A New Fast Algorithm for Fuzzy Rule Selection

Barbara Pizzileo, Kang Li

***Abstract*— This paper investigates the selection of fuzzy rules for fuzzy neural networks. The main objective is to effectively and efficiently select the rules and to optimize the associated parameters simultaneously. This is achieved by the proposal of a fast forward rule selection algorithm (FRSA), where the rules are selected one by one and a residual matrix is recursively updated in calculating the contribution of rules. Simulation results show that, the proposed algorithm can achieve faster selection of fuzzy rules in comparison with conventional orthogonal least squares algorithm, and better network performance than the widely used error reduction ratio method (ERR).**

I. Introduction

Fuzzy neural networks represent a large class of neural networks that combine the advantages of associative memory networks (e.g. B-splines, radial basis functions and support vector machines) with improved transparency, a critical issue for nonlinear modelling using conventional neural networks. For associative neural networks, the advantage is that the linear parameters can be trained online with good convergence and stability properties. However, they produce essentially black box models with poor interpretability. For fuzzy neural networks (FNNs), the basis functions are associated with some linguistic rules, and every numerical result can admit a linguistic interpretation [1].

One of the major issues with the FNN applications is that the network complexity can be extremely high. To tackle this problem, a number of FNN construction methods have been proposed in the literature either to determine the number of inputs, the number of membership functions or the number of rules. For example, the adaptive-network-based fuzzy inference system (ANFIS) [2], the radial basis function (RBF) based adaptive fuzzy system (RBFAFS) [3], and the self constructing neural fuzzy inference network [4], have been proposed mainly to address the problem of membership function selection. The orthogonal least squares (OLS), which is one of the most popular approaches for fast identification of nonlinear systems [5], [6], [7], [8], [9], was extended for the selection of fuzzy rules [10]. In addition, the Error Reduction Ratio [11] method has been widely used in fuzzy structure selection for the Minimal Resource Allocating Network (M-RAN) [12], Dynamic Fuzzy Neural Network (DFNN) [13] and the Generalized Dynamic Fuzzy Neural Networks (GDFNN) [14].

In [15], a Fast Recursive Algorithm (FRA) was proposed for the identification of nonlinear systems using linear-in-the-parameters models with improved efficiency and numerical stability. It was recently applied to the selection of fuzzy regressor terms for fuzzy neural networks [16]. In this paper, a fast rule selection algorithm (FRSA) is proposed by modifying and extending the FRA to the fuzzy systems, and a comparison between the FRSA, the OLS and the ERR will be provided.

The paper is organized as follows. Section II is the preliminary, and section III presents the fast rule selection algorithm (FRSA). Two numerical simulation examples are given in Section IV. Section V is the conclusion.

The authors are with the School of Electronics, Electrical Engineering & Computer Science, Queen's University Belfast, UK (Emails: bpizzileo01, k.li@qub.ac.uk). This work was supported by the U.K. EPSRC under Grant GR/S85191/01 to K. Li.

II. Preliminary

In Fuzzy Neural Networks, for a given set of m inputs and N samples, each input variable $x_i(t)$, $i = 1, \ldots, m$ is classified by k_i, $i = 1, \ldots, m$ fuzzy sets, denoted as $A_i(j_i)$, $j_i = 1, \ldots, k_i$ [17]. For every input value $x_i(t)$, $t = 1, \ldots, N$, its membership degree in $A_i(j_i)$ is denoted as $0 \leq \mu_i^{A_i(j_i)}(t) \leq 1$. The construction of a fuzzy neural network mainly involves the following three steps: i) fuzzification (each variable is classified into a certain number of fuzzy sets, and this involves choosing the number of the fuzzy sets and selecting the shapes of the membership functions); ii) rule evaluation (typically using the Takagi-Sugeno [17] or the Mamdani method [18]); iii) aggregation of rules and defuzzification (once the values of the basis function for each rule and for each data sample have been obtained, all the rules should be aggregated and the final value be defuzzified, i.e. to convert a fuzzy quantity to a precise, or crisp quantity [18]).

The defuzzification is achieved by the centroid technique, which produces the centre of gravity

$$y(t) = \frac{\sum\limits_{r=1}^{N_R} W_t^r y^r(t)}{\sum\limits_{r=1}^{N_R} W_t^r} = \sum_{r=1}^{N_R} \Phi_t^r y^r(t) \qquad (1)$$

where $y(t)$ is the crisp value for the t^{th} sample; N_R is the total number of rules; $\Phi_t^r = W_t^r \Big/ \sum\limits_{r=1}^{N_R} W_t^r$ is the fuzzy basis function [19] associated to the r^{th} rule, $\forall r = 1, ..., N_R$. In (1) $W_t^r = \prod\limits_{i=1}^{m} \mu_i^{A_i^r}(t)$ where A_i^r is the fuzzy set of the i^{th} variable associated with the r^{th} rule and $A_i^r \in \{A_i(j_i), j_i = 1, ..., k_i\}$, $y^r(t)$ is the output associated to the r^{th} rule and t^{th} sample whose expression depends on the particular choice of model structure. For example a linear model will have

$$y^r(t) = \sum_{i=1}^{m} g_i^r x_i(t), \ \forall r = 1, ..., N_R \qquad (2)$$

where g_i^r is the consequent parameter associated to the i^{th} input and the r^{th} rule [17].

From (1) and (2) it follows that $\forall t \in \{1, ..., N\}$ the crisp output can be reformulated as

$$\begin{aligned} y(t) = & W_t^1 \left(g_1^1 x_1(t) + \ldots + g_m^1 x_m(t)\right) + \ldots \\ & + W_t^r \left(g_1^r x_1(t) + \ldots + g_m^r x_m(t)\right) + \ldots \\ & + W_t^{N_R} \left(g_1^{N_R} x_1(t) + \ldots + g_m^{N_R} x_m(t)\right) \end{aligned} \qquad (3)$$

given that $\sum_{r=1}^{N_R} W_t^r = 1$ for (1).

The total number of unknown parameters g_i^r is $n = mN_R$ where $N_R = \prod_{i=1}^{m} k_i$ is the total number of rules.

