Chapter 4

System Identification with Neural Networks

The use of multilayer feedforward networks for the purpose of system identification is discussed. The identification consisting of forward modeling, which the forward model of the system is identified and inverse modeling, which the inverse model of the system is identified is provided in section 4.2 and the identification steps are described in the last section.

4.1 Introduction

In sampled-data control systems, a multivariable process can be generally represented by,

$$Y_k = f(Y_{k-1}, Y_{k-2}, Y_{k-3}, \ldots, Y_{k-m} : X_{k-1}, X_{k-2}, X_{k-3}, \ldots, X_{k-n}) \qquad (4.1)$$

where Y_k and X_k represent the vectors of outputs and inputs, the subscript k refers to the sampling instant, and f represents the functional relation between the inputs and outputs. In classical identification studies, a form of the function f representing the input-output behavior has to be specified and the parameters in that form have to be determined. Usually, several assumptions are made about the process, such as linearity, time-invariance, etc., and a separate noise model is often included to represent the unmodeled dynamics. But many important industrial chemical processes are nonlinear in nature and it is often not possible to specify the exact functional relationship between the inputs and the outputs.

Neural networks have been found to be extremely useful in this context. They do not require that a function be specified explicitly. Only the topology or the

structure of the network needs to be specified. The specification includes the number of neurons in the input layer, hidden layer, and the output layer.

4.2 Identification

The input and output of a time-invariant, causal discrete-time dynamical plant are $u(\cdot)$ and $y_p(\cdot)$ respectively, where $u(\cdot)$ is a uniformly bounded function of time. The plant is assumed to be stable with a known parameterization but with unknown values of the parameters. The objective is to construct a suitable identification model (Figure 4.1) which when subjected to the same input u as the plant, produces an output y^m which approximates y^p in a certain sense. The model will be used in the form of a neural network.

4.2.1 Forward Modeling

The procedure of training a neural network to represent the forward dynamics (i.e. obtain outputs given the inputs) of a system is referred to as forward modeling.

A few approaches can be utilized to model the forward dynamics using neural networks. These include using global recurrent networks or by introducing dynamic behavior into the neurons (local recurrent networks) (Su and McAvoy, 1992). But the most popular and straight forward approach is to augment the network inputs with corresponding discrete past input and past output data signals from the model or system being identified as seen in Figure 4.2.

The neural network model is placed in parallel with system and the error between the system and the network outputs (the prediction error) as the neural network training signal. A multilayer feedforward network is used in order to apply a backpropagation training algorithm.

Assume that the plant is governed by the following nonlinear discrete time difference equation:

$$y^p(t+1) = F\left[y^p(t),...,y^p(t-n+1);u(t),...,u(t-m+1)\right] \quad (4.2)$$

Thus, the plant output y^p at time $t+1$ depends on the past n output values and on the past m values of the input u. What is concentrated here is only on the dynamical part of the plant response; the model does not explicitly represent plant disturbances (for a method of including the disturbance see, e.g., Chen et al. (1990)). Special cases of the model (Eq. 4.2) have been considered by Narendra and Parthasarathy (1990).

