System Identification Using Neural Networks with Parametric Sigmoid Functions

Mahmoud HASHEMINEJAD*, Junichi MURATA*, Kotaro HIRASAWA* and Setsuo SAGARA**

Nonlinear systems can be modeled by neural networks. However, choice of suitable network architecture is the most important problem. And "how to find the best activation function" is a persistent aspect of the architecture design. Here we have proposed a sigmoid function with one parameter which provides us not only the reduction of error bound but also the opportunity of obtaining better insight into the systems. The proposed function has the ability of recognizing linear and/or nonlinear parts of the system under study. After automatic training of this parameter along the weights, more information about the system will be available. Using this additional knowledge about structure of the system, one will be well equipped to attack control problems such as controller design using neural network model.

Key Words: neural networks, identification, parametric sigmoid functions, nonlinear systems

1. Introduction

Studies on neural networks began with a mathematical model of a nerve cell found by McCulloch and Pitts in 1943. In recent years, neural networks have been tried in large variety of scientific fields because of their learning ability and capability of dealing with nonlinearity. System identification and controller design for nonlinear systems are among them.

Most of the physical systems are nonlinear. If the effect of system nonlinearity is negligible, the physical system can be approximated by a linear dynamical model. However, if it can not be neglected, large error will arise when a linear model is used. A neural network could be a good tool for treating such nonlinear systems. Nevertheless, it requires careful specification for fitting this tool to the subject. These tasks include determining the number of layers and neurons, specification of learning rate and choice of the shape of activation function. For determining the above parameters we should find automatic and reasonable methods which lead to appropriate results. Here we must define what is 'appropriate'. An appropriate neural network model of a nonlinear system should have good accuracy and allow us to obtain some insight into the system's inner structure e.g. whether the system can be dealt as composite of linear and nonlinear systems or not. In other words it should be an accurate gray-box model. The insight will be useful in, for instance, controller design where the established linear control theory maybe employed partly. Fuzzy-neural is another approach to gray-box neural network. However, the insight obtained there is a set of fuzzy rules which is different from our insight and can not be used in controller design based on linear control theory. In this paper we concentrate ourselves on the shape of activation functions: a parametric sigmoid function is exploited and utilized to obtain an appropriate model.

The idea of introducing additional parameters to sigmoid function, has been already investigated. An ordinary sigmoid function ranging over [-1, 1] can be written as equation 1 with $a=1$ and fixed $\sigma$:

$$f(x) = \frac{a}{1+e^{-\sigma x}}.$$  \hspace{1cm} (1)

However, Arai tried to change the slope $\sigma$ of the
sigmoid function\(^{11}\). Song proposed to tune the gain \(a\) of the function\(^{12}\). Nevertheless, the both aimed at accuracy only. The more adjustable parameters, the smaller learning error will result by neural network. Yamada has proposed both slope and gain tuning for sigmoid function, and applied it to controller design\(^{13}\). The remarkable point of his sigmoid function is that, it can cover from linear function to nonlinear threshold-like function by changing the parameter he introduced. However, his aim also lied in good accuracy.

In this article we do not intend to use neural networks as a black-box, but would rather to sensitize it to be able to bring out more information about inner structure of the system. Yamada’s sigmoid function can be a good tool; the parameter tells us some information on the structure, i.e. whether the neuron is linear or nonlinear\(^{14}\). However, there should be a major problem to be solved: how to learn sigmoid parameters along the network weights correctly and uniquely; too many parameters can not be determined uniquely.

2. Improved Neural Network

2.1 Modified Sigmoid Function

The aim for searching another kind of activation function is twofold:

- Firstly, we seek such a network which has good accuracy and high learning speed.
- Secondly, gaining more information about the system would be great help in identification and afterward controller design.

The more number of parameters are introduced in a network, the more describing ability it obtains. Especially introduction of additional parameters to the activation functions and changing their shape by adjusting the parameters will favor in giving more flexibility to the network: it will be able to represent a wide variety of input-output mappings which are different from each other in complexity and smoothness. Also we would like to obtain some insight into the system structure from the neural network model. Thus, we suggest to use a parametric sigmoid function

\[
f_p(x) = \frac{1}{\ln p} \cdot \tanh ((\ln p)x),
\]

as an activation function. Although devising the proposed function has mainly mathematical background, it has biological support as well. This function, as we will see in section 3.1, introduces the dependency between neurons and weights. Recent researches on Aplysia buccal ganglia, a biological creature, presents evidence that synaptic strengths are partially specified by postsynaptic neurons\(^{15}\).

