Long-term Forecasting of Electrical Load using
Gustafson-Kessel clustering algorithm on Takagi-Sugeno type MISO Neuro-
Fuzzy network

By: Felix Pasila
Electrical Engineering Department, Petra Christian University, Surabaya
felix@petra.ac.id

Keywords: Long-term forecasting, GK Clustering, TS-type MISO NF network,
LMA accelerated

1. Introduction

1.1. Problem definition: Neuro-Fuzzy Approach in Electrical Load Forecasting
Modeling and identification of electrical load processes are essential for the operation
and also planning of a utility either for a company or for a country. Electrical load
forecasting is needed because people intend to make important decision on generating
power generators, load switching, purchasing strategy and also infrastructure
development. Furthermore, load forecasts are extremely important for energy
suppliers, transmission, distribution and markets. In other words, load forecasts play a
fundamental role in the formulation of economic, reliable and secure operating
strategies for the power system.

Like other prediction in time series, load forecasts are dealing with sequential time. In
general, load forecasts are divided into two big categories: short-term forecasting
(STF), which can usually be defined as the capability of the network to forecast the
next several days to the some weeks, and long-term forecasting (LTF) which is dealing
with future forecasting. For example, if people only have several weeks of data for
training, how can they create Neuro-Fuzzy network to forecast what happened in the
next several weeks, next month or next year. How long they will believe that the
network still can or can not be trusted. Another important issue in LTF is the annual
peak demand for distribution of substations and feeders. Annual peak load is the most
important value to area planning, since peak load most strongly impacts capacity
requirements (Feinberg, 2003).

In addition, both categories have unique characteristic. STF and LTF can be determined
using the sampling interval and lead time of forecast from data time series. Here,
choosing the sampling interval and lead time will influence the result of forecasting
performance. In the past many researchers were working on STF scheme. Some of them
exposed very good result on training and forecasting performance (Palit, Computational
intelligence, 2005, p.257). Because of that inspiration, Neuro-Fuzzy network will be
used to achieve powerful training and forecasting performance on LTF-electrical load
applications.

Furthermore, Gustafson-Kessel (GK) clustering algorithm will be used to reduce model
complexity and provide initial parameters for Takagi-Sugeno-type MISO on NF network.
Choosing number of clusters is the key on this paper. By some experiments, some
number of clusters has been chosen to train and test the NF network. By using GK
clustering, error performance of the network can be reduce significantly by choosing no
of cluster = 5 (membership functions =5) instead of number of membership
functions=15.
1.2. Matrix Rearrangement of electrical load data

For *Long-term* model, MISO system will be used for training and forecasting. This case, 7 inputs and 1 output model for the given time series modeling and forecasting application the MISO Neuro-Fuzzy predictor should be arranged on $XIO$ matrix, as shown below:

$$XIO_{7,7} = \begin{bmatrix}
\text{Day}_1 & \text{Day}_2 & \text{Day}_3 & \text{Day}_4 & \text{Day}_5 & \text{Day}_6 & \text{Day}_7 \rightarrow \text{Day}_8 \\
\text{Day}_2 & \text{Day}_3 & \text{Day}_4 & \text{Day}_5 & \text{Day}_6 & \text{Day}_7 & \text{Day}_8 \rightarrow \text{Day}_9 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\text{Day}_8 & \text{Day}_9 & \text{Day}_{10} & \text{Day}_{11} & \text{Day}_{12} & \text{Day}_{13} & \text{Day}_{14} \rightarrow \text{Day}_{15}
\end{bmatrix}$$

(1.1)

As shown in equation (1.1), 7 input days are trained to produce 1 output day in NF network. Each input and output represents 1 day set of data. In case of electrical load data from ElectricalLoad.txt, 1 day has 96 data. Output from the first training $\text{Day}_8$ will replace the 7th input for the first forecasting. After 7 loops of forecasting, all the data are full input forecast data (means all input is coming from forecast outputs).