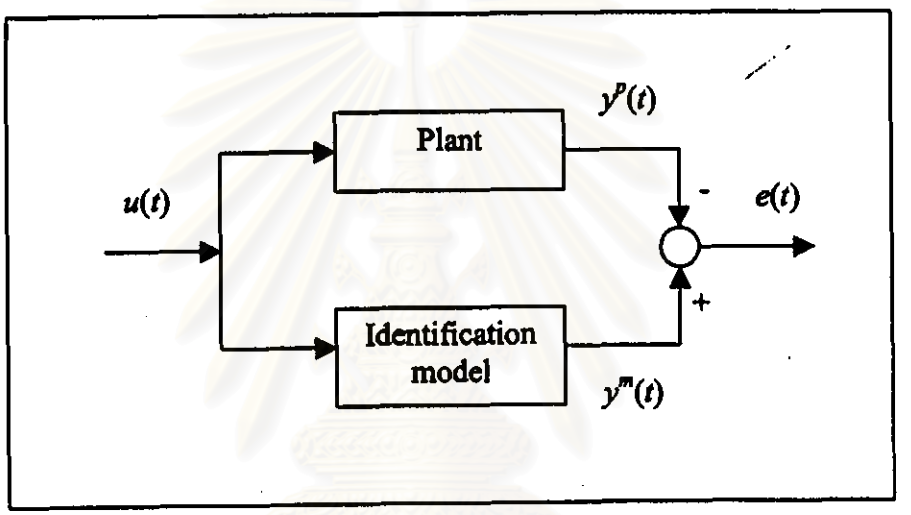

Figure 4.1: Identification.

An obvious approach for system modeling is to choose the input-output structure of the neural network to be the same as that of the system. Denoting the output of the network by y^n then it is obtained that

$$y^n(t+1) = \hat{F}\left[y^p(t),...,y^p(t-n+1);u(t),...,u(t-m+1)\right] \quad (4.3)$$

In the above, the mapping $\hat{F}(\cdot)$ represents the nonlinear input-output map of the network which approximates the plant mapping $F(\cdot)$. Note that the input to the network includes the past values of the plant output but not the past values of the network output (the network has no feedback). The learning statistical backpropagation algorithm is used to find the optimal values of the network weights. The structure of the model Eq. (4.3) is called series-parallel. The resulting

identification structure is illustrated in Figure 4.2. Note that Z^{-1} indicates a delay of one time unit.