This activation function pushes off the general function of Yamada toward the above mentioned goals. This new shape of the activation function is favorable because using \(\ln p\) instead of \(p\) prevents from excessive reduction of the absolute value of the function.

Expanding equation 2 can help to see the effect of changing \(p\) on the sigmoid function’s shape

\[
f_p(x) = \frac{(2x)^3}{2!} \ln p(x) + \frac{(\ln p)^3(2x)^3}{3!} \ldots
\]

\[
= 2 - (\ln p)x + \frac{2(\ln p)^3x^2}{2!} - \frac{4(\ln p)^3x^3}{3!} + \ldots
\]

(3)

Studying the above equation one can understand how easily a very changeable sigmoid function can be obtained. This variety starts from linear line and can develop to nonlinear ordinary sigmoid function. By setting \(p\) to 1 we can get a linear function instead of the sigmoid function. The shapes for different \(p\)s are depicted in Fig. 1

The proposed activation function will realize

- High flexibility which results in less error bound.
- Sensitizing the network to linear-nonlinear parts.

Figure 2 shows the neural network with parametric sigmoid functions which are drawn as circles in the figure. The round-corner rectangulars are weights.
U_k is a weight on the output, and usually it is equal to one. The jth hidden neuron accepts the weighted sum of the inputs x_k(j) and produces its outputs o_h(j) as,

$$o_h(j) = \frac{1}{\ln p_j} \tanh (\ln p_j \cdot x_k(j)).$$  (4)

The kth output neuron accepts x_o(k) and outputs o_o(k) in the same manner. In learning phase, error signals ∂E/∂o_o(k), ∂E/∂x_o(k), ∂E/∂o_h(j) and ∂E/∂x_h(j) propagate through the network backward.

The parameter p can be tuned along the weights so as to minimize the error between the network output and the teaching signal. If p=1 is obtained after tuning, it means that neuron is linear. Now the problem is "how to tune the parameter p".

2.2 Extended Backpropagation

After adding one parameter to the neuron’s function we should find one way for setting it properly. In the case of the weights, backpropagation has been used widely, by using that, in fact we push the weights to set in some way which causes minimum error. In the same way it is possible to use backpropagation to reach appropriate value of p. The parameter p is updated so as to minimize the squared sum of errors between the network outputs and their desired values:

$$E = \frac{1}{2} \sum \{t(k) - o_o(k)\}^2,$$  (5)

where t(k) is the desired value for the kth output. The update will be done as,

$$p_{\text{new}} = p_{\text{old}} - \beta \frac{\partial E}{\partial o_o(k)}$$  (6)

where β is learning rate. Let us define error signals as,

$$\delta o_o(k) = \frac{\partial E}{\partial o_o(k)} = o_o(k) - t(k),$$  (7)

$$\delta o_h(j) = \frac{\partial E}{\partial o_h(j)} = \frac{1}{\ln p_j} \tanh' (\ln p_j \cdot x_k(j)),$$  (8)

$$\delta x_h(j) = \frac{\partial E}{\partial x_h(j)} = \frac{1}{\ln p_j} \tanh' (\ln p_j \cdot x_k(j)).$$  (9)

These signals can be calculated in turn. This is nothing but the backpropagation. The gradient ∂E/∂p can be obtained using the above signals. For p in the kth output neuron,

$$\frac{\partial E}{\partial p_k} = \frac{1}{p_k \ln p_k} \delta o_o(k) \left( \frac{d}{dp_k} \frac{1}{\ln p_k} \cdot \tanh (\ln p_k \cdot x_o(k)) + \frac{1}{\ln p_k} \cdot \tanh' (\ln p_k \cdot x_o(k)) \frac{d}{dp_k} \ln p_k \cdot x_o(k) \right) - \frac{1}{p_k} \frac{1}{\ln p_k} \{\delta o_o(k) \cdot o_o(k) - \delta o_o(k) \cdot x_o(k)\}.$$  (10)

For p in the hidden neurons, the gradients can be calculated as,

$$\frac{\partial E}{\partial p_j} = \frac{1}{p_j \ln p_j} \left( \delta o_h(j) \cdot o_h(j) - \delta o_h(j) \cdot x_h(j) \right).$$  (11)

Then the parameter p can be updated using (6). Of course, all the weights can be updated using the above error signals, too.

