Prediction of Time Sequence Using Recurrent Compensatory Neuro-fuzzy Systems

ChiYung Lee[1] and ChengJian Lin[2,⋆]

[1] Dept. of Computer Science and Information Engineering
Nankai Institute of Technology
Nantou County, 542 Taiwan, China
[2] Dept. of Computer Science and Information Engineering
Chaoyang University of Technology
168 Gifeng E. Rd., Wufeng
Taichung County, 413 Taiwan, China
Tel: +886-4-23323000 Ext. 4408, Fax: +886-4-23742375
cjlin@mail.cyut.edu.tw

Abstract. In this paper, a recurrent compensatory neuro-fuzzy system (RCNFS) is proposed for prediction of time sequence. The compensatory-based fuzzy reasoning method is using adaptive fuzzy operations of neuro-fuzzy systems that can make the fuzzy logic systems more adaptive and effective. The recurrent network is embedded in the RCNFS by adding feedback connections in the second layer, where the feedback units act as memory elements. Also, an on-line learning algorithm is proposed to automatically construct the RCNFS. They are created and adapted as on-line learning proceeds via simultaneous structure and parameter learning.

1 Introduction

For a dynamic system, the output is a function of past input or past output or both, prediction of time sequence is not as direct as a static system, and to deal with temporal problem of dynamic system, the recurrent neural network and the recurrent neuro-fuzzy system have been attracting great interest [1]-[2].

In this paper, a recurrent compensatory neuro-fuzzy system (RCNFS) is proposed. The RCNFS is a recurrent multi-layer connectionist network for fuzzy reasoning and can be constructed from a set of fuzzy rules. In the RCNFS, adding feedback connections in the second layer develops the temporal relations. At the same time, the compensatory fuzzy inference method is using adaptive fuzzy operations of neuro-fuzzy system that can make the fuzzy logic system more adaptive and effective. An on-line learning algorithm is proposed to automatically construct the RCNFS. It consists of structure learning and parameter learning. The structure learning algorithm decides to add a new node which is satisfying the fuzzy partition of the input data. The back-propagation learning is then used for tuning input membership functions.

⋆ Corresponding Author.

J. Wang, X. Liao, and Z. Yi (Eds.): ISNN 2005, LNCS 3497, pp. 611–617, 2005.

2 The Compensatory Operator

Zimmermann [3] first defined the essence of compensatory operations. Zhang and Kandel [4] proposed more extensive compensatory operations based on the pessimistic operation and the optimistic operation. The compensatory operation can map the pessimistic input x_1 and the optimistic input x_2 to make the relatively compromised decision for the situation between the worst case and the best case. For example, $c(x_1, x_2) = x_1^{1-r} x_2^r$, where $r \in [0, 1]$ is called the compensatory degree. Many researchers [5]-[6] have used the compensatory operation to fuzzy systems successfully. Therefore, we can define the fuzzy if-then rule as follows:

$$R_j : [\text{ IF } x_1 \text{ is } A_{1j} \cdots \text{ and } x_n \text{ is } A_{nj}]^{1-r_j+r_j/n}, \text{ THEN } y' \text{ is } b_j \tag{1}$$

3 The Structure of RCNFS Model

The RCNFS realizes a fuzzy model of the following form:

$$Rule - j : [\text{ IF } h_{1j} \text{ is } A_{1j} \cdots \text{ and } h_{nj} \text{ is } A_{nj}]^{1-r_j+r_j/n}, \text{ THEN } y' \text{ is } w_j \tag{2}$$

where for $i = 1, 2, ..., n$, $h_{ij} = x_i + u_{ij}^{(2)}(t-1) \cdot \theta_{ij}$, $y'$is output variable, A_{nj} is linguistic term of the precondition part, w_j is constant consequent part, and n is number of input variables. That is, the input of each membership function is the network input x_i plus the temporal term $u_{ij}^{(2)}\theta_{ij}$. Therefore, the fuzzy system, with its memory (terms feed-back units), can be considered a dynamic fuzzy inference system.

