NEURO-FUZZY APPROACHES FOR FORECASTING ELECTRICAL LOAD USING ADDITIONAL MOVING AVERAGE WINDOW DATA FILTER ON TAKAGI-SUGENO TYPE MISO NETWORKS

FELIX PASILA
Electrical Engineering Department, Petra Christian University, Surabaya, Indonesia
Email address: felix@petra.ac.id

PALIT 1), THIELE 2)
1) Institut für Theoretische Elektrotechnik und Mikroelektronik,
2) Information and Automation Engineering, University of Bremen, Germany
Email address: palit@item.uni-bremen.de, thiele@iat.uni-bremen.de

ABSTRACT

The paper describes a Neuro-fuzzy approach with additional moving average window data filter and fuzzy clustering algorithm that can be used to forecast electrical load using the Takagi-Sugeno (TS) type multi-input single-output (MISO) neuro-fuzzy network efficiently. The training algorithm is efficient in the sense that it can bring the performance index of the network, such as the sum squared error (SSE), down to the desired error goal much faster than that the simple Levenberg-Marquardt algorithm (LMA). Finally, the above training algorithm is tested on neuro-fuzzy modeling and long-term forecasting application of Electrical load time series.

Key Words
Takagi-Sugeno model MISO-Neuro-fuzzy network, moving average filter, Gustafson-Kessel clustering algorithm, accelerated Levenberg-Marquardt algorithm

1. Introduction

We propose here a multi-input single-output (MISO) Takagi-Sugeno type Neuro-fuzzy (NF) network and neuro-fuzzy with additional moving average window data filter and fuzzy clustering based on Gustafson-Kessel clustering algorithm approach for modeling and forecasting application of electrical load time series. The idea behind applying moving average filter on TS-type MISO networks is an effort to find suitable and good starting point of parameters for the networks by implementing filtered electrical data (because electrical load data may has some noise on it) before entering the NF network ([6], p.42-p.71). The fuzzy clustering approach tries to reduce the complexity of the model by choosing a appropriate number of cluster as number of membership functions. The other hand, the neuro-fuzzy approach attempts to exploit the merits of both neural network and fuzzy logic based modeling techniques. For example, the fuzzy models are based on fuzzy if-then rules and are to a certain degree transparent to interpretation and analysis. Whereas, the neural networks based model has the unique learning ability. In this paper TS type MISO neuro-fuzzy network with moving average filter is constructed by multilayer feedforward network representation of the fuzzy logic system as described in section 2 and 2.1, whereas its training algorithm is described in section 2.2. NF modeling and long-term forecasting scenario, additional filter and GK algorithm are described in section 3. Simulation results are showed in section 4, and finally brief concluding remarks are presented in section 5.

2. Neuro-Fuzzy Systems selection for Forecasting

A NF network with an improved training algorithm for MIMO case was developed by Palit and Popovic [3, 4] and Palit and Babuška [2] 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 NF 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 Jang [1],
and Palit and Popovic [4], as shown in Figure 1 ([5], p.230).

For Long-term forecasting type MISO, m will be set to one. NF model as shown in Figure 1 is based on Gaussian membership functions. It uses TS type fuzzy rule, product inference, and weighted average defuzzification. The nodes in the first layer calculate the degree of membership of the numerical input values in the antecedent fuzzy sets.

![Fuzzy system MIMO feedforward Takagi-Sugeno-type Neural-Fuzzy network, m = 1](image)

The product nodes (x) represent the antecedent conjunction operator and the output of this node is the corresponding degree of fulfillment or firing strength of the rule. The division sign (/), together with summation nodes (+), join to make the normalized degree of fulfillment (z' / b) of the corresponding rule, which after multiplication with the corresponding TS rule consequent (cy'), is used as input to the last summation part (+) at the defuzzified output value, which, being crisp, is directly compatible with the actual data.

2.1. Neural Network Representation of Fuzzy Logic System (FLS)

The fuzzy logic system (FLS) considered in this paper for constructing the multi-input and single-output (MISO) neuro-fuzzy network is based on Takagi-Sugeno (TS) type fuzzy model, and with Gaussian membership functions (GMF), product inference rule, and a weighted average defuzzifier which has been explained clearly by Palit ([5], 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 NF 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 j-th rule from the above FLS can be written as

\[ R = \text{If } x_i \in G_i \text{ and } x_j \in G_j \text{ and } \ldots \text{ then } y_j' = W_{j1}x_1 + W_{j2}x_2 + \ldots + W_{jn}x_n \]  

(1)

