Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080151

Fuzzy Neural Network-based Time Delay Prediction for Networked Control Systems

Chang-Wook Han[*]

Department of Electrical Engineering, Dong-Eui University, Busan, South Korea

Received: 11 Jul. 2013, Revised: 24 Nov. 2013, Accepted: 25 Nov. 2013
Published online: 1 Jan. 2014

Abstract: In networked control systems, the varying transmission time delay of the transmitting signal is unavoidable. If the varying transmission time delay exceeds the fixed sampling time, the system will be unstable. To solve this problem, in this paper, the logic-based fuzzy neural networks are applied to the prediction of transmission time delay. The predicted time delay is used as a sampling period of the networked control systems. To show the usability of the proposed method, the transmission time delay data are collected from the real system, and these collected data are used to train and test the logic-based fuzzy neural networks.

Keywords: Networked Control Systems, Fuzzy Neural Network, Time Delay.

1 Introduction

Recently, the control systems have been complex because of many sensors and actuators. In point-to-point system the number of needed wire increases very much proportional to the number of sensor and actuator. Because of this problem networked control systems (NCS) are widely used in control areas [1,2,3].

NCS is designed as a feedback control system, i.e., the control loops are closed through a real time network. In NCS, network induced delays occur inevitably and they degrade the dynamic performance of the system. Therefore, one of the main design issues of the NCS is the transmission delay [2].

Lee et al. [4] designed single-input- single-output system based on remote fuzzy logic controller for the NCS using profibus-DP. Zheng et al. [5] used T-S fuzzy model to model the NCS having various type of network induced delay and data packet loss. To detect the sensor/actuator faults parity equation approach and fuzzy observer based approach were used.

In NCS, sampling time is fixed as constant. It is not desirable for the time varying network induced delay. If we can predict the delay, it can be used as a sampling time. Although, many researchers have been considering computational intelligence-based NCS, the time varying sampling time prediction is rarely considered. Therefore, this paper proposes the method to predict the sampling time of the NCS using logic-based fuzzy neural networks. The network induced delay data collected from the real system will be used to train and test the model.

2 Networked Control Systems

NCS is a distributed control systems with network. NCS is not control of networks, but control through networks. Real time transmission of short and frequent packet is different point of NCS from communication data network. Fig. 1 shows the overview of the NCS. As shown in Fig. 1, the wired/wireless communication medium exists between controller and sensor (actuator).

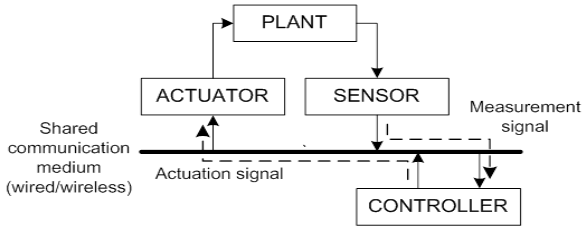

Fig. 1: Overview of the NCS.

[*] Corresponding author e-mail: cwhan@deu.ac.kr

The advantages of the NCS are no wire, low cost, low weight, low power, easy maintenance, etc. But time varying delay exists in NCS owing to the sharing of the network medium. We call this delay as network-induced delay. This network-induced delay varies widely depend on the message transmission time and overhead time. Transmission time through the medium heavily rely on network protocol. Therefore, it is necessary to reduce the network-induced delay without degradation of the NCS performance, called network scheduling method.

In NCS, transmitting sampled data within sampling time with guaranteed system stability is important. Short sampling period is desirable in most of the control systems, while sampling period of NCS can be extended to the bound guaranteeing the system stability notwithstanding the system degradation. This sampling period is called maximum allowable delay bound (MADB) [6]. Therefore, it is needed to find MADB guaranteeing stability of NCS, and scheduling method to restrict the network-induced delay within MADB.

3 Time Varying Sampling Period NCS [7]

In NCS, fixed sampling period model has the following problems:

-Time delay in real system is time varying and has no rule. It is very difficult to select adequate sampling time in NCS modeling.
-Time delay part of the NCS degrades the control performance very much.
-Compensation control technique is needed to compensate the time delay, and moreover, a special controller considering network-induced delay has to be designed.

