Chapter 4

Neuro-Fuzzy Model

4-1 Introduction

Control theory deals with the analysis and synthesis of dynamical systems in which one or more variables are kept within prescribed limits. Many real world applications need to describe models for unknown systems [Narendra'90]. In the past few decades, system modeling and identification attracted the attention of a considerable number of researchers [Narendra'90, Qin'92, Mastorocostas'02, Xu'87, Sugeno'93, Takagi'85, Azeem'OOb, Lee'00], the reason is its extensive application in practical life.

System identification plays a principal role in Input-Output data analysis, such that a better result can be obtained from better model. System identification includes two parts: structure identification and parameter identification. In structure identification, input variables and input-output relations are found. In parameter identification, the parameters of the model are adjusted by optimizing a performance index [Narendra'90, Sugeno'93].

The Parallel (P) and the Series-Parallel (S-P) configurations are the two common methods to identify parameters for the unknown model of dynamic systems [Narendra'90, Qin'92- Bernieri'94]. In Series-Parallel configuration, the output of the system (plant) is fed into
the model. Since there is no feedback of the model output to itself, a static learning algorithm is applied. In this configuration, the parameter learning will converge if the outputs are bounded for bounded inputs [Narendra'90].

In the Parallel configuration, the output of the model is feedback as inputs to the model. Identification using parallel configuration, the model feedback introduces dynamics to the model; but it can learn the system dynamics without assuming much knowledge about the structure of the system under consideration [Qin'92]. This model is suitable for long-term and multi-step prediction in forecasting problem. When information about system is less, this configuration is better; however, the learning convergence is not guaranteed [Narendra'90]. Since the output of the model can be carried out on-line, the Parallel configuration can be used for on-line learning approach [Bernieri'94].

Recently fuzzy system identification has attracted the researchers involved with systems modeling [Jang'93, Sugeno'93, Yager'94, Gebhardt'94, Wu'00, Azeem'03a, Klir'03]. In describing the behavior of many complex and ill-defined systems, precise mathematical models may fail to give satisfactory results. In such cases, fuzzy models are used to reflect the uncertainty of the systems in a proper way. Takagi and Sugeno introduced Takagi-Sugeno-Kang (TSK) fuzzy model [Takagi'85, Sugeno'88]. The basic idea in this approach is to decompose the complicated input space into subspaces and then approximate the system in each subspace by a linear/non-linear regression model called local model. The resulting fuzzy model is the aggregation of these local models. Later Shing and Jang proposed Adaptive Neuro-Fuzzy Inference System (ANFIS) as a powerful method for mapping input-output system modeling based on fuzzy inference system [Jang'93, Nauck'97].

In these application models, it is possible to use both parallel and parallel-series
configuration for estimation of unknown parameters of the model. The present work proposes an implementation of combining parallel and series-parallel configuration on TSK fuzzy model. It has advantages over both parallel and series-parallel configuration. Premise and consequent part of the rules in TSK models are learned by parallel and series-parallel configuration respectively, or vice versa. If the output of plant is feedback to premise part and output of the model feedback to consequent part, it results a Premise Series-Parallel (PS-P) configuration. In the same way if model output feedback to premise part and the plant output feedback to consequent part, we have a Consequent Series-Parallel (CS-P) configuration. Therefore in this way the advantage of both configuration, i.e. tracking the real output of the plant by series-parallel configuration and long-term or multi step prediction with less knowledge about the plant by parallel configuration are exploited together. Consequently, we obtain the best model that follow real output for long time prediction with less knowledge about the plant.

A lot of learning algorithm has been developed for recurrent models. Two specific algorithm that are based on GD are Back-Propagation Through Time (BPTT) [Rumelhart'86-Werbos'88] and Real Time Recurrent Learning (RTRL) [Williams'89]. However, these algorithms have two main problems: stability and slow rate of the convergence during learning procedure. The problem is that for stability, learning rate should be small but when that is small the speed of the convergence become low. To eradicate these two discrepancies large number of studied has been done for improving the speed of convergence [Wu'00, Yu'95a, Barbounis'06] in addition to incorporating the stability [Yu'01a, Yu'01b, Chen'94, Wang'97, Yu'95b, Li'06, Jin'99, Yi'06] to the parameter learning procedure. In [Yu'01b], the passivity theory has been applied to analyze the stability of the dynamic neural network for identification problem. Yi, etal., [Yi'06] carried out a comparative study for output convergence [Yi'01, Liu'04, Li'04] and
the state convergence \cite{Cao03a, Cao03b, Forti94, Forti95, Liang01} of a recurrent neural network. Yu, et al. \cite{Yu95b} have shown that the neuro-fuzzy model under certain condition is stable by applying the Lyapunov stability theorem and passivity theory. However, they have not ascertained any boundary for learning parameter. Based on Lyapunov-Krasovskii functional method, Li and Liao \cite{Li06} have proposed a robust learning algorithm for recurrent neural network under noise disturbance while Chen and Jain \cite{Chen94} have proposed a robust BP algorithm and shown that by improving the learning rate the algorithm is stable under noise effect. Wang et al. \cite{Wang97} have introduced a robust and fast learning algorithm for B-spline membership function using robust objective function and gradient descent method. Using Lyapunov stability theorem a mathematical way to calculate the upper bound of the learning rate for recurrent wavelet neural network \cite{Yoo06} and a mamdani fuzzy model \cite{Lee00}, based on the parameter of the network, have been introduced, respectively. Azeem, et al., \cite{Azeem03a} used an easy and understandable way for adaptive learning rate to increase speed of convergence rate. In this chapter, presented studies guarantee the stability and the speed of the learning procedure by applying the Lyapunov stability theorem and the adaptive learning rate to the learning procedure of neuro-fuzzy model.