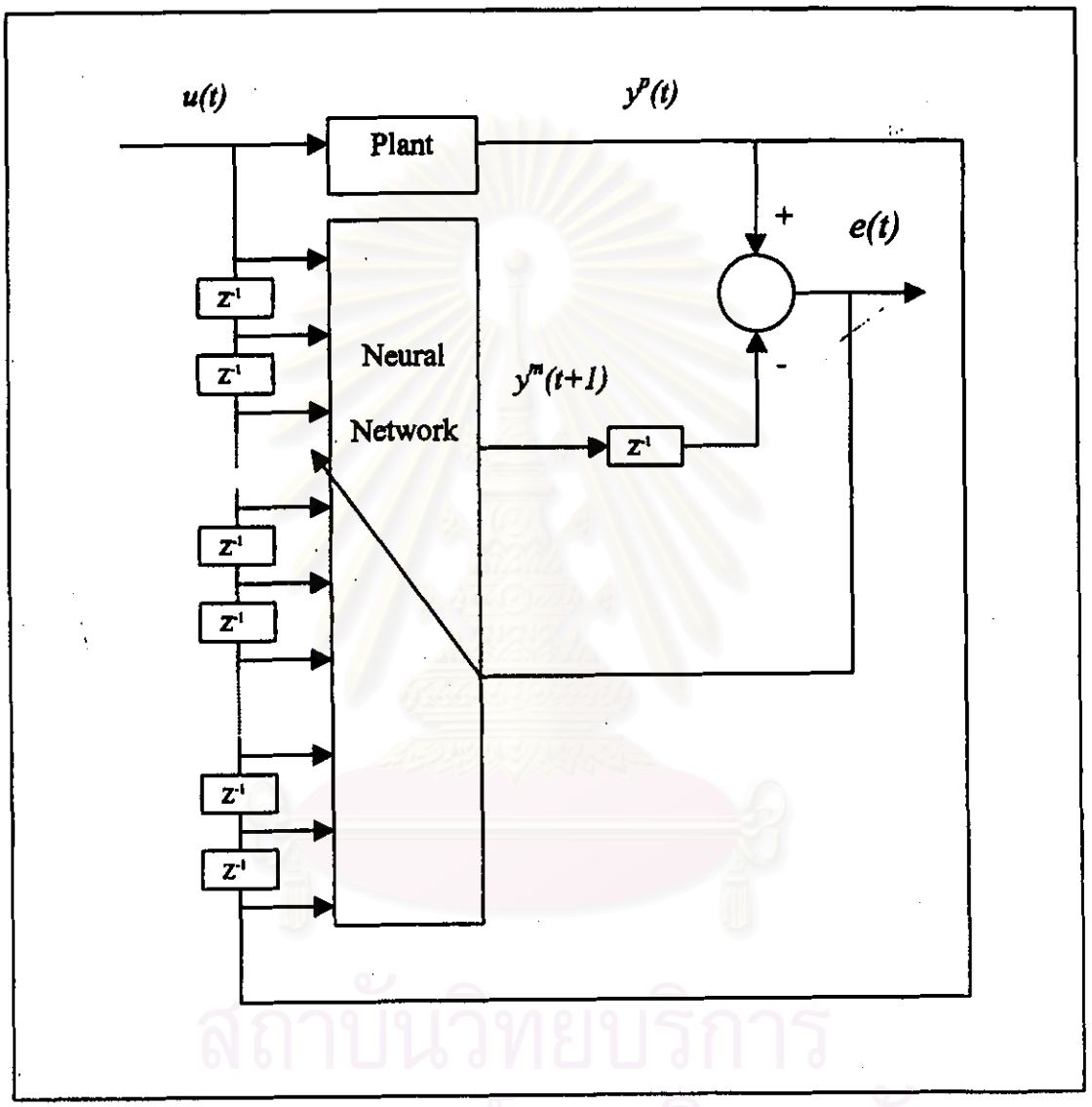

Figure 4.2: Series-parallel identification structure.

If it is assumed that after a suitable training period the network gives a good representation of the plant (i.e. $y^m \approx y^p$), then for subsequent post-training purposes the network output itself and its delayed values can be fed back and used as part of the network input. In this way the network can be used independently of the plant. Such a network is described as

$$y^m(t+1) = \hat{F}\left[y^m(t),....,y^m(t-n+1);u(t),...,u(t-m+1)\right] \qquad (4.4)$$

This structure may also be used from beginning that is during the whole process of learning. The structure of Eq. 4.4 is called parallel. Figure 4.3 depicts the parallel identification structure. It may be preferred when dealing with noisy systems since it avoids the problem of bias caused by noise on the plant output. On the other hand the series-parallel scheme (see Figure 4.2) is supported by stability results. Moreover, the parallel model requires a dynamical backpropagation training algorithm.