Where, \( x_i \), with \( i = 1,2,\ldots,n \) are the \( n \) system inputs, whereas \( f_j \), with \( j = 1,2,\ldots,m \), are its \( m \) outputs, and \( G_i \), with \( i = 1,2,\ldots,n \), and \( l = 1,2,\ldots,M \) are the Gaussian membership functions of the form (1) with the corresponding mean and variance parameters \( \mu_i \) and \( \sigma_i \) respectively and with \( y_j' \) as the output consequent of the \( j \)-th rule. It must be remembered that the Gaussian membership functions \( G_i \) actually represent linguistic terms such as low, medium, high, very high, etc. The rules as written in (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 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 \( \mu_i \) and also the variance \( \sigma_i \) parameters of Gaussian membership functions, along with \( w_{i1}, w_{i2} \) 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.

\[ f_j = \sum_{i=1}^{M} f_j' \cdot h' \]  

(2a)

\[ y_j' = W_{j1}x_1 + W_{j2}x_2 + \ldots + W_{jn}x_n \]  

(2b)

\[ h' = \{ \mu_j / b \} \text{, and } b = \sum_{i=1}^{M} z'_i \]  

(2c)

\[ z'_i = \prod_{i=1}^{M} \exp \left(-\frac{x_i - \mu_i}{\sigma_i} \right) \]  

(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 [5]. 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:

\[ \Delta w = -\left[ \nabla^2 V(w) \right]^{-1} \cdot \nabla V(w) \]  

(3a)

\[ w(k + 1) = w(k) + \Delta w \]  

(3b)
From equation (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:

\[
V(w) = 0.5 \cdot \sum_{i=1}^{N} e_i^2(w)
\]  

(4)

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

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

(5a)

\[
\nabla^2 V(w) = J^T(w) \cdot J(w) + \sum_{i=1}^{N} e_i(w) \nabla^2 e_i(w)
\]  

(5b)

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

\[
J(w) = \begin{bmatrix}
\frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \ldots & \frac{\partial e_1(w)}{\partial w_N}
\end{bmatrix}
\]  

(5c)

From (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 (5b) is assumed to be zero. Therefore, the update equations according to (3a) will be:

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

(6a)

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

\[
\Delta w = -J^T(w) \cdot J(w) + \mu \cdot I \nabla V(w)
\]  

(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 (3a)

\[
w(k+1) = w(k) - J^T(w) \cdot J(w) + \mu \cdot I \nabla V(w)
\]  

(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.

Furthermore, Xiaosong et al [8] 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:

\[
\text{VSS}_{\text{eq}}(w) = J^T(w) \left[ \begin{bmatrix} e(w) + \gamma \cdot (w - e_{\text{eq}}) \end{bmatrix} \right]
\]  

(7)

where \( e(w) \) is the column vector of errors, \( e_{\text{eq}} \) 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 \( \nabla V(w_{ij}) \) = SSE can be written as

\[
\nabla V(w_{ij}) = \left[ \frac{\partial S_i / \partial w_{ij}}{f_j - d_j} \right]
\]  

(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 (8) to (6a), where the gradient \( \nabla V(w) \) is expressed with the transpose of the Jacobian matrix multiplied with the network's error vector,

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

(9)

then the Jacobian matrix, the transpose of Jacobian matrix for the parameter \( w_{ij} \) of the NF network can be written by

\[
J^T(w_{ij}) = \left[ \frac{z_i}{b} \right]
\]  

(10a)

\[
J(w_{ij}) = \left[ J^T(w_{ij}) \right] = \left[ \frac{z_i}{b} \right]
\]  

(10b)

with prediction error of fuzzy network

\[
e_j = (f_j - d_j) b
\]  

(11)

But if the normalized prediction error on NF network is considered, then instead of equations (10a) and (10b), the equations will be

\[
J^T(w_{ij}) = \left[ \frac{z_i}{b} \right]
\]  

(12a)

\[
J(w_{ij}) = \left[ J^T(w_{ij}) \right] = \left[ \frac{z_i}{b} \right]
\]  

(12b)

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

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

(13)

The transpose of Jacobian matrix and Jacobian matrix for the parameter \( w_{ij} \) of the NF network can be written as

\[
J^T(w_{ij}) = \left[ \frac{z_i}{b} \right] \cdot x_i
\]  

(14a)

\[
J(w_{ij}) = \left[ J^T(w_{ij}) \right] = \left[ \frac{z_i}{b} \right] \cdot x_i
\]  

(14b)

Also, by considering normalized prediction error from (11), equations (14a)+(14b) then become:

\[
J^T(w_{ij}) = \left[ \frac{z_i}{x_i} \right]
\]  

(15a)

\[
J(w_{ij}) = \left[ J^T(w_{ij}) \right] = \left[ \frac{z_i}{x_i} \right]
\]  

(15b)