2. Neuro-Fuzzy Systems selection for Forecasting

A Neuro-Fuzzy network with an improved training algorithm for MIMO case was developed by Palit and Popovic (1999, 2000) and Palit and Babuška (2001) for electrical load time series forecasting. Compared to ANFIS, this similar model has achieved better model accuracy and faster training. Because of the reason above, some achievements have been reached in order to accomplish the optimum model accuracy based on a Takagi-Sugeno-type Neuro-Fuzzy network with MIMO model. In addition, this model is an upgrading version from Takagi-Sugeno-type multiple input single output Neuro-Fuzzy network. As continuity from MISO structure, feedforward multi input multi output is proposed by Palit and Popovic (2000) and Palit and Babuška (2001), as shown in Figure 2.1 (Palit, Springer, 2005, p.230)

![Figure 2.1 Fuzzy system MIMO feedforward Takagi-Sugeno-type Neural-Fuzzy network (Palit, 2005)](image)

For LTF type MISO, $m$ will be set to one.
2.1. Neural Network Representation of Fuzzy Logic System (FLS)

Neuro-fuzzy representation of the FLS is based on inference TS-type which has been explained clearly by Palit (2005, p.153, p.233). There are two important steps in this representation: calculating of the degree of fulfillment and normalized degree of fulfillment. The FLS considered here for constructing neuro-fuzzy structures is based on TS-type fuzzy model with Gaussian membership functions. It uses product inference rules and a weighted average defuzzifier defines as:

The corresponding $i^{th}$ rule from the above FLS can be written as

$$R_i = \text{If } x_1 \text{ is } G_i^1 \text{ and } x_2 \text{ is } G_i^2 \text{ and ... } x_n \text{ is } G_i^n \text{ then } y_i = W_{0ij} + W_{ij} x_1 + W_{2ij} x_2 + ... + W_{nj} x_n$$

(2.1)

Where, $x_i$ with $i=1,2,...,n$ are the $n$ system inputs, whereas $f_j$ with $j=1,2,...,m$ are its $m$ outputs, and $G_i^j$ with $i=1,2,...,n$ and $l=1,2,...,M$ are the Gaussian membership functions of the form (2.1) with the corresponding mean and variance parameters $c_i^j$ and $\tau_i^j$ respectively and with $y_i$ as the output consequent of the $i^{th}$ rule. It must be remembered that the Gaussian membership functions $G_i^j$ actually represent linguistic terms such as low, medium, high, very high, etc. The rules as written in (2.1) are known as Takagi-Sugeno rules.

It shows that the FLS can be represented as a three layer MIMO feedforward network as shown in Figure 2.1. Because of the implementation of the Takagi-Sugeno-type FLS, this figure represents a Takagi-Sugeno-type of MIMO neuro-fuzzy network, where instead of the connection weights and the biases in training algorithm such as BPA-NN, we have the mean $c_i^j$ and also the variance $\tau_i^j$ parameters of Gaussian membership functions, along with $W_{0ij}, W_{ij}$ parameters from the rules consequent, as the equivalent adjustable parameters of the network. If all the parameters for NF network are properly selected, then the FLS can correctly approximate any nonlinear system based on given data $XIO$ matrix.

$$f_j = \sum_{i=1}^{M} y_i y_i \cdot h^l$$

(2.2a)

$$y_i = W_{0ij} + W_{ij} x_1 + W_{2ij} x_2 + ... + W_{nj} x_n$$

(2.2b)

$$h^l = \left( z^l / b \right), \text{ and } b = \sum_{i=1}^{M} z_i$$

(2.2c)

$$z_i = \prod_{i=1}^{n} \exp \left( -\left( \frac{x_i - c_i^j}{\sigma_i^j} \right)^2 \right)$$

(2.2d)

2.2. Accelerated Levenberg-Marquardt algorithm (LMA)

To accelerate the convergence speed on neuro-fuzzy network training that happened in BPA, the Levenberg-Marquardt algorithm (LMA) was proposed and proved (Palit and Popovic, 1999).

If a function $V(w)$ is meant to minimize with respect to the parameter vector $w$ using Newton’s method, the update of parameter vector $w$ is defined as:
\[ \Delta w = -\left[ \nabla^2 V(w) \right]^{-1} \cdot \nabla V(w) \]  \hspace{1cm} (2.3a)

\[ w(k+1) = w(k) + \Delta w \]  \hspace{1cm} (2.3b)

From equation (2.3a), \( \nabla^2 V(w) \) is the Hessian matrix and \( \nabla V(w) \) is the gradient of \( V(w) \). If the function \( V(w) \) is taken to be SSE function as follows:

\[ V(w) = 0.5 \cdot \sum_{r=1}^{N} e_r^2(w) \]  \hspace{1cm} (2.4)

