Original Article
International Journal of Fuzzy Logic and Intelligent Systems
Vol. 14, No. 4, December 2014, pp. 276-287
http://dx.doi.org/10.5391/IJFIS.2014.14.4.276

ISSN(Print) 1598-2645
ISSN(Online) 2093-744X

On Neural Fuzzy Systems

Shun-Feng Su and Jen-Wei Yeh
Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

Abstract

Neural fuzzy system (NFS) is basically a fuzzy system that has been equipped with learning capability adapted from the learning idea used in neural networks. Due to their outstanding system modeling capability, NFS have been widely employed in various applications. In this article, we intend to discuss several ideas regarding the learning of NFS for modeling systems. The first issue discussed here is about structure learning techniques. Various ideas used in the literature are introduced and discussed. The second issue is about the use of recurrent networks in NFS to model dynamic systems. The discussion about the performance of such systems will be given. It can be found that such a delay feedback can only bring one order to the system not all possible order as claimed in the literature. Finally, the mechanisms and relative learning performance of with the use of the recursive least squares (RLS) algorithm are reported and discussed. The analyses will be on the effects of interactions among rules. Two kinds of systems are considered. They are the strict rules and generalized rules and have difference variances for membership functions. With those observations in our study, several suggestions regarding the use of the RLS algorithm in NFS are presented.

Keywords: Neural fuzzy systems, Structuring learning, Delay feedback networks, Recursive least squares

1. Introduction

System identification is a very important issue in system engineering. In recent decades, neural networks [1-3] and fuzzy systems [4-7] are often used to model complicated systems. In these modeling approaches, the task is to obtain a set of fuzzy rules or a neural network that can overall act like the system to be modeled. These approaches can construct systems directly from the input-output relationship without the use of any domain knowledge. Thus, they are often referred to as model-free estimators [1]. Neural fuzzy system (NFS) [8-10] is basically a fuzzy system that has been equipped with learning capability adapted from the learning idea used in neural networks. Due to their outstanding system modeling capability, NFS have been widely employed in various applications. Basically, the main concern of NFS is learning. As mentioned, NFS is a model-free estimator. Thus, it is natural to consider learning as the main issue for system performance. In fact, the structure used to learn may also bring significant differences for those approaches. This can be seen from [11, 12] about the comparison of fuzzy systems and neural networks. In this article, we intend to discuss several ideas regarding the learning of NFS for modeling systems.

The first issue discussed here is about structure learning techniques. It is easy to see that when the number of input variables or the number of fuzzy sets for a variable increase the

Received: Nov. 5, 2014
Revised : Dec. 2, 2014
Accepted: Dec. 7, 2014

Correspondence to: Corresponding author
(sfsu@mail.ntust.edu.tw)

fuzzy rule numbers will exponentially increase. But, usually, meaningful data patterns do not spread out in the whole region. Thus, it is not necessary to use all possible rules in the system for learning. Then to define which rules should be used can be conducted through the so-called self-organization process. This process is usually referred to as the structure learning stage. It is to define rules from data. Such an idea is first proposed in [13] and later on, various approaches were proposed. Those ideas are introduced in this article.

The second issue is about the use of recurrent networks in NFS to model dynamic systems. This kind of approach is often called recurrent NFS or RNFS in the literature. Recurrent networks are those networks of which the inputs of some nodes are from following layers so that it forms loops. RNFS is to have feedback links from some layers to some nodes (usually are input nodes). In this approach, the feedback links are with delay and it can be found that in this case, there is no signal chasing phenomenon as expected for recurrent networks. Thus, RNFS does not have various problems considered in recurrent networks, like stability. To distinguish this difference, in our study, it is referred to as delay feedback NFS instead of recurrent NFS. In this article, the discussion about the performance of such systems will be given. It can be found [14] that such a delay feedback can only bring one order to the system not all possible order as claimed in the literature.

Finally, we will consider the use of the issue of using recursive least square (RLS) algorithms for the learning of fuzzy rule consequences and the rule correlation effects in NFS. Usually the consequence parts of NFS are characterized by singletons or linear functions [13, 15, 16]. When linear functions are considered, two different kinds of update rules, Backpropagation (BP) and RLS algorithms can be used for updating those parameters. The BP algorithm is adapted from the learning concept of neural network [11, 17] and is easy to implement. In BP, the current gradient, which can be viewed as the local information is used to update parameters. Thus, BP algorithm may suffer from low convergence speed and/or being trapped in local minima. On the other hand, adaptive neuron-fuzzy inference system (ANFIS) [15] and self constructing neural fuzzy inference network (SONFIN) [13] are to use the RLS algorithm originally proposed in [16] in the learning process. However, in practical applications, there are problems and then various remedy mechanisms may be needed while using RLS for the learning process in NFS. We have analyzed the effects of those approaches in our previous work [18, 19]. In this article, the interaction between rules on consequent part and RLS algorithm will be analyzed. Furthermore, the operation of resetting the covariance matrix is also discussed.

2. General Description of NFS

NFS have been widely used in last two decades. ANFIS [15] is the most-often mentioned NFS. In the learning phase, ANFIS uses all possible combinations of fuzzy sets in defining rules and it can be expected that some rules may be useless. As being equipped with structure learning capability, the SONFIN is proposed in [13]. The structure of SONFIN is created dynamically in the learning process and the system only uses rules that are necessary. For the other parts, there is no different between SONFIN and ANFIS. They are to realize a fuzzy model of the following form:

$$\begin{aligned}&\textit{Rule } i:\\&\textit{If } x_1 \textit{ is } A_{i1} \textit{ and } \cdots \textit{ and } x_n \textit{ is } A_{in}\\&\textit{then } y_i \textit{ is } a_{i0}+a_{i1}x_1+a_{i2}x_2+\cdots+a_{in}x_n,\end{aligned} \tag{1}$$

where A_{ij} is a fuzzy set for input x_j and a_{ij}'s for $j = 1, \dots, n$ are the consequent parameters for the i-th rule. The output of the fuzzy system is to compute the overall output as the weighting sum of all incoming signals as

$$y = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}, \tag{2}$$

where f_i and w_i are the output and the firing strength, respectively, of the i-th rule. By using a hybrid learning procedure [11], ANFIS can tune both the membership function parameters of the premise part and all a_{ij}'s of the consequent part of the fuzzy rules. As mentioned before, SONFIN and ANFIS have the same structure. Since the membership function is tuned in the learning process, those membership functions used are Gaussian functions. With Gaussian functions, the system will tune their means (membership centers) and variances (membership widths).