III. The proposed method

For the convenience of notations, define 1) the scalars $\hat{\varphi}_i^r(t) = W_i^r x_i(t)$ for $t = 1, \ldots, N$ and $r = 1 \ldots N_R$; 2) the vectors $\hat{\varphi}_i^r = \left[\begin{array}{ccc} \hat{\varphi}_i^r(1) & \ldots & \hat{\varphi}_i^r(\mathrm{N}) \end{array}\right]^T$, $g^r = \left[\begin{array}{ccc} g_1^r & \ldots & g_m^r \end{array}\right]^T$; and 3) the full column rank submatrices $\hat{\varphi}^r = \left(\begin{array}{ccc} \hat{\boldsymbol{\varphi}}_1^r & \ldots & \hat{\varphi}_m^r \end{array}\right)$, where these N_R submatrices represent the corresponding initial N_R candidate fuzzy rules. Thus $\forall t \in \{1, ..., N\}$ the output in the fuzzy neural network can be expressed as:

$$y(t) = f\left(\mathbf{x}(t)\right) = \sum_{r=1}^{N_R} \sum_{i=1}^{m} \varphi_i^r(t) g_i^r + \varepsilon(t) \qquad (4)$$

with $\mathbf{x}(t) = \left[\begin{array}{ccc} x_1(t) & \ldots & x_m(t) \end{array}\right]^T$ the FNN neural input vector and $\varepsilon(t)$ the model residual sequence. If N data samples $\{\mathbf{x}(t), y(t)\}_{t=1}^N$ are used for network construction and training, (4) can then be reformulated as

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\Theta} + \boldsymbol{\Xi} \qquad (5)$$

where $\boldsymbol{\Phi} = \left[\begin{array}{ccc} \hat{\varphi}^1 & ... & \hat{\varphi}^{N_R} \end{array}\right]$ and $\boldsymbol{\Theta}^{\mathrm{T}} = \left[\begin{array}{ccc} \left(\boldsymbol{g}^1\right)^T & ... & \left(\boldsymbol{g}^{N_R}\right)^T \end{array}\right]$.

A cost function can be defined as

$$E = \sum\nolimits_{t=1}^{N} \left(y(t) - \sum_{r=1}^{N_R} \sum_{i=1}^{m} \varphi_i^r(t) g_i^r\right)^2$$

which represents the cost function associated to the whole network where all possible rule are included into the fuzzy model. Now, the objective is to select M out of the total N_R rules given a rule selection criterion, and calculate the corresponding M vectors $\boldsymbol{g}^j, \forall j = 1, \ldots, M$. To achieve this requires to select M different sub-matrices $\left(\hat{\varphi}^j\right)_{N \times m}$ from $\boldsymbol{\Phi}$ corresponding to the selected rules. To select the submatrices, a fast rule selection algorithm (FRSA) is proposed.

A. Fast Rule Selection Algorithm (FRSA)

At the beginning, suppose the first rule is to be selected from all candidate rules, i.e. $j = 1$, then at this stage, the major issues for FRSA are

1) Define N_R recursive matrices ${}^1\boldsymbol{M}^r \in \Re^{m \times m}$

$$ {}^1\boldsymbol{M}^r = \left({}^0\hat{\varphi}^r\right)^T \left({}^0\hat{\varphi}^r\right), \; r = 1, \ldots, N_R \qquad (6)$$

According to [15] it holds that

$$ {}^1\hat{\boldsymbol{\Theta}}^{\mathrm{r}} = \left({}^1\boldsymbol{M}^r\right)^{-1} \left({}^0\hat{\varphi}^r\right)^{\mathrm{T}} \boldsymbol{y} \qquad (7)$$

$$ {}^1E^r = \boldsymbol{y}^T\boldsymbol{y} - \left({}^1\hat{\boldsymbol{\Theta}}^{\mathrm{r}}\right)^T \left({}^0\hat{\varphi}^r\right)^T \boldsymbol{y} \qquad (8)$$

$$ {}^0\hat{\varphi}^r = \hat{\varphi}^r \qquad (9)$$

where ${}^1\hat{\boldsymbol{\Theta}}^{\mathrm{r}}$ is the vector containing the estimate of the m parameters and ${}^1E^r$ is the model error at the step $j = 1$, associated to the r^{th} rule.

Moreover define N_R residual matrices ${}^1\boldsymbol{R}^r \in \Re^{N \times N}$

$$ {}^1\boldsymbol{R}^r = \boldsymbol{I} - {}^0\hat{\varphi}^r \left({}^1\boldsymbol{M}^r\right)^{-1} \left({}^0\hat{\varphi}^r\right)^T \qquad (10)$$

which are obtained by updating the following matrix ($i = 1, \ldots, m$) for m times,

$$ {}^1\boldsymbol{R}_i^r \triangleq \boldsymbol{I} - {}^1\hat{\boldsymbol{\Phi}}_i^r \left({}^1\boldsymbol{M}_i^r\right)^{-1} \left({}^1\hat{\boldsymbol{\Phi}}_i^r\right)^T$$

where ${}^1\hat{\boldsymbol{\Phi}}_i^r \in \Re^{N \times i}$ is the matrix with the first i columns of $\hat{\varphi}^r$, i.e. ${}^1\hat{\boldsymbol{\Phi}}_{\mathrm{i}}^{\mathrm{r}} = \left(\begin{array}{ccc} \hat{\varphi}_1^r & \ldots & \hat{\varphi}_i^r \end{array}\right)$.

Moreover

$$ {}^1\boldsymbol{M}_i^r = \left({}^1\hat{\boldsymbol{\Phi}}_i^r\right)^T \left({}^1\hat{\boldsymbol{\Phi}}_i^r\right), \quad {}^1\boldsymbol{M}_m^r = {}^1\boldsymbol{M}^r$$

$$ {}^1\boldsymbol{R}_0^r = \boldsymbol{I}, \qquad {}^1\boldsymbol{R}_m^r = {}^1\boldsymbol{R}^r, \qquad {}^1\hat{\boldsymbol{\Phi}}_m^r = \hat{\varphi}^r$$

To update ${}^1\boldsymbol{R}_i^r$, according with [15] and from (9) it holds that:

$$ {}^1\boldsymbol{R}_{i+1}^r = {}^1\boldsymbol{R}_i^r - \frac{\left({}^1R_i^r\right)\left(\hat{\varphi}_{i+1}^r\right)\left(\hat{\varphi}_{i+1}^r\right)^T\left({}^1R_i^r\right)^T}{\left(\hat{\varphi}_{i+1}^r\right)^T\left({}^1R_i^r\right)\left(\hat{\varphi}_{i+1}^r\right)} \qquad (11)$$

