Long-term Forecasting in Financial Stock Market using accelerated LMA on Neuro-Fuzzy structure and additional Fuzzy C-Means Clustering for optimizing the GMFs.

Felix Pasila*, Sautma Ronni**, Thiang*, Lie Handra Wijaya*

* Electrical Engineering Department, Petra Christian University
Siwalankerto 121-131, Surabaya, Indonesia
** Management Department, Petra Christian University
Siwalankerto 121-131, Surabaya, Indonesia
E-mail: felix@petra.ac.id, sautma@petra.ac.id, thiang@petra.ac.id, lie@john.petra.ac.id

1. Introduction

We propose here a multi-input single-output (MISO) Takagi-Sugeno type neuro-fuzzy (NF) network and neuro-fuzzy with additional C-Means fuzzy clustering based approaches for modeling and forecasting application of stock market time series data. The fuzzy C-Means clustering approach helps in obtaining the initial parameters of fuzzy membership functions (e.g., mean and variance parameters of Gaussian membership functions) and also tries to reduce the complexity of the model by choosing an optimum number of clusters, which is same as number of gaussian membership functions to be used for fuzzy partitioning of data in neuro-fuzzy approach. The neuro-fuzzy system 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 exclusive learning ability. In this paper, TS type MISO neuro-fuzzy network is constructed by multilayer feedforward network representation of the fuzzy logic system as described in Section 2, whereas its training algorithm (an accelerated Levenberg-Marquardt Algorithm) is described in sub-section 2.2. Thereafter, neuro-fuzzy modeling and long-term forecasting scenario, and fuzzy C-Means clustering algorithm are described in Section 3. Simulation and results are shown in Section 4 and at the end of this paper, brief concluding remarks are presented in Section 5.

2. Neuro-fuzzy Systems Selection for modeling and forecasting

A neuro-fuzzy network with an improved training algorithm for MIMO case was developed by Palit and Popovic [3, 4] and used for forecasting of electrical load data. However, the same network for MISO case was developed by Palit and Babuška [2]. Compared to ANFIS by Jang [1], the neuro-fuzzy models of [2, 3, 4]
have achieved better model accuracy and faster training performance. In addition, the neuro-fuzzy model of [3, 4, 5] is an upgraded version of Takagi-Sugeno type multiple input single output NF network. Feedforward type MIMO-NF network is proposed by Palit and Popovic [4, 3], as shown in Fig. 1. For our stock market forecasting application we will also use the same model but with number of outputs equals to one.

Neuro-fuzzy model as shown in Fig. 1 is based on Gaussian membership functions (GMFs). It uses TS type fuzzy rules, product inferencing, and weighted average defuzzifier. The Gaussian nodes \( G_i \) to \( G_M \) calculate the degree of membership of the numerical input values in the antecedent fuzzy sets.

![Fig. 1. Takagi-Sugeno-type MIMO feedforward neuro-fuzzy network](image)

The product nodes \( (\times) \) represent the antecedent conjunction operator and the output of this node is the corresponding degree of fulfillment \( (z') \) with \( i = 1, 2, \ldots, M \) representing the number of rules. 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 \( (y_j') \), is used as input to the last summation part \( (+) \) at the defuzzified output value \( y_j \), which being crisp, is directly compatible with the actual stock market time series \( X \).

2.1. Neuro implementation of fuzzy logic system

The fuzzy logic system (FLS) structure with TS type MIMO NF network considered in Fig. 1 can be easily reduced to a MISO-NF network by setting number of outputs \( m = 1 \). Takagi-Sugeno (TS) type fuzzy model, and with Gaussian membership functions (GMFs), product inference rule, and a weighted average defuzzifier can be defined as (1) (see [4] for details).

\[
f_j = \sum_{i=1}^{M} \frac{y_j'}{b_i} \cdot h_i \tag{1a}
\]

where,

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

\[
h_i = \left( \frac{x_i}{b_i} \right), \quad \text{and} \quad b_i = \frac{1}{\sum_{j=1}^{M} w_{ji}}
\]

\[
z' = \prod_{i=1}^{M} \mu_{G_i}(x_i) \cdot G_i(x_i) = \exp\left( -\frac{(x_i - c_i)^2}{\sigma_i^2} \right)
\]