In the learning phase of SONFIN, rules are created dynamically as learning proceeds upon receiving training data. Three learning processes are conducted simultaneously in SONFIN to define both the premise and consequent structure identification of a fuzzy if-then rule. They are (A) input/output space partitioning, (B) construction of fuzzy rules, and (C) parameter identification. Processes A and B serve as the structure learning, and process C belongs to the parameter learning phase. In the structure learning phase, when a new fuzzy set is needed, its width (variance) can be defined by different initial values. When a small value is considered, it can be regarded as using the linguistic hedge "very." This kind of discussion about fuzzy membership functions is given in [20], in which different parameter identification methods are considered and discussed. The learning process (A) is to partition the input space based on the input data distribution. In the process, when the current existing rules of the system cannot sufficiently cover the new input pattern, the system will created a new rule from this input pattern. To sufficiently cover means the current input pattern cannot have a sufficiently large firing strength from all existing rules. This firing strength threshold will decide the number of input and output clusters generated in the SONFIN and in turn will determine the number of rules used in the system. In other words, this threshold will define the complexity of the system. However, as reported in our previous work [20], different thresholds (or different variances) affect not only the complexity of the system, but also the performance of the RLS algorithm. In this study, the correlation terms in the covariance matrix used in the RLS algorithm will be studied. As mentioned, the dimension of the RLS algorithm covariance matrix is $[(\text{input number}+1)\times(\text{rule number})]^2$. An obvious disadvantage of using RLS is the computational burden when the input number and/or the rule number are large. This is the reason that some approaches may tune the consequence parameters by assuming those rules are independent [13]. But our study shows that such ignorance of the correlation terms may degrade the learning performance. We will further discuss this issue in the next section. After a new rule is generated, the learning process (B) is to define fuzzy rules based on the current input data. The process is straightforward and the details can be found in [13].

Finally, the parameter-identification process is done concurrently with the structure identification process. For simplicity, a single-output case is considered here. The goal is to minimize the cost function $E = \frac{1}{2}(y(t) - y^d(t))^2$, where $y^d(t)$ is the desired output and $y(t)$ is the current output. SONFIN tunes the parameter of the consequent part (i.e., a in Eq. (1)) with the RLS algorithm [8] as

$$a(t+1) = a(t) + P(t+1)u(t+1)(y^d(t) - y(t)), \quad (3)$$

$$P(t+1) = \frac{1}{\lambda}\left[P(t) - \frac{P(t)u(t+1)u^T(t+1)P(t)}{\lambda + u^T(t+1)P(t)u(t+1)}\right], \quad (4)$$

where t is the iteration number, u is the current input vector, P is referred to as the covariance of the estimation for a, and λ is a forgetting factor in the range between 0 to 1 [21, 22]. In fact, the forgetting factor is originally employed to deal with time varying problems. However, the problem here is the parameters in the premise part are also tuned and thus the system matrix is no longer a constant and then a forgetting factor is employed to cope with this problem [11]. But, it can be expected that such a approach may also introduce other problems. Thus, a usual mean to avoid being trapped in local minima is to reset the covariance matrix P after a period of training.

As mentioned, in SONFIN, the parameters of the premise part (i.e., those means (m_{ij}) and variances (σ_{ij}) of the membership functions) are also tuned. The tuning algorithm used for those parameters is simply the BP learning algorithm. The details can be found in [13]. In this study, in order to reduce the effects from the change of the premise part, a small value of the learning constant in the BP algorithm is selected. The same idea is discussed in our previous work [20].

3. Structure Learning for NFS

In this section, several self-organization learning ideas used in the structure leaning for of NFS will be introduced. As mentioned, it is not necessary to use all possible rules in the system for learning. To construct a fuzzy model, the fuzzy subspaces required for defining fuzzy partitions in premise parts and the parameters required for defining functions in consequent parts must both be obtained. In the original Takagi-Sugeno-Kang (TSK) modeling approach [16], users must define fuzzy subspaces in advance, and then, the parameters in consequences are obtained through the RLS algorithm. This simple idea is then employed in ANFIS [15]. It can be found that those approaches must use all possible rules. In the following, we shall discuss ways of defining rules.

As mentioned in the above section, in the learning phase of SONFIN [13], rules are created dynamically as learning proceeds upon receiving training data. Three learning processes are conducted simultaneously in SONFIN to define both the premise and consequent structure identification of a fuzzy if-then rule. This kind of learning approach is somehow said to be an online learning approach. In that approach, even though in this structure learning phase, the process can be on line, the later learning algorithm, like BP is not suitable for online learning due to slow convergent property of the BP learning algorithm, which usually needs hundreds or thousands of epochs to converge. This effect can be seen from [23] that even a simple direct-generation-from-data approach [6] can outperform SONFIN in an online learning situation. Thus, it can be found that various structure learning approaches that are not online approaches were proposed in the later studies.