Next, we shall introduce the operation functions of the nodes in each layer of the RCNFS model are described. In the following description, $u^{(l)}$ denotes output of a node in the lth layer.

*Layer*1(*InputNode*): No computation is done in this layer. Each node in this layer is an input node, which corresponds to one input variable, only transmits input values to the next layer directly.

$$u_i^{(1)} = x_i \tag{3}$$

*Layer*2(*InputTermNode*): Nodes in this layer correspond to one linguistic label of the input variables in Layer1 and a unit of memory, i.e., the membership value specified the degree to which an input value and a unit of memory belongs a fuzzy set is calculated in Layer 2. The Gaussian membership function, the operation performed in Layer 2 is

$$u_{ij}^{(2)} = exp\{-\frac{[h_{ij} - m_{ij}]^2}{\sigma_{ij}^2}\} \tag{4}$$

where m_{ij} and σ_{ij} are, respectively, the mean and variance of Gaussian membership function of jth term of ith input variable x_i. In addition, the inputs of this

layer for discrete time t can be defined by $h_{ij}(t) = u_i^{(1)}(t) + u_{ij}^{(2)}(t-1) \cdot \theta_{ij}$, where $u_{ij}(t-1)$ denotes the feedback unit of memory, which store the past information of the system, and θ_{ij} denotes the link weight of the feedback unit.

*Layer*3(*CompensatoryRuleNode*): Nodes in this layer represents the precondition part of one fuzzy logic rule. And they receive the one-dimensional membership degrees of the associated rule from nodes of a set in Layer 2. Here we use a compensatory operator mentioned to perform IF-condition matching of fuzzy rules. As a result, the output function of each inference nodes is

$$u_j^{(3)} = [\prod_i u_{ij}^{(2)}]^{1-r_j+\frac{r_j}{n}} \tag{5}$$

where the $\prod_i u_{ij}^{(2)}$ of a rule node represents the firing strength of its corresponding rule, $r_j \in [0,1]$ is called the compensatory degree. By tuning r_j, the compensatory operator becomes more adaptive.

*Layer*4(*OutputNode*): This layer acts a defuzzifier. The node in this layer is labeled Σ and its sums all incoming signals to obtain the final inferred result

$$u_k^{(4)} = \sum_j u_j^{(3)} w_{jk} \tag{6}$$

where the weight w_{jk} is output action strength of the kth output associated with the jth rule and $u_k^{(4)}$ is the kth output of the RCNFS.

4 The On-Line Learning Algorithm

In this section, we present an on-line learning algorithm for constructing the RCNFS. The proposed learning algorithm consists of structure learning phase and parameter learning phase. The structure learning is based on the degree measure to determine the number of fuzzy rules. The parameter learning is base upon supervised learning algorithms.

4.1 The Structure Learning Phase

The first step in the structure learning is to determine whether or not to extract a new rule from training data as well as the number of fuzzy sets on the universal of discourse of each input variable. Since one cluster in the input space corresponds to one potential fuzzy logic rule, with m_{ij} and σ_{ij} representing the mean and variance of that cluster. For each incoming pattern x_i the strength a rule is fired can be interpreted as the degree the incoming pattern belongs to the corresponding cluster. For computational efficiency, we can use compensatory operation of the firing strength obtained from $[\prod_i u_{ij}^{(2)}]^{1-r_j+\frac{r_j}{n}}$ directly as this degree measure

$$F_j = [\prod_i u_{ij}^{(2)}]^{1-r_j+\frac{r_j}{n}} \tag{7}$$

where $F_j \in [0,1]$. Using this degree measure, we can obtain the following criterion for the generation of a new fuzzy rule of new incoming data is described as follows. Find the maximum degree

$$F_{max} = max_{1 \le j \le R_{(t)}} F_j \tag{8}$$

where $R_{(t)}$ is the number of existing rules at time t. If $F_{max} \le \overline{F}$, then a new rule is generated where $\overline{F} \in (0,1)$ is a prespecified threshold that decays during the learning process. Once a new rule is generated, the next step is to assign initial mean, variance, and weight of feedback for the new membership function. Since our goal is to minimize an objective function and the mean, variance, and weight of feedback are all adjustable later in the parameter learning phase. Hence, the mean, variance, and weight of feedback for the new membership function are set as follow: $m_{ij}^{(R_{(t+1)})} = x_i$, $\sigma_{ij}^{(R_{(t+1)})} = \sigma_{init}$, and $\theta_{ij}^{(R_{(t+1)})} = random$, where x_i is the new input data and σ_{init} is a prespecified constant.