2) Furthermore define a regression, and from (9) it holds that

$$\begin{aligned} {}^1a_{i,k}^r &= \left({}^1\boldsymbol{R}_{i-1}^r\hat{\varphi}_i^r\right)^T \left({}^1\boldsymbol{R}_{i-1}^r\hat{\varphi}_k^r\right) \\ {}^1a_{1,k}^r &= \left(\hat{\varphi}_1^r\right)^T \hat{\varphi}_k^r \\ {}^1a_{i,y}^r &= \left({}^1\boldsymbol{R}_{i-1}^r\hat{\varphi}_i^r\right)^T \boldsymbol{y} \\ {}^1a_{1,y}^r &= \left(\hat{\varphi}_1^r\right)^T \boldsymbol{y} \end{aligned} \qquad (12)$$

According to [15], the above quantities can be efficiently computed for $i = 1, \ldots, m$ $k = i, \ldots, m$ as

$$ {}^1a_{i,k}^r = \left(\hat{\varphi}_i^r\right)^T \hat{\varphi}_k^r - \sum_{h=1}^{i-1} \left({}^1a_{h,i}^r\,{}^1a_{h,k}^r\right) / {}^1a_{h,h}^r$$

$$ {}^1a_{i,y}^r = \left(\hat{\varphi}_i^r\right)^T \boldsymbol{y} - \sum_{h=1}^{i-1} \left({}^1a_{h,i}^r\,{}^1a_{h,y}^r\right) / {}^1a_{h,h}^r \qquad (13)$$

3) Finally define the net cost function contributed by each rule as ${}^1E^r$, which can be recursively updated as

$$ {}^1\delta E_{i+1}^r = -\frac{\left(y^T\varphi_{i+1}^r - \sum_{h=1}^{i}\left({}^1a_{h,y}^r\,{}^1a_{h,i+1}^r\right)/{}^1a_{h,h}^r\right)^2}{\left(\varphi_{i+1}^r\right)^T\varphi_{i+1}^r \sum_{h=1}^{i}\left({}^1a_{h,i+1}^r\right)^2/{}^1a_{h,h}^r} \qquad (14)$$

where ${}^1E_{i-1}^r + {}^1\delta E_i^r = {}^1E_i^r$, ${}^1\delta E_1^r = {}^1E_1^r$ and ${}^1E_m^r = {}^1E^r$.

4) Now it's possible to select among all the rules the one that gives the lowest cost function and start to build the matrix ${}^1\mathbf{S}$ corresponding to the selected rules and the matrix ${}^1\mathbf{U}$ of unselected rules. The updated full regression matrix will be

$$ {}^1\Phi = \left[\begin{array}{cc} {}^1\mathbf{S} & {}^1\mathbf{U} \end{array}\right]$$

where

$$^{1}\mathbf{S} = \left(^{1}\hat{\boldsymbol{\varphi}}^{s_1}\right)$$

and s_1 is the index of the selected rule; so if $(\hat{\boldsymbol{\varphi}}^{s_1})$ is the first corresponding selected submatrix from $\boldsymbol{\Phi}$ defined in (5), this matrix has to be updated as $^{1}\hat{\boldsymbol{\varphi}}^{s_1} = \left(^{1}\boldsymbol{R}^{s_1}\right)\hat{\boldsymbol{\varphi}}^{s_1}$ in the matrix of selected rules $^{1}\mathbf{S}$. In the meantime, the submatrices corresponding to the unselected rules are grouped together as

$$^{1}\mathbf{U} = \left[\ ^{1}\hat{\boldsymbol{\varphi}}^{u_1} \quad \dots \quad ^{1}\hat{\boldsymbol{\varphi}}^{u_{N_R-1}}\ \right]$$

where if u_r $r = 1, \dots, N_R - 1$ is the new index of the unselected rules then $^{1}\hat{\boldsymbol{\varphi}}^{u_r} = \left(^{1}\boldsymbol{R}^{s_1}\right)\hat{\boldsymbol{\varphi}}^{u_r}$ are the updated submatrices corresponding to the u_r^{th} unselected rules and $u_r \neq s_1$.

Now to generalize to the selection of the $(j+1)^{th}$ rule, the starting full regression matrix defined in (5), updated at the end of the step j becomes

$$^{j}\Phi = \left[\ ^{j}\mathbf{S} \quad ^{j}\mathbf{U}\ \right] \tag{15}$$

where

$$^{j}\mathbf{S} = \left[\ ^{1}\hat{\boldsymbol{\varphi}}^{s_1} \quad \dots \quad ^{j}\hat{\boldsymbol{\varphi}}^{s_j}\ \right] \tag{16}$$

and

$$^{j}\mathbf{U} = \left[\ ^{j}\hat{\boldsymbol{\varphi}}^{\mathbf{u_1}} \quad \dots \quad ^{j}\hat{\boldsymbol{\varphi}}^{\mathbf{u_{N_R-j}}}\ \right] \tag{17}$$

and at the end of the step j,

$$\left(^{j}\hat{\boldsymbol{\varphi}}^{s_j}\right) = \left(^{j}\boldsymbol{R}^{s_j}\right)\left(^{j-1}\hat{\boldsymbol{\varphi}}^{s_j}\right) \tag{18}$$

which is the final updated selected submatrix corresponding to the s_j^{th} selected rule and

$$\left(^{j}\hat{\boldsymbol{\varphi}}^{\mathbf{u_r}}\right) = \left(^{j}\boldsymbol{R}^{s_j}\right)\left(^{j-1}\hat{\boldsymbol{\varphi}}^{u_r}\right) \tag{19}$$

are the updated submatrices corresponding to the u_r^{th} unselected rules, $u_r \neq s_j$.