Now, comes to the computation of the rest parameters \( c_i \) and \( r_i \) by defining the terms \( D_{eq} \) and \( e_{eq} \)

\[
A = D_{eq} \cdot e_{eq} = \left( D_1 \cdot e_1 + D_2 \cdot e_2 + \ldots + D_n \cdot e_n \right)
\]  

(16)

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

\[
e_{eq} = \sqrt{e_1^2 + e_2^2 + \ldots + e_n^2}
\]  

(17)

Where, \( p = 1, 2, 3, \ldots, N \), corresponding to N as number of training data. From (6), the term \( D_{eq} \) can be determined as
\[ D_{eq} = A \left( e_{eq} \right)^{-1} \]  
This can also be written in matrix form using pseudo inverse as 
\[ D_{eq} = A \left( E_{eq} \cdot E_{eq}^{T} \right)^{-1} \]  
The terms \( e_{eq} \) (is the equivalent error vector), \( D_{eq} \) and \( A \) are matrices of size \((N\times1), (M\times N)\) and \((M\times 1)\) respectively. Now matrix \( A \) can be replaced with scalar product of \( e_{eq} \) and \( D_{eq} \). 
\[ A = D_{eq} \cdot e_{eq} \]  
Now, by considering normalized equivalent error in (13), taking into account the equation (9), the transposed Jacobian matrix, the Jacobian for the parameters \( c_i \) and \( c_j \) can be computed as: 
\[ J^T(c_i) = \frac{1}{2} \cdot D_{eq} \cdot z^T \cdot \{ x_i - c_i \} \cdot \{ e_{eq} \}^T \]  
\[ J^T(c_j) = \frac{1}{2} \cdot D_{eq} \cdot z^T \cdot \{ x_j - c_j \} \cdot \{ e_{eq} \}^T \]  
\[ J^T(\sigma^T) = \frac{1}{2} \cdot D_{eq} \cdot z^T \cdot \{ \sigma^T \} \cdot \{ e_{eq} \}^T \]  
\[ J^T(\sigma_i) = \frac{1}{2} \cdot D_{eq} \cdot z^T \cdot \{ \sigma_i \} \cdot \{ e_{eq} \}^T \]  

3. Identification and Modelling of Nonlinear Dynamics for Long-term Forecasting

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

\[
\begin{array}{cccccccc}
\text{Day}_1 & \text{Day}_2 & \text{Day}_3 & \text{Day}_4 & \text{Day}_5 & \text{Day}_6 & \text{Day}_7 & \text{Day}_8 \\
\text{Day}_1 & \text{Day}_2 & \text{Day}_3 & \text{Day}_4 & \text{Day}_5 & \text{Day}_6 & \text{Day}_7 & \text{Day}_8 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\text{Day}_7 & \text{Day}_8 & \text{Day}_9 & \text{Day}_{10} & \text{Day}_{11} & \text{Day}_{12} & \text{Day}_{13} & \text{Day}_{14} \\
\end{array}
\]

As shown in equation (21), 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 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). After day fifteenth, all data forecasting are full-long-term forecasting.

As further examples, the modeling of other electrical load time series were considered, and it has been also observed that multi-input and multi-output NF network trained with proposed algorithm can approximate the other electrical load time series with high accuracy, especially for short-term forecasting.

3.1. Gustafson-Kessel Clustering Algorithm

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.

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 Palit ([5], p.177-p.187). Given the data set:

\[ Z = [z_1, z_2, z_3, ..., z_N] \]

needs some parameters for GK clustering algorithm as following:

1. The number of clusters \(1 < c < N\)
2. The weighting exponent or fuzziness exponent parameter \(m > 1\)
3. The termination tolerance \(e > 0\)
4. The cluster volume \(S\) which must be selected.

How GK clustering algorithm can be applied to the TS-type MISO 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, \(\text{norm}\) inducing matrices and update the partition matrix again.
4. Declare parameters mean and sigma from number.
5. Calculate the remaining parameters 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 MISO NF network. Note that the initial performance SSE from NF should be same with the SSE result from the GK Clustering and LSE.

3.2. LMA with Moving Average Window Filter

In reality, Electrical Load data usually has a number of oscillations in them as shown in Figure 2 (blue). Putting these data to the NF networks usually makes the network work very hard to follow the real data in training. The idea of changing the data using moving average filter is
nothing but try to find the good starting value of the parameters which give performance function down to desired value and after having reached them, the original data can be reconstructed again by reducing the window size to the value until window size goes to one, which is same with original data. If electrical load data can be expressed as:

\[
X_N = [X_1, X_2, X_3, X_4, \ldots X_N]
\]

then electrical load filter can be written as:

\[
X_{NF} = [X_{f1}, X_{f2}, X_{f3}, X_{f4}, \ldots X_{fN}]
\]

with

\[
X_{fj} = \frac{X_j + X_{j+1} + X_{j+2} + X_{j+3}}{W},
\]

\[
X_{(N-wj)} = \frac{X_N + X_{N-1} + X_{N-2} + X_{N-3}}{W},
\]

where Window Size (in equation 25, WS = 5) can be chosen properly by some experiments.