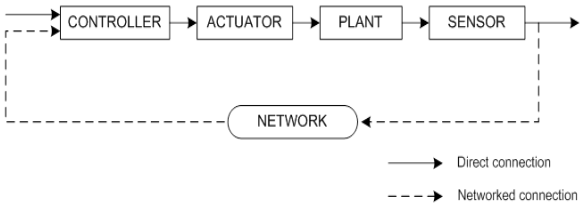

Fig. 2: Structure of the considered NCS.

To overcome these problems, in this paper, time varying sampling period (different sampling period for each sampling step) is considered. It will be reasonable to set time delay as sampling period. For the development of a new time varying sampling period model the following assumptions have to be considered (refer to Fig.2):

-NCS basically consists of plant, actuator, controller, sensor, and communication network.
-Actuator is directly connected to both of plant and controller.
-Controller-to-actuator, actuator-to-plant, there are no network, no time delay exist.
-Sensor is directly connected to plant. Sensor can be considered as a part of the plant.
-Network only exists between sensor and controller.
-Sensor, controller, and actuator are all event-driven.

The continuous plant model of the NCS considering network-induced delay is express in the following form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t). \end{aligned} \tag{1}$$

and the discrete controller is given as follows

$$y(kh) = -Kx(kh), \quad k = 0,1,2,\ldots \tag{2}$$

Sampling the system with a fixed sampling period h, and considering the time delay τ_k , we can get

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma_1(\tau_k)u(k) + \Gamma_l(\tau_k)u(k-1), \\ y(k) &= Cx(k), \\ u(k) &= -Kx(kh-\tau_k), \\ &\textit{where} \\ &\tau_k < h, \\ &\Phi = e^{Ah}, \\ \Gamma_1(\tau_k) &= \int_0^{h-\tau_k} e^{As}dsB, \\ \Gamma_1(\tau_k) &= \int_{h-\tau_k}^{h} e^{As}dsB. \end{aligned} \tag{3}$$

In fixed sampling period system, one upper limit of the time delay has to be selected. The common method is to select the maximum of the time delay and then turn it forward into integer time of the sampling period. It means that the time delay must be within one sampling period h or d sampling period dh. It is apparent that this method is conservative, and it does not accord with actual conditions. The system with time varying sampling period h_k can be described as follows:

$$\begin{aligned} x_{(k+1)} &= A_k(h_k)x_k + B_k(h_k)u_k, \\ y_k &= Cx_k, \\ u_k &= -Kx_k, \\ &\textit{where} \\ A_k &= e^{Ah_k}, \\ B_k &= \int_0^{h_k} e^{As}ds. \end{aligned} \tag{4}$$

As can be seen above, the sampling period is a parameter of A_k, B_k and the time delay disappears from

the controller. In this way, NCS becomes discrete time varying system without time delay. The advantage of replacing the fixed sampling period model with time varying sampling period model is that the stability analysis of time varying sampling period model becomes much easier owing to the disappearance of the time delay part. To analyze the stability we only need to find whether all the eigenvalues of $A_k - B_k K$ are in the unit circle at each sampling step k or not.

Further defining

$$z_k = \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \tag{5}$$

as the augmented state vector, then the augmented close-loop system can be describe as

$$z_{k+1} = \Phi_k z_k = \begin{bmatrix} A_k - B_k K & 0 \\ -K & 0 \end{bmatrix} \cdot \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}$$

where (6)

$$\Phi_k = \begin{bmatrix} A_k - B_k K & 0 \\ -K & 0 \end{bmatrix}$$

To apply this modeling scheme, however, we have to know the variable sampling period or the time delay at each sampling step. Owing to the characteristic of the NCS, we dont know the time delay before it occurs. Therefore, in this paper, we apply the logic-based fuzzy neural networks to predict the time delay at each sampling step.

4 Logic-based Fuzzy Neural Networks [8,9]

This paper is a new application version of the cascade architectures of fuzzy neural networks that is proposed by the author to predict the transmission time delay of the networked control systems. Therefore, the same version of cascade architectures of fuzzy neural networks and its optimization method in [8,9] are used in this paper. For this reason, all of this section directly refer to [8,9]. For more details about the cascade architectures of fuzzy neural networks, please refer to [8,9].