A simple idea is to use the original structure learning in [17]. In that approach, the input space is first divided into fuzzy subspaces through a clustering algorithm called the Kohonen self-organized network [24] according to only the input portion of training data. It can be found that the obtained fuzzy subspaces may not cover the entire input space. Note that this approach is similar to that in SONFIN and the fuzzy partition is only based on input data. In other words, rules are defined only based on the distribution of input portion of training data. Usually, in the literature, this structure learning stage is called the coarse tuning stage. After the fuzzy subspaces are defined, the system is approximated in each subspace by a linear function, through supervised learning algorithms, such as BP or least-square learning algorithms [2, 6, 7, 11]. In the meantime, the fuzzy subspaces may also be tuned usually through BP learning algorithms. This stage is called the fine tuning stage. Note that each subspace corresponding to one fuzzy rule is supposed to have a simple geometry in the input-output space, normally having the shape of ellipsoid [25]. In fact, other fuzzy clustering algorithms, such as the fuzzy C-mean (FCM) [26, 27] are also suitable to define fuzzy subspaces for fuzzy modeling. In general, the above mentioned approaches partition fuzzy subspaces based on only the clustering in the input space of training data and do not consider whether the output portion of the training data supports such clustering or not. In other words, such approaches do not account for the interaction between input and output variables.

Similar to the use of the AND operation in the fuzzy reasoning process, the match of a rule for a data set requires that the input portion and the output portions must be both matched with the premise part and the consequence part of the rule. Thus, to construct a rule, the input data and the output data must be both considered. Hence, the authors in [25, 27] considered the product space of input and output variables instead of only the input space in classical clustering algorithms for fuzzy modeling. However, these approaches and the above approaches still define fuzzy subspaces in a clustering manner and do not take into account the functional properties in TSK fuzzy models. In other words, in those approaches, training data that are close enough instead of having a similar function behavior are said to be in the same fuzzy subspace. Thus, if the consequence part is a fuzzy singleton, it is nice. But, if the consequence part is a linear function as usual be, such a structure learning behavior may not be proper. As a result, the number of fuzzy subspaces may tend to be more than enough.

In order to account for the linear function property, another approach is proposed in [28]. In the approach, fuzzy subspaces and the functions in consequent parts are simultaneously identified through the use of the fuzzy C-regression model (FCRM) clustering algorithm. Thus, not like to calculate a distance to a point (cluster center) in clustering algorithms, the approach is to calculate the distance to a line in a linear regression approach. This distance is then used to define an error function used in the cost function. The idea of this kind of approaches is to find a set of training data whose input-output relationship is somehow a linear function, and then, those training data can be clustered into one fuzzy subspace. Similar to other approaches, the structure learning behavior does not incorporate the optimization process in modeling and hence, the fine tuning stage supervised learning algorithms can further be used to adjust the model.

It should be note that in the above clustering or regression algorithms, users must assign the cluster number, which is supposed to be unknown. Another idea proposed in [29] is to employ the so-called robust competitive agglomeration (RCA) clustering algorithm used in computer vision and pattern recognition [30] to form subspace. In this approach, the cluster number is determined in the clustering process. Besides, the clustering process begins from the whole data set and then can reduce the effects of data sequence. As a result, it can have fast convergent speed and better performance as claimed in the literature.

Another consideration is about outliers in training data. The intuitive definition of an outlier [31] is "*an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.*" Outliers may occur due to various reasons, such as erroneous measurements or noisy data from the tail of noise distribution functions. When outliers exist, the networks may try to fit those improper data and thus, the obtained systems may have the phenomenon of overfitting [32-34]. The above structure learning algorithms are all based on the principle of least square error minimization and are easily affected by outliers, which should be degraded in the clustering process [28, 30, 35-37]. In the fine-tuning process, classical supervised learning algorithms such as gradient descent approaches are used. When training data are corrupted by large noise, such as outliers, traditional BP learning schemes usually cannot come up with acceptable performance [38, 39]. Based on the principle of robust statistics, various robust learning algorithms have been proposed in the neural network community [40-45]. Most of them are to replace the square term in the cost function by a so-called loss function. For the FCRM approach, it is difficult to adopt such robust learning concept. In [29], a novel approach termed as the robust fuzzy regression agglomeration (RFRA) clustering algorithm is shown to have robust learning effects against outliers. While the RFRA clustering algorithm determines the parameters in both premise and consequent parts, the approach also employs a robust learning algorithm to fine-tune the obtained fuzzy model. The simulation results shown in [29] have indeed shown superior performance of the proposed algorithm.

4. Delay Feedback and Dynamic Modeling

It can be easily found that NFS can only model static system due to no memory in keeping previous states. When a dynamic system is the modeling target, an often-used approach is to use all necessary past inputs and outputs of the system as explicit inputs. Such a structure is referred to as the nonlinear autoregressive with exogenous inputs (NARX) model [14, 46]. Another kind of approaches [14, 47-49] is to feedback the outputs of internal nodes in networks with a time delay. Those methods have shown to have nice modeling accuracy on modeling dynamical systems in some examples. Such networks are usually called recurrent networks in the literature [47-49]. However, it can be found that even though the system indeed have loop in the connections, the feedback is with a time delay and then the system does not have actual loop. Thus, in our study, it is called the delay feedback networks [14].

Delay feedback networks are to use internal memories to catch internal states. We can also introduce delay feedbacks to account for dynamical behaviors in SONFIN. However, where to put those delay links is not so straightforward because there are semantics associated with those layers. Different from that used in [47, 48], another delay feedback approach for SONFIN, termed as the additive delay feedback neural fuzzy network (ADFNFN) is proposed in [14]. The basic idea is to adopt the autoregression and moving-average (ARMA) type [50] of modeling approaches. In an ARMA model, the output is predicted as a linear combination of the current input and previous inputs and outputs. In other words, previous outputs are included into the prediction model in an additive manner. In fact, NARX models can generally be viewed as a nonlinear extension of ARMA models because they mix all inputs in an additive manner (linear integration functions in neural networks and linear consequence functions in SONFIN). From simulations, it is evident that the proposed ADFNFN can have better learning performances than those approaches proposed in [47] and in [48] do.