The chapter spread over six sections: Brief discussion about neuro-fuzzy model is given in section 4-2. In section 4-3, parameter identification configurations are devised. Section 4-4 deals with the learning algorithm and convergence analysis of S-W neuron models. Section 4-5 consists of simulation results and discussions. Finally, the conclusions are relegated to section 4-6.
4-2 Neuro-Fuzzy model

Each rule of a fuzzy model based on TSK fuzzy model mapped the input space \( A^n \subset R^n \) to a linear function in the output space \( W^m \subset R^m \), and has the form:

\[
R^m : \text{if } x_1 \text{ is } A_1^m \land x_2 \text{ is } A_2^m \land \ldots \land x_n \text{ is } A_n^m \text{ then } y \text{ is } W^m(x)
\]  
(4.1)

with \( m=1\ldots M, M \) being the number of rules. Each rule is premised on its own input vector \( x \in \mathbb{R}^n \), \( A_i^m \) is linguistic labels of fuzzy sets describing the qualitative nature of the input variable \( x_i \), \( \land \) and is a fuzzy conjunction operator (usually of T-norm).

The TSK model was introduced in \([\text{Takagi'85, Sugeno'88}]\) as a hybrid model, which integrates the fuzzy conditions in the input space with the functional relationships in the output space. TSK-model has a linear or nonlinear relationship of inputs \( W^m(x) \) in the output space.

Rules of TSK model are in the following form:

\[
R^m : \text{if } x \text{ is } A^m \text{ then } y \text{ is } W^m(x)
\]  
(4.2)

A linear form of \( W^m(x) \) in (4.1 & 4.2) is as follows:

\[
W^m(x) = W_0^m + W_1^m x_1 + \ldots + W_n^m x_n
\]  
(4.3)

where, \( W^m(x) \) defines a locally valid model on the support of the Cartesian product of fuzzy sets constituting the premise parts. The normalized firing strength for the normalized calculation or non-normalized firing strength for the non-normalized calculation is then multiplied with the output function \( W^m(x) \). By taking Gaussian membership function and equal number of fuzzy sets to rules with respect to the inputs, firing strength of rules (4.2) is written as:
where \( \bar{x}_m \) and \( \sigma_m \) are the center and the standard deviation of the Gaussian membership functions. Applying T-norm (product operator) of the membership functions of the premise parts of the rule and the weighted average gravity method for defuzzification, the output of the TSK model is defined as:

\[
\hat{y} = \frac{\sum_{m=1}^{M} \mu_{A^m}(x) \cdot w^m(x)}{\sum_{m=1}^{M} \mu_{A^m}(x)}
\]  

(4.5)

The functionally equivalent neuro-fuzzy model of TSK model is shown in Fig. 4.1. In the following description, \( u^j \) denotes the input to the \( j^{th} \) node in the \( l^{th} \) layer; \( O^l_j \) denotes the \( j^{th} \) node output in layer \( l \).

Layer 1: Nodes in layer 1 represent input variable. Every node accepts input values and transmits it to the next layer.

\[
O^1_m = u^1_m = x_i
\]  

(4.6)

Layer 2: Nodes in this layer represent the terms of the respective linguistic variables. Every node operates on incoming signal with Gaussian membership function expressed by (4.7). The parameters to be learned in this layer are \( \bar{x}_m \) and \( \sigma_m \). Corresponding to each rule the learning parameter are expressed in vector form as \( \bar{x}_m = \{\bar{x}_{m1}, \bar{x}_{m2}, \ldots, \bar{x}_{mn}\} \) and \( \sigma_m = \{\sigma_{m1}, \sigma_{m2}, \ldots, \sigma_{mn}\} \).

\[
\mu_{A^m}(x) = \prod_{i=1}^{n} \exp\left(-\left(\frac{x_i - \bar{x}_m}{\sigma_m}\right)^2\right)
\]

(4.4)
Layer 3: Each node in layer 3 represents a fuzzy rule. The output from the nodes in layer 2, specified for a fuzzy rule, is being input to the nodes, specified for that rule, in layer 3. The output of each node in layer 3 is the product of all inputs; it represents the firing strength of that rule. Thus, the firing strength of the $m^{th}$ rule is specified as (4.4, 4.8).

$$O_m^3 = \prod_i O_{mi}^2 = \mu_{A_m}(X)$$  \hspace{1cm} (4.8)
Layer 4: Nodes in layer 4 are called consequent nodes. Two inputs are applied to each node in this layer, namely the output from the layer 3 node and the output from its corresponding local model approximated by (4.3). The output of each node is the product of both input and given by (4.9). Where $X'$ is input for local model either from the system or from the model depending upon the configuration.

$$O_m^4 = y^m = w^m(X') \cdot O_m^3 = w^m(X') \cdot \mu_{a^n}(X) \quad (4.9)$$

Layer 5: Three nodes in this layer constitute the aggregation and defuzzification of fuzzy rules.