4.1 Logic processor (LP)

As originally introduced by Pedrycz et al. [8], fuzzy neurons emerge as result of a vivid synergy between fuzzy set constructs and neural networks. In essence, these neurons are functional units that retain logic aspects of processing and learning capabilities characteristic for artificial neurons and neural networks. Two generic types of fuzzy neurons are considered:

AND neuron is a nonlinear logic processing element with n-inputs $\mathbf{x}[0,1]^n$ producing an output y governed by the expression

$$y = AND(\mathbf{x};\mathbf{w}) = \mathbf{T}_{i=1}^{n}(w_i s x_i) \tag{7}$$

where $\mathbf{w}$ denotes an n-dimensional vector of adjustable connections (weights). The composition of $\mathbf{x}$ and $\mathbf{w}$ is realized by an t-s composition operator based on t- and s-norms, that is, s denoting some s-norm and t standing for a t-norm. As t- norms (s-norms) carry a transparent logic interpretation, we can look at as a two-phase aggregation process: first individual inputs (coordinates of $\mathbf{x}$) are combined or-wise with the corresponding weights and these results produced at the level of the individual aggregation are aggregated and-wise with the aid of the t-norm.

By reverting the order of the t- and s-norms in the aggregation of the inputs, we end up with a category of OR neurons,

$$y = OR(\mathbf{x};\mathbf{w}) = \mathbf{S}_{i=1}^{n}(w_i t x_i) \tag{8}$$

We note that this neuron carries out some and-wise aggregation of the inputs followed by the global or-wise combination of these partial results.

Some obvious observations hold:

-For binary inputs and connections, the neurons transform to standard AND and OR gates.
-The higher the values of the connections in the OR neuron, the more essential the corresponding inputs. This observation helps eliminate irrelevant inputs; the inputs associated with the connections whose values are below a certain threshold are eliminated. An opposite relationship holds for the AND neuron; here the connections close to zero identify the relevant inputs.
-The change in the values of the connections of the neuron is essential to the development of the learning capabilities of a network formed by such neurons; this parametric flexibility is an important feature to be exploited in the design of the networks.

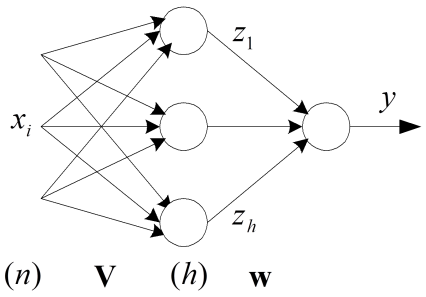

Fig. 3: Architecture of the LP.

The LP, described in Fig. 3, is a basic two-level construct formed by a collection of "h" AND neurons whose results of computing are then processed by a single OR neuron located in the output layer. Because of the

location of the AND neurons, we will be referring to them as a hidden layer of the LP.

Each LP is uniquely characterized by a number of parameters: a number of inputs (n), number of nodes in the hidden layer (h) and an array of connections of the AND neurons as well as the OR neuron in the output layer. Bearing in mind the topology of the LP, the connections of the AND neurons can be systematically represented in a matrix form V while the connections of the OR neuron are collected in a single vector form (w). We write the following detailed expressions

$$z_j = AND(\mathbf{x}, \mathbf{V}_j), \quad j = 1,2,...,h \quad y = OR(\mathbf{z}, \mathbf{w}) \quad (9)$$

where $\mathbf{z}$ is a vector of outputs of the AND neurons $\left(\mathbf{z} = [z_1 z_2 ... z_h]^T\right)$ while $\mathbf{V}_j$ denotes the j-th column of the connection matrix $\mathbf{V}$.

4.2 Cascade architectures of fuzzy neural networks

As LPs are our basic functional modules, there are several viable options to build an overall architecture. Here we discuss them and analyze its functional properties.