The whole algorithm for the generation of new fuzzy rules as well as fuzzy sets in each input variable is as follows. Suppose no rules are existent initially:
Step 1: IF x_i is the first incoming pattern THEN do
{ Generate a new rule
with mean $m_{i1} = x_i$, variance $\sigma_{i1} = \sigma_{init}$, weight of feedback $\theta_{i1} = random$,
compensatory degree $c_1 = random$, $d_1 = random$, weight $w_1 = random$
where σ_{init} is a prespecified constant.
}
Step 2: ELSE for each newly incoming x_i, do
{ Find $F_{max} = max_{1 \le j \le R_{(t)}} F_j$
IF $F_{max} \ge \overline{F}$
do nothing
ELSE
$R_{(t+1)} = R_{(t)} + 1$ generate a new rule
with mean $m_{ij}^{(R_{(t+1)})} = x_i$, variance $\sigma_{ij}^{(R_{(t+1)})} = \sigma_{init}$, weight of feedback $\theta_{i1}^{(R_{(t+1)})} = random$, compensatory degree $c_j^{(R_{(t+1)})} =$random, $d_j^{(R_{(t+1)})} =$ random, weight $w_j^{(R_{(t+1)})} =$ random
where σ_{init} is a prespecified constant.
}

4.2 The Parameter Learning Phase

After the network structure is adjusted according to the current training pattern, the network then enters the parameter learning phase to adjust the parameters of the net-work optimally based on the same training pattern. The learning process involves the determination of minimize a given cost function. The gradient of the cost function is computed and adjusted along the negative gradient. The idea of backpropagation algorithm is used for this supervised learning method. Considering the single output case for clarity, our goal is to minimize the cost function E is defined as

$$E = \frac{1}{2}[y - y^d]^2 \tag{9}$$

where y^d is the desired output and y is the current output. Using the steepest-descent gradient approach, the learning rule for a network weight in any one of the network layers is given by

$$\Delta W = -\frac{\partial E}{\partial W} \tag{10}$$

The weight is updated according to the following equation:

$$W(t+1) = W(t) + \eta_w \Delta W \tag{11}$$

where factor η_w is the learning rate parameter of the weight and t denotes the iteration number.

5 Prediction of Time Sequence

To clearly verify if the proposed RCNFS can learn the temporal relationship, a simple time sequence prediction problem found in [7] is used for test in the following example. The test bed used is shown in Figure 2(a). This is an 8 shape made up of a series with 12 points which are to be presented to the network in the order as shown. The RCNFS is asked to predict the succeeding point for every presented point. Obviously, a static network cannot accomplish this task, since the point at coordinate (0,0) has two successors: point 5 and point 11. The RCNFS must decide the successor of (0,0) based on its predecessor; if the predecessor is 3, then the successor is 5, whereas if the predecessor is 9, the successor is 11.

In this example, the RCNFS contains only two input nodes, which are activated with the two dimensional coordinate of the current point, and two output nodes, which represent the two dimensional coordinate of the predicted point. The learning rate $\eta_w = \eta_c = \eta_d = \eta_m = \eta_\sigma = \eta_\theta = 0.05$, and the prespecified threshold are chosen. After training, a root-mean-square (rms) error of 0.000237 is achieved, and the predicted values with 12 fuzzy logic rules ($\sigma = 0.08$) of RCNFS are shown in Figure 2(b). Simulation results show that we can obtain perfect prediction capability. Figure 2(c) shows the prediction results using the RFNN model [2]. In this figure, the RFNN also obtain prediction capability, but some time prediction points cannot be matched exactly. Figure 2(d) shows that a feedforward fuzzy neural network cannot predict successfully. Figure 2(e) shows the learning curves of the RCNFS model, the RFNN model and the FNN model. From the simulation results shown in Figure 2(d), we can see that the FNN is inappropriate for time sequence prediction because of its static mapping. To give a clear understanding of this performance comparison with the RFNN [2] and FNN [8] on the same problem is made in Table 1. Although the RCNFS needs more adjustable parameters than RFNN and FNN under the same fuzzy rules required, our model could obtain a smaller rms error and converge quickly.