3. The Uniqueness of Parameters

Regarding our aim the tuned parameters of the neurons are not only the passageway from inputs to outputs, but also they are indicators which are expected to show the same values in attempts to identifying the same system. This requires determination of the parameters uniquely through tuning or learning so that we can always say that ‘if p=1 then neuron is purely linear’ and if p>1 then neuron is nonlinear’.

3.1 Interdependent Weights

The neural network with the parametric sigmoid function can be represented in another but equivalent way. We can break the parametric sigmoid function, f_p, into an ordinary sigmoid function f, and two dependent weights ln p and 1/ln p. In other words, f_p contains parts of weights in it. This gives us an
alternative drawing of the network as shown in Fig. 3. The dependent weights are drawn as lozenges linked by a dash line. This figure helps us to compare our network with an ordinary neural network. We call this representation as a neural network with partially interdependent weights.

3.2 Uniqueness Analysis

Let us first examine whether the weights in an ordinary NN can be determined uniquely. The NN shown in Fig. 2 can be regarded as such an ordinary NN if \( f_p \) is replaced with ordinary sigmoid function \( f \) and if \( U_k \) is not fixed to 1.

If the output signals of all the neurons in the same layer are linearly independent each other, the weights on those signals can be determined uniquely. This condition is satisfied if sufficient number of independent teaching signals are available and if there is no redundant neuron. Comparing our proposed NN in Fig. 3 with that in Fig. 2 will show the number of parameters which can be determined uniquely.

We should be able to find that the new network has larger number of parameters than the old network. More specifically, the network in Fig. 2 has \( n_i n_h + n_h n_o \) parameters, while that in Fig. 3 has \( n_i n_h + 2n_h + n_h n_o + 2n_o \). Equating the corresponding parameters, we obtain the following relationships:

\[
\ln p_k = \frac{1}{U_k}; \quad k = 1, \ldots, n_o, \quad (13)
\]

\[
c_k = \frac{C_k}{\ln p_k}; \quad k = 1, \ldots, n_o, \quad (14)
\]

\[
\frac{U_k}{\ln p_k} = \frac{W_{kj}}{\ln p_j} = W_{kj}U_j; \quad k = 1, \ldots, n_o; \quad j = 1, \ldots, n_h, \quad (15)
\]

\[
b_j \ln p_j = B_j; \quad j = 1, \ldots, n_h, \quad (16)
\]

\[
\ln p_i = V_i; \quad j = 1, \ldots, n_h; \quad i = 1, \ldots, n_i. \quad (17)
\]

Once the parameters \( U_k, C_k, W_{kj}, B_j, \) and \( V_i \) in Fig. 2 are determined, we can specify the value of \( \ln p_k \) and \( c_k \) uniquely but can not for \( w_{kj}, \ln p_i, b_j \) and \( v_i \). This indicates that the neural network with interdependent weights is overparametrized. Thus, its parameters can not be determined uniquely. In this relation one suggestion can be setting some of the weights to fixed values. Here we choose all of the 1\( j \)th weights of the hidden-output connections to one:

\[
w_{1j} = 1; \quad j = 1, \ldots, n_h, \quad (18)
\]

Now the number of the unknown parameters is reduced to \( n_i + n_o n_h + n_h n_i \). By this procedure we are able to construct a unique network configuration for the system under examine.

4. Identification by Parametric Network

There are two approaches in treating nonlinear systems: The first one is to restrict the systems to particular set of nonlinear systems which have some favorable characteristics, that is a set of systems each of which consists of a linear (dominant) part and a nonlinear (additional) part; and the second one is to represent the systems in a general manner. The first approach is convenient to treat in controller design, etc., but does not cover a wide class of nonlinear systems. The second approach covers a wider class but results in complex representations of the systems which raises nuisance in analysis and controller design. We adopt approach just between the above two: we use a general representation tool of nonlinear systems, a neural network that can tell
Regarding our aims (see sec. 2.1) to use parametric network, the first task is to show how this network can reduce the error bound and/or learning time. Seeking this point we apply both of an ordinary network and our proposed network to identification of a system and show what is the probable advantage of the parametric NN. The system under study is:

\[ y(t) = 0.3\sin x(t) + 0.4y(t-1) + 0.5y(t-2), \]

(19)

where \( x(t) \) is random input varying between \([-1, 1]\).

Results of the simulations are indicated in Fig. 4. The upper curve shows MSE (Mean Squared Error) by ordinary back propagation, the middle is obtained by using Yamada’s activation function which uses \( p \) instead of \( \ln p \). The lower one is the MSE by using natural logarithm \( \ln p \). The results show the superiority of the proposed method by better network output or faster decreasing path of error with respect to both ordinary backpropagation and using only \( p \) for tuning the shape of sigmoid function instead of \( \ln p \).