The output of all nodes from layer 4 is the input to the first node and its output is the sum of all inputs and expressed as (4.10). The output of all nodes from layer 3 is the input to the second node and its output is the sum of all inputs and expressed as (4.11). Inputs to the third node in this layer are the output from first and second node. The output of the third node in the ratio of these two inputs is given in (4.12).

$$a = \sum_{m=1}^{M} O_m^4 = \sum_{m=1}^{M} (w^m(X') \cdot \mu_{a^n}(X)) \quad (4.10)$$

$$b = \sum_{m=1}^{M} O_m^3 = \sum_{m=1}^{M} \mu_{a^n}(X) \quad (4.11)$$

$$O_j^5 = \hat{y}_{NP} = \frac{a}{b} \quad (4.12)$$

**4-3 Configurations for Parameter Identification**

The problem of identification consists of selecting a suitable model and algorithm for learning its parameter. In this section, two well-known parameter identification methods, series-
parallel and parallel configurations are discussed to optimize the learning parameter of the neuro-fuzzy model. Two new configurations, which are combination of series-parallel and parallel configurations, applicable to TSK model are proposed.

4-3.1 Parallel Configuration

Parallel configuration for system identification is shown in Fig. 4.2. A Linear/nonlinear dynamic system models may be represented by mapping from the input space to the output space, which we call as function approximation. To construct a neuro-fuzzy model for a Multi-Input and Single-Output (MISO) system using parallel configuration, consider a Non-linear Auto-Regressive Moving Average (NARMA) model representing a MISO system.

\[
\hat{y}(t) = f\left( u_1(t), ..., u_t(t-\tau_{il}), u_2(t), ..., u_r(t-\tau_{ir}),
\begin{array}{c}
\hat{y}(t-1), \\
\hat{y}(t-2), \\
\vdots \\
\hat{y}(t-\tau_o)
\end{array}
\right)
\]

where \( u_q; (q=1,...,r) \) and \( \hat{y} \) denote the inputs and model outputs respectively. \( \tau_{iq} \) and \( \tau_o \) are the corresponding delays. Function 'f' in (4.13) may be linear or non-linear. Here it is supposed to be a neuro-fuzzy model. The output of model in parallel configuration is a function of the past output of the model as well as input delays. The premises of the rules, which represent delays as well as the order of dynamic systems, for a NARMA model of a complex system, are denoted by:

\[
X = \{x_1, ..., x_{(\tau_{i1}+...+\tau_{ir}+r)}, x_{(\tau_{i2}+...+\tau_{ir}+r+1)}, ..., x_n\} \\
= \{u_1(t), ..., u_t(t-\tau_{il}), u_2(t), ..., u_r(t-\tau_{ir}), \hat{y}(t-1), ..., \hat{y}(t-\tau_o)\}
\]

where \( n = \tau_{i1} + ... + \tau_{ir} + \tau_o + r \)

In Parallel configuration, input to the consequent part is \( X' = X \).
4-3.2 Series-Parallel configuration

In the series-parallel configuration, output of the model is a function of the input delays and past values of the plant output as shown in Fig. 4.3. Plant is a system that should be identified with neuro-fuzzy model.
We shall assume that output of unknown model in series-parallel configuration, which should be identified is as follows:

\[
\hat{y}(t) = g\left( u_1(t), ..., u_q(t - \tau_q), u_2(t), ..., u_r(t - \tau_r), y(t-1), y(t-2), ..., y(t-\tau_o) \right)
\] (4.15)

Function ‘\( g \)’ in (4.15), represented by neuro-fuzzy model. Where \( u_q; (q=1, ..., r) \) and \( y \) denote the inputs and model outputs, respectively. \( \tau_q \) and \( \tau_o \) are the corresponding delays. The premise inputs in this configuration are denoted by:

\[
X = \{ x_1, ..., x_{(\tau_1+...+\tau_q+r)}, x_{(\tau_1+...+\tau_q+r+1)}, ..., x_n \} = \{ u_1(t), ..., u_1(t - \tau_1), u_2(t), ..., u_r(t - \tau_r), y(t-1), ..., y(t-\tau_o) \}
\] (4.16)

where \( n = \tau_1 + ... + \tau_r + \tau_o + r \)

In S-P configuration, input to the consequent part is \( X' = X \).