LPs are basic functional modules of the network that are combined into a cascaded structure. The essence of this architecture is to stack the LPs one on another. This results in a certain sequence of input variables. To assure that the resulting network is homogeneous, we use LPs with only two inputs, as shown in Fig. 4. In this sense, with n input variables, we end up with (n-1) LPs being used in the network. Each LP is fully described by a set of the connections ($\mathbf{V}$ and $\mathbf{w}$). To emphasize the cascade-type of architecture of the network, we index each LP by referring to its connections as $\mathbf{V}[ii]$ and $\mathbf{w}[ii]$ with "ii" being an index of the LP in the cascade sequence.

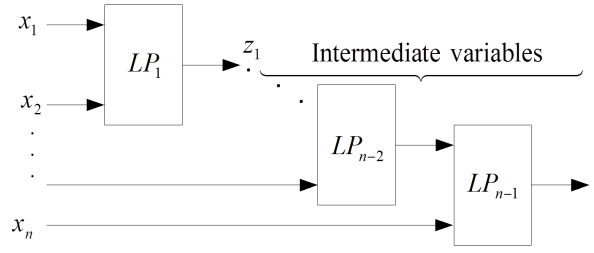

Fig. 4: A cascade network realized as a nested collection of LPs.

From the functional point of view, we regard the network as a realization of a decomposition of some function (either Boolean or fuzzy). It becomes obvious by noting that the output can be schematically represented as $y = f(x_i, g(\mathbf{x}_i^{\sim}))$ with x_i being the input to the LP under discussion and $g(\mathbf{x}_i^{\sim})$ describing the remaining part of the network ($\mathbf{x}_i^{\sim}$ indicates that the i-th input variable is nonexistent here). Subsequently, the same decomposition model applies to $g(\mathbf{x}_i^{\sim})$ which can be further decomposed. Fig. 5 underlines the effect of decomposition. It becomes obvious that an order of input variables affects the performance of the network.

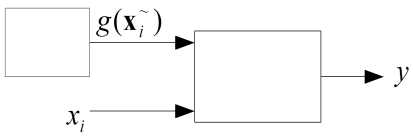

Fig. 5: Cascade network as a model of function decomposition.

To gain a better view at the mapping realized by the network, it is advantageous to discuss the functional aspects of a single logic processor. We start with the simplest possible topology. We assume that the number of AND neurons (hidden layer) is equal to 2 and consider a binary character of the connections, see Fig. 6.

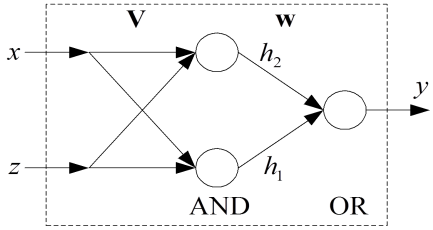

Fig. 6: An LP along with its possible logic descriptions.

Based on the values of the connection matrices ($\mathbf{w}$ and $\mathbf{V}$) we arrive at the following logic expressions (let us remind that the rows of $\mathbf{V}$ contain the values of the connections originating from a certain input node to the two AND neurons in the hidden layer; the columns are labeled by the corresponding AND nodes)

$$\mathbf{w} = [1\ 0], \mathbf{V} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix},$$
$$h1 = x, h2 = x, y = h1\ OR\ 0 = x$$
$$\mathbf{w} = [1\ 0], \mathbf{V} =, \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix},$$
$$h1 = x\ AND\ z, h2 = 1, y = h1 OR 0 = x\ AND\ z$$

In a nutshell, the proposed network realizes a successive realization of the logic model by incorporating more variables by augmenting its structure by a basic (generic) unit once at a time.

4.3 Development of the cascade architectures networks

The evolutionary optimization [10] is an attractive avenue to exploit in the development of the cascade network. In this learning scenario, we arrange all elements to be optimized (that is a sequence and a subset of input variables, and the connections of the logic processors) into a single chromosome and carry out their genetic optimization. The considered form of the chromosome for this optimization is described in Fig. 7. The input sequence of the variables (and thus is involved in the structural optimization of the network) and the connections (parametric optimization) are the phenotype of the chromosome. The sequence gene of input variables to be used in the model (we allow for a high level of flexibility by choosing only a subset of the input variables) consists of ”n” real numbers in the unit interval. These entries are assigned integer numbers that correspond to their ranking in the chromosome. The first ”p” entries of the chromosome (assuming that we are interested in ”p” variables) are then used in the structure of the network. For instance, if the chromosome consists of the following entries: 0.5 0.94 0.1 0.82 0.7 (n = 5) and we are interested in p=3 variables, the ranking leads to the sequence of integers 2 4 5 1 3 and we choose x_2, x_4, and x_5.