The corresponding \( i\)th rule from the above fuzzy logic system (FLS) can be written as

\[
R^i : IF \ x_i \ is \ G^i_1 \ \text{AND} \ \ldots \ \text{AND} \ x_n \ is \ G^i_m \ THEN \ y_j' = W_{jy}^i + W_{j1}^ix_1 + \ldots + W_{jn}^ix_n
\]

where, \( x_i \) with \( i = 1, 2, \ldots, n \); are the \( n \) system inputs, \( f_i \) with \( j = 1, 2, \ldots, m \); are its \( m \) outputs, and \( G^i_j \) with \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, m \) are the Gaussian membership functions of form (1d) with the corresponding mean and variance parameters \( c_i \) and \( \sigma_i \) respectively and with \( y_j' \) as the output consequent of the \( i\)th rule. It must be remembered that the Gaussian membership functions \( G_i \) actually represent linguistic terms such as low, medium, high, etc. The rules as written in (2) are known as Takagi-Sugeno rules.

Fig. 1 shows that the FLS can be represented as a three layer feedforward network. Because of the neuro 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 neural network, we have here the mean \( c_i \) and also the variance \( \sigma_i \) parameters of Gaussian membership functions, along with \( w_{ji}, w_{jy} \) parameters from the rules consequent, as the equivalent adjustable parameters of the network. If all these parameters of NF network are properly selected, then the FLS can correctly approximate any nonlinear system based on given data.

2.2. Accelerated Levenberg-Marquardt algorithm on neuro-fuzzy system

The FLS, once represented as the equivalent MIMO feedforward network similar to Fig. 1, can generally be trained using any suitable training algorithm. However, because of faster convergence speed the Levenberg-Marquardt algorithm (LMA) is proposed here [4, 5].

If a function \( V(w) \) is to be minimized with respect to the parameter vector \( w \) using Newton’s method, the updated parameter vector \( w \) is defined as:

\[
\Delta w = -\nabla^2 V(w)^{-1} \cdot \nabla V(w) \tag{3a}
\]

\[
w(k + 1) = w(k) + \Delta w \tag{3b}
\]

In 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 a SSE function as follows:

\[
V(w) = 0.5 \cdot \sum_{i=1}^{N} e_i^2(w), \tag{4}
\]
then the gradient of $V(w)$ and the Hessian matrix $V^2V(w)$ are generally defined as:

$$
V V(w) = J^T(w) \cdot e(w) \quad (5a)
$$

$$
V^2V(w) = J^T(w) \cdot J(w) + \sum_{i=1}^N e_i(w) \cdot V^2e_i(w) \quad (5b)
$$

where, the Jacobian matrix $J(w)$ is as follows:

$$
J(w) = \begin{bmatrix}
\frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \cdots & \frac{\partial e_1(w)}{\partial w_N} \\
\frac{\partial e_2(w)}{\partial w_1} & \frac{\partial e_2(w)}{\partial w_2} & \cdots & \frac{\partial e_2(w)}{\partial w_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial e_N(w)}{\partial w_1} & \frac{\partial e_N(w)}{\partial w_2} & \cdots & \frac{\partial e_N(w)}{\partial w_N}
\end{bmatrix} \quad (5c)
$$

From (5c), it is seen that the matrix dimension of the Jacobian matrix is $(N_p \times N_p)$, where $N_p$ 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. Consequently, the updated equations according to (3a) will be:

$$
\Delta w = -[J^T(w) \cdot J(w)]^{-1} \cdot J^T(w) \cdot e(w) \quad (6a)
$$

Now let us see the Levenberg-Marquardt’s modifications of the Gauss-Newton method:

$$
\Delta w = -[J^T(w) \cdot J(w) + \mu \cdot I]^{-1} \cdot J^T(w) \cdot e(w) \quad (6b)
$$

where, $I$ is the $(N_p \times N_p)$ identity matrix, and the parameter $\mu$ is multiplied or divided by some factor whenever the iteration step increases or decreases the value of $V(w)$.