### 4.3.3 Proposed Configurations

In the proposed configuration, output of the model depends upon the system history as well as the present and past output of the model as shown in the Fig. 4.4. System history means present and past input and output of the system. Our objective is that the model should track the actual system output. It means that the error between plant and model output decrease and results in an improved performance of the model. In S-P configuration, the main problem is selecting a model from a class of models and its structure determination. After selecting the model and deciding about its structure, the problem is reduced to parameter learning of the model. One important problem in S-P configuration is that model output is of no use during learning procedure except calculating the error. By using output of the model to learn the learning
parameter the significance of the model in learning procedure can be acquired. Parallel configuration has advantage that without much information about the system, it can learn the parameter of the model. In this configuration, the model output tracks the plant output by minimizing the error between them. If the convergence of the learning procedure is guaranteed the parallel configuration, is most suitable for long-term prediction.

Since, a neuro-fuzzy model is divided into two parts; i.e. premise and consequent parts, either the system output feedback to the premise part and the model output to the consequent part or vice versa. If the output of the system is feedback to the premise part and the output of the model feedback to the consequent part of the neuro-fuzzy system, it results a Premise Series-Parallel (PS-P) configuration. In the same way, if the output of the model feedback to the premise part, and the output of the plant feedback to the consequent part, we have a Consequent Series-Parallel (CS-P) configuration.

![Diagram of proposed parallel and Series-parallel configuration](image)

Fig. 4.4. Proposed parallel and Series-parallel configuration
This configurations can be used in special cases that model is combination of two part. Especially, as discussed before, fuzzy models are divided into two parts; premise and consequent parts. For the proposed configuration, the output of the model is written as:

\[
y(t) = h\left(y(t-1), y(t-2), \ldots, y(t-\tau_o), \right. \\
\left. \hat{y}(t-1), \hat{y}(t-2), \ldots, \hat{y}(t-\tau_o)\right)
\]  

(4.17)

Function ‘h’ in (4.17) here is represented by neuro-fuzzy model. Where \( u_q; (q = 1, \ldots, r) \) denote the inputs. \( \hat{y} \) and \( y \) are model and system outputs, respectively. \( \tau_u \) and \( \tau_o \) are the corresponding delays.

a) **Consequent Series-Parallel configuration (CS-P)**

In CS-P, the output of the model is fed back to the premise part as it is for parallel configuration and the output of the plant is fed back to the consequent part as it is for series-parallel configuration, which is shown in Fig. 4.5. It is a well-established fact that the premise part of each rule in fuzzy models exemplifies a local region in the input space in which consequent part act as a local model for the output space [Sugeno'93, Takagi'85- Sugeno'88, Zeng'94- Zeng'95]. These local models in the output space are approximated by linear or non-linear function of the premise variables. In CS-P configuration, the plant inputs and output with its delays are employed to approximate the local models in the consequent part of the TSK model, whereas the inputs and the delays of the model output are utilized to comprehend the input space region. The premise inputs in this configuration are denoted by:

\[
X = \left\{x_1, \ldots, x_{(\tau_1 + \ldots + \tau_u + r)}, x_{(\tau_1 + \ldots + \tau_u + r+1)}, \ldots, x_n\right\} = \left\{u_1(t), \ldots, u_1(t-\tau_1), u_2(t), \ldots, u_r(t-\tau_u), \hat{y}(t-1), \ldots, \hat{y}(t-\tau_o)\right\}
\]

(4.18)

where input to the consequent part in CS-P configuration is:
\[ X' = \{x_1, \ldots, x_{(r_1 + \ldots + r_r + r)}, x_{(r_1 + \ldots + r_{r+1})}, \ldots, x_n\} \]
\[ = \{u_1(t), \ldots, u_1(t - \tau_{\lambda 1}), u_2(t), \ldots, u_r(t - \tau_{\nu}), y(t - 1), \ldots, y(t - \tau_o)\} \] (4.19)

Fig. 4.5. Output of the plant feedback to the consequent part and the output of model feedback to the premise part

b) Premise Series-Parallel configuration (PS-P)

In PS-P, the output of the model feedback to the consequent part and the plant output is used for the premise part as shown in Fig. 4.6. In PS-P configuration, the model inputs and output with its delays are employed to approximate local models in the consequent part of the TSK model whereas the inputs and the plant output with their delays are utilized to comprehend the input space region. The premise inputs in this configuration are denoted by:

\[ X = \{x_1, \ldots, x_{(r_1 + \ldots + r_r + r)}, x_{(r_1 + \ldots + r_{r+1})}, \ldots, x_n\} \]
\[ = \{u_1(t), \ldots, u_1(t - \tau_{\lambda 1}), u_2(t), \ldots, u_r(t - \tau_{\nu}), y(t - 1), \ldots, y(t - \tau_o)\} \] (4.20)

and input to the consequent part in PS-P configuration is

\[ X' = \{x_1, \ldots, x_{(r_1 + \ldots + r_r + r)}, x_{(r_1 + \ldots + r_{r+1})}, \ldots, x_n\} \]
\[ = \{u_1(t), \ldots, u_1(t - \tau_{\lambda 1}), u_2(t), \ldots, u_r(t - \tau_{\nu}), \hat{y}(t - 1), \ldots, \hat{y}(t - \tau_o)\} \] (4.21)
4-4 Learning procedure

In this section, structure determination and initialization of the neuro-fuzzy model are presented. Discussion of different configurations in parameter learning of the neuro-fuzzy models also is including. In this section, an adaptive learning algorithm is introduced to learn the parameters of the model.