Then the gradient of \( V(w) \) and the Hessian matrix \( \nabla^2 V(w) \) are generally defined as:

\[ \nabla V(w) = J^T(w) \cdot e(w) \]  \hspace{1cm} (2.5a)

\[ \nabla^2 V(w) = J^T(w) \cdot J(w) + \sum_{r=1}^{N} e_r(w) \cdot \nabla^2 e_r(w) \]  \hspace{1cm} (2.5b)

where the Jacobian matrix \( J(w) \) as follows

\[
J(w) = \begin{bmatrix}
\frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \cdots & \frac{\partial e_1(w)}{\partial w_N} \\
\frac{\partial e_2(w)}{\partial w_1} & \frac{\partial e_2(w)}{\partial w_2} & \cdots & \frac{\partial e_2(w)}{\partial w_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial e_N(w)}{\partial w_1} & \frac{\partial e_N(w)}{\partial w_2} & \cdots & \frac{\partial e_N(w)}{\partial w_N}
\end{bmatrix}
\]  \hspace{1cm} (2.5c)

From (2.5c), it is seen that the dimension of the Jacobian matrix is \((N \times N_p)\), where \( N \) is the number of training samples and \( N_p \) is the number of adjustable parameters in the network. For the Gauss-Newton method, the second term in (2.5b) is assumed to be zero. Therefore, the update equations according to (2.3a) will be:

\[ \Delta w = -\left[ J^T(w) \cdot J(w) \right]^{-1} \cdot J^T(w) \cdot e(w) \]  \hspace{1cm} (2.6a)

Now let us see the LMA modifications of the Gauss-Newton method.

\[ \Delta w = -\left[ J^T(w) \cdot J(w) + \mu \cdot I \right]^{-1} \cdot J^T(w) \cdot e(w) \]  \hspace{1cm} (2.6b)

where dimension of \( I \) is the \((N_p \times N_p)\) identity matrix, and the parameter \( \mu \) is multiplied or divided by some factor whenever the iteration steps increase or decrease the value of \( V(w) \).

Here, the updated equation according to (2.3a)

\[ w(k+1) = w(k) - \left[ J^T(w) \cdot J(w) + \mu \cdot I \right]^{-1} \cdot J^T(w) \cdot e(w) \]  \hspace{1cm} (2.6c)

This is important to know that for large \( \mu \), the algorithm becomes the steepest descent algorithm with step size \( 1/\mu \), and for small \( \mu \), it becomes the Gauss-Newton method.

For faster convergence reason and also to overcome the possible trap at local minima and to reduce oscillation during the training (Palit, 2005. p.241), like in BPA, a small momentum term \( m_o \) (practically in electrical load forecasting, adding \( m_o \) around 5\% to 10\% will give better results) also can be added, so that final update (2.6c) becomes
\[ w(k+1) = w(k) - \left[ J^T(w) \cdot J(w) + \mu \cdot I \right]^{-1} \cdot J^T(w) \cdot e(w) + mo \cdot (w(k) - w(k-1)) \]  

(2.6d)

Furthermore, Xiaosong et al (1995) also proposed to add modified error index (MEI) term in order to improve training convergence. The corresponding gradient with MEI can now be defined by using a Jacobian matrix as:

\[ \nabla_{\text{SSE new}}(w) = J^T(w) \cdot \left[ e(w) + \gamma \cdot (e(w) - e_{\text{avg}}) \right] \]

(2.7)

where \( e(w) \) is the column vector of errors, \( e_{\text{avg}} \) is sum of error of each column divided by number of training, while \( \gamma \) is a constant factor, \( \gamma << 1 \) has to be chosen appropriately.

Now, comes to the computation of Jacobian Matrices. The gradient \( \nabla V(W^j_0) \equiv \text{SSE} \) can be written as

\[ \nabla V(W^j_0) = \left( \partial S / \partial W^j_0 \right) = \left\{ z^j / b \right\} \cdot (f_j - d_j) \]

(2.8)

Where \( f_j \) and \( d_j \) are the actual output of the Takagi-Sugeno type MIMO and the corresponding desired output from matrix input-output training data. And then by comparing (2.8) to (2.6a), where the gradient \( \nabla V(w) \) is expressed with the transpose of the Jacobian matrix multiplied with the network’s error vector

\[ \nabla V(w) = J^T(w) \cdot e(w) \]

(2.9)

then the Jacobian matrix, the transpose of Jacobian matrix for the parameter \( W^j_0 \) of the NF network can be written by

\[ J^T(W^j_0) = \left( z^j / b \right) \]