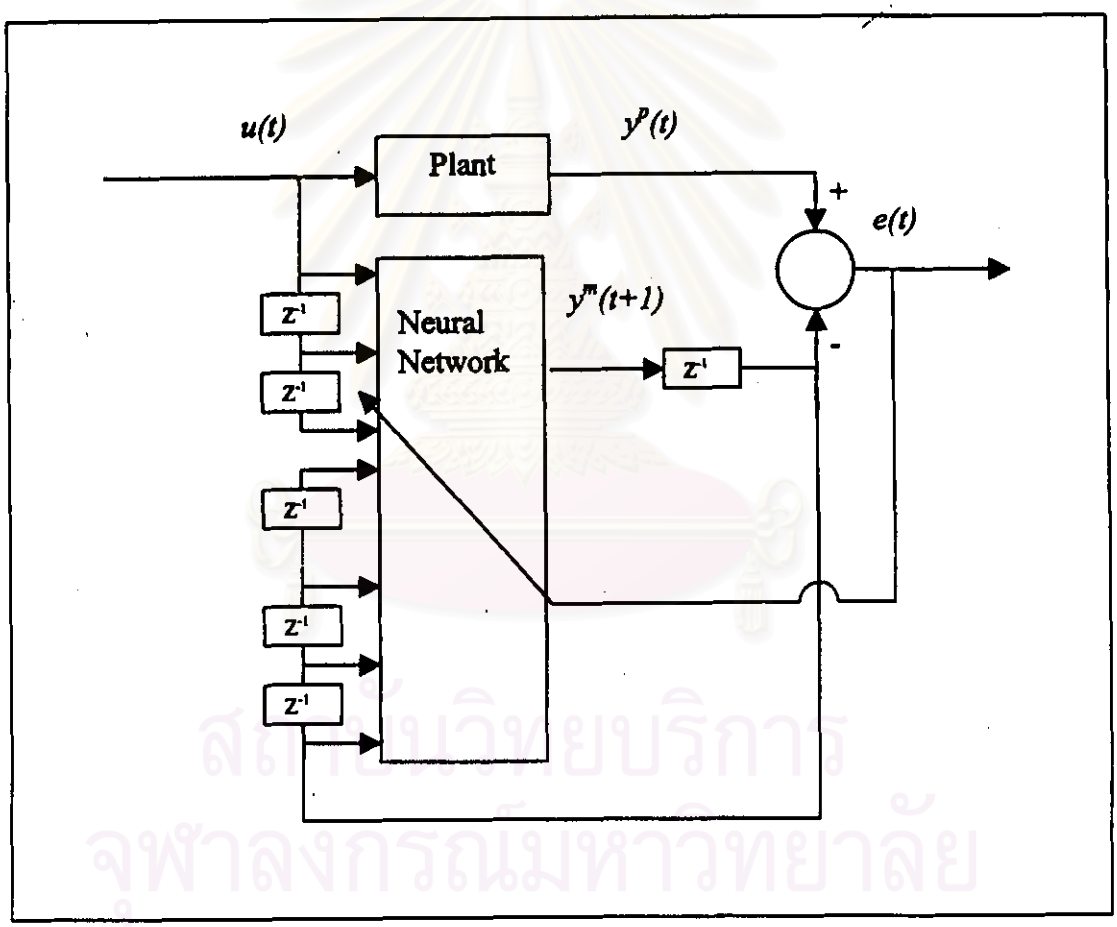

Figure 4.3: Parallel identification structure.

4.2.2 Inverse Modeling

The inverse model of dynamical system yields input for given output. The models play a crucial role in a range of control structures. However, obtaining inverse models

raises several important issues. Conceptually the simplest approach is *direct inverse modeling* as shown in Figure 4.4. Here, a synthetic training signal (the plant input) is introduced to the system. The plant output is then used as input to the network. The network output is compared with the training signal (the system input) and this error is used to train the network. This structure will clearly force the network to represent the inverse of the plant. However, there are some drawbacks:

- the learning procedure is not "goal directed"; the training signal must be chosen to sample over a wide range of system inputs, and the actual operational inputs may be hard to define a *prior*. The actual goal in the control context is to make the system output behave in a desired way, and thus the training signal in direct inverse modeling does not correspond to the explicit goal;
- if the nonlinear system is not one-to-one, then an incorrect inverse can be produced.

The first point is strongly related with the general concept of persistent excitation.