Here, the updated equation according to (3a)

$$
w(k+1) = w(k) - [J^T(w) \cdot J(w) + \mu \cdot I]^{-1} \cdot J^T(w) \cdot e(w) \quad (6c)
$$

It is important to recognize 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, Xiao Song et al. [9] 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:

$$
\text{VSSF}_{\text{ave}}(w) = J^T(w) \cdot [e(w) + \gamma \cdot e_{\text{ave}}] \quad (7)
$$

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

Now, the computation of Jacobian matrix can be performed as follows. The gradient $V V(w_{ij})$ can be written as:

$$
V V(w_{ij}) = [\frac{\partial e_i(w)}{\partial w_{ij}}] = \frac{[e_i/b]}{j} \cdot (f_j - d_j) \quad (8)
$$

where, $f_j$ and $d_j$ are respectively the actual output and desired output of the Takagi-Sugeno type MIMO neuro-fuzzy network. Now, by comparing (8) to (5a), where the gradient $V V(w)$ is expressed as the transpose of the Jacobian matrix multiplied with the network’s error vector, i.e.

$$
V V(w) = J^T(w) \cdot e(w) \quad (9)
$$

the corresponding Jacobian matrix and the transpose of Jacobian matrix for the parameter $w_{ij}$ of the NF network can be written as:

$$
J[w_{ij}] = \left[ \begin{array}{c}
\frac{\partial e_i}{\partial w_{ij}} \\
\frac{\partial e_j}{\partial w_{ij}} \\
\vdots \\
\frac{\partial e_N}{\partial w_{ij}}
\end{array} \right] = \left[ \begin{array}{c}
[e_i/b] \\
[e_j/b] \\
\vdots \\
[e_N/b]
\end{array} \right] \quad (10a)
$$

$$
J[w_{ij}]^T = \left[ \begin{array}{c}
\frac{\partial e_i}{\partial w_{ij}} \\
\frac{\partial e_j}{\partial w_{ij}} \\
\vdots \\
\frac{\partial e_N}{\partial w_{ij}}
\end{array} \right] = \left[ \begin{array}{c}
[e_i/b] \\
[e_j/b] \\
\vdots \\
[e_N/b]
\end{array} \right] \quad (10b)
$$

where prediction error of NF network is written as:

$$
e_j = (f_j - d_j) \quad (11)
$$

If the normalized prediction error on NF network is considered, then instead of equations (10a) and (10b), the corresponding Jacobian and transpose of Jacobian matrix will be as follows:

$$
J[w_{ij}] = \left[ \begin{array}{c}
\frac{\partial e_i}{\partial w_{ij}} \\
\frac{\partial e_j}{\partial w_{ij}} \\
\vdots \\
\frac{\partial e_N}{\partial w_{ij}}
\end{array} \right] = \left[ \begin{array}{c}
[e_i/b] \\
[e_j/b] \\
\vdots \\
[e_N/b]
\end{array} \right] \quad (12a)
$$

$$
J[w_{ij}]^T = \left[ \begin{array}{c}
\frac{\partial e_i}{\partial w_{ij}} \\
\frac{\partial e_j}{\partial w_{ij}} \\
\vdots \\
\frac{\partial e_N}{\partial w_{ij}}
\end{array} \right] = \left[ \begin{array}{c}
[e_i/b] \\
[e_j/b] \\
\vdots \\
[e_N/b]
\end{array} \right] \quad (12b)
$$

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

$$
e_j\text{ (normalized)} = (f_j - d_j)/b \quad (13)
$$

In the same way, the transpose of Jacobian matrix and Jacobian matrix itself for the parameter $w_{ij}$ of the NF network can be written as:

$$
J[w_{ij}] = \left[ \begin{array}{c}
\frac{\partial e_i}{\partial w_{ij}} \\
\frac{\partial e_j}{\partial w_{ij}} \\
\vdots \\
\frac{\partial e_N}{\partial w_{ij}}
\end{array} \right] = \left[ \begin{array}{c}
[e_i/b] \\
[e_j/b] \\
\vdots \\
[e_N/b]
\end{array} \right] \quad (14a)
$$

$$
J[w_{ij}]^T = \left[ \begin{array}{c}
\frac{\partial e_i}{\partial w_{ij}} \\
\frac{\partial e_j}{\partial w_{ij}} \\
\vdots \\
\frac{\partial e_N}{\partial w_{ij}}
\end{array} \right] = \left[ \begin{array}{c}
[e_i/b] \\
[e_j/b] \\
\vdots \\
[e_N/b]
\end{array} \right] \quad (14b)
$$