4-4.1 Structure determination and Initialization

Structure determination in neuro-fuzzy models means determination of the number of rules and input membership function. Initialization of the neuro-fuzzy models means that initializes center and standard deviation of membership function and initializes each linear function in consequent parts. In present work, Gaussian membership function is used. To determine number of necessary rules Modified Mountain Clustering (MMC) is applied [Azeem'03a, Yager'94, Chiu'96]. The purpose of clustering is to do natural grouping of large set of data, producing a concise representation of system's behavior. Azeem et.al., [Azeem'03a] have proposed a simple and easy way to implement, MMC for estimating the number and location of cluster centers. A brief discussion about MMC and its parameter is covered in Appendix B.
4-4.2 Training

To adjust the learning parameter of the model, the performance index $J$ as given in (2.18) is minimized by Gradient Descent (GD) algorithm. In this section, the GD based algorithm is applied. Since the parallel, CS-P and PS-P configurations include external recurrent to the model during learning procedure; criterion for learning stability and convergence has been evolved. To learn the parameters of the recurrent network, based on the gradient descent, different methods are presented in literature. All learning methods are the same as of back-propagation-through-time [Rumelhart’86- Werbos’88] or real-time recurrent learning algorithm [Williams’89] and it can be applied to adjust parameters of the recurrent network. In this work, by applying Lyapunov theorem, the learning stability and the convergence of learning procedure is guaranteed. To guarantee the speed of the convergence an adaptive learning rate with upper bound is applied.

a) Gradient Descent Technique of the parameters

For fine-tuning of initialized model/network parameters, a GD technique with momentum update and forgetting factor, as discussed in chapter 2, is applied to modify the parameters $\bar{x}, \sigma$ and $w$ in (4.4 - 4.5). The parameter update formula for $p^{th}$ data set is as follows:

$$\Delta_p W^m(q) = -\eta_w \cdot \frac{\partial J}{\partial W^m} = \eta_w \cdot \frac{1}{P} \cdot y_r^2 \cdot e \cdot \frac{\partial \hat{y}}{\partial W^m} \bigg|_p \quad (4.23)$$

$$\Delta_p \sigma_{mi} = -\eta \cdot \frac{\partial J}{\partial \sigma_{mi}} = \eta \cdot \frac{1}{P} \cdot f_r^2 \cdot e \cdot \frac{\partial \hat{y}}{\partial \sigma_{mi}} \bigg|_p \quad (4.24)$$

$$\Delta_p \bar{x}_{mi} = -\eta \cdot \frac{\partial J}{\partial \bar{x}_{mi}} = \eta \cdot \frac{1}{P} \cdot f_r^2 \cdot e \cdot \frac{\partial \hat{y}}{\partial \bar{x}_{mi}} \bigg|_p \quad (4.25)$$

where $e = y - \hat{y}$ is the error between the plant output and the model output. By applying the
chain rule to the above equation, \( \frac{\partial \hat{Y}_{NF}}{\partial w_m}, \frac{\partial \hat{Y}_{NF}}{\partial \sigma_{mi}} \) and \( \frac{\partial \hat{Y}_{NF}}{\partial \sigma_{mi}} \) for different configurations are derived as follows: Define \( \beta_m = \frac{\mu_{a^*}(X)}{\sum_{m=1}^{M} \mu_{a^*}(X)} \) then,

\[
\frac{\partial \hat{Y}_{NF}}{\partial w_{m0}} = \beta_m \tag{4.26}
\]

\[
\frac{\partial \hat{Y}_{NF}}{\partial w_{mi}} = x_i \cdot \beta_m \tag{4.27}
\]

\[
\frac{\partial \hat{Y}_{NF}}{\partial \sigma_{mi}} = w_m(X') \cdot \frac{\beta_m}{\mu_{a^*}} \cdot (1 - \beta_m) \cdot \frac{2 \cdot (x_i - \bar{x}_{mi})}{\sigma^2_{mi}} \tag{4.28}
\]

\[
\frac{\partial \hat{Y}_{NF}}{\partial \sigma_{mi}} = w_m(X') \cdot \frac{\beta_m}{\mu_{a^*}} \cdot (1 - \beta_m) \cdot \frac{2 \cdot (x_i - \bar{x}_{mi})^2}{\sigma^3_{mi}} \tag{4.29}
\]