It is noted that the prediction for the next step in dynamic systems is always based on previous data for a series-parallel identification scheme [50]. Then, when the step size is small, the possible error generated in one step will also be small. This correspondence will make the traditional root mean square errors (RMSE) not able to truly capture the ideas about how accurate the current model can predict. Another evaluating index called the non-dimensional error index (NDEI) is considered in [51] to evaluate modeling errors. The NDEI is defined as:

$$\text{NDEI} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(T(i) - O(i)\right)^2}}{\delta(T)}, \qquad (5)$$

where N is the number of data, $T(i)$ is the desired output, $O(i)$ is the predicted output and $\delta(T)$ is the averaged change in one sampling time in the target series. In some reports such as [47, 52], the performances shown in their applications seem nice. However, their NDEI is 4.2370 (RMSE $= 0.2585$ (MSE $=$ 0.0668) and $\delta(T) = 0.0610$) in [52] and 3.4426 (RMSE $=$ 0.21 and $\delta(T) = 0.0610$) in [47]. That means their averaged prediction errors are about 423.7% and 344.26%, respectively, of the averaged change in one step. It is unacceptable. In other words, their learning in fact did not converge. As a matter of fact, in [14], by tuning some parameters, the NDEI can converge to 0.4344 (as shown in Table 1) (RMSE $= 0.0265$, $\delta(T) = 0.0610$), which is only 43.44%. Such a result still is not good enough. In [14], there is another approach, in which the NDEI is only 0.2148 (21.48%), which then can be viewed as a nice prediction.

Intuitively, the use of delay feedback could model any order of dynamic systems because those delay elements can be al kind of signals including those delayed ones. However, from [14], it can be found that those delay feedback models can only achieve the accuracy level of order-2 NARX models. In other words, those delay feedback networks seem not able to model the systems as accurately as the NARX models with proper orders do. It is because delay feedback models only use one delay in the modeling approaches, it is somewhat similar to NARX models with order 2. As shown in [14], if two feedback connections are used for each internal state; one is with one delay and the other is with two delays, it can be found that the errors have been significantly reduced when the considered systems are order-3 and order-4 systems. However, the modeling accuracy for the order-4 system is still not good enough. It is clearly evident that the role of the used delay number in a delay feedback model is similar to that of the system order in an NARX model. Thus, it can be concluded that delay feedback network is not necessary because it is more complicated than an NARX model with a proper order and cannot provide better performance.

5. Analysis of the Use of RLS

As mentioned, ANFIS [15] and SONFIN [13] employ the RLS algorithm originally proposed in [16] in the learning process. However, in practical applications, there are problems and then various remedy mechanisms may be needed while using RLS for the learning process in NFS. In this article, the interaction between rules on consequent part and RLS algorithm will be analyzed and the operation of resetting the covariance matrix is also discussed. In the literature, RLS algorithms have been widely used in adaptive filtering, self-tuning control systems and system identification [53]. From the literature, it can be found that there are several advantages in using RLS algorithms, such as fast convergence speed and small estimation errors, especially while the system considered is simple and time invariant. Theoretically, the estimation of RLS is the best estimation under the assumption of Gaussian noise in ideal cases. In practical applications, there are some possible remedies. The advantage of those approaches can be obviously but not always while the system considered becomes complicated. Recently, IRSFNN [49] also present two types of parameter identification steps, which we referred to as the reduced and full covariance matrices for RLS algorithms. To explain the results in [49], the overlap coefficient is employed to define the intersection between fuzzy sets. Similarly in our previous work [20], two types of membership functions are considered to manifest what the differences between full and reduce covariance matrix are. When the system has more interaction between fuzzy sets, it can be found that the off-diagonal parts of the covariance matrix should not be ignored.

In our study, SONFIN [13] is employed as the NFS. In this section, the learning algorithms used in the original SONFIN are considered as a basis. The structure of SONFIN is created dynamically in the learning process and only uses rules that are necessary. It is easy to see that SONFIN can have very nice performance especially when the number of input variables is large. In the learning process, when the firing strength of the current input feature is lower than a threshold, the system will generate a new fuzzy rule for this input feature. In other words, if necessary, SONFIN can generate new fuzzy rules for new input features. After constructing fuzzy rules, the parameters of the premise part and of the consequent part of the NFS are updated through BP and RLS, respectively. It can be found that there are two kinds of interactions; interaction between input data and interaction between consequent structures. In order to have the efficient computation, the consequence parameters of SONFIN are calculated independently among rules [13]. However, it is expected that with the consideration of those rule interactions, it can help in discovering new knowledge among rules and improving the reasoning accuracy [13]. If the membership functions are of less overlapping, no matter what kinds of RLS algorithms are used, there is no much difference in the learning performance [20].

As mentioned in [54-56], those pre-defined membership functions can be modified (or tuned) by some operations, like "very" or "more-or-less". When membership functions are tuned, linguistic variables can bring more human-like thinking with different kinds of membership functions. In SONFIN, the system also has the capability of changing the cores and shapes of membership functions. Consequently, SONFIN can have nice performance in data learning. However, those tuning effects may destroy original features of those membership functions. Since SONFIN only creates rules that are necessary, those original membership functions may contain some characteristics for that data pair. While the membership functions are tuned in the later learning process, those characteristics may be altered and more complex interactions emerge. As mentioned in [57], redundancy interaction usually cannot have significant improvement in performance. Besides, the correlation terms in the covariance matrix required in RLS is another problem. When the dimension of the system to be modeled becomes very large, the computational time required may become infeasible because of the dimension of the covariance matrix is $(\text{the number of rules}\times\text{the number of input variable}+1)^2$. Also, the interaction between BP (used for the tuning of input fuzzy membership functions) and RLS (used for the tuning of the consequence parameters) will be unexpected. In this study, the interaction between rules on consequent part and RLS algorithm will be analyzed and the operation of resetting the covariance matrix is also discussed.

