INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 10 (10): 1999-2003, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.1999.2003

Fuzzy Process Neural Network based on Orthogonal Basis Function

Cheng Ma and Ye San
Control and Simulation Center, Harbin Institute of Technology, Harbin, China

Abstract: In order to widen the range of processing signals for fuzzy neural network, a kind of fuzzy process neural network based on orthogonal basis function is proposed. By inducting the orthogonal basis function into input space, the input function can be feature expanded. Meanwhile the weight vectors are also expanded under the same orthogonal basis function. The operation process of space aggregation and time cumulative can be simplified by using the orthogonality of basis function. The back propagation algorithm is used as learning algorithm. Simulation results show the great approximation ability of the fuzzy process neural network. And further experiments show that the network is sensitive to the number of fuzzy rules. The best approximation accuracy can be obtained only by choosing the proper values, such as k = 4 in this study.

Key words: Network structure, process neuron, time cumulative, orthogonality, back propagation, feature expanded

INTRODUCTION

Fuzzy Neural Networks (FNNs) have been widely used in economic and industry (Amjady, 2006; Pindoriya *et al*., 2008; Lin and Chou, 2009) that require modeling uncertain and imprecise system in recent years, due to the merge of Fuzzy Interference System (FIS) and Neural Network (NN) (Lin *et al*., 2005). Because of its logic inference and adaptive learning ability, FNNs have attracted a lot of attention. For example, A novel FNNs approach is proposed and it can adjust the structures by growing and pruning fuzzy rules during the learning process (Han and Qiao, 2010). In order to improve the computational efficiency, the record of each neuron's firing strength for all data previously clustered was used (Coyle *et al*., 2009). Evolution computation was also applied to the parameters and structures of FNNs, such as the order of polynomial, the number of membership function (Roh *et al*., 2007).

While facing with practical problems, system outputs usually depend on spatial aggregation and temporal cumulative of input signals. However, FNNs of all the above models can only deal with spatial related inputs. To solve the issues in a time-varying system, common method has to be replaced by spatial relation (time series) methods. In the year 2000, a Process Neural Network (PNN) model (He and Liang, 2000) was proposed which can deal with spatial-temporal information synchronously. Later, a PNN and traditional neural network combined model (He and Xu, 2003) was presented with time-varied input and output function. Because of the good localization property of the wavelet transform in time domain and frequency domain, a wavelet process neural network (Gang *et al*., 2008) was introduced and applied to the time series prediction.

In order to improve the ability of dealing with time-varying system, this paper proposed a kind of fuzzy process neural network based on orthogonal basis function. The inputs of the new network could be the fuzzy process information, the time-varying numeric, or the combination of them. Meanwhile, the network can receive the time/space signals or sequences directly. This means that the input signal scopes are enlarged. Because the computation of time cumulative is complicated, a set of the orthogonal basis functions in the input space was used. By the nature of the orthogonal basis functions, it's easy to simplify the computation and the aggregation operation process.

PROCESS NEURON

Process neuron is similar to the traditional neuron, it consists of three parts: weight function, converge and activation threshold. The differences between them are that the weight function, activation threshold of the PN are time-varying and the converging operation includes not only the multi-inputs space aggregation but also time cumulative. The topology of single process neuron is shown in Fig. 1.
where, X (t) = (x_2 (t), x_2 (t),..., x_n (t)) is the vector of inputs, y is the output, W (t) = (w_1 (t), w_2 (t),..., w_n (t)) is the vector of connection weighting function and f (.) is the activation

Corresponding Author: Cheng Ma, Control and Simulation Center, Harbin Institute of Technology, Harbin, China
Tel: +86-15004523136 Fax: +86-451-86416571

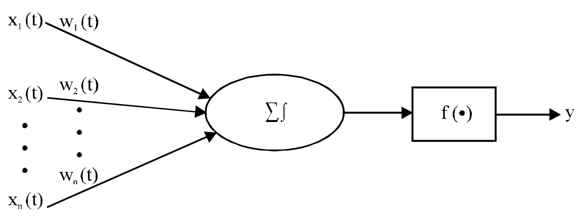

Fig. 1: Topology of single process neuron

function, [0 T] is the sample range and the symbols Σ and ∫ denote spatial and time aggregation operation, respectively. According to the topology, the relation between inputs X (t) and output y can be written as:

$$y = f\left(\sum_{i=1}^{n}\left(\int_0^T w_i(t)x_i(t)\right)\right) \tag{1}$$

In a function space, there are many sets of orthogonal functions (infinite dimensional), such as the trigonometric function space system in L^2 (0, T) function space, the Legendre polynomials in L^2 (-1, 1) function space, Walsh function systems in L^2 (0, 1) function space and so on.