\( X \) and \( X' \) in above equation are determined by (4.14) and (4.16) for P and S-P configurations, respectively. In CS-P configurations, \( X \) and \( X' \) are obtained by (4.18) and (4.19), respectively. With PS-P configuration, (4.20) and (4.21) are used to extract \( X \) and \( X' \), respectively. Fig. 4.7 shows the learning algorithm for TSK Neuro-Fuzzy model with different configuration. Using performance indexes \( J \) as in (2.18) convergence theorem of the learning procedure is stated as follows:

b) Learning Convergence theorems

Small value of learning rate \( \eta \) results in slower speed of convergence. Large value of \( \eta \) causes the learning procedure non-stable. Therefore learning rate should be chosen in such a way that the stability and convergence be guaranteed. To guarantee the stability during the learning procedure, Lyapunov stability theorem is applied. This formulates the appropriate range of
learning rate. Following Theorems guarantees the convergence stability of the neuro-fuzzy models:

Theorem 4.1: The asymptotic learning convergence of S-P and CS-P configurations (since local models have same variables i.e. $X$) are guaranteed if the learning rate for different learning parameters follows the upper bound as mentioned below:

$$0 < \eta_w < 2 \cdot P \cdot y^2_x$$  \hspace{1cm} \text{(4.30)}

$$0 < \eta_\sigma < \frac{2 \cdot P \cdot y^2_x}{\max_n |w(X)|^2 \left( \frac{2}{\sigma^3_{\text{max}}} \right)^2}$$  \hspace{1cm} \text{(4.31)}

$$0 < \eta_\bar{x} < \frac{2 \cdot P \cdot y^2_x}{\max_n |w(X)|^2 \left( \frac{2}{\sigma^3_{\text{min}}} \right)^2}$$  \hspace{1cm} \text{(4.32)}

Theorem 4.2: The asymptotic learning convergence of P and PS-P configurations (since local models have same variables i.e. $X'$) are guaranteed if the learning rate for different learning parameters follows the upper bound as mentioned below:

$$0 < \eta_w < 2 \cdot P \cdot y^2_x$$  \hspace{1cm} \text{(4.33)}

$$0 < \eta_\sigma < \frac{2 \cdot P \cdot y^2_x}{\max_n |w(X')|^2 \left( \frac{2}{\sigma^3_{\text{max}}} \right)^2}$$  \hspace{1cm} \text{(4.34)}

$$0 < \eta_\bar{x} < \frac{2 \cdot P \cdot y^2_x}{\max_n |w(X')|^2 \left( \frac{2}{\sigma^2_{\text{min}}} \right)^2}$$  \hspace{1cm} \text{(4.35)}

Stability analysis and convergence is carried out in Appendix C.
c) Adaptive learning rate

The learning rate is adaptive with the lower and upper bounds as mentioned in the above stated Theorem. Whether the learning rate \( \eta \) is increased or decreased, it depends on the change in the value of performance index \( J \). A two-phase adaptive scheme, to make the learning rate adaptive, is used in the GD technique. The initial value of learning rate is kept at 0.1 for all applications. In the first phase either it increases or decreases by a factor of “10”. When it reaches within bounds, in a very few epochs (i.e. < 10), then the second phase starts. This increase or decrease is dependent upon the acceptance or rejection, respectively, for updating the parameters. In the second phase, involving the operation \( \eta \leftarrow \gamma \eta \); we choose \( \gamma = 1.05 \) for the acceptance of parameter updates and \( \gamma = 0.7 \) for the rejection of the same. In the first phase, if the learning rate is continuously decreasing due to the rejection for update of the parameter, and the learning rate reaches with in a bound, the update of the parameter is accepted. This acceptance forces the learning rate to increase according to the second phase. If the learning rate is continuously increasing, in the first phase due to the acceptance for update of the parameter, and this increase in learning rate goes beyond the upper bound, the update of the parameter is rejected. This rejection forces the learning rate to decrease according to the second phase. Once first phase finishes learning rate follows the rule of second phase until the learning last.
Fig. 4.7. Learning algorithm for Neuro-Fuzzy model
4-5 Simulation Results

In this section, different types of dynamic systems that have been discussed in chapter 1 is considered. The selected dynamic examples are different nonlinear equation with different dynamic order. First 4 example are dynamical equations and Example 5 is a general benchmark problem.

Revisited Example 1: Linear regression with nonlinear input

By applying modified clustering and cluster validity function [Azeem'03a, Xie'87], five rules are obtained. Figure 4.8 illustrates the learning pattern and Table 4.1 shows the value of performance index of models obtained for all configurations. In this figure, the parallel-series category has shown by solid-blue, parallel with dot-black, PS-P with dot-slash & green and CS-P with slash-dot. The S-P and the CS-P configurations have better performance and CS-P is the best. It means that when the actual output of the system feedback to consequent part it yield better result. The initial fuzzy rules for the models are listed below:

\[ R^1 : \text{if } u(k) \text{ is } A_1^1 \text{ and } \hat{y}(k-1) \text{ is } A_1^1 \text{ and } \hat{y}(k) \text{ is } A_1^1 \text{ then } y'(k+1) = w_1(x) \]

\[ R^2 : \text{if } u(k) \text{ is } A_1^2 \text{ and } \hat{y}(k-1) \text{ is } A_2^2 \text{ and } \hat{y}(k) \text{ is } A_2^2 \text{ then } y^2(k+1) = w_2(x) \]