Now, at this stage, the major issues for FRSA are

1) Update of the residual matrix. Similarly to (6), define $N_R - j$ recursive matrices, $^{j+1}\boldsymbol{M}^{u_r} \in \Re^{m\times m}$

$$^{j+1}\boldsymbol{M}^{u_r} = \left[\ ^{j}\mathbf{S} \quad ^{\mathbf{j}}\hat{\boldsymbol{\varphi}}^{\mathbf{u_r}}\ \right]^T \left[\ ^{j}\mathbf{S} \quad ^{\mathbf{j}}\hat{\boldsymbol{\varphi}}^{\mathbf{u_r}}\ \right]$$

where $u_r = 1, \dots, N_R - j$. From (7) and (8) it follows that

$$^{j+1}\hat{\boldsymbol{\Theta}}^{u_r} = \left(^{j+1}\boldsymbol{M}^{u_r}\right)^{-1}\left[\ ^{j}\mathbf{S} \quad ^{\mathbf{j}}\hat{\boldsymbol{\varphi}}^{\mathbf{u_r}}\ \right]^T \boldsymbol{y}$$

$$^{j+1}\boldsymbol{E}^{u_r} = \boldsymbol{y}^T\boldsymbol{y} - \left(^{j+1}\hat{\boldsymbol{\Theta}}^{u_r}\right)^T \left[\ ^{j}\mathbf{S} \quad ^{j}\hat{\boldsymbol{\varphi}}^{u_r}\ \right]^T \boldsymbol{y}$$

Moreover, similar to (10), the $N_R - j$ residual matrices becomes $^{j+1}\boldsymbol{R}^{u_r}$, which can be obtained by updating each of them m times, i.e., according to (11),

$$^{j+1}\boldsymbol{R}_1^{u_r} = {}^{j}\boldsymbol{R}^{s_j} - \frac{\left(^{j}R^{s_j}\right)\left(^{j}\hat{\varphi}_1^{u_r}\right)\left(^{j}\hat{\varphi}_1^{u_r}\right)^T\left(^{j}R^{s_j}\right)^T}{\left(^{j}\hat{\varphi}_1^{u_r}\right)^T\left(^{j}R^{s_j}\right)\left(^{j}\hat{\varphi}_1^{u_r}\right)}$$

$$^{j+1}\boldsymbol{R}_{i+1}^{u_r} = {}^{j+1}\boldsymbol{R}_i^{u_r} - \frac{\left(^{j+1}R_i^{u_r}\right)\left(^{j}\hat{\varphi}_{i+1}^{u_r}\right)\left(^{j}\hat{\varphi}_{i+1}^{u_r}\right)^T\left(^{j+1}R_i^{u_r}\right)^T}{\left(^{j}\hat{\varphi}_{i+1}^{u_r}\right)^T\left(^{j+1}R_i^{u_r}\right)\left(^{j}\hat{\varphi}_{i+1}^{u_r}\right)}$$

$$^{j+1}\boldsymbol{R}_m^{u_r} = {}^{j+1}\boldsymbol{R}^{u_r}$$

where $i = 1, \dots, m$, $^{j}\hat{\boldsymbol{\varphi}}_i^{u_r}$ is the i^{th} vector of the matrix $^{j}\hat{\boldsymbol{\varphi}}^{u_r}$.

2) Update of the regression context. From (12)

$$\begin{aligned} ^{j+1}a_{i,k}^{u_r} &= \left[\left(^{j+1}R_{i-1}^{u_r}\right)\left(^{j}\hat{\varphi}_i^{u_r}\right)\right]^T\left(^{j+1}R_{i-1}^{u_r}\right)\left(^{j}\hat{\varphi}_k^{u_r}\right), \\ ^{j+1}a_{1,k}^{u_r} &= \left[\left(^{j}R^{s_j}\right)\left(^{j}\hat{\varphi}_1^{u_r}\right)\right]^T\left(^{j}R^{s_j}\right)\left(^{j}\hat{\varphi}_k^{u_r}\right) \end{aligned}$$

and

$$\begin{aligned} ^{j+1}a_{i,y}^{u_r} &= \left[\left(^{j+1}R_{i-1}^{u_r}\right)\left(^{j}\hat{\varphi}_i^{u_r}\right)\right]^T y, \\ ^{j+1}a_{1,y}^{u_r} &= \left[\left(^{j}R^{s_j}\right)\left(^{j}\hat{\varphi}_1^{u_r}\right)\right]^T y \end{aligned}$$

Finally, according to (13), the above quantities can be efficiently computed for $i = 1, \dots, m$, $k = i, \dots, m$ as

$$^{j+1}a_{i,k}^{u_r} = \left(^{j}\hat{\boldsymbol{\varphi}}_i^{u_r}\right)^T\left(^{j}\hat{\boldsymbol{\varphi}}_k^{u_r}\right) - \sum_{h=1}^{i-1}\left(^{j+1}a_{h,i}^{u_r}\ ^{j+1}a_{h,k}^{u_r}\right)\Big/ ^{j+1}a_{h,h}^{u_r}$$

$$^{j+1}a_{i,y}^{u_r} = \left(^{j}\hat{\boldsymbol{\varphi}}_i^{u_r}\right)^T\boldsymbol{y} - \sum_{h=1}^{i-1}\left(^{j+1}a_{h,i}^{u_r}\,^{j+1}a_{h,y}^{u_r}\right)\Big/ ^{j+1}a_{h,h}^{u_r}$$

3) Computation of the net contribution to the cost function. From (14)

$$^{j+1}\delta E_{i+1}^{u_r} = -\frac{\left(y^T\,^{j}\hat{\varphi}_{i+1}^{u_r} - \sum_{h=1}^{i}\left(^{j+1}a_{h,y}^{u_r}\,^{j+1}a_{h,i+1}^{u_r}\right)/^{j+1}a_{h,h}^{u_r}\right)^2}{\left(^{j}\hat{\varphi}_{i+1}^{u_r}\right)^T\,^{j}\hat{\varphi}_{i+1}^{u_r}\sum_{h=1}^{i}\left(^{j+1}a_{h,i+1}^{u_r}\right)^2/^{j+1}a_{h,h}^{u_r}}$$

where $^{j}E^{s_j} + {}^{j+1}\delta E_1^{u_r} = {}^{j+1}E_1^{u_r}$, $^{j+1}E_{i-1}^{u_r} + {}^{j+1}\delta E_i^{u_r} = {}^{j+1}E_i^{u_r}$ and $^{j+1}E_m^{u_r} = {}^{j+1}E^{u_r}$