If the input space of process neuron is [0 T], $\{b_l(t)\}^L_{l=1}$ is a set of orthogonal basis function which is also included in the input function space of the model, the input vector X (t) can be feature expanded into the following form:

$$X(t) = \left(\sum_{l=1}^{L} a_{1l}b_l(t), \sum_{l=1}^{L} a_{2l}b_l(t), \cdots, \sum_{l=1}^{L} a_{nl}b_l(t)\right) \tag{2}$$

where, a_{il} is a corresponding coefficient in the expanded equation. Let:

$$Z(t) = \left(\sum_{l=1}^{L} z_{1l}b_l(t), \sum_{l=1}^{L} z_{2l}b_l(t), \ldots, \sum_{l=1}^{L} z_{nl}b_l(t)\right)$$

which is also in the input function space. According to the nature of the orthogonal functions, the flowing equations can be established:

$$\begin{aligned} X(t) * Z(t)^T &= \left(\sum_{l=1}^{L} a_{1l}b_l(t), \sum_{l=1}^{L} a_{2l}b_l(t), \ldots, \sum_{l=1}^{L} a_{nl}b_l(t)\right) \\ &* \left(\sum_{l=1}^{L} z_{1l}b_l(t), \sum_{l=1}^{L} z_{2l}b_l(t), \ldots, \sum_{l=1}^{L} z_{nl}b_l(t)\right)^T \\ &= \sum_{i=1}^{n}\sum_{l=1}^{L} a_{il}z_{il} \end{aligned} \tag{3}$$

$$\begin{aligned} (X(t)-Z(t))^2 &= (X(t)-Z(t)) * (X(t)-Z(t))^T \\ &= \left(\sum_{l=1}^{L}(a_{1l}-z_{1l})b_l(t), \sum_{l=1}^{L}(a_{2l}-z_{2l})b_l(t), \ldots, \sum_{l=1}^{L}(a_{nl}-z_{nl})b_l(t)\right)^2 \\ &= \sum_{i=1}^{n}\sum_{l=1}^{L}(a_{il}-z_{il})^2 \end{aligned} \tag{4}$$

STRUCTURE OF FUZZY PROCESS NEURAL NETWORK

Here, introduces the structure of the FPNN. The proposed network structure is shown in Fig. 2 which has a total of five layers. The following sections present mathematical functions of each layer in detail.

Layer 1 (input layer): No computation is performed in this layer. Each node in this layer corresponds to one input variable. The input vector is $X_k(t) = (x_{k1}(t), x_{k2}(t), \ldots, x_{kn}(t))$, where k = 1, 2,..., k is the number of the samples and n is the number of input variables. And one of the input $x_{ki}(t)$ can be feature expanded as followed:

$$x_{ki}(t) = \sum_{l=1}^{L} a^l_{ki} b_l(t) \tag{5}$$

where, a^l_{ki} is a corresponding coefficient in the expanded equation. The inputs of this layer can be numerical time-verified function or the fuzzy variable with process information.

Layer 2 (fuzzification layer): Layer 2 acts as the fuzzification layer of the PFNN, where the values of the activated fuzzy Membership Functions (MFs) for a given current values are calculated. Each node in this layer represents a membership function. For input variable x_{ki}, the following Gaussian membership function is used:

$$\mu_{ij}(x_{ki}) = \exp[-\frac{(x_{ki}(t)-c_{ij}(t))^2}{\sigma_j^2}] \quad i = 1,2,\ldots,n;\ j = 1,2,\ldots,m \tag{6}$$

where, $\mu_{ij}(x_{kj})$ is the output of this layer, $c_{ij}(t)$ and σ_j denote the center and width of the Gaussian membership functions, respectively and m is the number of rules. Only one width σ_j is assigned to each fuzzy rule j, so the network size could be reduced.