\[ R^3 : \text{if } u(k) \text{ is } A_1^3 \text{ and } \hat{y}(k-1) \text{ is } A_2^3 \text{ and } \hat{y}(k) \text{ is } A_3^3 \text{ then } y^3(k+1) = w_3(x) \]

\[ R^4 : \text{if } u(k) \text{ is } A_1^4 \text{ and } \hat{y}(k-1) \text{ is } A_2^4 \text{ and } \hat{y}(k) \text{ is } A_4^4 \text{ then } y^4(k+1) = w_4(x) \]

\[ R^5 : \text{if } u(k) \text{ is } A_1^5 \text{ and } \hat{y}(k-1) \text{ is } A_2^5 \text{ and } \hat{y}(k) \text{ is } A_5^5 \text{ then } y^5(k+1) = w_5(x) \]

where,

\[ w_1(x) = 0.0200 - 0.0356u(k) - 0.7385y(k-1) + 1.7168y(k) \]

\[ w_2(x) = -0.2127 + 0.5018u(k) - 0.7315y(k-1) + 1.7663y(k) \]

\[ w_3(x) = 0.0727 + 0.0387u(k) - 0.9743y(k-1) + 1.8446y(k) \]

\[ w_4(x) = -1.4026 + 3.6808u(k) - 0.1376y(k-1) + 1.0485y(k) \]

\[ w_5(x) = 0.4522 - 0.6490u(k) - 1.0283y(k-1) + 1.8407y(k) \]
The premise variable membership functions $A_1^1 - A_1^5$, $A_2^1 - A_2^5$ and $A_3^1 - A_3^5$ for inputs $u(k)$, $\hat{y}(k-1)$ and $\hat{y}(k)$ are shown in Fig. 4.9. The fuzzy rules corresponding to the learned network are listed below:

\[ R^1': \text{if } u(k) \text{ is } A_1^1 \land \hat{y}(k-1) \text{ is } A_1^1 \land \hat{y}(k) \text{ is } A_1^1 \text{ then } y'(k+1) \text{ is } w^1(X) \]

\[ R^2': \text{if } u(k) \text{ is } A_2^1 \land \hat{y}(k-1) \text{ is } A_2^1 \land \hat{y}(k) \text{ is } A_2^1 \text{ then } y^2(k+1) \text{ is } w^2(X) \]

\[ R^3': \text{if } u(k) \text{ is } A_3^1 \land \hat{y}(k-1) \text{ is } A_3^1 \land \hat{y}(k) \text{ is } A_3^1 \text{ then } y^3(k+1) \text{ is } w^3(X) \]

\[ R^4': \text{if } u(k) \text{ is } A_4^1 \land \hat{y}(k-1) \text{ is } A_4^1 \land \hat{y}(k) \text{ is } A_4^1 \text{ then } y^4(k+1) \text{ is } w^4(X) \]

\[ R^5': \text{if } u(k) \text{ is } A_5^1 \land \hat{y}(k-1) \text{ is } A_5^1 \land \hat{y}(k) \text{ is } A_5^1 \text{ then } y^5(k+1) \text{ is } w^5(X) \]

where,

\[ w^1(X) = 0.0246 - 0.0248u(k) - 0.7424y(k-1) + 1.7156y(k) \]

\[ w^2(X) = -0.2026 + 0.5045u(k) - 0.7499y(k-1) + 1.7484y(k) \]

\[ w^3(X) = 0.0757 + 0.0399u(k) - 0.9761y(k-1) + 1.8385y(k) \]

\[ w^4(X) = -1.4054 + 3.6797u(k) - 0.1360y(k-1) + 1.0498y(k) \]

\[ w^5(X) = 0.4420 - 0.6545u(k) - 1.020y(k-1) + 1.8430y(k) \]

The learned premise variable membership functions $A_1^U - A_1^V$, $A_2^U - A_2^V$ and $A_3^U - A_3^V$ for inputs $u(k)$, $\hat{y}(k-1)$ and $\hat{y}(k)$ are shown in Fig. 4.10. Figure 4.11 shows the actual output, model output and model error with CS-P configuration. In this figure, actual output of the plant is solid-blue and the model output is dot-red. The error is solid-blue. The value of performance index $J=1.8694 \times 10^{-7}$ for the model obtained for CS-P configuration.
Fig. 4.8. Learning pattern of all configurations for Example 1

Fig. 4.9. Initial membership functions of the normalized inputs for Example 1
Fig. 4.10. Learned membership functions of the normalized inputs for Example 1

Fig. 4.11. Actual output & model output with CS-P configuration and the error for Example 1
Revisited Example 2: Non-linear regression with random input