(2.10a)

\[ J(W^j_0) = \left[ J^T(W^j_0) \right]^T = \left[ z^j / b \right]^T \]

(2.10b)

with prediction error of fuzzy network

\[ e_j = (f_j - d_j) \]

(2.11)

But if the normalized prediction error on NF network is considered, then instead of equations (2.10a) and (2.10b), the equations will be

\[ J^T(W^j_0) = \left( z^j \right) , \]

(2.12a)

\[ J(W^j_0) = \left[ J^T(W^j_0) \right]^T = \left[ z^j \right]^T \]

(2.12b)

This is because the normalized prediction error of the MIMO-NF network is

\[ e_j(\text{normalized}) = (f_j - d_j) / b \]

(2.13)

The transpose of Jacobian matrix and Jacobian matrix for the parameter \( W^j_0 \) of the NF network can be written as

\[ J^T(W^j_0) = \left( z^j / b \right) \cdot x_j \]

(2.14a)

\[ J(W^j_0) = \left[ J^T(W^j_0) \right]^T = \left[ \left( z^j / b \right) \cdot x_j \right]^T \]

(2.14b)

Also, by considering normalized prediction error from (2.11), equations (2.14a)-(2.14b) then become:
\[ J^T(W_{ij}) = (\mathbf{z}^t \cdot \mathbf{x}_i) \]  
(2.15a)

\[ J(W_{ij}) = [J^T(W_{ij})]^T = [\mathbf{z}^t \cdot \mathbf{x}_i]^T \]  
(2.15b)

Now, comes to the computation of the rest parameters \( c_j^i \) and \( \tau_j^i \) by defining the terms \( D_{eqv} \) and \( e_{eqv} \) as:

\[ A \equiv D_{eqv} \cdot e_{eqv} = (D_1 \cdot e_1 + D_2 \cdot e_2 + \cdots + D_m \cdot e_m) \]  
(2.16)

With \( e_{eqv} \) as the same amount of sum squared error that can be found by all the errors \( e_j \) from the MIMO network.

\[ e_{eqv}^p = \sqrt{e_1^p + e_2^p + \cdots + e_m^p} \]  
(2.17)

Where, \( p = 1, 2, 3, \ldots, N \) ; corresponding to \( N \) as number of training data. From (2.16), the term \( D_{eqv} \) can be determined as:

\[ D_{eqv} = A \cdot (e_{eqv})^{-1} \]  
(2.18a)

This can also be written in matrix form using pseudo inverse as:

\[ D_{eqv} = A \cdot (E_{eqv}^T \cdot E_{eqv} \cdot E_{eqv}^T)^{-1} \]  
(2.18b)

The terms \( E_{eqv} \) (is the equivalent error vector), \( D_{eqv} \) and \( A \) are matrices of size (\( Nx1 \)), (\( MxN \)) and (\( Mx1 \)) respectively. Now matrix \( A \) can be replaced with scalar product of \( e_{eqv} \) and \( D_{eqv} \).

\[ A = D_{eqv} \cdot e_{eqv} \]  
(2.19)

Now, by considering normalized equivalent error in (2.13), taking into account the equation (2.9), the transposed Jacobian matrix, the Jacobian for the parameters \( c_j^i \) and \( \tau_j^i \) can be computed as:

\[ J^T(c_j^i) = \left\{ 2 \cdot D_{eqv} \cdot z^t \cdot (x_i - c_j^i) / (\sigma_j^i)^3 \right\} \]  
(2.20a)

\[ J(c_j^i) = [J^T(c_j^i)]^T = \left[ 2 \cdot D_{eqv} \cdot z^t \cdot (x_i - c_j^i) / (\sigma_j^i)^3 \right]^T \]  
(2.20b)

\[ J^T(\sigma_j^i) = \left\{ 2 \cdot D_{eqv} \cdot z^t \cdot (x_i - c_j^i)^2 / (\sigma_j^i)^3 \right\} \]  
(2.20c)

\[ J(\sigma_j^i) = [J^T(\sigma_j^i)]^T = \left[ 2 \cdot D_{eqv} \cdot z^t \cdot (x_i - c_j^i)^2 / (\sigma_j^i)^3 \right]^T \]  
(2.20d)

It is to be noted that normalized prediction error is considered for computation of Jacobian matrices for the free parameters \( W_{0j}^i \) and \( W_{ij}^i \). Meanwhile normalized equivalent error has been considered for the computation of transposed Jacobian matrices and their Jacobian matrices respectively for the free parameters mean \( c_j^i \) and variance \( \tau_j^i \).
3. LMA with Fuzzy Clustering

The purpose of doing such fuzzy clustering before data enters the network is similar with model reduction. Because of the complexity of data, usually some number of membership functions ($M$) should be enough to bring the SSE as low as possible. In Chapter 4, it can be seen that using fuzzy clustering, such as GK clustering algorithm will reduce $M$ from 15 to 3, 5 or 7.