With the same orthogonal basis functions $\{b_l(t)\}^L_{l=1}$ of input vector, the center c_{ij} can be feature expanded to the following form:

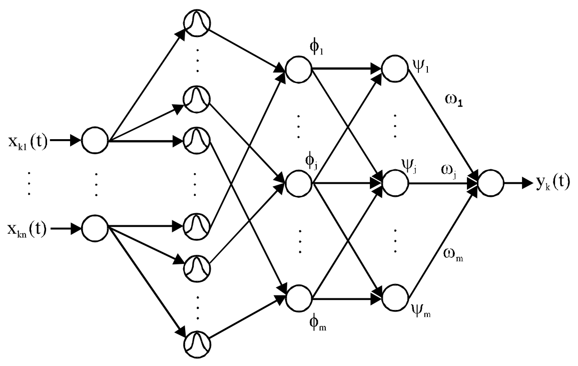

Fig. 2: The structure of fuzzy process neural network

$$c_{ij}(t)=\sum_{l=1}^{L} q_{ij}^{l}\, b_l(t) \tag{7}$$

where, q_{ij}^{l} is a corresponding coefficient in the expanded equation. The Eq. 5 can be rewritten as:

$$\mu_{ij}(x_{ki})=\exp[-\frac{\sum_{l=1}^{L}(a_{ki}^{l}-q_{ij}^{l})^2}{\sigma_j^2}]\; i=1,2,\dots,n;\, j=1,2,\dots,m \tag{8}$$

Layer 3 (rule layer): Each node in this layer performs a t-norm operation on inputs from Layer 2 using an algebraic product operation to obtain a spatial firing strength. The number of nodes in this layer equals the number of fuzzy rules. So there are m nodes. The output of the jth node is ϕ_j which can be written as:

$$\phi_j=\exp[-\frac{\sum_{i=1}^{n}(x_{ki}(t)-c_{ij}(t))^2}{\sigma_j^2}]=\exp[-\frac{\sum_{i=1}^{n}\sum_{l=1}^{L}(a_{ki}^{l}-q_{ij}^{l})^2}{\sigma_j^2}] \tag{9}$$

Layer 4 (normalized layer): This layer consists of normalized nodes. The number of nodes is equal to that of Rule layer:

$$\psi_j=\frac{\phi_j}{\sum_{j=1}^{m}\phi_j}\; j=1,2,\dots,m \tag{10}$$

where, Ψ_j is the output of the jth node in this layer.

Layer 5 (output layer): This layer performs defuzzification (weighted average) of the output as follows:

$$y_k(t)=\sum_{j=1}^{m}\omega_j\psi_j \tag{11}$$

where, y_k (t) is the output of the network, ω_j denotes the connecting weight between the output layer and the normalized layer. The weight can be expressed as follows:

$$\begin{aligned}\omega_j&=\alpha_{j0}(t)+\alpha_{j1}(t)x_{k1}(t)+\cdots+\alpha_{jn}(t)x_{kn}(t)\\&=(\alpha_{j0}(t),\alpha_{j1}(t),\cdots,\alpha_{jn}(t))*(1,x_{k1}(t),\cdots,x_{kn}(t))^T\end{aligned} \tag{12}$$

Suppose (α_{j0} (t), α_{j1} (t),..., α_{jn} (t)) are also in the input space, each of them can be feature expanded into:

$$\alpha_{ji}(t)=\sum_{l=1}^{L} w_{ji}^{l}\, b_l(t) \tag{13}$$

where, w_{ji}^{l} is a corresponding coefficient in the expanded equation. The corresponding coefficient α_{kj}^{l} in Eq. 5 is replaced by α_{ki}^{*1}. The weight can be rewritten as:

$$\begin{aligned}\omega_j&=\left(\alpha_{j0}(t),\alpha_{j1}(t),\cdots,\alpha_{jn}(t)\right)*(1,x_{k1}(t),\cdots,x_{kn}(t))^T\\&=\sum_{i=1}^{n+1}\sum_{l=1}^{L} w_{ji}^{l}\, a_{ki}^{*1}\end{aligned} \tag{14}$$