Figure 4.12 and Table 4.1, show the performance index for different identification configuration. The CS-P configuration model yields better result with five rules, obtained by MMC and cluster validity function. The initial fuzzy rule of the CS-P configuration is listed below:

\[ R^1: \text{if } u(k) \text{ is } A^1 \land \hat{y}(k-1) \text{ is } A^2 \land \hat{y}(k) \text{ is } A^3 \text{ then } y'(k+1) \text{ is } w'(X) \]

\[ R^2: \text{if } u(k) \text{ is } A^1 \land \hat{y}(k-1) \text{ is } A^2 \land \hat{y}(k) \text{ is } A^3 \text{ then } y^2(k+1) \text{ is } w^2(X) \]

\[ R^3: \text{if } u(k) \text{ is } A^3 \land \hat{y}(k-1) \text{ is } A^2 \land \hat{y}(k) \text{ is } A^3 \text{ then } y^3(k+1) \text{ is } w^3(X) \]

\[ R^4: \text{if } u(k) \text{ is } A^4 \land \hat{y}(k-1) \text{ is } A^4 \land \hat{y}(k) \text{ is } A^4 \text{ then } y^4(k+1) \text{ is } w^4(X) \]

where,

\[ w'(X) = 0.3527 + 0.6184u(k) - 0.5301y(k-1) + 0.5573y(k) \]

\[ w^2(X) = 0.3308 + 0.7160u(k) - 0.3063y(k-1) - 0.1194y(k) \]

\[ w^3(X) = 0.6575 + 0.6492u(k) - 0.371y(k-1) - 0.2846y(k) \]

\[ w^4(X) = 0.5092 + 0.7051u(k) - 0.6106y(k-1) - 0.4889y(k) \]

The premise variable membership function \( A_1 \sim A_4 \) and \( A_1 \sim A_4 \) for inputs \( u(k), \hat{y}(k-1) \) and \( \hat{y}(k) \) are shown in Fig. 4.13. The fuzzy rules corresponding to the learned non-normalized network are listed below:

\[ R'^1: \text{if } u(k) \text{ is } A'^1 \land \hat{y}(k-1) \text{ is } A'^2 \land \hat{y}(k) \text{ is } A'^3 \text{ then } y'^1(k+1) \text{ is } w'^1(X) \]

\[ R'^2: \text{if } u(k) \text{ is } A'^1 \land \hat{y}(k-1) \text{ is } A'^2 \land \hat{y}(k) \text{ is } A'^3 \text{ then } y'^2(k+1) \text{ is } w'^2(X) \]

\[ R'^3: \text{if } u(k) \text{ is } A'^3 \land \hat{y}(k-1) \text{ is } A'^2 \land \hat{y}(k) \text{ is } A'^3 \text{ then } y'^3(k+1) \text{ is } w'^3(X) \]

\[ R'^4: \text{if } u(k) \text{ is } A'^4 \land \hat{y}(k-1) \text{ is } A'^4 \land \hat{y}(k) \text{ is } A'^4 \text{ then } y'^4(k+1) \text{ is } w'^4(X) \]

127
where,

\[
W^{1f}(X) = 0.4130 + 0.7532u(k) - 0.4786y(k-1) + 0.4839y(k)
\]

\[
W^{2f}(X) = 0.3017 + 0.7805u(k) - 0.3881y(k-1) - 0.1744y(k)
\]

\[
W^{3f}(X) = 0.6266 + 0.5967u(k) - 0.1121y(k-1) - 0.2307y(k)
\]

\[
W^{4f}(X) = 0.5018 + 0.8116u(k) - 0.4932y(k-1) - 0.6007y(k)
\]

The learned premise variable membership function functions \(A_1^V\), \(A_2^V\), \(A_3^V\) and \(A_4^V\) - \(A_3^V\) for inputs \(u(k), y(k-1)\) and \(y(k)\) are shown in Fig. 4.14. Figure 4.15 shows actual output and model output of the CS-P model. The error for learning and prediction section is shown in Fig. 4.15. The value of performance Index \(J = 1.1040 \times 10^{-6}\) is obtained for CS-P configuration.

![Fig. 4.12. Learning pattern of all configurations for Example 2](image-url)
Fig. 4.13. Initial membership functions of the normalized inputs for Example 2

Fig. 4.14. Learned membership functions of the normalized inputs for Example 2
Revisited Example 3: Non-Linear Regression with Non-Linear Input

By applying MMC and cluster validity, three rules are generated. Figure 4.16 and Table 4.1, show performance index for different configuration of the identification models. CS-P configuration model yield better result with performance Index $J=3.2015\times10^{-6}$. Next on S-P model is better. The initial fuzzy rule of the CS-P configuration is listed below:

- $R^1$: if $u(k)$ is $A_1^1$ and $\hat{y}(k)$ is $A_3^1$ then $y'(k+1)$ is $w^1(X)$
- $R^2$: if $u(k)$ is $A_2^1$ and $\hat{y}(k)$ is $A_3^2$ then $y^2(k+1)$ is $w^2(X)$
- $R^3$: if $u(k)$ is $A_3^3$ and $\hat{y}(k)$ is $A_2^3$ then $y^3(k+1)$ is $w^3(X)$

where,

$$w^1(X) = -1.3762 + 2.5888u(k) + 1.0779y(k)$$
$$w^2(X) = -3.8700 + 7.7564u(k) - 0.0247y(k)$$
$$w^3(X) = -2.9392 + 8.0787u(k) + 0.0428y(k)$$