In this section, we will rearrange the analysis in [20] to analyze the performance of using the RLS algorithm with the full covariance matrix and with a reduced covariance matrix. The RLS algorithm with the full covariance matrix is the original approach as Eqs. (3) and (4) where P is a full matrix. To use a reduced matrix is to assume the consequence parameters among rules are independent and the correlation terms between rules are all assumed zeros. Define $[a_{o1}\ a_{11}\ \cdots\ a_{1I}\ \ a_{o2}\ a_{21}\ \cdots\ a_{2I}\ \cdots\ a_{oJ}\ a_{J1}\ \cdots\ a_{JI}]^T$ as the parameter vector, where I is the input dimension + 1 and J is the rule number. For Eq. (3), u is the input vector for the whole consequence part and can be written as.

$$u = \Big[\mathrm{LI}_1^{(5)}(1\ \ x_1 \cdots x_j)\ \ \mathrm{LI}_2^{(5)}(1\ \ x_1 \cdots x_j)\ \ \cdots\ \ \mathrm{LI}_i^{(5)}(1\ \ x_1 \cdots x_j)\Big]^T$$

where $\mathrm{LI}_i^{(5)}$ is the firing strength of the ith rule and the supscript (5) means it is on the layer 5 of the whole structure [13]. For the reduced matrix case, the consequence parameters among rules are assumed to be independent. Thus, for Eq. (3), the input vector of u for the ith rule is

$$u_i = \Big[\mathrm{LI}_i^{(5)}(1\ \ x_1 \cdots x_j)\Big]^T$$

Then the covariance matrix P for each rule is with dimension Jind×Jind, where Jind= I =(the input dimension + 1). There are J (=the rule number) such covariance matrices and the total dimension is $J \times I^2$ while it is $(J \times I)^2$ in the full matrix case.

In [57], three kinds of interaction for sensory inputs are defined. They are *redundancy/negative synergy*, *complementarity/positive synergy* and *independency*. Those interaction types define different situations of the actual value compared to the combination of individual sensors. Those types can also be considered in rule interactions of SONFIN while using the RLS algorithm. While it is in the case of redundancy/negative synergy, the system learning performance will better using the independent RLS algorithm than that of using the full-rule RLS algorithm. While it is in the case of complementarity/positive synergy the full-rule RLS algorithm will perform better than the independent RLS algorithm does. Finally, while they are Independent, two RLS algorithms have no significant difference.

SONFIN updates the membership function and consequent parts simultaneously in the learning process. Interaction in each rule is difficult to manipulate. When BP changes the membership function and the consequent part change by RLS algorithm in one step, new membership functions will change the Layer 5 input, which makes the premise part used in previous RLS calculation changed. Somehow such a change can be viewed as the system is time varying. Thus, there are several methods proposed in the literature to resolve this problem, like to use a forgetting factor in the RLS algorithm, to reduce the learning constant in BP.

Another issue in the use of forgetting factor in the RLS algorithm is about resetting the covariance matrix P. It can be found that resetting the covariance matrix can somehow improve the training performance from our previous study [18]. However the overfitting phenomenon may occur in some cases. By setting the covariance matrix with a large diagonal values, the RLS algorithm will have the capability to focus on the present data. It is called bootstrap [58]. In other words, we set the system turn into local learning phase to reduce the effect from previous learning.

Two functions are considered for illustration for this part of study. First, a simply function ***Sinc*** is considered. The second one is the identification of the ***Macky-Glass*** chaotic series. It is know that by using different thresholds in SONFIN. SONFIN can generate several NFS with different rule complexity. Here we will have two types of fuzzy rules; generalized rules and strict rules. They can be viewed as the interactions in those two types are heavy and slight, respectively.

We continue the simulations in [19]. In the case of $sinc$, 7 and 25 rules are considered for two type fuzzy set membership functions with initial variance value 0.1 and 1. Their corresponding learning performances are given in Table 2. It can easily be found that the same performance for strict 7-rule system. This system type they have less interaction on rules. However, in 25-rule system, using the full P matrix can have better performance than that of using the reduce P matrix. For the generalized system, It is clearly evident that using the full P matrix has better performance than that of using the reduce system. Thus, when the system has less interaction between rule, two different RLS algorithms with the whole system or calculating for each rule individually will have similar performance. But it could not be confirmed the system is in local minimum. Next, the resetting P is taken into account and the results are given in Table 3. From the results, it is observed that except the strict 7 rules system, all systems have better performance than those in Table 2.

While the rule system is of generalized rules cases and the interaction among rules are significant, the original (Full) covariance matrix must be used. However, it can be expected that when the full matrix is considered in the RLS algorithm, the computation problem owing to a large dimension of the covariance matrix may be there. In order to have a better computational efficiency, the learning can use different RLS algorithms after resetting the P matrix. Two simulations are shown in Table 1. Except the strict 7-rule system, for the change from full to reduce P matrix case after resetting, the performance in Table 1 is between with reset and non-reset full P matrix system compared to those in Tables 2 and 3. But in the case of change from reduce to full P matrix after resetting, it has very nice performance in generalized cases.

The second simulation is the identification of the *Macky-Glass* chaotic series. It has 200 2-dimension points for input feature. The same cases situations as above are considered. Tables 4-6 show those learning performances after 1100 epochs. The same conclusions can be observed in this example too. Thus, we can claim that the use of change from reduce to full P matrix after resetting can have nice learning performance on error and can achieve computational efficiency also.

Now, more complexity for the Macky-Glass function is considered. In [49], four past values are used for training. 1000 patterns are generated in this study. First 500 patterns are used for training and the other 500 patterns are reserved for testing. After 100 epochs, the error for the generalized system with the use of full matrix is significantly better than others. In other words, most researchers recommend to use the reduce covariance matrix to save computational burden, it can be found that the error difference is not slight. In practice, the change of using different covariance matrices after resetting can be employed to improve learning performance. In this study, the covariance matrix is reset to the initial status at 10 epochs. The results are shown in Table 7. Comparing the performance between resetting and non-resetting, two cases become worse; strict rules with full covariance matrix system and generalized rules with reduce covariance matrix system. Some approached are considered here. First, the initial value of the covariance matrix diagonal parts is reduced to increase the effects from previous learning. The simulation results are shown in Table 8. Secondly, we reset the system with the full covariance matrix and the simulation result shown in Table 9.