Equation 11 can be rewritten as:

$$y_k(t)=\sum_{j=1}^{m}\omega_j\psi_j=\sum_{j=1}^{m}\left(\exp[-\frac{\sum_{i=1}^{n}\sum_{l=1}^{L}(a_{ki}^{l}-q_{ij}^{l})^2}{\sigma_j^2}]\left(\sum_{i=1}^{n+1}\sum_{l=1}^{L} w_{ji}^{l}a_{ki}^{*1}\right)\right)\Bigg/\sum_{j=1}^{m}\exp[-\frac{\sum_{i=1}^{n}\sum_{l=1}^{L}(a_{ki}^{l}-q_{ij}^{l})^2}{\sigma_j^2}] \tag{15}$$

LEARNING FOR FUZZY PROCESS NEURAL NETWORK

The back propagation algorithm is used as learning algorithm. Assume that we have K learning functions samples and each learning function has n inputs:

$$\begin{bmatrix} x_{11}(t),x_{12}(t),\dots, & x_{1n}(t), & y_1(t)\\ x_{21}(t),x_{22}(t),\dots, & x_{2n}(t), & y_2(t)\\ \vdots\quad\vdots & \vdots & \vdots\\ x_{K1}(t),x_{K2}(t),\dots, & x_{Kn}(t), & y_K(t)\end{bmatrix}$$

where, the input function or sequence is x_{ij} (t), y_i (t) is the desired output. Suppose $\tilde{y}_k(t)$ is the corresponding actual output function of the FPNN, then the mean square error of the FPNN output can be written as:

$$\begin{aligned}E&=\frac{1}{2}\sum_{k=1}^{K}(y_k(t)-\tilde{y}_k(t))^2\\&=\frac{1}{2}\sum_{k=1}^{K}\left(y_k(t)-\sum_{j=1}^{m}\left(\exp[-\frac{\sum_{i=1}^{n}\sum_{l=1}^{L}(a_{ki}^{l}-q_{ij}^{l})^2}{\sigma_j^2}]\left(\sum_{i=1}^{n+1}\sum_{l=1}^{L} w_{ji}^{l}\, a_{ki}^{*1}\right)\right)\Bigg/\sum_{j=1}^{m}\exp[-\frac{\sum_{i=1}^{n}\sum_{l=1}^{L}(a_{ki}^{l}-q_{ij}^{l})^2}{\sigma_j^2}]\right)^2\end{aligned} \tag{16}$$

According to the gradient descent method, the modifications of the network connection weights and corresponding coefficient are:

$$w^{1}_{ij} = w^{i}_{ij} + \beta \Delta w^{l}_{ij} \tag{17}$$

$$q^{1}_{ij} = q^{i}_{ij} + \eta \Delta q^{l}_{ij} \tag{18}$$

$$\sigma_j = \sigma_j + \lambda \Delta \sigma_1 \tag{19}$$

where, β, γ, λ are the learning rates constants and the modifications Δw^{l}_{ij}, Δq^{l}_{ij}, Δσ can be calculated as follows:

$$\Delta w^{1}_{ij} = -\frac{\partial E}{\partial w^{1}_{ij}} = -\sum_{k=1}^{K} (y_k - d_k)\psi_j a^{*1}_{ij} \tag{20}$$

$$\Delta q^{1}_{ij} = -\frac{\partial E}{\partial q^{1}_{ij}} = -2\sum_{k=1}^{K} (y_k - d_k)(\omega_j \sum_{j=1}^{m} \phi_j - \sum_{j=1}^{m} \omega_j \phi_j)\phi_j (a^{*1}_{ij} - q^{1}_{ij}) / \left(\sigma_j \sum_{j=1}^{m} \phi_j\right)^2 \tag{21}$$

$$\Delta \sigma_j = -\frac{\partial E}{\partial \sigma_j} = -2\sum_{k=1}^{K} (y_k - d_k)(\omega_j \sum_{j=1}^{m} \phi_j - \sum_{j=1}^{m} \omega_j \phi_j)\phi_j / \left(\sigma_j^3 \left(\sum_{j=1}^{m} \phi_j\right)^2\right) \tag{22}$$

In general, the major steps of the FPNN learning algorithm are as follows:

Step 1: Select the standard orthogonal basis functions in the input space. If the inputs vectors are continue variables, the number of basis function should make the expansion of the basis function satisfy the required precision. Otherwise, divide the input interval [0 T] equally and determine the partition points $t_1, t_2, ..., t_l$

Step 2: Initialize the connection weights, the center and the width of the Gaussian functions, the maximal learning times is MaxGen, the learning time is gen which is equal to 1 at the beginning

Step 3: Calculate the error function according to Eq. 16. If gen>MaxGen, go to step 5

Step 4: Modify the connection weight, the center and the width of the Gaussian functions according to Eq. 20-22. Then gen = gen+1, go back to step 3

Step 5: Output the learning result and stop the progress

SIMULATION

To validate the proposed fuzzy process neural network algorithm, this section gives some numerical results.

In Fig. 3, continuous process neural networks are used to validate the effectiveness of the FPNN. The input internal is [0 1]. The sets of simulated input signals are generated by the following function: {sin (2kπt); cos (2kπt)}. Where k = 1, 2..., 10 is the number of the input set. The output is k for the corresponding kth input set. Using

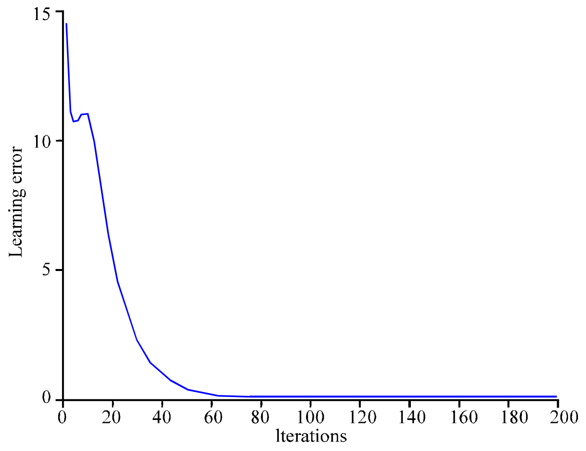

Fig. 3: Learning error curse

the FPNN shown in Fig. 2 and taking n = 2, m = 4, the number of each FPNN layer is 2-8-2-2-1, respectively. The learning rate constants are β = 0.005, γ = 0.07; the maximal learning time is 200. The basis function is the Walsh orthogonal function and the number of basis function is 32. The mean square error of the FPNN output is 5.7732e-007 after the 200th iteration. The learning error curse is shown in Fig. 3.

The result shows that there is powerful approximation ability of the fuzzy process neural network. As shown in the Fig. 3, there is a little glitch at the beginning of the curve. This means that the fuzzy process neural network may be sensitive to some parameters and this will be verified in Fig. 4.

In Fig. 4, discrete process neural networks are used to validate the number of fuzzy rule's effect on the FPNN. Construct 9 input sample function belonging to 3 sequences with process input interval [0 1]. The first sequence has 4 sample functions: sin (2π (t-0.5)), sin (2.1π (t-0.5)), sin (2.2π (t-0.5)), sin (2.4π (t-0.5)). Suppose that the corresponding expected output is 0.3333. The second sequence has 4 sample functions: 1.2 sin (3π (t-0.667)), 1.2 sin (3.2π (t-0.667)), 1.2 sin (3.4π (t-0.667)) and 1.2 sin (3.6π (t-0.667)). Suppose that the corresponding expected output is 0.6667. The third sequence has 4 sample functions: 1.4 sin (4π (t-0.25)), 0.14 sin (4.3π (t-0.25)), 1.4 sin (4.6π (t-0.25)) and 1.4 sin (4.8π (t-0.25)). Suppose that the corresponding expected output is 1.0000. The sample function are dispersed as {sin (2kπ (t_i-0.5))}, where t_i = i/128 for i = 0, 1,..., 127. Use the FPNN shown in Fig. 2. The number of the input and output are 3 and 1, respectively. The learning rate constants are β = 0.006, γ = 0.055, λ = 0.08; the maximal learning time is 200. A Walsh transform is implemented for discrete data and the transformed data are submitted to