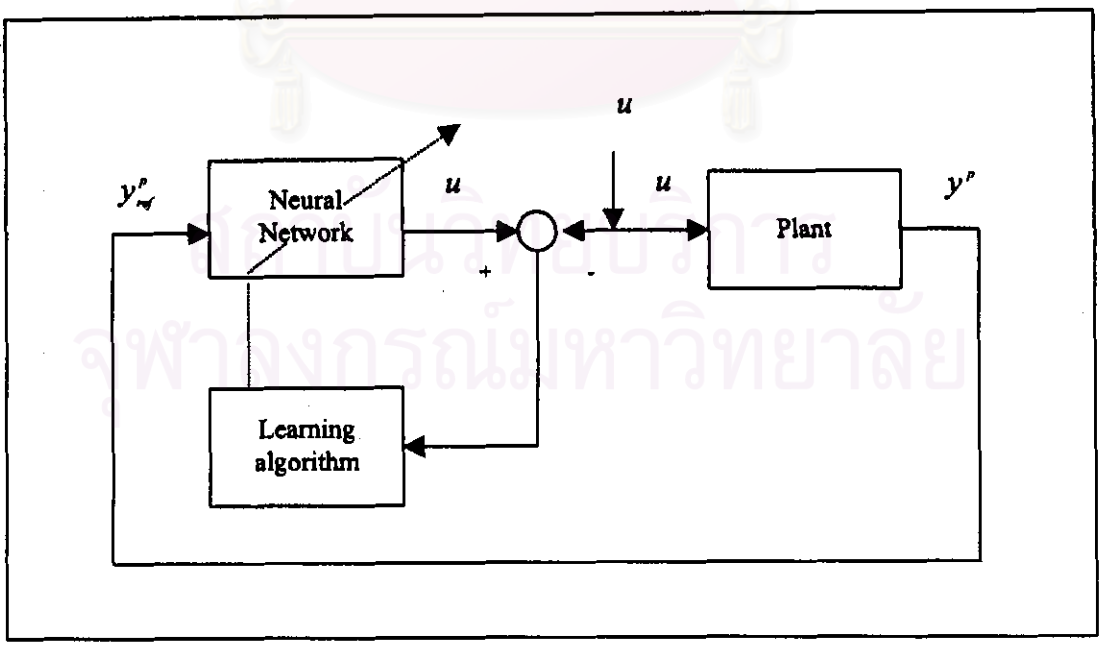

Figure 4.4: Direct inverse modeling.

A second approach to inverse modeling which aims to overcome these problems is known as specialized inverse learning (Psaltis, Sideris and Yamamura, 1988). The specialized inverse learning structure is shown in Figure 4.5. In this approach the network inverse model precedes the system and receives as input a training signal which spans the desired operational output space of the controlled system (i.e. it corresponds to the system reference signal). This learning structure also contains a train forward model of the system (e.g. a network trained as described in the section 4.2) placed in parallel with the plant. The error signal for the training algorithm in this case is the difference between the training signal and the system output (it may also be the difference between the training signal and the forward model output if the system is noisy). It can be shown that using the plant output an exact inverse even when the forward model is not exact can be produced; this is not the case when the forward model output is used. The error may then be propagated back through the forward model and the inverse model; only the inverse network model weights are adjusted during this procedure. Thus, the procedure is effective at learning and identifies mapping across the inverse model and the forward model; the inverse model is learned as a side effect. In comparison with direct inverse modeling, the specialized inverse learning approach possesses the following features:

- The procedure is goal directed since it is based on the error between desired system outputs and actual outputs. In other word, the system receives inputs during training which correspond to the actual operational inputs it will subsequently receive.

- In case in which the system forward mapping is not one-to-one a particular inverse (pseudo-inverse) will be found. The problem of bias can also be handled.