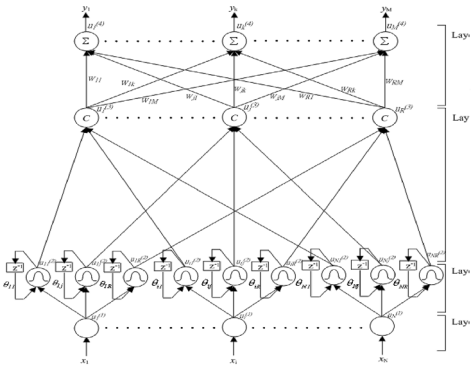

Fig. 1. Structure of the proposed RCNFS

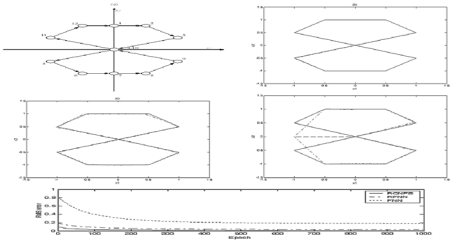

Fig. 2. Simulation results of time sequence prediction

Table 1. Performance comparison of various existing models

	RCNFS	RFNN [2]	FNN [8]
Rule numbers	12	12	12
Nodes	40	40	66
Parameter	120	96	108
RMS error	0.000237	0.0063	0.1758
Epochs	1000	1000	1000

6 Conclusion

A recurrent compensatory neuro-fuzzy system (RCNFS) is proposed in this paper. The compensatory operators are used to optimize fuzzy logic reasoning and select optimal fuzzy operators. Therefore, an effective neuro-fuzzy system should be able not only to adaptively adjust fuzzy membership functions but also to dynamically optimize adaptive fuzzy operators. An on-line learning algorithm is proposed to per-form the structure learning and the parameter learning. The simulation results show that the proposed learning algorithm converges quickly and requires a small number of tuning parameters.

References

1. Narendra, K. S., Parthasarathy, K.: Identification and Control of Dynamical Systems Using Neural Networks. IEEE Trans. on Neural Networks, **1** (1990) 4-27

2. Lee, C. H., Teng, C. C.: Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks. IEEE Trans. on Fuzzy Systems, **8** (2000) 349-366
3. Zimmermann, H. J., Zysno, P.: Latent Connective in Human Decision. Fuzzy Sets and Systems, **4** (1980) 31-51
4. Zhang, Y. Q., Kandel, A.: Compensatory Neurofuzzy Systems with Fast Learning Algorithms. IEEE Trans. on Neural Networks, **9** (1998) 83-105
5. Lin, C. J., Chen C. H.: Nonlinear System Control Using Compensatory Neuro-Fuzzy Networks. IEICE Trans. On Fundamentals of Electronics, Communications and Computer Sci-ences, E86-A (2003) 2309-2316
6. Lin, C. J., Ho, W. H.: A Pseudo-Gaussian-Based Compensatory Neural Fuzzy System. Proceedings of the IEEE International Conference on Fuzzy Systems (2003)
7. Santini, S., Bimbo, A. D., Jain, R.: Block-Structured Recurrent Neural Networks. Neural Networks, **8** (1995) 135-147
8. Chao, C. T., Chen, T. J., Teng, C. C.: Simplification of fuzzy-neural systems using similarity analysis. IEEE Trans. Syst., Man, Cybern., pt. B. **26** (1996) 344-354