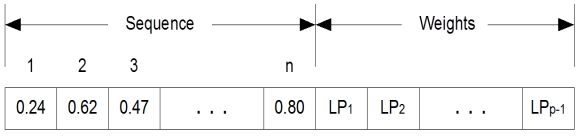

Fig. 7: Structure of a chromosome for the optimization of cascade architectures of fuzzy neural networks.

The resulting connections of GA are binary. Instead of going ahead with the continuous connections, the intent of GA is to focus on the structure and rough (binary) values of the connections. This is legitimate in light of the general character of genetic optimization: we can explore the search space however there is no guarantee that the detailed solution could be found. The promising Boolean solution can be next refined by allowing for the values of the connections confined to the unit interval. Such refinement is accomplished by the RSL that is quite complementary to the GA; while RSL could be easily trapped in a local minimum, it leads to a detailed solution. The complete learning mode is composed then as a sequence of GA followed by the RSL, let us express as GA-RSL (two-step optimization). Owing to space limitations, RSL is not mentioned in this paper. For more details about RSL, please refer to [11,12].

5 Experimental Results

Generally, the current time delay is affected by the past time delay. Therefore, time delay can be regarded as a Markov chain, and we can assume that there is a correlation between the current time delay and the past time delay. Based on this theoretical foundation we apply the cascade fuzzy neural networks to predict the current time delay of the NCS. To collect the time delay data we used Bluetooth-based NCS as shown in Fig. 8. The wired transmission time has been compared to the wireless (Bluetooth) transmission time to collect the time delay data for the same signal transmission.

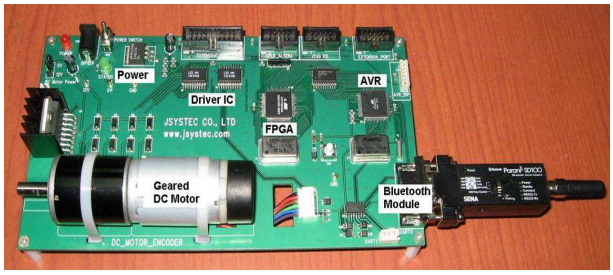

Fig. 8: Considered target NCS.

For the prediction of the time delay 800 data have been collected as a time series using the system in Fig. 8. We set fixed sliding data window as 20, and then 780 feature vectors were acquired. Among them 50% (390) were used for the train and test the cascade architectures of networks, respectively. Each feature is discretized using 3-uniformly distributed Gaussian membership function with an overlap of 0.5. We again exercised the learning scenarios that GA optimizes the input sequence and binary connections for different number of input and then find the best result of them, respectively [8]. For the best result the RSL refines the connections of the neurons positioned in the unit interval. All logic processors have two AND neurons in their hidden layers. In the experiments, we limit the number of input variables and consider only 7 number of input. The selection of these variables is also genetically optimized and in this way we can reduce the size of the model as well as reveal a hierarchy of the input fuzzy sets [8].

The used parameters, acquired from the trial and error method, for the optimization are described as follows:

GA parameters

-Population size: 100
-Maximum generation: 200
-Crossover rate: 0.8
-Mutation rate: 0.03

RSL parameters

-Learning rate: 0.05
-λ: 5
-Maximum iteration number: 2000

Once the cascade architectures of fuzzy neural networks have been optimized using the training data, the testing data (independent from the training data) were considered to verify the trained networks.

The optimization results shown as Root Mean Square Error(RMSE) are described in Table 1.

Table 1: RMSE after the optimization of GA and RSL.