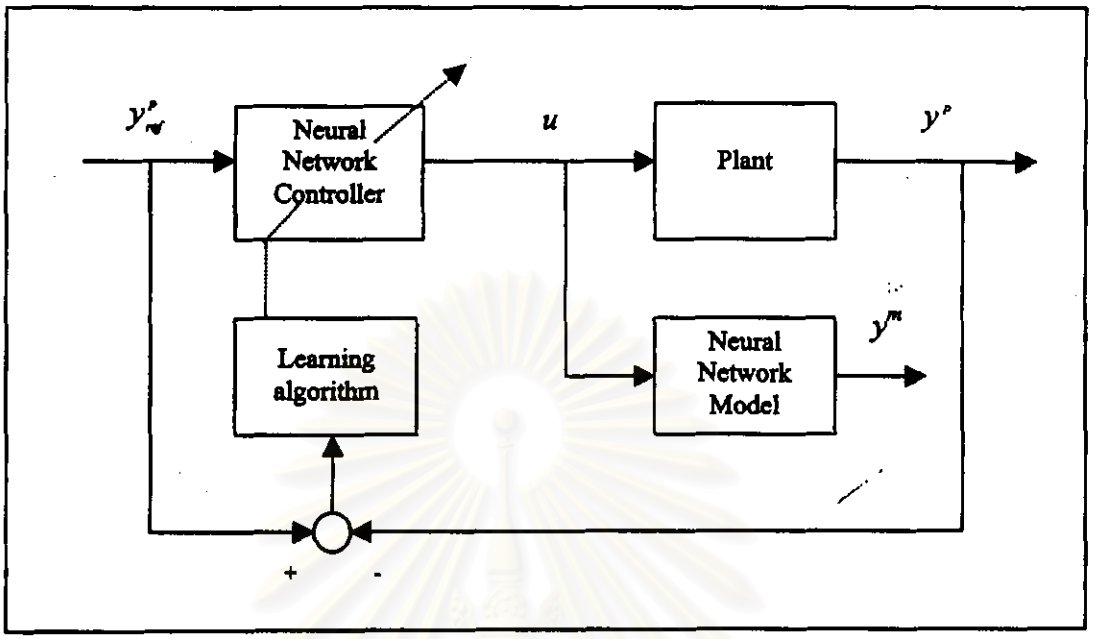

Figure 4.5: Specialized inverse modeling.

Next, the input-output structure of network modeling the system inverse is considered. From Eq. 4.2 the inverse F^{-1} leading to the generation of $u(t)$ would require knowledge of the future value $y^p(t+1)$. To overcome this problem we replace this future value with the value $y^p_{ref}(t+1)$ which is assumed to be available at time t. This seems to be a reasonable assumption since $y^p_{ref}(t+1)$ is typically related to the reference signal which is normally known one step ahead. Thus, the nonlinear input-output mapping relation of the network modeling the plant inverse is

$$u(t) = F^{-1}(y^p(t),....,y^p(t-n+1); y^p_{ref}(t+1); u(t-1),....,u(t-m+1)) \quad (4.5)$$

that is the inverse model network receives as inputs the current and past system outputs, the training (reference) signal, and the past values of the system outputs. Where it is desirable to train the inverse without the plant the values of y^p in the above relation are simply replaced by the forward model outputs y^m.

4.3 System Identification Steps

As in the other conventional techniques, various steps and criteria for choosing the most appropriate neural network model for any given set of input and output data has to be followed. These steps play a major role in obtaining the best possible model for any particular application. These basic steps and criteria, which applies for both forward and inverse modeling approaches, are as follows:

4.3.1 Model Structure and Size

No standard method has been known to determine the structure and the number of nodes of a network required for any particular application. Although there are some guidelines and heuristics suggested in the literature the actual choice still remains on a case-to-case basis. In this work the feedforward fully connected structure with one hidden layer, in all cases they are adequate was utilized. The normal procedure for selecting the hidden nodes is to fix an initial size and then check if this model satisfies the error requirement when the identification process is stopped. If not, the size is revised and the whole procedure repeated until it satisfies the tolerance for the prediction error. Although the choice of the number of hidden nodes here is done by trial-and-error, normally within a few trials it becomes quite easy to constrain it in an optimum range (within some upper and lower limit) required for achieving acceptable training. The choice is also made keeping in view of one of the objective of this work which is to select parsimonious models i.e. models which contain the smallest number of free parameters such as the connection weights, required to represent the time system adequately.

4.3.2 Data Set

Data set collection either from available model simulation or actual on-line data is a fundamental step in all identification procedure. In utilizing neural networks, the data set is normally split into various sets. One is the initial training set, which is the data used to train the network weights and normally span the operating region of the

model. Next is the cross validation data set, which is used to assess the generalization capability of the network. This set is normally of similar quality to the training data set. Training can also be switched between these two sets from time to time to improve on its identification process. Finally a testing data set is needed, which is new, unseen data set which is used for final validation of the trained neural network.

4.3.3 Input Excitation

Secondly to obtain an adequate model, it is important to train the network with input signals which are representative of the types of signals anticipated during normal operation. Furthermore for nonlinear identification, the input excitation signal is not only required to be of a particular frequency but also to be appropriate magnitude so that the nonlinear nature of the system can be observed. Some researches have found that the random step input sequence is more suitable than the Pseudo-Random-Binary-Sequence (PRBS) input signal for identification of nonlinear systems, in particular for neural net modeling (Scott and Ray, 1993). This is mainly due to the fact that the PRBS applies only two input amplitudes to the system and usually is not enough to cover the range of important input magnitudes. It has been suggested that the input be a pseudo-random signal with random amplitudes, whose range covers the important values of the input signals. In this study large and small step range with random amplitude signals are used for the training and cross validation, respectively. A ramp-input signal is used for testing since it is entirely of different characteristics from these signals.