4) The submatrix with the lowest cost function $^{j+1}E^{u_r}$ has the index $u_r = s_{j+1}, (u_r = 1, , N_r - j)$.
Therefore, similar to (15), the full regression matrix can be updated as

$$^{j+1}\Phi = \left[\ ^{j+1}\mathbf{S} \quad ^{j+1}\mathbf{U}\ \right]$$

where from (16) and (17)

$$\begin{aligned} ^{j+1}\mathbf{S} &= \left(\ ^{1}\hat{\boldsymbol{\varphi}}^{s_1} \quad \dots \quad ^{j+1}\hat{\boldsymbol{\varphi}}^{s_{j+1}}\ \right) \\ ^{j+1}\mathbf{U} &= \left[\left(\ ^{j+1}\hat{\boldsymbol{\varphi}}^{\mathbf{u_1}}\right) \quad \dots \quad \left(^{j+1}\hat{\boldsymbol{\varphi}}^{\mathbf{u_{N_R-j-1}}}\right)\ \right] \end{aligned}$$

and from (18) and (19), $u_r = 1, \dots, N_R - j - 1$

$$\left(^{j+1}\hat{\boldsymbol{\varphi}}^{s_{j+1}}\right) = \left(^{j+1}\boldsymbol{R}^{s_{j+1}}\right)^{j}\hat{\boldsymbol{\varphi}}^{s_{j+1}}$$

$$\left(^{j+1}\hat{\boldsymbol{\varphi}}^{u_r}\right) = \left(^{j+1}\boldsymbol{R}^{s_{j+1}}\right)^{j}\hat{\boldsymbol{\varphi}}^{u_r}$$

The above procedure can be explained in a compact way, $\forall j = 1, \dots, M$, which is summarized as follows.

1) Define the matrix

$$^{j}\boldsymbol{\Phi}^{u_r} = \left(\ ^{j-1}\mathbf{S} \quad ^{j-1}\hat{\boldsymbol{\varphi}}^{u_r}\ \right) \tag{20}$$

where if $j = 1, u_r = r$, then $^{j-1}\mathbf{S}$ is an empty matrix and $^{j-1}\hat{\boldsymbol{\varphi}}^{u_r} = \hat{\boldsymbol{\varphi}}^{u_r} = \hat{\boldsymbol{\varphi}}^{r}$
Moreover, define

$$^{j}\hat{\boldsymbol{\Theta}}^{u_r} = \left(^{j}\boldsymbol{M}^{u_r}\right)^{-1}\left(^{j}\boldsymbol{\Phi}^{u_r}\right)^T\boldsymbol{y}$$

$$^{j}\boldsymbol{E}^{u_r} = \boldsymbol{y}^T\boldsymbol{y} - \left(^{j}\hat{\boldsymbol{\Theta}}^{u_r}\right)^T\left(^{j}\boldsymbol{\Phi}^{u_r}\right)^T\boldsymbol{y}$$

$$ {}^{j}R_{i+1}^{u_r} = {}^{j}R_{i}^{u_r} - \frac{\left({}^{j}R_{i}^{u_r}\right)\left({}^{j-1}\hat{\varphi}_{i+1}^{u_r}\right)\left({}^{j-1}\hat{\varphi}_{i+1}^{u_r}\right)^T\left({}^{j}R_{i}^{u_r}\right)^T}{\left({}^{j-1}\hat{\varphi}_{i+1}^{u_r}\right)^T\left({}^{j}R_{i}^{u_r}\right)\left({}^{j-1}\hat{\varphi}_{i+1}^{u_r}\right)} $$

$\forall i = 0,\ldots,m$ where ${}^{1}\boldsymbol{R}_0^{u_r} = \boldsymbol{I}$ and ${}^{j}\boldsymbol{R}_0^{u_r} = {}^{j-1}\boldsymbol{R}^{s_j-1}$

2) Define

$$ {}^{j}a_{i,k}^{u_r} = \left[\left({}^{j}\boldsymbol{R}_{i-1}^{u_r}\right)\left({}^{j-1}\hat{\boldsymbol{\varphi}}_i^{u_r}\right)\right]^T\left({}^{j}\boldsymbol{R}_{i-1}^{u_r}\right)\left({}^{j-1}\hat{\boldsymbol{\varphi}}_k^{u_r}\right) $$
$$ {}^{j}a_{1,k}^{u_r} = \left[\left({}^{j-1}\boldsymbol{R}^{s_{j-1}}\right)\left({}^{j-1}\hat{\boldsymbol{\varphi}}_1^{u_r}\right)\right]^T\left({}^{j-1}\boldsymbol{R}^{s_{j-1}}\right)\left({}^{j-1}\hat{\boldsymbol{\varphi}}_k^{u_r}\right) $$

with ${}^{0}R^{s_0} = \boldsymbol{I}$ and

$$ {}^{j}a_{i,y}^{u_r} = \left[\left({}^{j}\boldsymbol{R}_{i-1}^{u_r}\right)\left({}^{j-1}\hat{\boldsymbol{\varphi}}_i^{u_r}\right)\right]^T\boldsymbol{y}, $$
$$ {}^{j}a_{1,y}^{u_r} = \left[\left({}^{j-1}\boldsymbol{R}^{s_{j-1}}\right)\left({}^{j-1}\hat{\boldsymbol{\varphi}}_1^{u_r}\right)\right]^T\boldsymbol{y} $$

which can be fast computed for $i = 0,\ldots,m$, $k = i,\ldots,m$ as

$$ {}^{j}a_{i,k}^{u_r} = \left({}^{j-1}\hat{\boldsymbol{\varphi}}_i^{u_r}\right)^T\left({}^{j-1}\hat{\boldsymbol{\varphi}}_k^{u_r}\right) - \sum_{h=1}^{i-1}\left({}^{j}a_{h,i}^{u_r}\,{}^{j}a_{h,k}^{u_r}\right)\Big/{}^{j}a_{h,h}^{u_r} $$
$$ {}^{j}a_{i,y}^{u_r} = \left({}^{j-1}\hat{\boldsymbol{\varphi}}_i^{u_r}\right)^T\boldsymbol{y} - \sum_{h=1}^{i-1}\left({}^{j}a_{h,i}^{u_r}\,{}^{j}a_{h,y}^{u_r}\right)\Big/{}^{j}a_{h,h}^{u_r} $$