From those results, it can be found that in this case only resetting the covariance matrix is not enough to improve the learning performance. In Table 8, the learning performance at 100 epochs 0.004281 is better than that in Table 7. Also for generalized rules with reduce covariance matrix system case, to reduce the value of covariance matrix diagonal parts can indeed prevent the system unstable so as to catch up with the generalized rules with the reduce covariance matrix system results in Table 10.

Table 1. Learning errors (RMSE) for ***Sinc*** with change algorithms

Rule #/Epoch	Strict		Generalized	
	500	600	500	600
Full to reduce P matrix				
7	0.0791	0.0791	0.0104	0.0084
25	0.0135	0.0113	0.0037	0.0026
Reduce to full P matrix				
7	0.0791	0.0791	0.0580	0.0121
25	0.0163	0.0120	0.0159	0.0023

RMSE, root mean square errors.

Table 2. Learning errors (RMSE) for ***Sinc***

Rule #/Epoch	Strict		Generalized	
	500	600	500	600
Full P matrix				
7	0.0791	0.0791	0.0104	0.0095
25	0.0135	0.0127	0.0037	0.0032
Reduce P matrix				
7	0.0791	0.0791	0.0580	0.0561
25	0.0163	0.0154	0.0159	0.0150

RMSE, root mean square errors.

Table 3. Learning errors (RMSE) for ***Sinc*** with reset

Rule #/Epoch	Strict		Generalized	
	500	600	500	600
Full P matrix				
7	0.0791	0.0791	0.0104	0.0073
25	0.0135	0.0111	0.0037	0.0012
Reduce P matrix				
7	0.0791	0.0791	0.0580	0.0303
25	0.0163	0.0123	0.0159	0.0118

RMSE, root mean square errors.

Table 4. Learning errors (RMSE) for ***Macky-Glass***

Rule #/Epoch	Strict		Generalized	
	1000	1100	1000	1100
Full P matrix				
7	0.0176	0.0176	0.0064	0.0061
15	0.0081	0.0079	0.0050	0.0047
Reduce P matrix				
7	0.0176	0.0176	0.0096	0.0095
15	0.0083	0.0081	0.0081	0.0079

RMSE, root mean square errors.

6. Conclusions

NFS is a nice modeling technique. In this article, we make a brief survey about its use in various aspects. For the structure

Table 5. Learning errors (RMSE) for ***Macky-Glass*** with reset

Rule #/Epoch	Strict		Generalized	
	1000	1100	1000	1100
Full P matrix				
7	0.0176	0.0175	0.0064	0.0056
15	0.0081	0.0072	0.0050	0.0040
Reduce P matrix				
7	0.0176	0.0175	0.0096	0.0091
15	0.0083	0.0073	0.0081	0.0065

RMSE, root mean square errors.

Table 6. Learning errors (RMSE) for ***Macky-Glass*** with change algorithm

Rule #/Epoch	Strict		Generalized	
	1000	1100	1000	1100
Full to reduce P matrix				
7	0.0176	0.0175	0.0064	0.0059
15	0.0081	0.0072	0.0050	0.0040
Reduce to full P matrix				
7	0.0176	0.0175	0.0096	0.0075
15	0.0083	0.0072	0.0081	0.0047

RMSE, root mean square errors.

Table 7. Learning errors (RMSE) for 4-input ***Macky-Glass*** with reset after 10 epochs

Epochs	Strict	Generalized
	100	100
Full P matrix		
Training	0.009606	0.000874
Testing	0.014053	0.001479
Reduce P matrix		
Training	0.004258	4671.207
Testing	0.007109	7306.332

RMSE, root mean square errors.

Table 8. Learning errors (RMSE) for 4-input ***Macky-Glass*** with reset and reduce the covariance matrix diagonal values from 1000 to 1

The RMS error at 100 epoch	
Strict rules full covariance matrix system	
Training	0.004281
Testing	0.0071
Generalized rulesreduce covariance matrix system	
Training	0.005222
Testing	0.006421

RMSE, root mean square errors.

Table 9. Learning errors (RMSE) for 4-input ***Macky-Glass*** with full covariance matrix reset

The RMS error at 100 epoch	
Training	0.000913
Testing	0.00153

RMSE, root mean square errors.

Table 10. Learning errors (RMSE) for 4-input ***Macky-Glass***

Epochs	Strict		Generalized	
	10	100	10	100
Full P matrix				
Training	0.004289	0.004259	0.002299	0.001523
Testing	0.007097	0.007105	0.003141	0.002385
Reduce P matrix				
Training	0.004289	0.004259	0.00617	0.005027
Testing	0.007094	0.007103	0.007501	0.006355

RMSE, root mean square errors.

learning, from the basic idea used in the original approach to the several different approaches are introduced. Hopefully, readers can understand those ideas and can select a suitable approach for their applications. For the dynamic system modelling, the idea of recurrent network, which has been widely used in the literature, is discussed. From the results reported in [14], it is clearly evident that such a methodology is not a good approach even though lots of researchers have used this idea in their approaches. A simple approach use a sufficient order in a traditional NARX model will have the best results. Finally, the effects on rule interaction in the use or RLS are reported in this article. It can be observe that to use the reduced matrix, the computational burden can be reduced significantly but the error may be large especially for complicated systems. To reserve the computational efficiency and to have nice learning performance on error, we propose to add one final step in SONFIN by resetting the P matrix and then performing the RLS algorithm with the full covariance matrix.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

References

[1] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Englewood Ciffs, NJ: Prentice Hall, 1992.

[2] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, Upper Saddle River, NJ: Prentice Hall PTR, 1996.