For detecting clusters of different geometrical shapes in one data set such as electrical data, Gustafson-Kessel clustering algorithm (GK) will be proposed according to Panchariya (Panchariya et al., 2003) and Palit (Springer, 2005, p.177-p.187).

Given the data set:

$$Z = \{Z_1, Z_2, Z_3, \ldots, Z_N\},$$  \hspace{1cm} (3.1)

needs some parameters for GK clustering algorithm as following:

- The number of clusters $1 < c < N$
- The weighting exponent or fuzziness exponent parameter $m > 1$
- The termination tolerance $\varepsilon > 0$
- The cluster volume $S$

which must be selected.

Furthermore, GK clustering algorithm is proposed with some steps:

With $U^{(l=0)} = \text{random}$, repeat for iterations $l = 1, 2, 3, \ldots$

Step 1 compute the clustering centers (mean)

$$V_g^l = \frac{\sum_{s=1}^{N} (\mu^{(l-1)}_s)^2 * Z_s}{\sum_{s=1}^{N} (\mu^{(l-1)}_s)^2}; \hspace{0.5cm} 1 \leq g \leq c$$  \hspace{1cm} (3.2a)

Step 2 determine the cluster covariance matrices

$$P_g = \frac{\sum_{s=1}^{N} (\mu^{(l-1)}_s)^2 * (Z_s - V_g^l) * (Z_s - V_g^l)^T}{\sum_{s=1}^{N} (\mu^{(l-1)}_s)^2}; \hspace{0.5cm} 1 \leq g \leq c$$  \hspace{1cm} (3.2b)

Step 3 calculate the distance

$$D_{gs}^2 = (Z_s - V_g^l)^T * \left[ S * \text{det}(P_g)^{1/n} * P_g^{-1} \right] * (Z_s - V_g^l),$$  \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} 1 \leq g \leq c, \hspace{0.5cm} 1 \leq s \leq N$$  \hspace{1cm} (3.2c)

Step 4 update the partition matrix

For $1 \leq s \leq N$

If $D_{gs}^2 > 0 \hspace{0.5cm} \text{for all} \hspace{0.5cm} g = 1, 2, \ldots, c$;
How GK clustering algorithm can be applied to the TS-type MIMO NF network for modeling and forecasting electrical load data?

Here are the steps:
1. Create $XIO$ data from electrical load
2. Random Partition matrix $U$
3. Calculate cluster centers, cluster covariance, distances, norm inducing matrices and update the partition matrix again
4. Declare parameters mean $(c)$ and sigma $(\sigma)$ from number 3
5. Calculate the remaining parameters $W_{ij}^l, W_{ij}^o$ by using least squared error algorithm (LSE)
6. Calculate SSE from LSE algorithm and save all parameters
7. Using parameters from number 6 as input parameters of TS-type MIMO NF network. Note that the initial performance $SSE$ from NF should be same with the $SSE$ result from the GK Clustering and LSE.

4. Results and Discussion

4.1. Performance of NF Networks using Clustering

In case of electricalload.txt, Clustering and LSE result can be seen on Figure 4.1a. The parameters which are determined from this part bring the $SSE = 0.5026$, $MSE = 0.0021$ and $RMSE = 0.046$. Initial $SSE$ for NF network. In addition, NF training, with 500 data training reduced initial $SSE$ from Clustering-LSE to 0.3431, $MSE = 0.0014$ and $RMSE = 0.037$. It means, NF network reduced the $SSE$ training 32% lower compared to initial $SSE$ using 499 epochs. Figures 4.1b and 4.1d show the transition of GMF from Clustering-LSE and GMF from NF network. Furthermore, for forecasting purposes, the 2800 data of electrical load as illustrated in Figures 4.1f give the result of $SSE$ forecasting = 77.8518 with $MSE$ forecasting = 0.0104 and $RMSE$ forecasting = 0.102. Full forecasting input start from data 763 (comes from 7 days multiply to 96 data plus 1) and full LTF from output NF network can follow until the next 3000 actual data (around 30 days from electrical load data).
Figure 4.1a  GK Fuzzy Clustering plus LSE Performance from electricalload.txt, n= 7, m= 1, d=96, 480 data Training, c = 5, Fuzziness exponent m=2.