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

$$
J[w_{ij}] = \left[ \begin{array}{c}
\frac{\partial e_i}{\partial w_{ij}} \\
\frac{\partial e_j}{\partial w_{ij}} \\
\vdots \\
\frac{\partial e_N}{\partial w_{ij}}
\end{array} \right] = \left[ \begin{array}{c}
[e_i/b] \\
[e_j/b] \\
\vdots \\
[e_N/b]
\end{array} \right] \quad (15a)
$$

$$
J[w_{ij}]^T = \left[ \begin{array}{c}
\frac{\partial e_i}{\partial w_{ij}} \\
\frac{\partial e_j}{\partial w_{ij}} \\
\vdots \\
\frac{\partial e_N}{\partial w_{ij}}
\end{array} \right] = \left[ \begin{array}{c}
[e_i/b] \\
[e_j/b] \\
\vdots \\
[e_N/b]
\end{array} \right] \quad (15b)
$$

Now, the Jacobian matrix computation of the remaining parameters $e_i^\dagger$ and $e_j^\dagger$ are performed by defining the terms $D_\text{ave}$ and $e_\text{ave}$ as

$$
A = D_\text{ave} \cdot e_\text{ave} = (D_1 \cdot e_1 + D_2 \cdot e_2 + \cdots + D_m \cdot e_m) \quad (16)
$$

Where, $D_j = (y_j - f_j)$ and $e_j = (f_j - d_j)$ with $j = 1, 2, ..., m$ and the term $e_\text{ave}$ is such that it contributes the same amount of sum squared error that can be obtained jointly by all the errors $e_j$ from the MIMO network.

Therefore,

$$
e_\text{ave} = \sqrt{(e_1^2 + e_2^2 + \cdots + e_m^2)} \quad (17)
$$

Where, $p = 1, 2, ..., N_l$ corresponding to $N$ as number of training samples/data. From (16), the term $D_\text{ave}$ can be determined as
\[ D_{eq} = A \cdot (e_{eq})^T \]  
(18a)

This can also be written in matrix form using pseudo inverse as
\[ D_{eq} = A \cdot (E_{eq})^\dagger \cdot (E_{eq} \cdot e_{eq})^T \]  
(18b)

The terms \( e_{eq} \) (is the equivalent error vector), \( D_{eq} \) and \( A \) are matrices of size \((N \times 1)\), \((M \times N)\) and \((M \times 1)\) respectively. Now matrix \( A \) can be replaced with scalar product of \( e_{eq} \) and \( D_{eq} \).
\[ A = D_{eq} \cdot e_{eq} \]  
(19)

Note that for MISO neuro-fuzzy network, i.e., for \( m = 1 \) and \( A = D_1 \cdot e_1 \), \( D_{eq} = D_1 \) and \( e_{eq} = e_1 \) hold. This means that in this case equations (16) - (19) need not be computed.

Now, by considering normalized equivalent error in (13), taking into account the equation (9), the transposed Jacobian matrix, the Jacobian for the parameters \( e_i \) and \( \sigma_i \) can be computed as:
\[ J^T[e_i] = \left[ 2 \cdot D_{eq} \cdot z' \cdot (v_i - c_i) / \sigma_i \right] \]  
(20a)
\[ J^T[\sigma_i] = \left[ 2 \cdot D_{eq} \cdot z' \cdot (v_i - c_i) / \sigma_i \right] \tau \]  
(20b)
\[ J^T[e_i] = \left[ 2 \cdot D_{eq} \cdot z' \cdot (v_i - c_i) / \sigma_i \right] \]  
(20c)
\[ J^T[\sigma_i] = \left[ 2 \cdot D_{eq} \cdot z' \cdot (v_i - c_i) / \sigma_i \right] \tau \]  
(20d)

The above procedure describes actually layer by layer computation of Jacobian matrices for various parameters of neuro-fuzzy network from the backpropagation results (see [4] for details).