3) Define

$$ {}^{j}\delta E_{i+1}^{u_r} = -\frac{\left(y^{T\,j-1}\hat{\varphi}_{i+1}^{u_r} - \sum_{h=1}^{i}\left({}^{j}a_{h,y}^{u_r}\,{}^{j}a_{h,i+1}^{u_r}\right)/{}^{j}a_{h,h}^{u_r}\right)^2}{\left({}^{j-1}\hat{\varphi}_{i+1}^{u_r}\right)^{T\,j-1}\hat{\varphi}_{i+1}^{u_r}\sum_{h=1}^{i}\left({}^{j+1}a_{h,i+1}^{u_r}\right)^2\Big/{}^{j}a_{h,h}^{u_r}} $$

where ${}^{j}E_1^{u_r} = {}^{j-1}E^{s_{j-1}} + {}^{j}\delta E_1^{u_r}, {}^{0}E^{s_0} = 0, {}^{j}E_i^{u_r} = {}^{j}E_{i-1}^{u_r} + {}^{j}\delta E_i^{u_r}$ and ${}^{j}E_m^{u_r} = {}^{j}E^{u_r}$

4) Update the regression matrix as

$$ {}^{j}\Phi = \left[\begin{array}{cc} {}^{j}\mathbf{S} & {}^{j}\mathbf{U} \end{array}\right] $$

where

$$ \begin{array}{rcl} {}^{j}\mathbf{S} & = & \left(\begin{array}{ccc} {}^{1}\hat{\boldsymbol{\varphi}}^{s_1} & \ldots & {}^{j}\hat{\boldsymbol{\varphi}}^{s_{j+1}} \end{array}\right) \\ {}^{j}\mathbf{U} & = & \left[\begin{array}{ccc} \left({}^{j}\hat{\boldsymbol{\varphi}}^{\mathrm{u}_1}\right) & \ldots & \left({}^{j}\hat{\boldsymbol{\varphi}}^{\mathrm{u}_{N_R-j}}\right) \end{array}\right] \end{array} $$

$$ \left({}^{j}\hat{\boldsymbol{\varphi}}^{s_j}\right) = \left({}^{j}\boldsymbol{R}^{s_j}\right)^{j-1}\hat{\boldsymbol{\varphi}}^{s_j} $$
$$ \left({}^{j}\hat{\boldsymbol{\varphi}}^{u_r}\right) = \left({}^{j}\boldsymbol{R}^{s_j}\right)^{j-1}\hat{\boldsymbol{\varphi}}^{u_r} $$

This rule selection procedure will continue, until a fuzzy neural network structure selection criterion is satisfied, such as the net contribution of the selected rule is below certain threshold, etc [15] and the output of the FNN can be estimated as

$$ \hat{\boldsymbol{y}} = {}^{M}\mathbf{S}\left[\begin{array}{ccc} \boldsymbol{g}^1 & \ldots & \boldsymbol{g}^M \end{array}\right]^T $$

B. Computational Complexity

Since for each step $j = 1,\ldots,M$ the FRA needs to be apploed for [15] $N_R - j + 1$ times, the total computation of the FRSA is

$$ C^{(FRSA)} = \left(\sum_{i=N_R-M+1}^{N_R} i\right)C^{(FRA)} $$

According to [15], if $N >> m$, for the FRA the computational effort mainly comes from the term $= m^2N$. Therefore the total cost for the FRSA is

$$ C^{(FRSA)} = \left(\frac{(2N_R - M + 1)(M)}{2}\right)m^2N $$

where N_R is the total number of rules, M is the desired number of selected rules, m is the number of inputs, N is the number of samples.

IV. Numerical Examples

Example 1- The membrane function was approximated using a fuzzy network [1]. The function inputs are x_1 and x_2 which are within the range of [0 1]. The membership functions are 1-D piecewise quadratic B-splines, which were generated using the recursive Cox-De Boor algorithm [20]:

$$ \begin{array}{l} \left[\mu_i^z(t)\right]_d = \frac{x_i(t)-\tau_z}{\tau_{z+m}-\tau_z}\cdot\left[\mu_i^z(t)\right]_{d-1} \\ +\frac{\tau_{z+m+1}-x_i(t)}{\tau_{z+m+1}-\tau_{z+1}}\cdot\left[\mu_i^{z+1}(t)\right]_{d-1} \\ \left[\mu_i^z(t)\right]_0 = \left\{\begin{array}{ll} 1, & \tau_z \le x_i(t) \le \tau_{z+1} \\ 0, & otherwise \end{array}\right. \end{array} \quad (21) $$

where d is the degree of the B-spline, τ is a knot vector with $k_i + d + 1$ dimensions, $\left[\mu_i^z(t)\right]_d$ is the B-spline for the input value $x_i(t)$ in the iteration step z. In this example, the fuzzy neural network is fuzzified by $k_i = 6\ (i = 1, 2)$, $d = 2$, $N_R = k_1 \times k_2 = 6 \times 6 = 36$ and $\tau = [-0.2, 0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4]$.

Figure 1 shows the membership functions. Table II summarizes the cardinality of rules associated to their linguistic interpretations. Among a total of 2601 samples, 1989 samples were used for the training and 612 for the validation.

The specifications of the PC used in the simulation are summarized as follows - CPU: Intel(R) Pentium(R) 4 CPU 3.20GHz; Speed: 512 MB; max bandwidth 266MHz.

A. Selection of fuzzy rules

Using the proposed FRSA and the stop criteria $[MSE]_j - [MSE]_{j+1} = 1 \times 10^{-4}$ (MSE_j is the mean squared error with j rules), 18 rules were selected.

B. Comparison with other approaches

For comparison purpose, the OLS was also applied to the fuzzy rule selection using the similar procedures described above.