Figure 4.1b  5 GMF tuned for Input 1, produced by GK cluster + LSE, Data training = 500, c=5, m=2, Input=7, output=1
Figure 4.1c Graph of SSE vs Epochs of TS-type MIMO NF network with LMA accelerate for electricalload.txt, n= 7, m= 1, Epochs= 499, d=96, Learning Rate for LMA =65, WF = 1.005, Gamma =0.05, mo = 0.05, 500 data Training, c = 5, Fuzziness exponent m=2.

Figure 4.1d Training Performance of TS-type MIMO NF network with LMA accelerate for electricalload.txt, n= 7, m= 1, Epochs= 499, d=96, Learning Rate for LMA =65, WF = 1.005, Gamma =0.05, mo = 0.05, 500 data Training, c = 5, Fuzziness exponent m=2.
Figure 4.1e 5 GMF tuned for Input 1, produced by NF network, Data training = 500, c=5, m=2, Input=7, output=1, Epochs=499, d=96, Learning Rate for LMA =65, WF = 1.005, Gamma =0.05, mo = 0.05

Figure 4.1f Forecasting Performance of TS-type MIMO NF network with LMA accelerate for electricalload.txt, n= 7 Inputs, m= 1 output, Epochs= 499, Lead Time d=96, Learning Rate for LMA =65, Wildness Factor WF = 1.005, Gamma =0.05, Momentum mo = 0.05, 4000 data Forecasting, No of clusters c = 5, Fuzziness exponent m =2
5. Summary and Conclusion

Performance results from Section 4 prove that the trained NF network using LMA is found to be very efficient in modeling and prediction of the various nonlinear dynamics. An efficient training algorithm based on combination of LMA with additional modified error index extension (MEI) and adaptive version of learning rate (momentum) have been developed to train the Takagi-Sugeno type multi-input single-output (MISO) Neuro-fuzzy network, improving the training performance. The good result from this report is combination LMA accelerated with Fuzzy Clustering (based on Gustafson-Kessel clustering algorithm). The proposed Fuzzy Clustering will reduce number of membership functions (M) and also reduces sum squared error training compared to LMA accelerate results.

Now, what can be inferred about these overall results?

First, there is still a big question in dealing with optimization. How do we know that choosing such the combination among determination of input-output and their structure, choosing the parameters, and dealing with over-/under fitting in Takagi-Sugeno-type Neuro-Fuzzy network in the same time will give the optimized performance?. Answering this question is beyond the scope of this report.

Second, another thing which is important here is that the fuzzy rules generated from these results are occasionally found to be non-transparent or less interpretable. This is due to the fact that some of the membership functions finally tuned through neuro-fuzzy network training are highly similar or overlapping on each other, giving rise to a difficult situation to interpret. To improve the transparency of fuzzy rules, set theoretic similarity measures [15] should be computed for each pair of fuzzy sets and the fuzzy sets which are highly similar should be merged together into a single one. This idea of transparency of course will increase the cost of sacrificing of the model accuracy.

The last conclusion is about long-term forecasting result. The proposed rearrangement of matrix XIO according to the appropriate lead time gives much better results compared to small lead time. By re-arranging the XIO matrix, NF network can follow the actual output until some hundreds of data. The problem in this scenario is to find the optimal lead time with the optimum number of training which gives the minimum global error in forecasting, and also the possibility to get maximum time of forecasting (long-term) before the network can not follow the actual output. Here a lot of combination and repetition of simulation is needed in order to find the suitable result in long-term forecasting. By repeating the simulation, Monte Carlo procedure can be applied to study the distribution and the statistics from data that give information on the long term distribution of time series [16]. This method reduces the risk of simulation by finding constraints simulation such as training algorithm for NF networks and reduces or deletes unwanted results of simulation. For the future, the possibility research to get optimal result is combining Hybrid Monte-Carlo procedure (HMC) with Takagi-Sugeno-type Neuro-Fuzzy network, or Hybrid HMC-TS-type NF network.
13. Palit AK, Popovic D (2005), Computational Intelligence in Time Series Forecasting, Theory and Engineering Applications, Springer