, RES ERR=  .513E-01
J= 1     I= 3
.454     .145      .134;  COMBINED=  .416E+00, RES ERR=  .122E+00
J= 1     I= 4
.425     .213      .120;  COMBINED=  .218E+00, RES ERR=  .657E-01
J= 1     I= 5
.455     .069      .268;  COMBINED=  .279E+00, RES ERR=  .103E+00
J= 2     I= 3
.434     .155      .179;  COMBINED=  .907E-01, RES ERR=  .406E-01
J= 2     I= 4
.405     .629     -.376;  COMBINED=  .215E+00, RES ERR=  .137E+00
J= 2     I= 5
.458     .052      .284;  COMBINED=  .226E+00, RES ERR=  .997E-01
J= 3     I= 4
.452     .203      .133;  COMBINED=  .207E+00, RES ERR=  .589E-01
J= 3     I= 5
.465     .073      .260;  COMBINED=  .266E+00, RES ERR=  .102E+00
J= 4     I= 5
.466     .008      .329;  COMBINED=  .257E+00, RES ERR=  .109E+00
---------------------
LAYER=   2        (m2= 5)
J= 1     I= 2
.024     1.097     -.151;  COMBINED=  .154E+00, RES ERR=  .523E-01
J= 1     I= 3
.033     2.313    -1.363;  COMBINED=  .144E+00, RES ERR=  .609E-01
J= 1     I= 4
-.033     .208      .822;  COMBINED=  .295E+00, RES ERR=  .527E-01
J= 1     I= 5
.004     -.451     1.423;  COMBINED=  .113E+00, RES ERR=  .412E-01
J= 2     I= 3
-.079     .186      .933;  COMBINED=  .224E+00, RES ERR=  .523E-01
J= 2     I= 4
-.054     .076      .989;  COMBINED=  .165E+00, RES ERR=  .536E-01
J= 2     I= 5
.020     -.099     1.045;  COMBINED=  .102E+00, RES ERR=  .443E-01
J= 3     I= 4
-.019    -.665      1.664;  COMBINED=  .263E+00, RES ERR=  .598E-01
J= 3     I= 5
.020     -.437     1.388;  COMBINED=  .613E-01, RES ERR=  .381E-01
J= 4     I= 5
.023     -.794     1.747;  COMBINED=  .581E-01, RES ERR=  .417E-01
```

Table 1: Network structure with threshold objective function.

```
LAYER=   3        (m3= 5)
J= 1     I= 2
.008    1.439    -.472;  COMBINED=  .919E-01, RES ERR=  .390E-01
J= 1     I= 3
.001     .098     .886;  COMBINED=  .548E-01, RES ERR=  .374E-01
J= 1     I= 4
-.008     .399     .596;  COMBINED=  .119E+00, RES ERR=  .399E-01
J= 1     I= 5
.008    4.123   -3.144;  COMBINED=  .453E-01, RES ERR=  .406E-01*
J= 2     I= 3
.000     .047     .939;  COMBINED=  .642E-01, RES ERR=  .374E-01
J= 2     I= 4
-.013     .146     .858;  COMBINED=  .111E+00, RES ERR=  .404E-01
J= 2     I= 5
.003    -.456    1.430;  COMBINED=  .128E+00, RES ERR=  .411E-01
J= 3     I= 4
.004    1.154    -.174;  COMBINED=  .969E-01, RES ERR=  .372E-01
J= 3     I= 5
.001     .929     .055;  COMBINED=  .537E-01, RES ERR=  .373E-01
J= 4     I= 5
-.009     .715     .281;  COMBINED=  .105E+00, RES ERR=  .406E-01
---------------------
LAYER=   4        (m4= 5)
J= 1     I= 2
.004    -.390    1.372;  COMBINED=  .896E-01, RES ERR=  .353E-01
J= 1     I= 3
.004    -.400    1.385;  COMBINED=  .699E-01, RES ERR=  .353E-01
J= 1     I= 4
-.007     .713     .283;  COMBINED=  .918E-01, RES ERR=  .363E-01
J= 1     I= 5
.002    -.172    1.156;  COMBINED=  .121E+00, RES ERR=  .351E-01
J= 2     I= 3
.001     .001     .986;  COMBINED=  .636E-01, RES ERR=  .350E-01
J= 2     I= 4
.000     .867     .121;  COMBINED=  .636E-01, RES ERR=  .350E-01
J= 2     I= 5
.002    2.012   -1.025;  COMBINED=  .819E-01, RES ERR=  .351E-01
J= 3     I= 4
.001     .992    -.005;  COMBINED=  .636E-01, RES ERR=  .350E-01
J= 3     I= 5
.001    1.118    -.130;  COMBINED=  .716E-01, RES ERR=  .350E-01
J= 4     I= 5
-.002     .253     .738;  COMBINED=  .669E-01, RES ERR=  .351E-01
```

Table 1: *Continued.*

```
LAYER=   5       (m5= 5)
J= 1      I= 2
.004     1.419    -.436;  COMBINED=  .971E-01, RES ERR=  .352E-01
J= 1      I= 3
.003     3.864   -2.879;  COMBINED=  .105E+00, RES ERR=  .354E-01
J= 1      I= 4
.001      .337     .649;  COMBINED=  .484E-01, RES ERR=  .350E-01
J= 1      I= 5
.001     -.137    1.123;  COMBINED=  .484E-01, RES ERR=  .350E-01
J= 2      I= 3
.004     -.585    1.567;  COMBINED=  .113E+00, RES ERR=  .351E-01
J= 2      I= 4
.004     -.438    1.421;  COMBINED=  .983E-01, RES ERR=  .352E-01
J= 2      I= 5
.004     -.446    1.429;  COMBINED=  .964E-01, RES ERR=  .352E-01
J= 3      I= 4
.003    -2.476    3.461;  COMBINED=  .814E-01, RES ERR=  .353E-01
J= 3      I= 5
.003    -2.602    3.587;  COMBINED=  .935E-01, RES ERR=  .353E-01
J= 4      I= 5
.001     -.172    1.158;  COMBINED=  .340E+01, RES ERR=  .350E-01
---------------------
LAYER=   6       (m6= 5)
J= 1      I= 2
-.004     -.141    1.132;  COMBINED=  .836E-01, RES ERR=  .364E-01
J= 1      I= 3
.004     -.555    1.539;  COMBINED=  .899E-01, RES ERR=  .353E-01
J= 1      I= 4
.004     -.557    1.542;  COMBINED=  .883E-01, RES ERR=  .353E-01
J= 1      I= 5
.003    -7.773    8.758;  COMBINED=  .983E-01, RES ERR=  .352E-01
J= 2      I= 3
.004     -.456    1.439;  COMBINED=  .972E-01, RES ERR=  .352E-01
J= 2      I= 4
.004     -.445    1.428;  COMBINED=  .982E-01, RES ERR=  .352E-01
J= 2      I= 5
-.004      .666     .323;  COMBINED=  .895E-01, RES ERR=  .363E-01
J= 3      I= 4
.001      .492     .494;  COMBINED=  .483E-01, RES ERR=  .350E-01
J= 3      I= 5
.004     1.659    -.675;  COMBINED=  .870E-01, RES ERR=  .353E-01
J= 4      I= 5
.004     1.677    -.693;  COMBINED=  .888E-01, RES ERR=  .354E-01
```

Table 1: *Continued.*

```
LAYER=   1
J= 1     I= 2
.411     .186      .147;    RES ERR=  .513E-01
J= 2     I= 3
.434     .155      .179;    RES ERR=  .406E-01
J= 3     I= 4
.452     .203      .133;    RES ERR=  .589E-01
J= 4     I= 5
.466     .008      .329;    RES ERR=  .109E+00
J= 5     I= 1
.455     .268      .069;    RES ERR=  .103E+00
---------------------
LAYER=   2
J= 1     I= 2
-.500   -1.100     2.489;    RES ERR=  .223E-01*
J= 2     I= 3
-.477    1.803    -.436;     RES ERR=  .336E-01
J= 3     I= 4
-.489    1.989    -.613;     RES ERR=  .328E-01
J= 4     I= 5
-.856     .115     1.709;    RES ERR=  .102E+00
J= 5     I= 1
-.757    1.304      .402;    RES ERR=  .824E-01
---------------------
LAYER=   3
J= 1     I= 2
-.484    1.052      .329;    RES ERR=  .242E-01
J= 2     I= 3
-.456    1.464    -.117;     RES ERR=  .368E-01
J= 3     I= 4
-.614     .960      .577;    RES ERR=  .393E-01
J= 4     I= 5
-.722    1.497      .169;    RES ERR=  .764E-01
J= 5     I= 1
-.488     .158     1.229;    RES ERR=  .249E-01
```

Table 2: Network structure with sigmoid function.

```
LAYER=   4
J= 1     I= 2
-.458     .905      .454;   RES ERR=  .438E-01
J= 2     I= 3
-.441    1.641     -.304;   RES ERR=  .492E-01
J= 3     I= 4
-.502    1.757     -.349;   RES ERR=  .410E-01
J= 4     I= 5
-.467     .290     1.080;   RES ERR=  .456E-01
J= 5     I= 1
-.436   -4.405     5.736;   RES ERR=  .465E-01
---------------------
LAYER=   5
J= 1     I= 2
-.437    1.088      .253;   RES ERR=  .643E-01
J= 2     I= 3
-.455     .954      .405;   RES ERR=  .591E-01
J= 3     I= 4
-.476     .560      .830;   RES ERR=  .586E-01
J= 4     I= 5
-.422   19.103   -17.783;   RES ERR=  .642E-01
J= 5     I= 1
-.426    -.459     1.786;   RES ERR=  .651E-01
---------------------
LAYER=   6
J= 1     I= 2
-.427    1.159      .175;   RES ERR=  .778E-01
J= 2     I= 3
-.428    -.111     1.444;   RES ERR=  .741E-01
J= 3     I= 4
-.441    1.755     -.406;   RES ERR=  .729E-01
J= 4     I= 5
-.413     .121     1.195;   RES ERR=  .796E-01
J= 5     I= 1
-.417    -.037     1.358;   RES ERR=  .791E-01
```

Table 2: *Continued.*

References

[1] I. Aleksander (editor), *Neural Computing Architectures: The Design of Brain-like Machines* (Cambridge, MIT press, 1989).

[2] M. A. Arbib and S. Amari (editors), *Dynamic Interactions in Neural Networks: Models and Data* (New York, Springer Verlag, 1989).

[3] R. L. Barron, "Adaptive Transformation Networks for Modeling, Predictions and Control," *IEEE SMC Group Annual Symposium Record* (1971), 254–263.

[4] E. Basar, H. Flohr, H. Haken, and J. Mandell (editors), *Synergetics of the Brain* (New York, Springer Verlag, 1983).

[5] G. M. Edelman, *Neural Darwinism: The Theory of Neural Group Selection* (Boston, Basic Books, 1987).

[6] S. J. Farlow (editor), *Self Organizing Methods in Modeling: GMDH Type Algorithms* (New York, Marcel Dekker, 1984).

[7] F. Fogelman Soulie, Y. Robert, and M. Tchuente (editors), *Automata Networks in Computer Science: Theory and Applications* (Princeton, Princeton University Press, 1987).

[8] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Sciences*, **79** (1982) 2554–2558.

[9] A. G. Ivakhnenko, "Polynomial Theory of Complex Systems," *IEEE Transactions on Systems, Man and Cybernetics*, **1**(4) (1971) 364–378.

[10] T. Kohonen, *Self Organization and Associative Memory*, second edition (New York, Springer Verlag, 1988), 312.

[11] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, April 1987, 4–22.

[12] H. Madala, "Layered Inductive Learning Algorithms and Their Computational Aspects," *Proceedings—IEEE Tools for Artificial Intelligence*, (TAI-89) (1989) 448–456.

[13] W. S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, **5** (1943) 115–133.

[14] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry* (Cambridge, MIT Press, 1969).

[15] D. H. Nguyen and B. Widrow, "Neural Networks for Self Learning Control Systems," *IEEE Control Systems Magazine*, April 1990, 18–23.

[16] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, **65**(6) (1958) 386–408.

[17] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms* (Spartan Books, 1962).

[18] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Volume 1, Foundations* (Cambridge, MIT Press, 1986).

[19] B. Widrow and M. E. Hoff, Jr., "Adaptive Switching Circuits," *Western Electronic Show and Convention Rec 4, Institute of Radio Engineers*, (1960) 96–104.

[20] B. Widrow, R. G. Winter, and R. A. Baxter, "Layered Neural Nets for Pattern Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **36**(7) (1988) 1109–1118.

[21] N. Wiener, *Cybernetics* (Cambridge, MIT Press, 1947).