Let see how moving average filter works in detail. Figure 3 shows the flowchart of the filter in LMA scheme. This figure needs three steps of data filter process until coming to the original data. The first data filter with window size = 5 entered the NF network and all parameters is created by random only once. After the end of the loop of iteration, UpdateWindowSize should be reduced by subtract it with StepWindowSize=2 until the value of UpdateWindowSize equals to one. The original data entered the NF networks after UpdateWindowSize=1. This idea tries to reduce performance function by finding the best parameters using the filter. If the initial parameters using filter found somewhere are suitable for the networks, then there is possibility to find performance of SSE as low as it can be in original data.

4. Results

By using data from electricalload.txt, Clustering and LSE result can be seen on Figure 2. The parameters which are determined from this part bring the SSE = 0.5026, MSE = 0.0021 and RMSE = 0.046. They are put as 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 3 and 5 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 9 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). By using 2nd configuration of clustering, another performance is showed in Figures 10 and 11 produces training \( \text{SSE} = 0.2397, \text{MSE} = 0.00096 \) and \( \text{RMSE} = 0.022 \). Full LTF from output NF network can follow until the next 4000 data (around 40 days). We can say that each configuration of clustering will give different result of performance.

**Fig. 4.** GK Fuzzy Clustering plus LSE Performance from electricalload.txt, \( n = 7, m = 1, d = 96, 480 \) data Training, \( c = 5, \text{WS} = 5 \), Fuzziness exponent \( m = 2 \).

**Fig. 5.** 5 GMF tuned for Input 1, produced by GK cluster + LSE, Data training = 500, \( c = 5, m = 2, \text{Input} = 7, \text{output} = 1, \text{WS} = 5 \)

**Fig. 6.** Graph of SSE vs Epochs of TS-type MISO NF network with LMA, \( n = 7, m = 1, \text{Epochs} = 499, d = 96, \) Learning Rate = 65, \( \text{WF} = 1.005, \text{Gamma} = 0.05, \text{mo} = 0.05, 500 \) data Training, \( c = 5 \).

**Fig. 7.** Training Performance of TS-type MISO NF network with LMA accelerate for electricalload.txt, \( n = 7, m = 1, \text{Epochs} = 499, d = 96, \) Learning Rate for LMA = 65, \( \text{WF} = 1.005, \text{Gamma} = 0.05, \text{mo} = 0.05, 500 \) data Training, \( c = 5, \text{WS} = 5 \)

**Fig. 8.** GMF tuned for Input 1, produced by NF network, Data training = 500, \( c = 5, m = 2, \text{Input} = 7, \text{output} = 1, \) Epochs=499, \( d = 96, \) Learning Rate for LMA = 65, \( \text{WF} = 1.005, \text{Gamma} = 0.05, \text{mo} = 0.05 \)
Fig. 9. Forecasting Performance of TS-type MISO NF, 1st Configuration, n= 7 Inputs, m= 1 output, Epochs= 499, learning = 65, WF = 1.005, Gamma = 0.05, mo = 0.05, c.a. 4000 data Forecasting, c = 5, WS=5

Fig. 10. GMF tuned for Input 1, produced by NF network, 2nd Configuration, Data training = 500, c=5, m=2, Input=7, output=1, Epochs=499, d=96, Learning Rate = 65, WF = 1.002, Gamma =0.05, mo = 0.05

Fig. 11. Forecasting Performance of TS-type MISO NF, 2nd Configuration, n= 7 Inputs, m= 1 output, Epochs= 499, Lead Time d=96, Learning =65, WF = 1.002, Gamma =0.05, mo = 0.05, c.a. 5000 data Forecasting, c = 5, WS=5

5. Conclusion

Results from Section 4 demonstrate that the trained NF network using LMA is found to be very efficient in modeling and forecasting of the various nonlinear dynamics. An efficient training algorithm based on combination of LMA with additional moving average data filter, 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 case is combination LMA accelerated with Window size filter =5 and Fuzzy Clustering (no. of cluster = 5). The proposed Fuzzy Clustering will reduce number of membership functions (M) and also reduces sum squared error training compared to LMA accelerate results.

Fuzzy rules generated from these results are occasionally found to be non-transparent or less interpretable. To improve the transparency of fuzzy rules, set theoretic similarity measures [7] 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.

REFERENCES