4.3.4 Input and Output Data

The choice of input data fed into the network is an important consideration in the utilization of neural networks for any particular application. For steady state application, the choice of inputs to the network basically depends on the relevant variables likely to have an effect on the predicted output variable. For modeling the dynamic behavior of a system, it would not only depend on these relevant variables but also the time history of these variables as well as the time history of the output

variables. In this work we choose the inputs based on the relevant input and state variables as well as their time history which might have an effect on the output. The knowledge of the system such as the model order is use as the initial guide to decide on the time history. Furthermore all variables are measured in different units having different magnitudes and normally variables having a larger magnitude are given unequal importance due to the nature of the weight update or optimization procedure. Hence these variables are scaled between some upper and lower bound to give appropriate weighting to all the variables. The outputs are anyway bounded due to the nature of the activation functions utilized.

4.3.5 Weight Initialization

It is well known that initial weight specification has a pronounced effect on the speed and quality of neural network training. It is best to initialize the weights with small, random numbers e.g. in the range –0.5 to 0.5 (Bhat and McAvoy, 1990) so that each connection responds slightly differently during training and has the effect of breaking the symmetry and promotes faster convergence to the global minimum. In the identification step followed in this work, if the final prediction does not satisfy the error tolerance during training, other than reconfiguring the network, the weights are also re-initialized and the identification process repeated. This has been found to improve the performance of the neural network training.

4.3.6 Training Methodology

Training is a procedure to determine the optimal values of the connection weights and bias weights. It begins by initially assigning arbitrary small random values (both positive and negative) to the weights. Training proceeds iteratively until a satisfactory model is obtained. In each iteration, called an epoch, the actual outputs corresponding to all the sets of inputs in the training set are predicted, and the weights are adjusted in the direction in which the output prediction error decreases. For training to be complete many iterations are necessary. The weights are incrementally adjusted for every pattern in every iteration and they gradually converge on the optimal values. If

n_t is the total number of data patterns in the X, Y data set, where X is the matrix of inputs and Y is the matrix of targets, then one iteration corresponds to feeding all the n_t patterns once. Actual outputs are not available for the hidden units. Therefore, to adjust the hidden layer weights, error from the output layer is propagated back to the hidden layer, and their weights adjusted to decrease the prediction error.

Different network architectures require different training algorithms and training times can be significantly reduced by the use of suitable algorithms. However backpropagation with its variants remain the mainstay of performing neural network (multilayer feedforward) learning. Hence training or optimization of the weights to achieve the required prediction, is performed in this work by the backpropagation technique with a momentum term. Although there are many other variants of this method to improve the speed of training, this approach is deemed sufficient to produce the required results and accuracy for our application in this work.

4.3.7 Model Validation

Overlearning, which occurs when the model starts to learn the presented pattern in a pointwise fashion instead of learning the functionality, is a potential problem that can easily occur in process identification. During overlearning the performance of the network training continues to improve on the learning data set but starts to degrade on the testing set i.e. poor generalization capability at this instance. It can however be dealt with by proper training and validation.

Most of the quantitative validation tests available are based on correlation approaches, for linear systems, intended to check whether the residuals are correlated to the input signal or among themselves (autocorrelated). Other information criteria methods such as Akaike Information Criteria, Final Prediction Error and Bayesian Information Criteria etc., (Ljung, 1987; Cherkassky, Gehring, and Mulier, 1995) attempt to measure how well a model fits the data set provided as well as penalizing complex models by accounting for the number of parameters in the model. However the model can also be validated, as in mainly done in this study, by predicting the

output in data sets not used in the identification procedure and the quality of the fit can be observed in terms of its sum-squared error.

The major steps required to be followed in performing neural network based systems identification are outlined in the chart of Figure 4.6. The use of the multilayer feedforward networks for system identification, function approximation, and advanced control is demonstrated in the Chapter 5, 6 and 7, respectively.