[3] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks,* vol. 2, no. 5, pp. 359-366, 1989. http://dx.doi.org/10.1016/0893-6080(89)90020-8

[4] L. A. Zadeh, “Outline of a new approach to the analysis of complex systems and decision processes,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-3, no. 1, pp. 28-44, Jan. 1973. http://dx.doi.org/10.1109/TSMC.1973.5408575

[5] W. Pedrycz, “Structured fuzzy models,” *Cybernetics and Systems*, vol. 16, no. 1, pp. 103-117, Jan. 1985. http://dx.doi.org/10.1080/01969728508927757

[6] L. X. Wang and J. M. Mendel, “Generating fuzzy rules by learning from examples,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 6, pp. 1414-1427, Nov. 1992. http://dx.doi.org/10.1109/21.199466

[7] R. R. Yager and D. P. Filev, *Essentials of Fuzzy Modeling and Control*, New York, NY: Wiley, 1994.

[8] S. Horikawa, T. Furuhashi, and Y. Uchikawa, “On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm,” *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 801-806, Sep. 1992. http://dx.doi.org/10.1109/72.159069

[9] K. Tanaka, M. Sano, and H. Watanabe, “Modeling and control of carbon monoxide concentration using a neuro-fuzzy technique,” *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 3, pp. 271-279, Aug. 1995. http://dx.doi.org/10.1109/91.413233

[10] Y. Lin and G. A. Cunningham, III, “A new approach to fuzzy-neural system modeling,” *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 2, pp. 190-198, May 1995. http://dx.doi.org/10.1109/91.388173

[11] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Upper Saddle River, NJ: Prentice Hall, 1997.

[12] S. F. Su and K. Y. Chen, “Conceptual discussions and benchmark comparison for neural networks and fuzzy systems,” *Differential Equations and Dynamical Systems*, vol. 13, no. 1, pp. 35-61, 2005.

[13] C. F. Juang and C. T. Lin, “An online self-constructing neural fuzzy inference network and its applications,” *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 12-32, Feb. 1998. http://dx.doi.org/10.1109/91.660805

[14] S. F. Su and F. Y. P. Yang, “On the dynamical modeling with neural fuzzy networks,” *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1548-1553, Nov. 2002. http://dx.doi.org/10.1109/TNN.2002.804313

[15] J. S. R. Jang, “ANFIS: adaptive-network-based fuzzy inference system,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665-685, May 1993. http://dx.doi.org/10.1109/21.256541

[16] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, no. 1, pp. 116-132, Jan. 1985. http://dx.doi.org/10.1109/TSMC.1985.6313399

[17] C. T. Lin and C. S. G. Lee, “Neural-network-based fuzzy logic control and decision system,” *IEEE Transactions on Computers*, vol. 40, no. 12, pp. 1320-1336, Dec. 1991. http://dx.doi.org/10.1109/12.106218

[18] J. W. Yeh, S. F. Su, J. T. Jeng, and B. S. Chen, “On learning analysis of neural fuzzy systems,” in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ)*, Barcelona, Italy, July 18-23, 2010, pp. 1-6. http://dx.doi.org/10.1109/FUZZY.2010.5584389

[19] J. W. Yeh, S. F. Su, and I. Rudas, “Analysis of using RLS in neural fuzzy systems,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Anchorage, AK, October 9-12, 2011, pp. 1831-1836. http://dx.doi.org/10.1109/ICSMC.2011.6083937

[20] J. W. Yeh and S. F. Su, “Learning analysis for correlation of fuzzy rules in applying RLS for neural fuzzy systems,” in *Proceedings of the IEEE International Conference on Granular Computing*, Hangzhou, China, August 11-13, 2012, pp. 609-613. http://dx.doi.org/10.1109/GrC.2012.6468690

[21] Y. Zhang and X. R. Li, “A fast U-D factorization-based learning algorithm with applications to nonlinear system modeling and identification,” *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 930-938, Jul. 1999. http://dx.doi.org/10.1109/72.774266

[22] Y. S. Cho, S. B. Kim, and E. J. Powers, “Time-varying spectral estimation using AR models with variable forgetting factors,” *IEEE Transactions on Signal Processing*, vol. 39, no. 6, pp. 1422-1426, Jun. 1991. http://dx.doi.org/10.1109/78.136549

[23] S. R. Huang, "Analysis of model-free estimators: applications to stock market with the use of technical indices," M.S. thesis, National Taiwan University of Science and Technology, Taipei, Taiwan, 1999.

[24] T. Kohonen, *Self-organization and Associative Memory*, 3rd ed., New York, NY: Springer-Verlag, 1989.

[25] J. A. Dickerson and B. Kosko, "Fuzzy function approximation with ellipsoidal rules," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 4, pp. 542-560, Aug. 1996. http://dx.doi.org/10.1109/3477.517030

[26] A. Kroll, "Identification of functional fuzzy models using multidimensional reference fuzzy sets," *Fuzzy Sets and Systems*, vol. 80, no. 2, pp. 149-158, Jun. 1996. http://dx.doi.org/10.1016/0165-0114(95)00140-9

[27] F. Klawonn and R. Kruse, "Constructing a fuzzy controller from data," *Fuzzy Sets and Systems*, vol. 85, no. 2, pp. 177-193, Jan. 1997. http://dx.doi.org/10.1016/0165-0114(95)00350-9

[28] E. Kim, M. Park, S. Ji, and M. Park, "A new approach to fuzzy modeling," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 3, pp. 328-337, Aug. 1997. http://dx.doi.org/10.1109/91.618271

[29] C. C. Chuang, S. F. Su, and S. S. Chen, "Robust TSK fuzzy modeling for function approximation with outliers," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 6, pp. 810-821, Dec. 2001. http://dx.doi.org/10.1109/91.971730

[30] H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 450-465, May 1999. http://dx.doi.org/10.1109/34.765656

[31] D. M. Hawkins, *Identification of Outliers*, New York, NY: Chapman and Hall, 1980.

[32] M. Smith, *Neural Networks for Statistical Modeling*, New York, NY: Van Nostrand Reinhold, 1993.

[33] W. S. Sarle, "Stopped training and other remedies for overfitting," in *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, Pittsburgh, PA, June 21-24, 1995, pp. 352-360.

[34] P. L. Bartlett, "For valid generalization, the size of the weights is more important than the size of the network," *Advances in Neural Information Processing Systems*, vol. 9, pp. 134-140, 1996.

[35] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, New York, NY: Plenum Press, 1981.

[36] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Englewood Cliffs, NJ: Prentice Hall, 1988.

[37] R. N. Dave and R. Krishnapuram, "Robust clustering methods: a unified view," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 2, pp. 270-293, May 1997. http://dx.doi.org/10.1109/91.580801

[38] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, New York, NY: J. Wiley, 1993.

[39] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*, New York, NY: Wiley, 1987.

[40] D. S. Chen and R. C. Jain, "A robust backpropagation learning algorithm for function approximation," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 467-479, May 1994. http://dx.doi.org/10.1109/72.286917

[41] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 240-254, Mar. 1994. http://dx.doi.org/10.1109/72.279188

[42] K. Liano, "Robust error measure for supervised neural network learning with outliers," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 246-250, Jan. 1996. http://dx.doi.org/10.1109/72.478411

[43] V. D. Sánchez A, "Robustization of a learning method for RBF networks," *Neurocomputing*, vol. 9, no. 1, pp. 85-94, Sep. 1995. http://dx.doi.org/10.1016/0925-2312(95)00000-V

[44] L. Huang, B.-L. Zhang, and Q. Huang, "Robust interval regression analysis using neural networks," *Fuzzy Sets and Systems*, vol. 97, no. 3, pp. 337-347, Aug. 1998. http://dx.doi.org/10.1016/S0165-0114(96)00325-9

[45] C. C. Chuang, S. F. Su, and C. C. Hsiao, "The annealing robust backpropagation (ARBP) learning algorithm," *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1067-1077, Sep. 2000. http://dx.doi.org/10.1109/72.870040

[46] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 1999.

[47] C. F. Juang and C. T. Lin, "A recurrent self-organizing neural fuzzy inference network," *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 828-845, Jul. 1999. http://dx.doi.org/10.1109/72.774232

[48] C. H. Lee and C. C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 4, pp. 349-366, Aug. 2000. http://dx.doi.org/10.1109/91.868943

[49] Y. Y. Lin, J. Y. Chang, and C. T. Lin, "Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 2, pp. 310-321, Dec. 2013. http://dx.doi.org/10.1109/TNNLS.2012.2231436

[50] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27, Mar. 1990. http://dx.doi.org/10.1109/72.80202

[51] J. Espinosa and J. Vandewalle, "Constructing fuzzy models with linguistic integrity from numerical data-AFRELI algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp. 591-600, Oct. 2000. http://dx.doi.org/10.1109/91.873582

[52] P. S. Sastry, G. Santharam, and K. P. Unnikrishnan, "Memory neuron networks for identification and control of dynamical systems," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 306-319, Mar. 1994. http://dx.doi.org/10.1109/72.279193

[53] S. S. Haykin, *Adaptive Filter Theory*, 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 1991.

[54] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-I," *Information Sciences*, vol. 8, no. 3, pp. 199-249, 1975. http://dx.doi.org/10.1016/0020-0255(75)90036-5

[55] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-II," *Information Sciences*, vol. 8, no. 4, pp. 301-357, 1975. http://dx.doi.org/10.1016/0020-0255(75)90046-8

[56] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-III," *Information Sciences*, vol. 9, no. 1, pp. 43-80, 1975. http://dx.doi.org/10.1016/0020-0255(75)90017-1

[57] M. Grabisch, "The representation of importance and interaction of features by fuzzy measures," *Pattern Recognition Letters*, vol. 17, no. 6, pp. 567-575, May 1996. http://dx.doi.org/10.1016/0167-8655(96)00020-7

[58] R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee, P. S. Lewis, and S. Qian, "Function approximation and time series prediction with neural networks," in *Proceedings of the IJCNN International Joint Conference on Neural Networks*, San Diego, CA, June 17-21, 1990, pp. 649-665. http://dx.doi.org/10.1109/IJCNN.1990.137644

Shun-Feng Su received the B.S. degree in electrical engineering, in 1983, from National Taiwan University, Taiwan, R.O.C., and the M.S. and Ph.D. degrees in electrical engineering, in 1989 and 1991, respectively, from Purdue University, West Lafayette, IN.

He is now a chair professor of the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan, R.O.C. He is an IEEE fellow and CACS fellow. He has published more than 190 refereed journal and conference papers in the areas of robotics, intelligent control, fuzzy systems, neural networks, and non-derivative optimization. His current research interests include computational intelligence, machine learning, virtual reality simulation, intelligent transportation systems, smart home, robotics, and intelligent control.

Dr. Su is very active in various international/domestic professional societies. He is now the president of the Taiwan Association of System Science and Engineering and the president-elect of International Fuzzy Systems Association. He now is also in the boards of governors of the Chinese Automatic Control Society, the Taiwan Society of Robotics, and the Taiwan Fuzzy System Association of. Dr. Su also acted as program chair, program co-chair, or PC members for various international and domestic conferences. Dr. Su currently serves as associate editors of IEEE Transactions on Cybernetics, IEEE Transactions on Fuzzy Systems, and IEEE Access. Since 2015, he will be the editor-in-chief of International Journal of Fuzzy Systems.

Jen-Wei Yeh was born in Taipei, Taiwan, R.O.C., in 1986. He received the B.S. degree in National Taipei University of Technology, Taipei, Taiwan, R.O.C. He is currently working toward the Ph.D. degree at National Taiwan University of Science and Technology, Taipei, Taiwan. His research interests include fuzzy logic systems, adaptive control, and intelligent control.