3. Identification and Modeling

3.1 Modeling of Nonlinear Dynamics for Long-term Forecasting

For long-term forecasting of stock market TS type MISO neuro-fuzzy predictor will be used. Therefore, the given time series data have been organized as seven inputs and one output XIO data matrix of (21).
\[ XIO_{7 \times 1} = \begin{bmatrix} \text{Day}_1 & \text{Day}_2 & \text{Day}_3 & \text{Day}_4 & \text{Day}_5 & \text{Day}_6 & \text{Day}_7 \\ \text{Day}_2 & \text{Day}_3 & \text{Day}_4 & \text{Day}_5 & \text{Day}_6 & \text{Day}_7 & \text{Day}_8 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{Day}_{16} & \text{Day}_{17} & \text{Day}_{18} & \text{Day}_{19} & \text{Day}_{20} & \text{Day}_{21} & \text{Day}_{22} \end{bmatrix} \]  
(21)

As can be seen from the first row of XIO matrix in (21), five days of stock market data (five inputs mean 5 days active stock market, Saturday and Sunday are not included) are applied as network input to produce sixth day data as predicted output (indicated by Day6 with hat symbol) of NF network. Likewise, in the second row of XIO matrix input data consisting of Day2 till Day5 and the previous prediction of Day6 data have been applied to the NF network to further predict the Day7 data. Therefore, input data of sixth row of XIO matrix will consist of previous predictions of Day6 till Day10 data to further predict the output data Day11. Likewise, further rows of XIO matrix can be built up. Therefore, long term forecasting of stock market starts actually from sixth row onwards, where each input data consist of previously predicted data by NF network. Note that for short-term stock market forecasting only the desired output data (i.e., sixth column of XIO matrix) from the network are replaced by the predicted data, whereas all input data consist of original data values obtained from the stock market time series. Here, each day (24 hrs) represents actually only one data point, as in the case of JCI series data have been recorded at the sampling interval of every 24 hours. This implies that only one JCI data have been recorded within one day. Note that for training purpose of neuro-fuzzy network all the predicted data (i.e., Day6 until Day10 and onwards) in XIO matrix are actually replaced by their original data values. Usually, a few hundreds of rows of XIO matrix are used for network training, whereas remaining data (also a few hundreds of rows of XIO data) are used for short term or long-term forecasting of stock market data.

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 series with high accuracy, especially for short-term forecasting [3, 4].

3.2. C-Means Clustering Algorithm

The C-Means fuzzy clustering is actually performed on the given training data set even before the neuro-fuzzy network’s training. The purpose of applying fuzzy clustering algorithm on the stock market data is to basically find the optimum or near optimal number of clusters (Gaussian) membership functions and their initial parameters (mean and variance), which will be used as the good starting parameters of neuro-fuzzy networks during its training. For detecting clusters of different geometrical shapes in a given data set such as time series data, Gustafson-Kessel clustering algorithm (GK) is also can be generally proposed [4, 7].

Given the data set:
\[ Z = [Z_1, Z_2, Z_3, ..., Z_n] \]  
(22)
where, \( Z \) is represented by \( n \times N \) pattern matrix, then select the number of clusters \( 1 < c < N \), the weighting exponent or fuzziness exponent parameter \( m > 1 \), the termination tolerance \( \varepsilon > 0 \).
Furthermore, C-Means clustering algorithm proceeds with randomly generated initial partition matrix $U^{(i)}$ where, $U = [\mu_g]_{c \times N}$.

- Create XIO matrix; Example XIO for 3 inputs and 1 output on z representation:
  $$Z_s(XIO) = [Z_{s1}(input_1), Z_{s2}(input_2), Z_{s3}(input_3), Z_{s4}(output)]$$
- Set fuzziness parameter $m = 2$ ($m$ equals to 2 means not to Fuzzy).
- Set Termination tolerance $\epsilon = 0.001$
- Set number of cluster $c$.
- Determine $A$ matrix, $A$ is matrix inverse from matrix covariance, $A = R^{-1}$, where
  $$R = \frac{1}{\sum_{i=1}^{N}N} \sum_{j=1}^{N} (Z_{s} - Z)(Z_{s} - Z)^T$$
  $All\ N$ is representation for all data JCI.