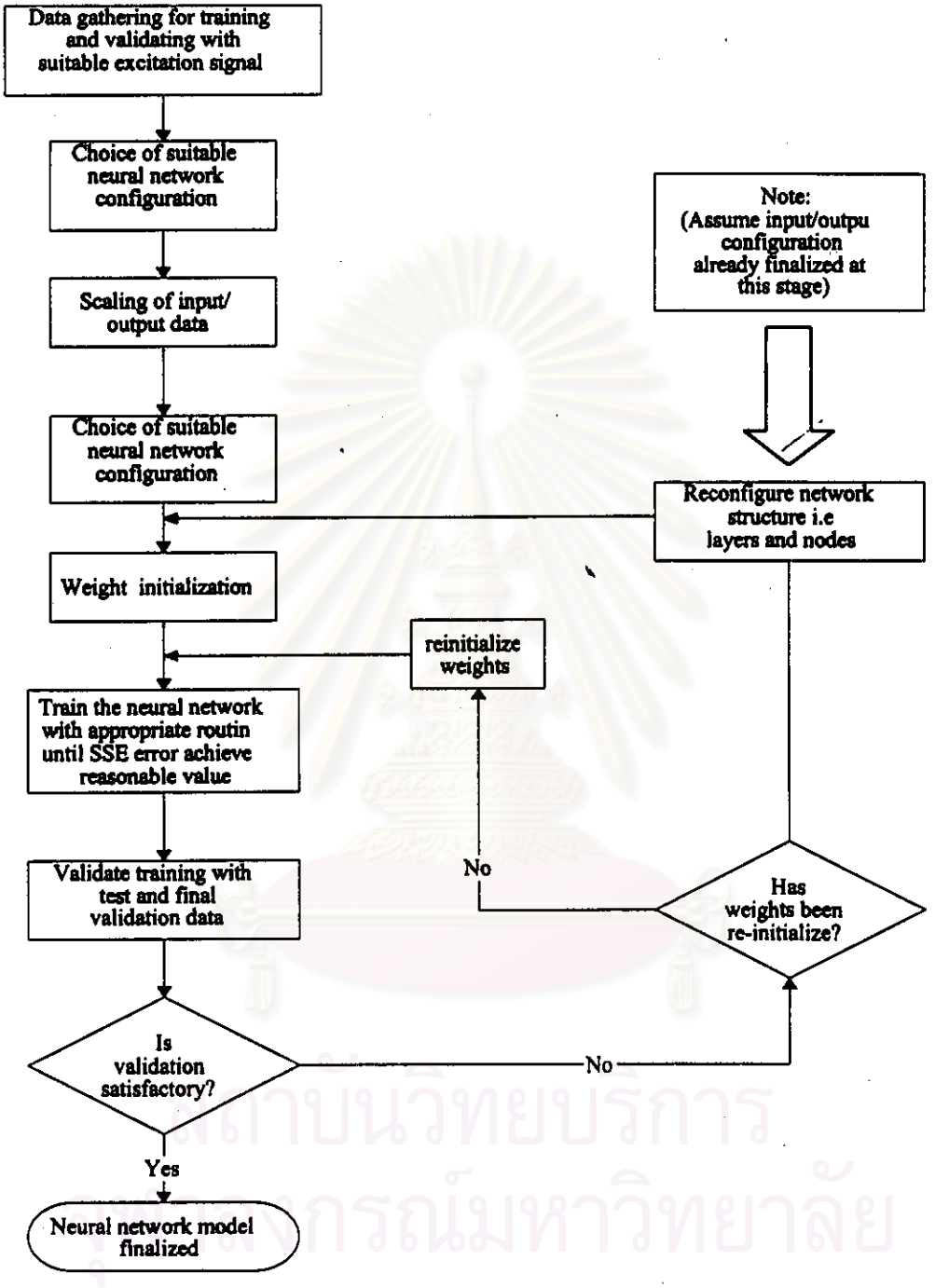

Figure 4.6: Basic steps - Neural network system identification.