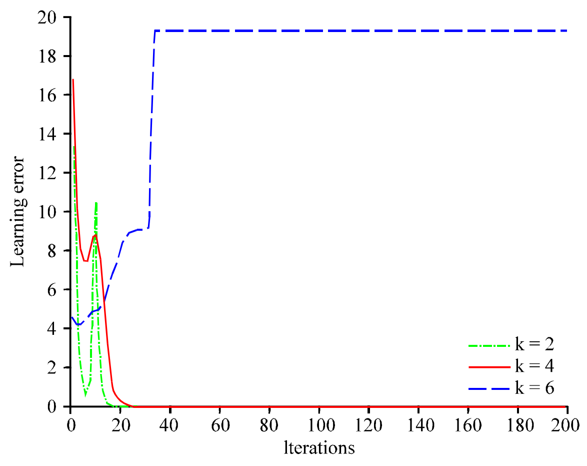

Fig. 4: Learning error curse with different number of the fuzzy rules

Table 1: The mean square error of the FPNN output

No. of the fuzzy rules	Mean square error
k = 2	2.52e-15
k = 4	1.50e-15
k = 7	19.27

the network for training. FPNN with the different number of the fuzzy rules are trained. After the 200th iteration, the mean square error of the FPNN output is shown in Table 1. The learning error curses are shown in Fig. 4.

The results show that the number of fuzzy rules is very important to the generalization ability of the FPNN model. If the number of the rules is large (k = 7), the learning error may decrease at fist but at some point it would increase. If it is small (k = 2), the amplitude of the oscillate part on the learning curse may be large. This indicates that the FPNN is sensitive to the number of the fuzzy rules. So we can choose suitable number of the fuzzy rules to optimize the performance, for example, here we choose k = 4.

CONCLUSION

In this study, fuzzy process neural network is proposed which combine the advantage of fuzzy neural network and process neuron, so the fuzzy neural network has the ability of achieving space aggregation and time cumulative. The network structure is introduced in detail, the orthogonal functions are used as time integral operators and the learning algorithm is given. Simulation results show that the fuzzy process neural network has strong approximation ability and the network is sensitive to the number of the fuzzy rules. Finally some issues which the FPNN may have are discussed, such as the unequal signal duration, rule extraction and the possible solutions are also considered.

ACKNOWLEDGMENT

Present study is supported by the National Natural Science Foundation of China under Grant No. 61074127.

REFERENCES

Amjady, N., 2006. Day-ahead price forecasting of electricity markets by a new fuzzy neural network. IEEE Trans. Power Syst., 21: 887-896.

Coyle, D., G. Prasad and T.M. McGinnity, 2009. Faster self-organizing fuzzy neural network training and a hyperparameter analysis for a Brain-computer interface. IEEE Trans. Syst. Man Cybern. Part B: Cybern., 39: 1458-1471.

Gang, D., Z. Shi-Sheng and L. Yang, 2008. Time series prediction using wavelet process neural network. Chin. Phys. B, 17: 1998-2003.

Han, H. and J. Qiao, 2010. A self-organizing fuzzy neural network based on a growing-and-pruning algorithm. IEEE Trans. Fuzzy Syst., 18: 1129-1143.

He, X.G. and J.Z. Liang, 2000. Some theoretical issues on procedure neural networks. Eng. Sci., 2: 40-44.

He, X.G. and S.H. Xu, 2003. Process neural network with time-varied input and output functions and its applications. J. Software, 14: 746-769.

Lin, C.T., W.C. Cheng and S.F. Liang, 2005. An on-line ica-mixture-model- based self-constructing fuzzy neural network. IEEE Trans. Circuits Syst. I: Regul. Pap., 52: 207-221.

Lin, F.J. and P.H. Chou, 2009. Adaptive control of two-axis motion control system using interval type-2 fuzzy neural network. IEEE Trans. Ind. Electron., 56: 178-193.

Pindoriya, N.M., S.N. Singh and S.K. Singh, 2008. An adaptive wavelet neural network-based energy price forecasting in electricity markets. IEEE Trans. Power Syst., 23: 1423-1432.

Roh, S.B., W. Pedrycz and S.K. Oh, 2007. Genetic optimization of fuzzy polynomial neural networks. IEEE Trans. Ind. Electron., 54: 2219-2238.