In addition, the Error Reduction Ratio (ERR) combined with the sensitivity analysis of fuzzy rules [11]-[14] has been another popular approach in pruning the insignificant rules for the fuzzy models. This approach is summarized as follows:

1) Error Reduction Ratio: given N samples from (4) and (5), the matrix $\boldsymbol{\Phi}$ can be transformed into a set of orthogonal basis vector by the QR decomposition, therefore the output can be expresses as

$$ \boldsymbol{y} = \boldsymbol{\Psi}\boldsymbol{A}\boldsymbol{\Theta} + \boldsymbol{\Xi} = \boldsymbol{\Psi}\boldsymbol{G} + \boldsymbol{\Xi} $$

where

$$\hat{\mathbf{\Theta}} = \mathbf{A}^{-1}\hat{\mathbf{G}},\ \mathbf{\Psi} = \mathbf{\Phi}\mathbf{A}^{-1} = \left[\ \psi_1 \quad \cdots \quad \psi_n\ \right]$$

and $\mathbf{A}$ is a unit upper triangular matrix. The Least Square solution of G is given by

$$\hat{\mathbf{G}} = [\hat{g}_1 \cdots \hat{g}_n]^T = \left[\mathbf{\Psi}^T\mathbf{\Psi}\right]^{-1}\mathbf{\Psi}^T\mathbf{Y}$$

or

$$\mathrm{g_i} = \frac{\boldsymbol{\psi}_i^T\boldsymbol{y}}{\boldsymbol{\psi}_i^T\boldsymbol{\psi}_i},\ i = 1, \ldots, n \tag{22}$$

As $\boldsymbol{\psi}_i$ and $\boldsymbol{\psi}_j$ are orthogonal for i $\neq$ j , the variance of y is given by

$$\frac{\boldsymbol{y}^T\boldsymbol{y}}{N} = \frac{\left(\sum_{i=1}^{n}\mathrm{g_i^2}\boldsymbol{\psi}_i^T\boldsymbol{\psi}_i\right) + \mathbf{\Xi}^T\mathbf{\Xi}}{N}$$

Thus, based on (22) the ERR due to ψ_i can be defined as

$$\mathrm{err_i} = \frac{\mathrm{g_i^2}\boldsymbol{\psi}_i^T\boldsymbol{\psi}_i}{\boldsymbol{y}^T\boldsymbol{y}} = \frac{\left(\boldsymbol{\psi}_i^T\boldsymbol{y}\right)^2}{\boldsymbol{\psi}_i^T\boldsymbol{\psi}_i\boldsymbol{y}^T\boldsymbol{y}} \tag{23}$$

2) Sensitivity analysis of fuzzy rules: define the ERR matrix

$$\mathbf{\Delta} = \left(\ \rho_1 \quad \ldots \quad \rho_{N_R}\ \right) \in \Re^{mxN_R}$$

whose elements are obtained from (23) and the r^{th} column of $\mathbf{\Delta}$ is the total ERR corresponding to the r^{th} rule. Furthermore define the significance of the r^{th} rule as

$$\eta_r = \sqrt{\frac{\rho_{\mathrm{r}}^{\mathrm{T}}\rho_{\mathrm{r}}}{m}},\ r = 1, \ldots, N_R$$

If $\eta_{\mathrm{r}} < \mathrm{k_{err}}$, $r = 1, \ldots, N_R$ where $\mathrm{k_{err}}$ a pre-specified threshold, then the r^{th} is deleted.

This above approach was also applied to the same problem. Table I summarizes the performances of the three methods, where MSE is the mean squared error, MSE_T is MSE for training data, MSE_V is MSE for validation data. It is shown that in comparison with the OLS, FRSA selected the same rules with the same accuracy, however required less computation effort. In comparison with the ERR, the FRSA was less computationally efficient but produced more accurate results.

TABLE I
COMPARISON OF PERFORMANCE FOR FRSA, OLS AND ERR FOR EXAMPLE1

Method	Index of Rules	Simul. Time	MSE
FRSA	20;8;21;14;22;15 27;13;2;31;23;35 26;19;25;28;7;1	15.9 sec	MSE_T=5.6×10^{-4} MSE_V=3.7×10^{-4}
OLS	20;8;21;14;22;15 27;13;2;31;23;35 26;19;25;28;7;1	80.7 sec	MSE_T=5.6×10^{-4} MSE_V=3.7×10^{-4}
ERR	13;19;7;20;14;25 2;8;21;26;1;32;15 27;22;33;3;9	2.5 sec	MSE_T=2.61×10^{-3} MSE_V=1.7×10^{-3}

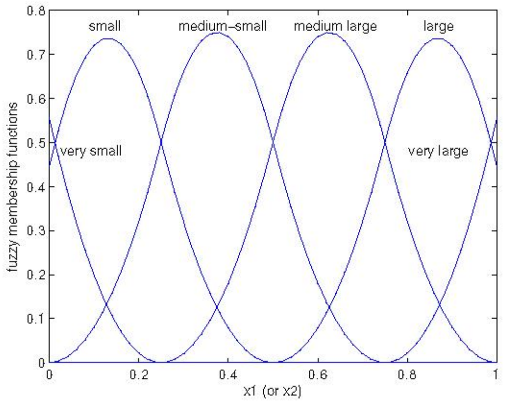

Fig. 1. Piecewise quadratic B-spline fuzzy membership functions for example 1

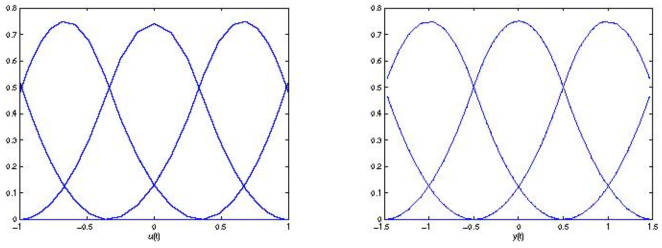

Fig. 2. Membership functions of the inputs for example 2

Example 2: Nonlinear Dynamical System-The following system was modelled using FNN [4]

$$y(t+1) = \frac{y(t)}{1 + y^2(t)} + u^3(t)$$

where $u(t) = \sin(0.04\pi \cdot t)$. One hundred training data were generated using $t = 1, \ldots, 100$. The initial condition of the output was set to be zero.

Figure 2 shows the inputs membership functions. Choosing the stop criterion $MSE \leq 10^{-3}$, the FRSA selected 19 rules. With the same number of rules being selected, again the FRSA is computationally more efficient than the OLS and more accurate than the ERR, as shown in Table III. Figures 3 and 4 illustrate the simulation results of the above different approaches.

TABLE II
ASSOCIATION OF RULES TO FUZZY SETS IN EXAMPLE 1

Rule	X_1					
	V-S	S	M-S	M-L	L	V-L
X_2						
V-S	1	7	13	19	25	31
S	2	8	14	20	26	32
M-S	3	9	15	21	27	33
M-L	4	10	16	22	28	34
L	5	11	17	23	29	35
V-L	6	12	18	24	30	36

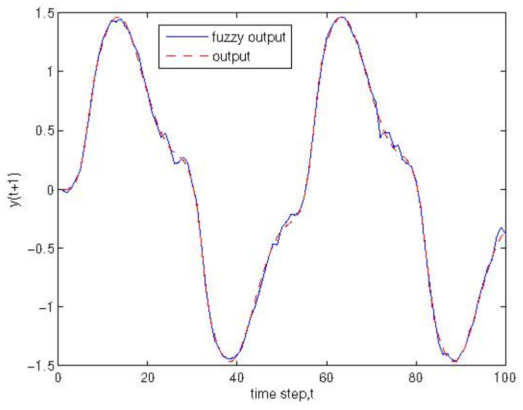

Fig. 3. Simulation result of the fuzzy model by FRSA in example 2

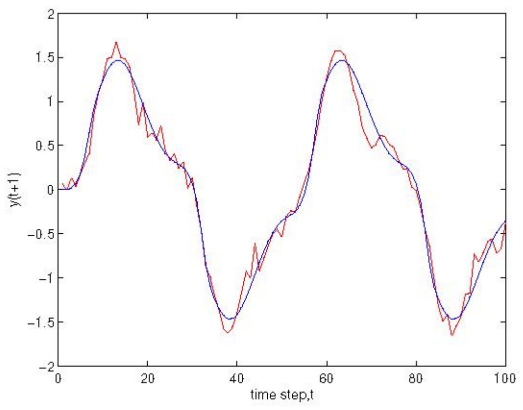

Fig. 4. Simulation result of the fuzzy model by ERR in example 2

V. Conclusion

A fast rule selection algorithm has been proposed. The simulation results show that, in comparison with the classical Orthogonal Least Square method, it leads to the same modelling accuracy but requires less computational effort. In comparison with the Error Reduction Ratio method, it gives higher accuracy.

TABLE III
COMPARISON OF PERFORMANCE FOR FRSA, OLS AND ERR IN EXAMPLE2

Method	Index of Rules	Simul. Time	MSE
FRSA	8;18;19;7;17;4;23 14;25;2;21;12;6 10;24;22;3;20;9	1.3 sec	$MSE{=}1.2\times10^{-3}$
OLS	8;18;19;7;17;4;23 14;25;2;21;12;6 10;24;22;3;20;9	10.2 sec	$MSE{=}1.2\times10^{-3}$
ERR	1;25;21;4;16;11;5 15;20;23;6;22;2 10;14;12;24;3;13	0.2 sec	$MSE{=}2.61\times10^{-3}$

References

[1] X. Hong, C. Harris, and S. Chen, "Robust neurofuzzy rule base knowledge extraction and estimation using subspace decomposition combined with regularization and d-optimality," *IEEE Trans Syst Man Cybern Part B Cybern*, vol. 34, pp. 598–608, 2004.
[2] J. Jang, "Anfis: adaptive-network-based fuzzy inference system," *IEEE Trans. Systems Man Cybernet*, vol. 23, pp. 665–684, 1993.
[3] K. Cho and B. Wang, "Radial basis function based adaptive fuzzy systems and their applications to identification and prediction," *Fuzzy Sets and Systems*, vol. 83, pp. 325–339, 1996.
[4] C. Juang and C. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Systems*, vol. 6, pp. 12–32, 1998.
[5] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to nonlinear system identification," *International Journal of Control*, vol. 50, pp. 1873–1896, 1989.
[6] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function network," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, 1991.
[7] S. Chen and J. Wigger, "Fast orthogonal least squares algorithm for efficient subset model selection," *IEEE Trans. Signal Processing*, vol. 43, pp. 1713–1715, 1995.
[8] Q. M. Zhu and S. A. Billings, "Fast orthogonal identification of nonlinear stochastic models and radial basis function neural network," *Int. J. Contr.*, vol. 64, pp. 871–886, 1996.
[9] K. Z. Mao, "Fast orthogonal forward selection algorithm for feature subset selection," *IEEE Trans. Neural networks*, vol. 13, pp. 1218–1224, 2002.
[10] X. Hong and C. J. Harris, "A neurofuzzy network knowledge extraction and extended gram-schmidt algorithm for model subspace decomposition," *IEEE Trans Fuzzy Syst*, vol. 11, pp. 528–541, 2003.
[11] L. Wang and J. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Networks*, vol. 3, pp. 807–814, 1992.
[12] N. S. Y. Lu and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Computation*, vol. 9, pp. 461–478, 1997.
[13] S. Wu and M. J. Er, "Dynamic fuzzy neural networks-a novel approach to function approximation," *IEEE Trans. Systems Man Cybernet., P. B: Cybernet.*, vol. 30, pp. 358–364, 2000.
[14] S. Wu, M.J.Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," *IEEE Trans. Fuzzy Systems*, vol. 9, pp. 578–594, 2001.
[15] K. Li, J. Peng, and G. Irwin, "A fast nonlinear model identification method," *IEEE Transactions on Automatic Control*, vol. 50, no. 8, pp. 1211–1216, 2005.
[16] B. Pizzileo, K. Li, and G. Irwin, "A fast fuzzy neural modeling method for nonlinear dynamic system," in *International Symposium on Neural Networks*, June 2007.
[17] J. D. Saez, "Takagi-sugeno fuzzy model structure selection based on new sensitivity analysis," in *The 2005 IEEE International Conference on Fuzzy Systems*, pp. 501–506, 2005.
[18] P. T. J. Ross, *Fuzzy Logic with engineering application*. John Wiley & Son Ltd, 2004.
[19] H. M. Kim and J. M. Mendel, "Fuzzy basis functions: Comparison with other basis function," *IEEE Trans Fuzzy Syst*, vol. 3, no. 2, pp. 158–168, 1995.
[20] C. Wang, W. Wang, T. Lee, and P. seng, "Fuzzy b-spline membership function (bmf) and its applications in fuzzy-neural control," *IEEE Trans Syst Man Cybern*, vol. 25, pp. 841–851, 1995.