	After GA	After RSL
Training	0.412	0.283
Testing	0.431	0.297

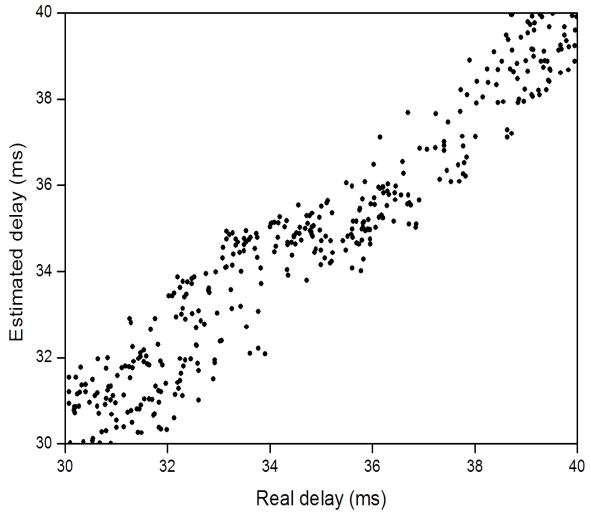

Fig. 9: The prediction results for the testing data.

Fig. 9 shows the testing data prediction results comparing the real delay and the predicted delay. The closer point to the diagonal has the small prediction error.

6 Conclusion

This paper applied the cascade architectures of fuzzy neural networks optimized by GA-RSL to predict the time delay of the NCS. As mentioned above, time varying sampling period has an advantage over the fixed sampling period with respect to the cancelation of the time delay part in the system equation. As is shown in the experiment results, the considered method has a potential ability to predict the time delay of the NCS.

For further research the variety of dada acquired from the different experimental circumstances (different wireless communication paths) are needed. Moreover, this trained predictor need to be included in the NCS to decide the time varying sampling time and control the real systems.

References

[1] T. Jia, Y. Niu and X. Wang, H control for networked systems with data packet dropout, International Journal of Control, Automation, and Systems, **8**, 198-203 (2010).
[2] W. Zhang, M. S. Branicky, and S. M. Phillips, Stability of networked control systems, IEEE Control Systems Magazine, **21**, 84-99 (2001).
[3] G. C. Walsh and H. Ye, Scheduling of networked control systems, IEEE Control Systems Magazine, **21**, 57-65 (2001).
[4] K. C. Lee, S. Lee, and M. H. Lee, Remote fuzzy logic control of networked control system via Probibus-DP, IEEE Trans. Industrial Electronics, **50**, 784-792 (2003).
[5] Y. Zheng, H. Fang, and H. O. Wang, Takagi-Sugeno fuzzy-model-based fault detection for networked control systems with Markov delays, IEEE Trans. Systems, Man, and Cybernetics-Part B, **36**, 924-929 (2006).
[6] D. S. Kim, Y. S. Lee, W. H. Kwon, and H. S. Park, Maximum allowable delay bounds of networked control systems, Control Engineering Practice, **11**, 1301-1313 (2003).
[7] J. Yi, Q. Wang, D. Zhao, and J. T. Wen, BP neural network prediction-based variable-period sampling approach for networked control systems, Applied Mathematics and Computation, **185**, 976-988 (2007).
[8] W. Pedrycz, M. Reformat, and C. W. Han, Cascade architectures of fuzzy neural networks, Fuzzy Optimization and Decision Making, **3**, 5-37 (2004).
[9] Y. H. Lee, C. W. Han, and T. S. Kim, 3D facial recognition with soft computing, Lecture Notes in Computer Science, **4650**, 194-205 (2008).
[10] D. E. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading, MA, (1989).
[11] C. W. Han and J. I. Park, Design of a Fuzzy Controller using Random Signal-based Learning Employing Simulated Annealing, Proc. of the 39th IEEE Conference on Decision and Control, 396-397 (2000).
[12] C. W. Han and J. I. Park, A Study on Hybrid Genetic Algorithms using Random Signal-based Learning Employing Simulated Annealing, Proc. of the 2001 American Control Conference, 198-199 (2001).

Chang-Wook Han received the B.S., M.S., and Ph.D. degrees in Electronic Engineering from Yeungnam University in 1994, 1996, and 2002, respectively. He joined the faculty of the Department of Electrical Engineering, Dong-Eui University in 2008, where he has been an assistant professor. His research interests include fuzzy-neural networks, intelligent control, computational intelligence and its applications.