**Repeat iteration** $L = 1, 2, 3$

1. Calculate means from cluster
   $$\mu^{(i)} = \frac{\sum_{j=1}^{N} (\mu_g^{(i-1)}) \cdot Z_s}{\sum_{j=1}^{N} (\mu_g^{(i-1)})^2}; 1 \leq g \leq c$$
   (23)
2. Calculate cluster distance
   $$D_{gst} = (Z_s - v_g^{(i)})^T A (Z_s - v_g^{(i)})$$
   (24)
3. Update partition matrix:
   If $D_{gst} > 0$, for $g = 1, 2, 3, \ldots, c$
   $$\mu^{(i)} = \frac{1}{\sum_{k=1}^{c} (\mu_g^{(i-1)})}; 1 \leq g \leq c; 1 \leq s \leq N$$
   else,
   $$\mu^{(i)} = 0 \text{ and } \mu^{(i)} \in [0,1] \text{ with } \sum_{g=1}^{c} \mu_g^{(i)} = 1$$
   (25)
   (26)

Until
   $$||U^{(i)} - U^{(i-1)}|| < 0.001$$
   (27)

How fuzzy C-Means clustering algorithm is proposed on NF structure can be seen on Fig.2.

In this scheme, starting parameters (means, variance and rule consequent parameters) are coming not from random, but those are approaching from fuzzy C-Means and Least Squared Error mechanism (LSE).

On the other hand Fig. 3 shows that fuzzy C-Means uses 5 GMF inputs (means number of cluster equals to 5) to find suitable starting parameter before enter the network.
4. Simulation Experiments and Results

For modeling purposes, training and validation will use data from September 2003 to December 2006, the rest 10 months will use for long-term forecasting. The parameters which are determined from fuzzy cluster C-Means and LSE bring the $SSE_{\text{Cluster}} = 0.0819$, $MSE_{\text{Cluster}} = 3.2760e-004$ and $RMSE_{\text{Cluster}} = 0.0181$, display of actual and cluster’s results can be seen on Fig. 5. In addition, these performances are used as initial performance functions of NF network.

Figs. 4 and 6 show graph of SSE versus epochs and NF network’s training performance with 500 data points. Training algorithm has reduced the SSE down to 0.0468 from its initial SSE of 0.0819. The corresponding final MSE = 1.8738e-004 and RMSE = 0.0137. By using only some hundreds epochs, SSE can found down to minimum. The figure also tent to go to minimum if we choose another hundreds epochs.

![Plot of SSE Vs. Epoch](image)

**Fig. 4.** Graph of SSE vs Epochs of TS-type MISO NF network training with LMA, $In=5$, $Out=1$, Epochs=300, training data=500, $c=5$.

![Output from Fuzzy C-Means + LSE](image)

**Fig. 5.** Fuzzy C-Means and LSE Performance, Data training $= 500$.

Furthermore, for long-term prediction purposes, the 200 data of JCI as illustrated in Fig. 7 give the result of SSE LTF forecasting = 0.0388 with MSE prediction = 3.8000e-004 and RMSE forecasting only = 0.0196, means each average error prediction from NF results only 1.96%. Compare to previous paper from Pasila et.al [6], NF network results without fuzzy C-Means was 2.55%.

Every result from this paper is still un-optimized results. This means, it is possible to find another combination parameters such as number of cluster, number of input/output fuzzy, number of training data, etc, that gives better results (by hundred experiments could find only small improvement) compare to the previous one.
5. Conclusion

Neuro-fuzzy approaches with fuzzy clustering algorithm have been presented for long term forecasting of stock market in Indonesia. It has been demonstrated that fuzzy clustering of stock market data, followed by neuro-fuzzy network and trained with accelerated Levenberg-Marquardt algorithm, is very efficient in modeling and forecasting of the stock market series in the next 10 months in advanced. The better result has been obtained when Fuzzy Clustering (with number of cluster $c = 5$, training data = 500) combined with accelerated training of neuro-fuzzy network is used will bring percentage of RMSE = 1.96%.

Several issues need to be addressed in the future works which includes mainly transparency and interpretability of generated fuzzy model (rules). For the latter issue set theoretic similarity measures [8] 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.

Additional information such as fundamental economy data of the company or the country will bring the performance of stock market forecasting more accurate and representative. This issue is known as multivariate input of Neuro-Fuzzy network.

References: