6-1-1989

Self-Organizing Neural Network for Optimum Supervised Learning

Manoel Fernando Tenorio
Purdue University, tenorio@ee.ecn.purdue.edu.ARPA

Wei-Tsih Lee
Purdue University, lwt@ee.ecn.purdue.edu

Self Organizing Neural Network for Optimum Supervised Learning

Manoel Fernando Tenorio
Wei-Tsih Lee

TR-EE 89-30
June, 1989

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

Self-Organizing Neural Network for Optimum Supervised Learning

Manoel Fernando Tenorio
School of Electrical Engineering
Purdue University
W. Lafayette, IN. 47907
tenorio@ee.ecn.purdue.edu

Wei-Tsih Lee
School of Electrical Engineering
Purdue University
W. Lafayette, IN. 47907
lwt@ed.ecn.purdue.edu

ABSTRACT

This work introduces a new algorithm called the Self-Organizing Neural Network (SONN), and demonstrates its use in a system identification task. The algorithm constructs a network, chooses the neuron functions, and adjusts the weights. Here, it is compared to the Back-Propagation algorithm in the identification of the chaotic time series. The results show that SONN constructs a simpler, more accurate model, requiring less training data and epochs. The algorithm can also be applied as a classifier.

I. INTRODUCTION

I.1 THE SYSTEM IDENTIFICATION PROBLEM

In various engineering applications, it is important to be able to estimate, interpolate, and extrapolate the behavior of an unknown system when only its input-output measurement pairs are available. Algorithms which produce an estimation of the system behavior based on these pairs fall under the category of system identification techniques. These techniques can be divided into two major groups: *parameter estimation* and *functional estimation*. These two groups are not mutually exclusive, but can be described in a continuum between the two. The parameter estimation approach is relatively simple, if one can safely assume a known system function with only unknown parameters. This calls for a large body of a priori knowledge about the underlying process being observed, and so it is not very general. The second approach, functional estimation, deals with the estimation of the system function as well as its parameters, and it can be simplified if assumptions about the function can be made. The larger the number of assumptions, the more the parameter estimation process it resembles.

I.2 SYSTEM IDENTIFICATION USING NEURAL NETWORKS

Neurocomputing techniques can be applied to the system identification problem using adaptive algorithms for either parameter or functional estimation. When both input and output are observable and usable by the algorithm, the learning process is called supervised since the output signal works as a teacher signal. The Generalized Delta Rule (GDR), the most popular form of this type of learning process, has been used for system identification successfully [Lapedes and Farber, 1987a&b]. GDR falls somewhere between a pure parameter based approach, and a functional estimator where the order of the system complexity is known (number of hidden units). The system identification problem can appear in a diverse number of ways, varying in the types of relationships the input-output pairs can have within themselves and with each other. If no relationship is ascribed to subsequent pairs in time, the system can be thought of as performing a static classification over the input space. When a time structure is found in the input sequence with implications for the output sequence, the system is said to be acting on and producing time dependent sequences. Such systems can have a constant internal structure in time. This means that these systems would always produce the same output sequence for a given input sequence regardless of its previous history, and they would be called static or time independent. Some systems might also produce an output value based solely on past values of the input-output pairs. In these cases, the systems are said to be causal. This work describes a new supervised learning algorithm for self - organizing neuromorphic structures which minimizes the number of assumptions about the underlying process. Although it does not necessarily have to be so, the examples presented here are of static, causal systems, operating on time varying signals.

Several different applications of identification techniques can be found in fields as diverse as computer vision [Marroquin, 1985], and system adaptive modeling [Widrow, 1985]. According to [Zadeh, 1962], the system identification problems can be defined as: "the determination, on the basis of input and output, of a system with a specified class of systems, to which the system under test is equivalent." The three key components to system identification are: (1) the class of systems, (2) a class of signals, and (3) a criterion of equivalence. Åström and Eykhoff [1971] pointed out that the identification problem can be transformed into the optimization problem, if the criterion of equivalence is defined in terms of the error function.

A general form to represent systems, both linear and nonlinear, is the Kolmogorov-Garbor polynomial [Garbor et al., 1961] shown below:

$$y = a_0 + \sum_i a_i x_i + \sum_i \sum_j a_{ij} x_i x_j + \cdots \qquad (1)$$

where the y is the output, and x is the input to the system. Garbor [1961] proposed a learning method that adjusted the coefficients of (1) by minimizing the mean square error between each desired output sample and the actual output. Roy and Sherman [1967] suggested the concept of "hypersurfaces" as an alternative way to think about the identification problem. In this case, the identification of a system, especially of a nonlinear one, is viewed as the construction of an N+1 dimension hypersurface; if the highest order of the polynomial in (1) is N. Lapedes and Farber [1987a] also explored the hypersurface interpretation of the functional relationship in (1). Their work concentrated on the use of feedforward neural networks in which the neuron transfer function is the sigmoid, or the logistic function. The structure of the network is fixed, and the weights adjusted by the GDR algorithm. The elementary structure of the system is based on n neurons in the first layer connected to the neurons in the second, and third layers according the particular connectivity (figure 1). This substructure forms a hypersurface in n+1 dimension, which they call a "bump" [Lapedes and Farber, 1987a]. Combinations of such bumps are used to approximate the desired input-output relationship in (1). Here n is the dimensionality of the input, which in this case represents past samples of the input sequence x(t), which should completely characterize y(t).

The approaches described above suffer from the rapid explosion of the possible combination of terms as the order of the polynomial is increased. The number of samples also need to be much larger than the dimensionality of the hypersurface used, which for practical purposes can be difficult to achieve. They also require repeated presentation of the training data, or more preferably infinite sequences.

To address some of the difficulties described above, a heuristic algorithm called the Group Method of Data Handling (GMDH) was developed [Ivakhnenko, 1971]. This method constructs a feedforward network as it tries to estimate the system function. The neuron transfer function consists of a quadratic polynomial of two variables, and its parameters are obtained through regression. At each stage of the algorithm, neurons are created pairwisely connecting the output of one layer to form a new layer, starting with the input nodes. This process is exhaustively done for all possible input pairs in each layer, and the connections are always in feedforward, n-to-n+1 layer form. The power of this method comes from the use of simple elementary functions (low dimensionality producing low complexity, linear regression on the parameters, and low training data requirements), and the ability of the algorithm to discard unpromising neurons (elementary hypersurfaces which represent terms in (1)). This selection process is based on a performance criterion which evaluates how close the new surface describes the output data in a least-mean-square sense. The GMDH method estimates the order and the complexity of the plant (low a priori knowledge requirements), and generates suboptimal estimates at each neuron output. Thus the algorithm could be stopped at any point to obtain a model of the process. A heuristic stopping criterion picks the output of the last node before the increase of the fitting error is detected. This technique has drawbacks because of its heuristic nature. There were various improvements made to overcome the problems associated with these heuristics with some success [Duffy and Franklin, 1975; Ikeda, Ochiai, and Sawarogi, 1976; Tamura and

Kondo, 1980]. However, due to the constraints in the possible connectivity pattern in the nodes, the set of possible functions which can be expressed before the termination criteria is applied is at best limited. Thus, the models estimated by the GMDH are suboptimal by nature in expressivity, search power, richness of elementary functions, and suffering from gradient descent hill-climbing problems.

This paper describes a supervised learning algorithm for structure construction and adjustment. Here, systems which can be described by (1) are presented. The computation of the function for each neuron performs a choice from a set of possible functions previously assigned to the algorithm, and it is general enough to accept a wide range of both continuous and discrete functions. In this work, the set is taken from variants of the 2-input quadratic polynomial for simplicity, although there is no requirement making it so. This approach abandons the simplistic mean-square error for performance control in favor of a modified Minimum Description Length (MDL) criterion [Rissanen, 1978], with provisions to measure the complexity of the model generated. The algorithm searches for the simplest model which generates the best estimate. The modified MDL, from hereon named the Structure Estimation Criterion (SEC), is applied hierarchically in the selection of the optimal neuron transfer function from the function set, and then used as an optimality criterion to guide the construction of the structure. The connectivity of the resulting structure is arbitrary, and under the correct conditions [Geman and Geman, 1984] the estimation of the structure is optimal in terms of the output error and low function complexity. This approach shares the same spirit of GMDH-type algorithms. However, the concept of parameter estimation from Information Theory, combined with a stochastic search algorithm - Simulated Annealing, was used to create a new tool for system identification.

This work is organized as follows: section II presents the problem formulation and the Self-Organizing Neural Network (SONN) algorithm description; section III describes the results of the application of SONN to a well known problem tested before using other neural network algorithms [Lapedes and Farber, 1987a; Moody, 1988]; section IV compares this approach to the Generalized Delta Rule applied to feedforward networks; and finally, section V presents a discussion of the results and future directions for this work.

II. THE SELF-ORGANIZING NEURAL NETWORK ALGORITHM

II.1 SELF-ORGANIZING STRUCTURES

As mentioned before, the problem with system identification algorithms is the size to the possible system function space, which creates a need for the reliance of the algorithms on large amounts of a priori knowledge or strong assumptions about the process which might not hold true. It is necessary therefore to create algorithms which are not dependent on this type of knowledge, but rely mostly on the observed data behavior and could incorporate this knowledge when available. This leads to the adoption of self-organizing structures in the design of neural networks, since fixed structure algorithms present the same problems as parameter estimation algorithms, in which the assumed model is either an under or overestimation of the true process. If the elementary functions of the neurons are permitted to be general enough to minimize the prior assumptions, and a self-organized rule is rich enough to capture the complexity of the underlying process, then the simplest model can be found.

The efficiency of the search for the correct model is directly proportional to the amount of knowledge available about the process. One would like to steer away from heavy dependence on this knowledge by allowing, for example, arbitrary connectivity, and number of neurons as is the case with SONN. On the other hand, if such a knowledge is available, it is of primary importance to be able to transparently incorporate such information into the algorithm. There have been very few attempts to design self-organizing structure algorithms due to the complexity of the search space, and only now there are a few tentatives using modifications of the GDR [Hansen and Pratt, 1988]. Algorithms with fixed structure depend on a good guess of the set of initial parameters and structure order, having their final performance dependent on the ability and knowledge of the designer.

The Self-Organizing Neural Network (SONN) algorithm performs a search on the model space by the construction of hypersurfaces. A network of nodes, each node representing a hypersurface, is organized to be an approximate model of the real system. SONN can be fully characterized by three major components, which can be modified to incorporate knowledge about the process: (1) a generating rule of the primitive neuron transfer functions, (2) an evaluation method which accesses the quality of the model, and, (3) a structure search strategy. Below, the components of SONN are discussed.

II.2 THE ALGORITHM STRUCTURE

II.2.1 The Generating Rule

Given a set of observations S:

$S = \{(X_1, Y_1),(X_2, Y_2),\cdots,(X_N, Y_N)\}$ generated by

$$Y_i = f(X_i) + \eta \qquad (2)$$

where f(.) can be represented by a Kolmogorov-Garbor polynomial, and the random variable η is normally distributed, N(0,1). The dimension of output variable $Y = [y_1, y_2, .., y_m]$ is m, and the dimension of input variable $X = [x_1, x_2, .., x_n]$ is n. Every component y_l of Y forms a hypersurface, $y_l = f_l(X)$ in the space of n + 1. The problem is to find $f(.) = [f_1, f_2, ..., f_m]$, given the observations S.

The approach taken here is to estimate the simplest model which best describes f(.) by generating an optimal function at each neuron. This can be viewed as the construction of a hypersurface based on the observed data. It can be described as follows: given a set of observations S; use p components of the n dimensional space of X to create a hypersurface which best describes $y_l = f_l(X)$ through a three step process. To explain the process, the following terms are defined: the terminals γ_j (i=1,n) defined as the ordered set whose elements initially are the jth components of all input variables X_i (i=1,N), the set A as the set of terminals, and the mapping $\pi_j : A \to \gamma_j$ is the projection on jth terminal of set A. The selection mapping $\psi_k^p : A_k \to \{\pi_i, ..., \pi_j\}$ $1 \le i,j \le |A_k|$ consists of p such π's

Initialize k=1, $A_1 = \{ \gamma_1, \gamma_2, ..., \gamma_n \}$ whose size corresponds to the dimension of X.

Step 1. From a set of prototype function F, select a function h in the kth iteration. Construct the hypersurface $h_k(\psi_k^p(A_k))$.

Step2. If the global optimality criterion is reached by the construction of $h_k(\psi_k^p(A_k))$, then stop, otherwise continues to the third step.

Step3. $A_{k+1} = A_k \cup h_k(\psi_k^p(A_k))$, k=k+1, go to step 1.

The resulting model is a multi-layered neural network whose topology is arbitrarily complex and created by a stochastic search guided by an optimality criterion (Structure Estimation Criterion). Each neuron describes a function h of p variables.

For simplicity in this work, the set of prototype functions (F) is restricted to be 2-input quadratic surfaces or smaller, with only four possible types:

$$y = a_o+a_1x_1+a_2x_2 \quad (3)$$

$$y = a_o+a_1x_1+a_2x_2+a_3x_1x_2 \quad (4)$$

$$y = a_o+a_1x_1+a_2x_1^2 \quad (5)$$

$$y = a_o+a_1x_1+a_2x_2+a_3x_1x_2+a_4x_1^2+a_5x_2^2 \quad (6)$$

Type (3) indicates a linear relationship between the input and output variables, types (4) and (5) indicate a 2nd order relationship, whereas type (6) is a complete quadratic polynomial in 2 variables. There is no restriction on the use of higher order polynomials, functions of more variables, or other functions such as the sigmoid (logistic) or sinusoidal functions.

II.2.2 Evaluation of the Model Based on the MDL Criterion

The selection rule (T) of the neuron transfer function was based on a modification of the Minimal Description Length (MDL) information criterion. In [Rissanen, 1978], the principle of minimal description for statistical estimation was developed. This principle chooses the best model as the one that minimizes the total number of binary digits required to encode the observations. The reason for the choice of such a criterion is that, in general the accuracy of the model can increase at the expense of simplicity in the number of parameters. The increase of complexity might also be accompanied by the overfitting of the model. To overcome this problem, the MDL provides a trade-off between the accuracy and the complexity of the model by including the structure estimation term of the final model. The final model (with the minimal MDL) is optimum in the sense of being a consistent estimate of the number of parameters while achieving the minimum error [Rissanen, 1980]. Given a sequence of observation $x_1, x_2, \ldots, x_n$ from the random variable X, the dominant term of the MDL in [Rissanen, 1978] is:

$$\mathrm{MDL} = - \log f(x|\theta) + 0.5\, k \log N \qquad (7)$$

where $f(x|\theta)$ is the estimated probability density function of the model, k is the number of parameters, and N is the number of observations. The first term is actually the negative of the maximum likelihood (ML) with respect to the estimated parameter. The second term describes the structure of the models and it is used as a penalty for the complexity of the model. As indicated in [Rissanen, 1983], the ML estimate is a special case of the MDL, as is also the Maximum Entropy Principle [Rissanen,1983; Feder,1986]. A related estimator is the AIC derived by [Akaike, 1974] which has a similar structure defined as:

$$\mathrm{AIC} = - 2 \log f(x|\theta) + 2\, k \qquad (8)$$

For the linear polynomial regression, as in the set F {(4)-(6)}, the AIC reduces to $\log(s_n^2) + 2\,k$, where s_n^2 is the mean square error, and k is the number of parameters in the model [Akaike, 1972]. In spite of the success of the AIC estimator in various applications such as AR model estimation, and river environment identification, the estimator has been proven inconsistent [Kashyap, 1980]. In the case of linear polynomial regression, the MDL is:

$$\mathrm{MDL} = 0.5\, N \log S_n^2 + 0.5\, k \log N \qquad (9)$$

where k is the number of coefficients in the model selected.

In the SONN algorithm, the MDL criterion is modified to operate both recursively and hierarchically. First, the concept of the MDL is applied to each candidate prototype surface for a given neuron. Second, the acceptance of the node, based on Simulated Annealing, uses the MDL measure as the system energy. However, since the new neuron is generated from terminals which can be the output of other neurons, the original definition of the MDL is unable to compute the true number of system parameters of the final function. Recall that due to the arbitrary connectivity it is non trivial to compute the number of parameters in the entire structure. In order to reflect the hierarchical nature of the model, a modified MDL called Structure Estimation Criterion (SEC) is used in conjunction with an upper bound on the number of parameters in the system at each stage of the algorithm. The expression to compute the estimated upper bound of the number of parameters in the system, $\hat{k}$, is:

$$\hat{k} = \binom{r+d}{r} \qquad (10)$$

where r is equal to 2^l, l is the number of neurons between the input and the output (layers), and d is the number of different variables in the model. The proof of the upper bound is given in the appendix. The upper bound of the number of parameters in the example of the figure 2 can be estimated as follows:

$$\widehat{k_7} = \binom{2^2+4}{2^2} = \binom{8}{4} = 70. \qquad (11)$$

Another computationally efficient heuristic for the estimation of the number of parameters in the model is based on the fact that SONN creates a tree-like structure with multiple roots at the input terminals. Then k, in expression (12), can be estimated recursively by:

$$\hat{k} = \hat{k}_L + \hat{k}_R + \text{(no. of parameters of the current node)} \qquad (12)$$

where $\hat{k}_L$ and $\hat{k}_R$ are the estimated number of parameters of the left and right parents of the current node, respectively. This heuristic estimator is neither a lower bound nor an upper bound of the true number of parameter in the model.

II.2.3 The SONN Algorithm

To explain the algorithm, the following definitions are necessary:

NODE - a neuron and the associated function, connections, and SEC.

BASIC NODE - A node for the system input variable.

FRONT NODE - A node without children.

INTERMIDIATE NODE - The nodes that are neither front or basic nodes.

STATE - The collection of nodes, and the configuration of their interconnection.

INITIAL STATE (S_I) - The state with only basic nodes.

PARENT AND CHILD STATE - The child state is equal to the parent state except for a new node and its interconnection generated on the parent state structure.

NEIGHBOR STATE - A state that is either a child or a parent state of another.

ENERGY OF THE STATE (SEC-S_i) - The energy of the state is defined as the minimum SEC of all the front nodes in that state.

In the SONN algorithm, the search for the correct model structure is done via Simulated Annealing. Therefore the algorithm at times can accept partial structures that look less than ideal. In the same way, it is able to discard partially constructed substructures in search of better results. The use of this algorithm implies that the node accepting rule (R) varies at run-time according to a cooling temperature (T) schedule. The SONN algorithm is as follows:

```
Initialize T, and SI
Repeat
        Repeat
            $S_j$ = generate ($S_i$),                 - application of P.
            If accept ( SEC_$S_j$, SEC_$S_i$, T) then $S_i = S_j$,
                                                      - application of R.
        until the number of new neurons is greater than N.
        Decrease the temperature T. ( cooling sequence )
until the temperature T is smaller than $T_{end}$ - (terminal temperature for
Simulated Annealing).
```

Each neuron output and the system input variables are called terminals. Terminals are viewed as potential dimensions from which a new hypersurface can be constructed. Every terminal represents the best tentative to approximate the system function with the available information, and are therefore treated equally. The rule P which produces new neurons can be defined as:

```
generate ( $S_i$ ):
Randomly select two terminals from $A_{i+1} = A_i \cup h_i(\psi_i^2(A_i))$
For each prototype surface in the set F, fit the surface and calculate SEC.
Choose the surface with the smallest SEC (best fitting and less complex).
Construct the neuron using the prototype surface chosen if the SEC of the
neuron is smaller than the SEC of parents.
```

The generation function can be further refined by keeping records of the combination of nodes being produced, and checking the new combination against the records. By such a checking process, we can avoid the useless nodes being re-generated during the search procedure. Such a scheme corresponds to a memory version of Simulated Annealing.

The rule R represents the test for acceptance of a new neuron. It searches the model space using Simulated Annealing, where the SEC is viewed as the state energy. There is a non zero probability that a previously generated neuron can be destroyed, thus returning to a previous state. The rule can be defined as:

accept (SEC_*S_j, SEC_*S_i*, T):*

If the new state SEC is smaller than the SEC of the current state, accept the neuron, else accept the neuron with probability:

$$p = Exp(-(\frac{\Delta\ State\_SEC}{T_{Current}}))) \qquad (13)$$

There have been different annealing sequences proposed in the literature [Kirkpatrick et al., 1983; Gelfand and Mitter, 1985]. We adopted the geometrical annealing sequence in the SONN algorithm.

$$T_{new} = \alpha * T_{old} \qquad (14)$$

II.2.4 Structuring the State Search

The SONN algorithm performs the organization of the model structure. Each node is generated competitively and cooperatively with one another. The competition occurs between front nodes competing for lower SEC values. The cooperation occurs between parent nodes and their child. The initial set of terminals contains only the input nodes. From this set, candidates are chosen randomly (in this case 2) to construct a new terminal (new node for the list), which becomes the front node. This process of the combining of nodes starts at a high temperature T, and an initial state S (only input terminals). In the outer loop of the algorithm that defines SONN, the temperature is decreased after a certain number of new nodes are created. This slow cooling procedure is motivated by the analogy with the annealing process used in physical systems to minimize the system's potential energy [Kirpatrick et al, 1983]. A final state S_f is reached when the low temperature T_{end} is present. S_f corresponds to the state of low energy. The energy of the state in terms of the SEC is defined to be the energy function of the Simulated Annealing algorithm.

The application of the P rule produces a state S_j by a perturbation in the state S_i. The transition probability between states is determined by the probability function (13) as part of the acceptance rule R. Figure 3 shows the generation of state S_j from state S_i. The difference between the two states is the new neuron H, the connections from the previous structure into H, and the new terminal (the output of H). Every terminal is an estimate of the output of the system; in other words, the structure relevant to the terminal is an approximated suboptimal model of (1). The optimal model which represents the global minimum of the objective function will be embedded in the final state S_f.

The definition of the system's SEC is consistent with the state generation mechanism. Given the state S_i and the state S_j derived after the structure of S_i, then:

1. State S_j has the same energy value of S_i, if the new terminal generated has a higher SEC than the front nodes of S_i. This is true since the structures in S_i are subsumed in S_j.
2. State S_j has a lower energy than S_i, if the new terminal generated has a lower SEC than the front nodes of S_i. In this case, there is a structure in S_j which better estimates the model and it is not present in S_i.

III. EXAMPLE - THE CHAOTIC TIME SERIES

Randomness is a prevalent characteristic in natural systems such as economical or physical systems. Traditionally, randomness has been viewed as a characteristic attributed to the inherent complexity in the large number of degrees of freedom present in the observed system. However, recently chaos has been proposed as a possible source of randomness in dynamical systems [Farmer and Sidorowich, 1987]. A system is said to be chaotic if the evolutionary trajectory of the system is generated by a deterministic mechanism, but it is very sensitive to the system's initial condition. Examples of chaotic systems can be found in turbulent fluid flow analysis [Packard, Crutchfield, Farmer, and Shaw, 1980], and biological systems [Mackey and Glass, 1977]. Since under certain conditions a chaotic system behaves randomly, the identification of such systems is difficult. Under those conditions, a model capable of identifying the underlying deterministic mechanism can greatly improve system performance, predictability and control.

Lapedes and Farmer [1987a] were the first to explore the use of neural networks for the identification of a chaotic time series. A two hidden layer neural network with the sigmoid transfer function in the individual neurons was used to identify the chaotic time series generated from the Mackey-Glass differential equations [Mackey and Glass, 1977]. The two hidden layer neural network using the Generalized Delta Rule (GDR) has been proven successful as a tool to identify such systems. This architecture, together with the GDR, is then referred to as the "Nonlinear Signal Processing Method." Other neurocomputing methods have been applied to the same problem. For example, the work of Moody [1988] used a variation of the CMAC algorithm proposed by Albus [1981].

In the following results, the same chaotic time series was used. The SONN with the SEC, and its heuristic variant, were used to obtain the approximate model of the system. The result is compared with those obtained by using the Nonlinear Signal Processing Method [Lapedes and Farber, 1987a]. The advantages and disadvantages of both approaches are analyzed in the next section.

III.1 STRUCTURE OF THE PROBLEM

The Mackey-Glass differential equation used here can be described as:

$$\frac{\partial x(t)}{\partial t} = \frac{a\,x(t-\tau)}{1 + x^{10}(t-\tau)} - b\,x(t) \qquad (15)$$

By setting $a = 0.2$, $b = 0.1$, and $\tau = 17$, a chaotic time series with a strange attractor of fractal dimension about 3.5 will be produced [Farmer and Sidorowich, 1987].

If the points on the attractor of dimension A are related by :

$$x(t+P) = f(x_1(t), x_2(t), \cdots, x_M(t)) \qquad (16)$$

then the necessary condition [Takens, 1981] for f(.) to be a smooth mapping is:

$$d_A \leq d_M \leq 2\,d_A + 1 \qquad (17)$$

where P is the prediction time, d_A is the dimension of the attractor, and d_M is the embedding dimension of the dynamic system of (15). Takens' theorem guarantees the existence of f (.), but it does not provide a constructive method. The Nonlinear Signal Processing Method utilizes the GDR in a four layer feedforward neural network to estimate f (.). To facilitate the comparison between the SONN and the GDR, we chose the embedding dimension to be 4.

This also specifies that four data points should be chosen as state variables: x(t), x(t -τ), x(t - 2τ), x(t -3τ).The prediction time P is chosen according to the need for long term or short term prediction. However, for practical engineering use, the short term prediction is often sufficient. In this example, P was chosen to be 6. To compare the accuracy of prediction the normalized root mean square error is used as a performance index:

$$\text{Normalized RMSE} = \frac{\text{RMSE}}{\text{Standard Deviation}} \qquad (18)$$

III.2 Using the SONN Algorithm for the Time Series Identification

The SONN algorithm is based on the idea that all neurons generated try to solve the problem by best tuning themselves and the structure to the system's response. The "quality" of such estimation is given directly by the SEC, and the "quality" of the moves between states by the difference between state SEC measures. Therefore, each and every terminal (neuron output) produces of itself the best possible estimate for the system given the present structure. The user can select any suboptimal stopping criterion for the algorithm to obtain an estimate of the system, or he/she can allow the algorithm to converge to its global optimum. Since the search space is very large, it is sometimes undesirable to spend the computing time for the optimal estimate. Therefore, by considering the computation resource available, the designer can stop the algorithm by accepting a feasible but suboptimal solution.

III.2.1 SONN with the SEC

In this subsection, the SEC is used as the criterion in the choice of the model for the chaotic time series, using the estimator of (10). Recall that this estimator is the upper bound of the number of parameters in the model. The resultant model tends to be an underestimate of the number of parameters of the real system. Since the SONN is basically a stochastic search algorithm, to compare the result of the SONN with the Nonlinear Signal Processing Method, 10 runs of the SONN were initiated. The averaged statistics of the performance parameter were computed. In the example of the chaotic time series, the first 100 points of the series were used as training data. One example of a non linear model which has been obtained by SONN for this series is a two layer network with five nodes (figure 4). There are 15 weights in the model, 4 connections. The output of the network overlapped with the output of the system and is shown in figure 5. The performance index over the next 400 points (101st-500th) used as testing data is equal to 0.137. The averaged number of regressions (the corresponding number of ephocs is 64.1) used to obtain the model is 320.5. The standard deviation of the number of regressions is 29.4. The SONN using the SEC as an estimator did not produce satisfactory results. The SEC produces severe constraints on the number of parameters the model can have (underestimation), therefore drastically limiting the search space.

III.2.2 SONN with the Heuristic SEC (SONN-h)

In the following examples, a modified heuristic version of the SEC is used to try to overcome the limitations described before. The estimator of the number of parameters is switched from the expression (10) to the estimator of (12). To compare the performance, the same averaging method used in last section is adopted. The first run was initiated. Two suboptimal models were obtained during the search procedure. The same state-SECs were then used as the criterion to choose the corresponding suboptimal models in the next several runs. The averaged number of the regressions (i.e. the epochs) was calculated.

III.2.2.1 Node 19

First, the SONN is allowed to generate up to the 19th accepted node. In this example, the first one hundred points of the time series was used for training, and samples 101 through 400 used for prediction testing. The total number of weights in the network is 27. The performance index average $0.07\bar{7}$. The output of the network is overlapped in figure 6 with the original time series, and the averaged number of the regressions spent was 225.8(i.e. 45.16 epochs).

For comparison purposes, a GDR network with the structure used in [Lapedes and Farber, 1987a] is trained for 6,500 epochs. The training data consisted of the first 500 points of the time series, and the testing data ran from the 501st sample to the 832nd. The total number of weights is 165, and the final performance index is equal to 0.165. Notice that the number of epochs in the GDR training was smaller than the work of Lapedes and Farber. This was done to give both algorithms similar computational resources. Figure 7 shows the original time series overlapped with the GDR network output.

III.2.2.2 Node 37

The second model chosen was formed by the 37th accepted node. Remember that Simulated Annealing requires a non zero probability of returning to a previously visited state, and therefore, in this case, withdrawing part of the structure. The network was trained in a similar manner to the first example, since it was actually part of the same run. The final number of weights was 40, the performance index was 0.018, and the average number of the regressions spent was 1765.8 (i.e. 353.13 epochs). Figure 8 shows the output of the network with the original time series. Figure 9 shows the GDR with 11,500 epochs. Notice that in both cases, the GDR network requires 150 connections and 150 weights, as compared to 12 connections and 27 weights for the first example and 10 connections and 40 weights for the second example. Figure 10 shows the final state of the algorithm; some of the connections were omitted for clarity.

Another interesting aspect of the final state is that not all, but only 3 of the inputs contribute to the model.

III.4 COMPARISON OF THE GDR AND SONN RESULTS

In figure 12 the performance indexes versus sample points for the 2 experiments with GDR and SONN are shown. The table 1 summarizes the comparison data.

Table 1 Comparison between the four experiments

Algorithm	Ephocs	Connections	Weights	Performance
SONN	320.5	4	34	0.137
GDR	6,500	150	150	0.165
SONN-h	225.8	12	27	0.077
GDR	11,500	150	150	0.038
SONN-h	1765.8	10	40	0.018

IV. ADVANTAGES AND DISADVANTAGES

In this section, the advantages and disadvantages of SONN as compared with GDR are outlined.

IV.1 ADVANTAGES OVER GDR

The SONN algorithm requires considerably less samples to acquire an estimate of the system. In this exercise, GDR was given five times more samples than SONN. GDR assumes that the order of the model is known through the size of the hidden units; SONN estimates the complexity of the model at run-time. It constructs the connectivity pattern, and the neuron transfer functions during the identification process. The resulting network is more sparsely connected, requiring less hardware, and achieving better results for a given amount of computation. The SONN algorithm is not subject to the difficulties encountered with local minima, nor with the initial set of weights given to the network. The learning process does not depend on ad-hoc error assignment mechanisms.

The algorithm produces estimates of the system model at every new node, permitting the user to trade-off the accuracy of the model for learning time. The final model is more accurate, and at the same time less complex and is dependent on fewer parameters. The prediction error revealed to be almost constant over a wide range of samples. The arbitrary connectivity, with the two-input function used here, can be easily extended to accommodate functions of an arbitrary number of variables. Furthermore, the connectivity pattern can be restricted by incorporating knowledge about the problem, such as neuro-sensorial spatio-temporal mappings present on biologic hearing systems, or biological visual systems. SONN can be used as an investigation tool to hypothesize the connectivity pattern in such cases.

Different sets of transfer functions can be mixed in the same network, with the choice subject to a problem dependent criterion. This allows for another level of a priori knowledge incorporation. The functions can have temporal behavior, or can be simple linear discriminant functions. This work can easily be extended for the use of SONN as a classifier. Although not tested yet, the set of quadratic functions used here should work quite well for classification tasks, since they are special cases of the Mahalanobis distance.

The SONN algorithm can be used in cases where the connectivity pattern between the input and the output is unknown, such as sensori-neuronal pathways. The algorithm can be restricted to apply a set of functions known to be present in the problem, and have the choice of new terminals restricted by high level knowledge. This same idea can be used to integrate symbolic knowledge about the problem to restrict the search with the numeric knowledge being developed during the search. The SONN algorithm could reason about its experiments on the model, with the choice of the experiment being guided by a priori knowledge, as opposed to only using the outcome of the experiment. In this case the search can be viewed as a symbol-numeric planning strategy. The algorithm can then be used to produce a symbolic model of the underlying system.

Overall the algorithm has low computational demands as compared to the GDR, and reduces the learning problem at each neuron to a simple linear regression, therefore avoiding local minima problems. At every stage, a sub-optimal partial structure is generated, allowing for variations on the algorithm stopping criterion.

IV.2 DISADVANTAGES OVER GDR

The SONN learning algorithm, based on a stochastic search, is non deterministic, and therefore the learning time cannot be known a priori. For the same reason, successive runs of the algorithm do not generate the same partial structures, but they do generate the same optimal final structure if the algorithm is allowed to run to the end. Similarly, the connectivity cannot be known a priori for the partial structures, which demands flexibility of communication for parallel hardware architectures.

V. CONCLUSION AND FUTURE WORK

In this study, we proposed a new approach for the identification problem based on a flexible, self-organizing neural network (SONN) structure. The variable structure provides the opportunity to search and construct the optimal model based on input-output observations. The hierarchical version of the MDL, called the structure estimation criteria, was used to guide the trade-off between the model complexity and the accuracy of the estimation. The SONN approach demonstrates potential usefulness as a tool for system identification through the example of modeling a chaotic time series.

An alternative interpretation of the competition between the nodes in different layers is the feature selection process in the pattern recognition literature [Fukunaga, 1972]. The quadratic node functions are specializations of the Mahalanobis distance, and can easily be changed to accommodate other discriminant functions. Thus it is not unreasonable to treat the final approximated model as a classifier. Polynomial classifiers have been successfully used before [Sanz and Hinkle, 1988].

Future work on the SONN algorithm includes the development of a better estimator for the SEC, the use of multiple sets of functions, the generalization to functions of an arbitrary number of variables (arbitrary neuron branching), the use of high level symbolic knowledge about the problem domain to aid the search, and a connection to a symbolic module to analyze the hypotheses, and generate model closed form solutions for symbolic manipulation. Possible future applications include: use as a static pattern classifier, extensions of the functions to operate on time varying patterns, knowledge based restriction of the connectivity pattern to search for neurobiologically plausible nerve connectivity, naive and experimental physics reasoning (model building and hypothesis testing).

Appendix :

The proof of expression (10) is the upper bound of the number of parameters in the layered system is followed from the induction. First, we prove that the nodes in layer 1 satisfy the expression. Let the node X_1 in layer 1 receive the inputs x1, x2 from layer 0, then l=1, d=2. The maximum number calculated according to expression (10) will be 6. Since the highest order polynomial in propotype functions (F) is second order. he number of parameters is 6. Hence, the expression holds for nodes in the 1st layer. Suppose the expression (10) holds for the two nodes , $X_{k,1}$, $X_{k,2}$, with highest order k1,k2 separately, in the kth layer, then k1,k2 must be smaller than 2^k. Observe that there are four possible combinations of the variables according to (F). The highest order terms of these formulas are of the following three types:

(1). $a_1x_1^2$

(2). $a_2x_2^2$

(3). $a_1x_1x_2$

all of these three formulas of the combination of variables will produce the polynomial of a degree no more than 2^{k+1}. Since d of the expression (10) is the union of the variables from the nodes in kth layer. By the expression derived in [Roy and Sherman, 1967] , all the possible combinations of terms appearing in the resultant polynomial can be computed according the expression (10). Since the actual order of the polynomial,d, of node x_{k+1} is no more than 2^{k+1}. The $\hat{k}$ computed by expression (10) will be the upper bound for the number of parameters in the system. Q.E.D.

References :

J. L. Marroquin, "Probabilistic solution of Inverse Problems," Ph.D. dissertation, M.I.T., Sept. 1985.

B. Widrow, S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Inc. 1985.

L. A. Zadeh, "From Circuit to System Theory," Proc. IRE 50, pp. 856-865, 1962.

K. J. Åström, P. Eykhoff,"System Identification - A Survey," Automatica, vol.7, pp. 123-162, 1971.

D. Garbor et. al. "A Universal Nonlinear Filter, Predictor and Simulator Which Optimizes Itself by a Learning Process," IEE Proc., 108B, pp. 422-438, 1961.

R. J. Roy, J. Sherman, "A Learning Technique for Volterra Series Representation," IEEE Trans. on Automatic Control, pp. 761-764, December 1967.

A. Lapedes and R. Farber,"Nonlinear Signal Processing Using Neural Networks; Prediction and System Modeling," TR LA-UR-87-2662, 1987.

A. Lapedes and R. Farber,"How Neural Network Work," TR LA-UR-88-418, 1987.

A. G. Ivakhnenko,"Polynomial Theory of Complex Systems," IEEE Trans. S.M.C, Vol. SMC-1, no.4, pp. 364-378, Oct. 1971.

J. J. Duffy and M. A. Franklin, "A Learning Identification Algorithm and its Application to an Environmental System," IEEE Trans. S.M.C., Vol. SMC-5, no. 2, pp. 226-240, 1975.

S. Ikeda, M. Ochiai and Y. Sawarogi, "Sequential GMDH Algorithm and its Application to River Flow Prediction," IEEE Trans S.M.C., Vol. SMC-6, no.7, pp. 473-479, July, 1976.

H. Tamura, T. Kondo, "Heuristics Free Group Method of Data Handling Algorithm of Generating Optimal Partial Polynomials with Application to Air Pollution Predication," Int. J. Systems Sci., 11,no.9, pp. 1095-1111, 1980 .

J. Rissanen "Modeling by Shortest Data Description," Automatica, Vol.14, pp. 465-471, 1978.

J. Rissanen, "Consistent Order Estimation of Autoregression Processes by Shortest Description of Data," *Analysis and Optimation of Stochastic System*, Jacobs et al eds. NY Academic, 1980.

J. Rissanen,"A Universal Prior for Integers and Estimation by Minimum Description Length," Annuals of Statistics, Vol.11, no. 2, pp. 416-431,1983.

M. Feder, "Maximum Entropy as a Special Case of the Minimum Description Length Criterion," IEEE Trans. of Information Theory, vol. IT-32, no.6, pp. 847-849, Nov. 1986.

S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributation, and the Bayesian Restoration of Images," IEEE Pattern Analysis and Machine Intelligence, PAMI-6, pp. 721-741, 1984.

J. Moody, "Fast Learning In Multi-Resolution Hierarchies," *Advances in Neural Information Processing Systems I*, David S. Touretzky ed., pp. 29-39, 1989.

J. S. Albus, *Brain, Behavior and Robotics*. Byte Books, 1981.

H. Akaike, "A New Look at the Statistical Model Identification", IEEE Trans. on Automatic Control, Vol-AC-19, no. 6, pp. 716-722, Dec.,1974.

H.Akaike, "Automatic Data Structure Search by the Maximum Likehood," Comput. Biomed. Suppl., 5th Hawaii Int. Conf. Syst., pp. 99-101, 1972.

R.L. Kashyap, "Inconsistency of AIC rule," IEEE Trans. on Automatic Control, Vol-AC-25, no. 5, pp. 997-998, 1980.

J. D. Farmer, J.J. Sidorowich,"Predicting Chaotic Time Series," Physical Review Letters, 59(8): 845-848,1987.

N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a Time Series", Physical Review Letters, 45(9):712-716, 1980.

F.Takens, "Detecting Strange Attractor In Turbulence," Lecture Note in Mathematics, PP. 366, D. Rand, L. Young Ed., Spring Berlin, 1981.

M. Mackey, L. Glass, "Oscillation and Chaos in Physiological Control System," Science , pp. 197-287 ,1977.

S. J. Hansen and Y. Pratt, "Comparing Biases for Minimal Network Construction with Back-Propagation," *Advances in Neural Information Processing Systems I*, David S. Touretzky ed., pp. 177-185, 1989.

S.Kirkpatrick, C.D. Gelatt, M.P. Vecchi,"Optimization by Simulated Annealing," Science, vol.220, pp. 671-680, May 1983.

S. B. Gelfand., S.K. Mitter, "Analysis of Simulating Annealing for Optomization," Proc. 24th Conf. on Decision and Control, F.T. Lauderdale, pp. 779-786, December, 1985.

K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, New York, 1978.

Jorge L. C. Sanz, Eric B. Hinkle, "Note on Contrast Measure and Polynomial Classifiers," Proceedings of the IEEE, vol.76, no. 3, pp. 256-259March ,1988.

M. F. M. Tenorio and W.-T. Lee, "Self-Organizing Neural Network for the Identification Problem," *Advances in Neural Information Processing Systems I*, David S. Touretzky ed., pp. 57-64, 1989.

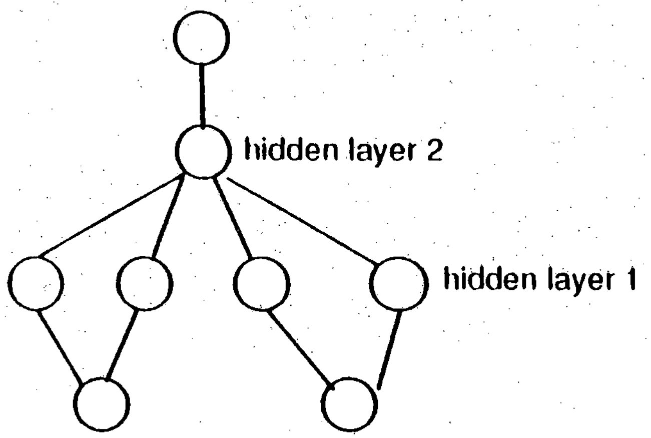


Fig.1 the substructure of neural network of nonlinear signal processing method

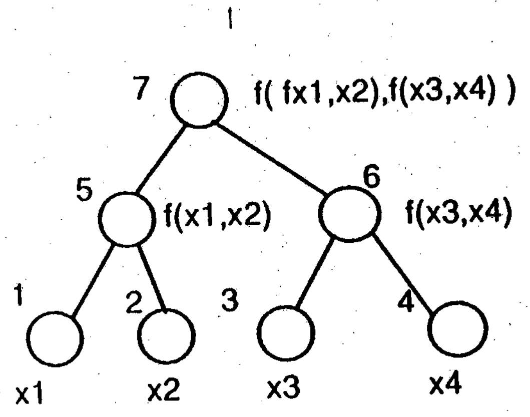


Fig. 2 computation of the $\widehat{K}$ for the system

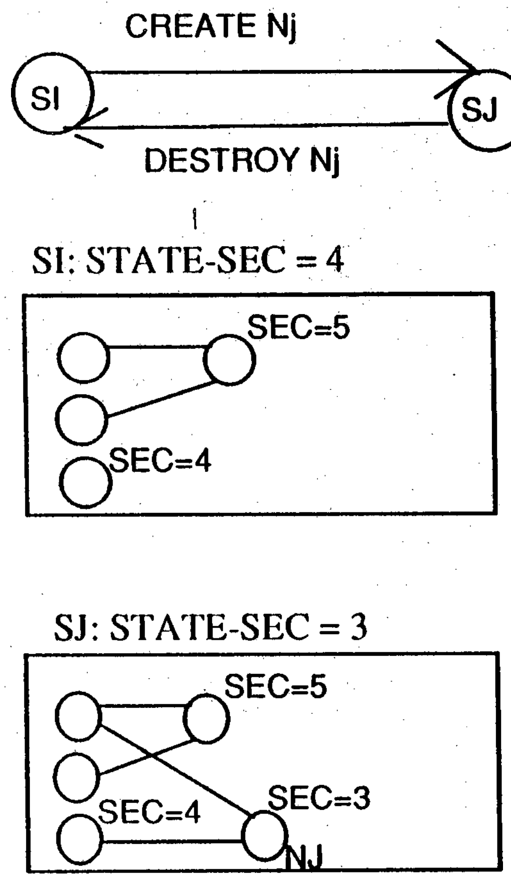
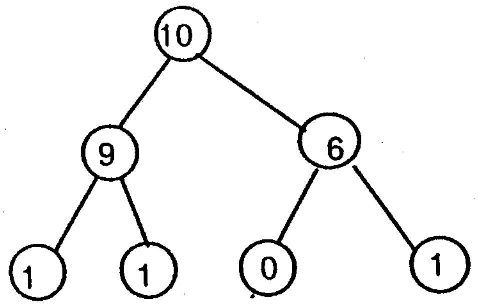


Fig. 3 state transition and calculation of the State_SEC



$x_{10} = 0.484497*x_9^2 + 0.078320*x_6^2 + -0.766702*x_9*x_6 + 0.894671*x_9 + 0.476700*x_6 - 0.177909$

$x_9 = -2.006957*x_1^2 + 2.968429*x_1 - 0.004738$

$x_6 = -2.409953*x_0^2 + -2.218073*x_1^2 + 1.037223*x_0*x_1 + 2.259852*x_0 + 2.259852*x_1 - 0.852424$

$x_1 = x(t-6)$

$x_0 = x(t)$

Fig. 4 model from SONN with the SEC

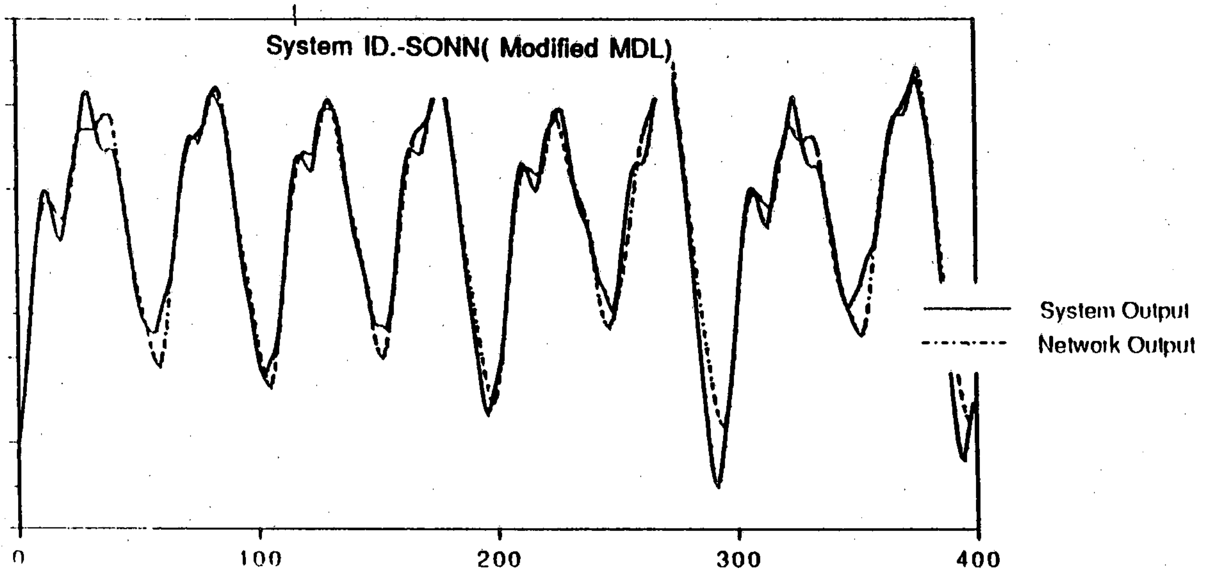


Fig. 5 data from SONN with SEC

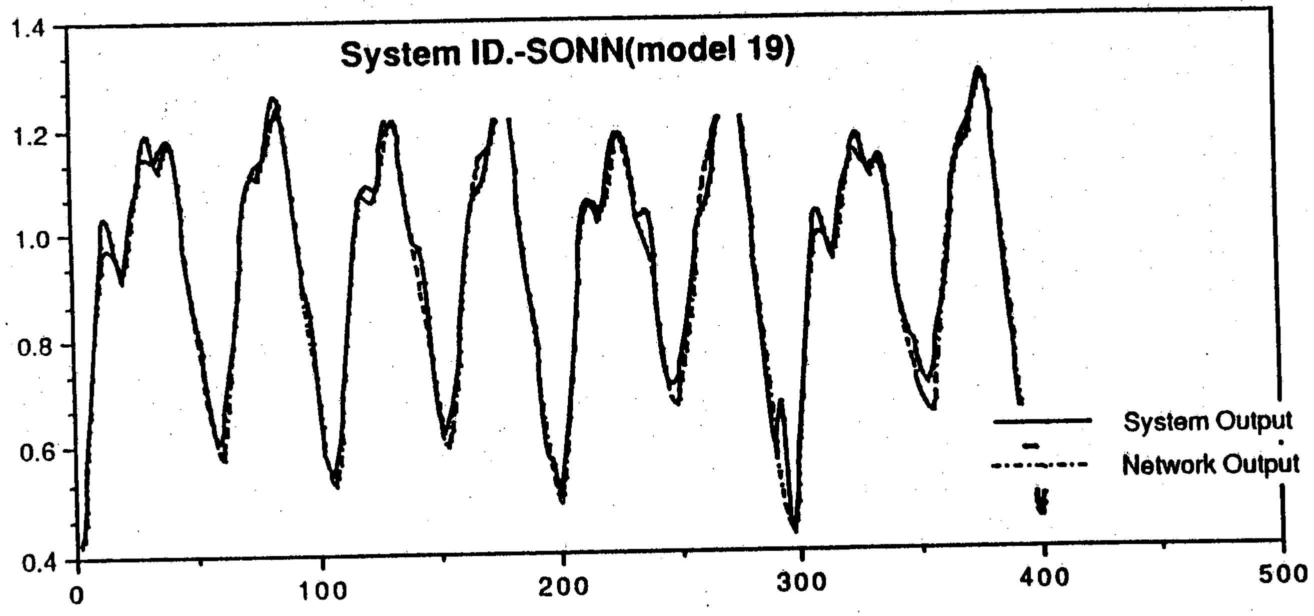


Fig. 6 data from node 19

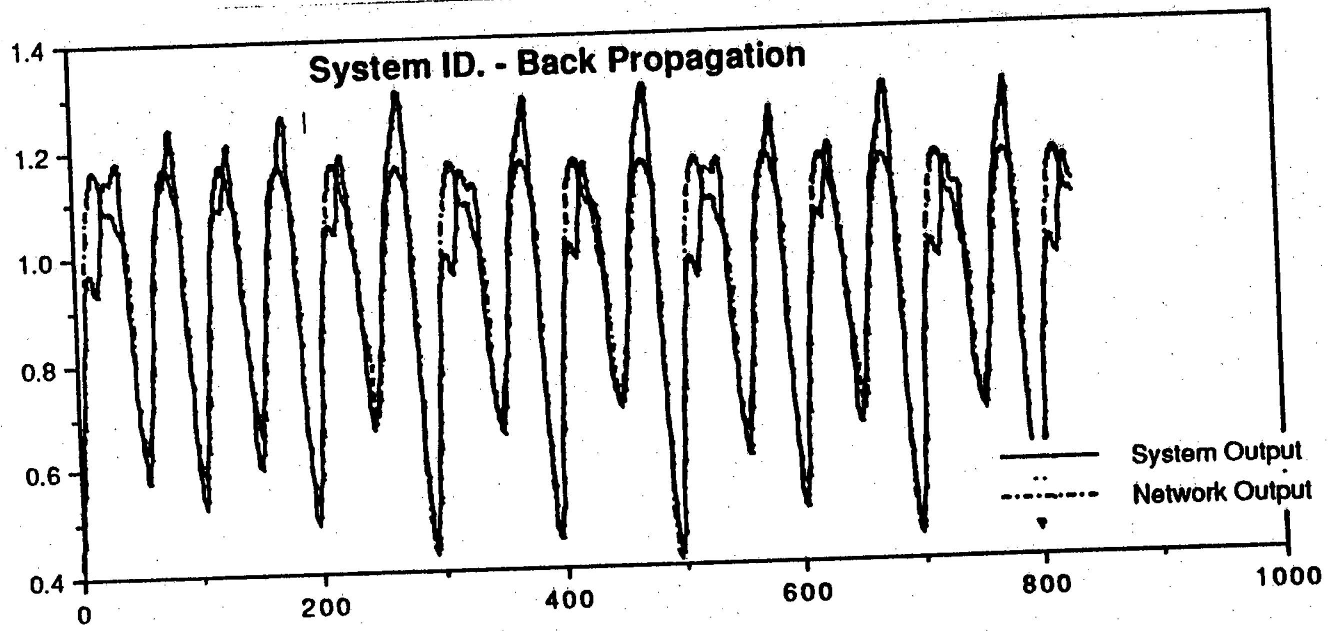


Fig. 7 data from nonlinear signal processing method with 6,500 epochs

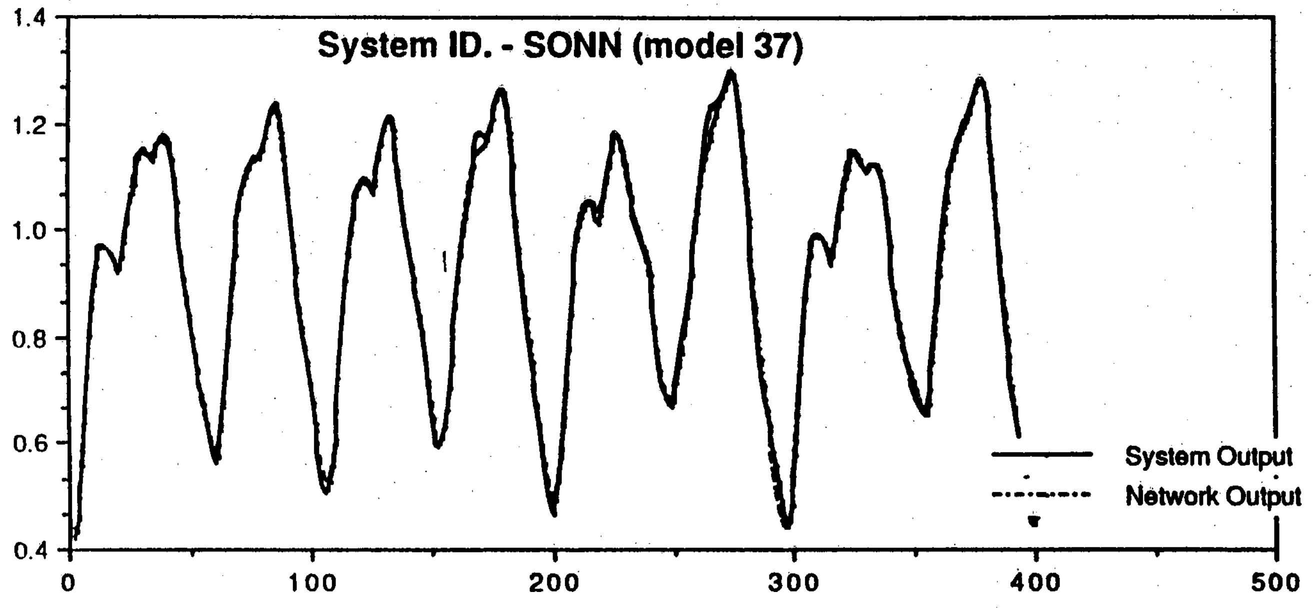


Fig. 8 data from node 37

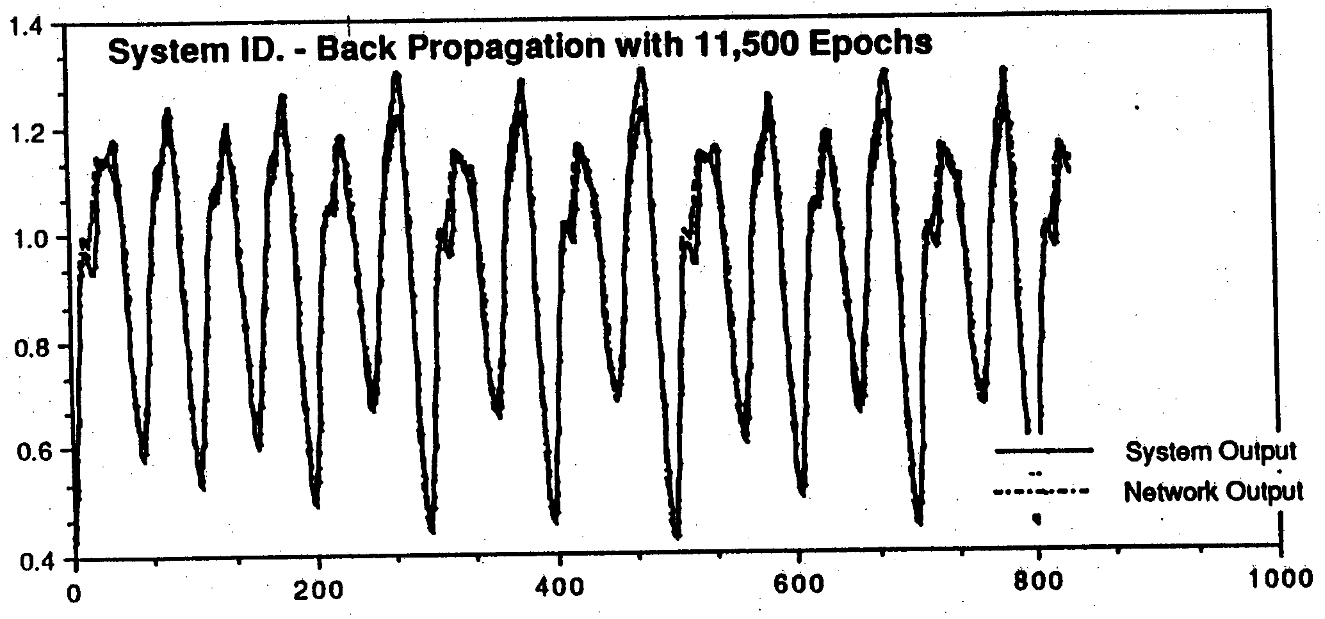


Fig. 9 data from nonlinear signal processing method
with 11,500 epochs

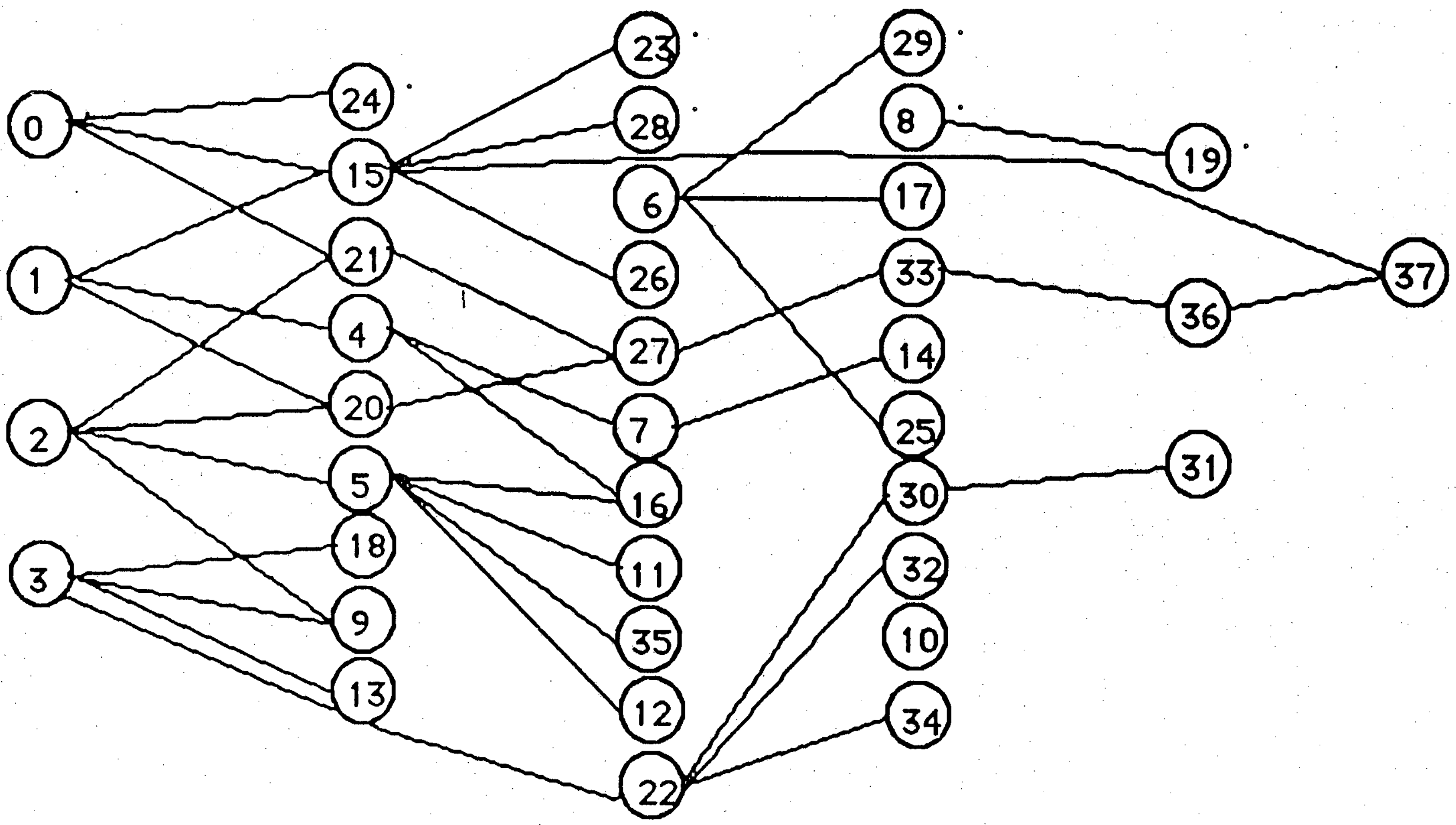


Fig. 10 final state of SONN

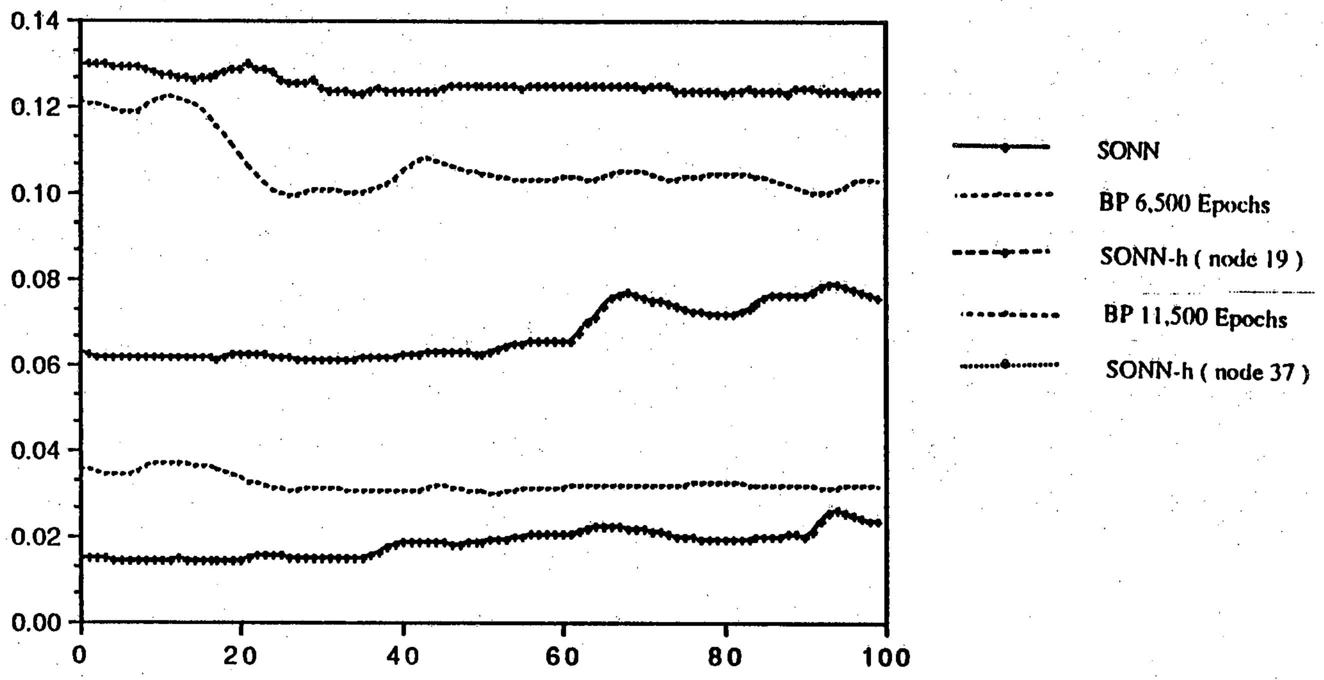
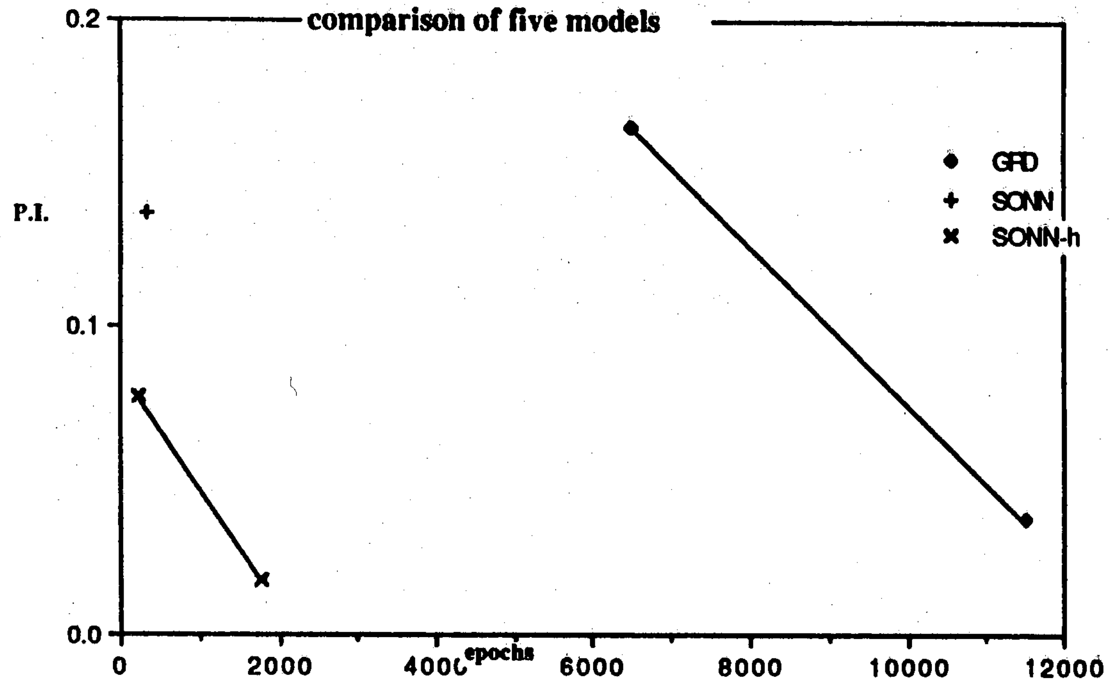


Fig 11.performance comparison of models



comparison of five models
P.I.
0.2
0.1
0.0
0
2000
4000
6000
8000
10000
12000
epochs
GRD
SONN
SONN-h