The result shown in Fig. 4 can be explained as follows. Introducing an additional parameter into the ordinary sigmoid functions enhances the function’s flexibility and leads to smaller learning error. However, the parametric sigmoid function with large \( p \) is similar to a step function, and its derivative or gradient is almost zero everywhere except for in the vicinity of the origin. Thus, if the parameter \( p \) becomes large in an early stage of the learning, the learning can not be effective any more. The proposed function with \( \ln p (p \geq 1) \) can prevent this to some extent; its derivative is equal to \( 1/p \leq 1 \) times of that of Yamada’s function, which means the learning rate of \( p \) is reduced if \( p \) is getting larger.

### 4.2 Linear System

We intend to show what is the reaction of the trained parametric network to the pure linear system. For this purpose let us begin with:

\[ y(t) = 0.3x(t) + 0.4y(t-1) + 0.5y(t-2), \]

(20)

This is a linear dynamical system. For modeling this system we have selected a 3-layer network with six neurons in the hidden layer. Table 1 shows how the parameters, \( p \), can provide more information about the corresponding system; as we have expected, the parameters, \( p \), are near one and this is in harmony with the linearity of the system under consideration.

### 4.3 Mixed Linear and Non Linear System

We intend to study the partly linear and partly nonlinear system such as:

\[ y(t) = 0.3x(t) + 0.4y(t-1) + 0.5y^2(t-2), \]

(21)

Figure 5 shows the MSE. System’s corresponding network Fig. 6 will reveal some details of the system.
under examine, where linear neurons with $\beta=1$ are drawn as blank ellipses and nonlinear neurons as shaded ellipses. This figure teaches us which part of the NN is linear and which part is nonlinear. Here the first, the third and the fourth hidden neurons make up the hidden nonlinear part and the rest belong to the linear part.

5. Conclusion

In this article we had proposed another shape of sigmoid function instead of ordinary sigmoid function. In this new shape we could give more flexibility to neurons' functions to adapt for different cases.

When we had a dynamical nonlinear system we could teach the network to follow the teaching signal better.

Another important point in this method is: by looking into the parameter $\beta$ of the neurons we can recognize which part of the network is linear and which part is nonlinear. This information will be useful in making suitable and accurate model and suitable controller.

References

1) W. S. McCulloch and W. Pitts: A Logical Calculus of Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics, 9, 127/147 (1943)

[Authors' Profiles]

Mahmoud HashemiNejad

Mahmoud HashemiNejad was born in Tehran, Iran in 1964. He received the B. S. from Univ. of Chamran in 1988. He worked for Sairan Co. as a control designer during 1988-1990. Since 1990 he has been studying in the department of Electrical Engineering, Kyushu University, Japan, where he is now working toward the Ph. D. degree. His research has been focused primarily on neural networks, with the special emphasis on identification and control.

Junichi Murata (Member)

Junichi MURATA graduated from Kyushu University in 1981. He received Master of Engineering and Doctor of Engineering degrees from Kyushu University in 1983 and 1986, respectively. He then became a Research Associate at Faculty of Engineering, Kyushu University and now is an Associate Professor. His current research interests are neural networks and self-organizing systems and their applications to control and identification.
Kotaro Hirasawa (Member)

He was born on 26, January, 1942. He graduated from Kyushu University in 1964, and received the M.S. degree in electrical engineering from the same university in 1966. From April 1966 to July 1992, he served in Hitatch Lab., and from August 1992 to November 1992 he served in Omika Factory of Hitatch. Since December 1992, he has been a professor in faculty of engineering, Kyushu University. He is a member of SICE, a member of IPSJ, and a member of IEEE.

Setsuo Sagara (Member)

He was born on 5 May 1928. He graduated from Kyushu University, Japan, in 1953, and received the Ph. D. degree in electrical engineering from the same university in 1962. He has been a Lecturer, Associate Professor and Professor in electrical engineering at Hiroshima University from 1956 to 1968, Professor in electrical engineering at Kyushu University from 1968 to 1992, and Professor in communication and computer engineering at Fukuoka Institute of Technology since 1992. His main interests are system identification, adaptive control and prediction in electrical power systems. Dr. Sagara is a member of the Institute of Electrical Engineers of Japan, the Institute of Systems, Control and Information Engineers of Japan, and a fellow of the Society of Instrumental and Control Engineers of Japan.