Chaotic Time Series Forecasting using Locally Quadratic Fuzzy Neural Models

Mohammad J. Mahjoob
Assistant Professor, School of Mechanical Engineering, University of Tehran, Tehran, Iran

Majid Abdollahzade
Grad. Student, School of Mechanical Eng., University of Tehran, Tehran, Iran

Reza Zarringhalam
Grad. Student, School of Mechanical Eng., K. N. Toosi University of Technology, Tehran, Iran

Ahmad Kalhor
Electrical and Computer Engineering Department, Control and Intelligent Processing Center of Excellence, University of Tehran, Iran

Abstract: Time series forecasting in highly nonlinear and chaotic systems is a challenging research area with a variety of applications in economics, environmental sciences and various fields of engineering. This paper presents a novel Locally Quadratic Fuzzy Neural Model (LQFNM) to forecast the behavior of highly nonlinear and chaotic time series. It is based on the idea of approximating a nonlinear function with interpolated local quadratic models using a tree construction algorithm. A fast heuristic learning algorithm is integrated in the model to derive the structure as well as the parameters of the Locally Quadratic Models. Four different case studies are conducted in which the performance of the method is evaluated through comparisons with other techniques available in literature. The results confirm the accuracy and reliability of the presented method. The proposed LQFNM can be applied to time series forecasting in a wide range of real world applications.

Keywords: Locally Quadratic Neural Fuzzy Model, Forecasting, Chaotic Time Series

1 Introduction
The tendency to discover unknown aspects of different phenomena has occupied human being’s mind all over the history. Rather than being a matter of curiosity, forecasting is admired to be used as a basis to perform indispensable preparedness for an upcoming event. In its early ages and as a superstitious concept, forecasting was prematurely aimed to reinforce human against the unpleasant natural catastrophes and is nowadays concerned as a scientific capability with a wide range of applications with a variety of purposes.

In recent years, time series forecasting has been regarded as an interesting as well as a challenging research area by the experts of different disciplines including economics, environmental sciences, astronomy and also various fields of engineering, e.g. artificial intelligence, decision making, system identification, intelligent processing and control.

The chaotic dynamical systems in nature are characterized by high sensitivity to initial conditions which results in long term unpredictability. The dynamical reconstruction of such systems seems to be impracticable not because of computational complexity, but due to inaccessibility of perfect inputs and state variables. This fact is the main reason of using data driven approaches, especially the learning and optimization techniques based on general function approximators like neural networks and neurofuzzy models.

Fuzzy modeling is a powerful and practical tool for modeling and control of nonlinear systems and complex processes. The Takagi-Sugeno and Kang (TSK) fuzzy model is a major topic in the field of fuzzy control and modeling [1]. Through using this fuzzy reasoning method, input data space is decomposed into fuzzy regions and each region is approximated by a simple
linear function ([2], [3] and [4]). The output of this network of interpolated subsystems can be defined as a weighted sum of all outputs generated by the subsystems [5]. The TSK model offers a computationally efficient and relatively precise solution to a wide range of prediction and control applications, e.g., [1], [6], [7], [8].

However, this model encounters a dimensionality problem dealing with complex nonlinear systems. In fact, these systems require numerous, if not infinite, locally linear subsystems to achieve good fitness especially in complex manifolds of the system. A possible solution can be using more flexible locally linear subsystems instead of simple linear ones, an approach that decreases the number of required subsystems for satisfactory performance of the model.

Quadratic models are simple and straightforward types of polynomial models, which have been applied for modeling different nonlinear systems ([9], [10], [11], [12] and [13]). These models have also been used as local subsystems in nonlinear system identification ([14], [15] and [16]). However, the range of applications of Quadratic Models is limited to specific cases and this approach has not been considered as a general modeling tool.

In this paper, a novel fuzzy neural TSK system constructed by locally quadratic models is introduced for forecasting purposes in highly nonlinear and chaotic time series. A fast heuristic model tree algorithm ([16][17]) is proposed to partition the input space to validation regions of locally quadratic models, and locally weighted least squares optimization is applied to estimate the parameters of the subsystems. The algorithm, termed as Locally Quadratic Fuzzy Neural Model (LQFNM), is applied to perform forecasting in four highly nonlinear and chaotic time series case studies: Behavior of analytical Mackey-Glass chaotic time series, monthly smoothed sunspot numbers forecasting, forecasting the exchange rate between Mexican Peso and US Dollar, and electricity price forecasting in a competitive power market. Each case study contains comparisons with available approaches proposed in literature. Superior performance of the presented method is demonstrated in all of the four forecasting applications, verifying the LQFNM method to be a reliable and efficient technique for forecasting applications in real word forecasting applications.

2 Locally Quadratic Fuzzy Neural Model
Locally Quadratic Fuzzy Neural Model (LQFNM) is based on the idea of approximating a nonlinear function with interpolated local quadratic models using a tree construction algorithm [17]. The algorithm has two main stages at every epoch: at the first stage, the structure of the model (premise parts of rules) is determined and in the second one, consequent parameters of quadratic models are calculated. The structure of model evolves through splitting a Local Quadratic Model (LQM) to two new LQMs at each epoch. LQMs (or sometimes-called neurons) are interpolated using normalized orthogonal Gaussian weighting (membership) functions. Output of the LQFNM is calculated by summing up the weighted outputs of all local quadratic models.

If input of the system has p dimensions \((x_1, x_2,..., x_p)\), then input of LQFNM will be as follows:

\[ u_i = x_i \quad \text{for} \quad i = 1, 2, ..., p \]

and \( u_i = x_k x_j \quad \text{for} \quad i = p, p+1, ..., p(p+3)/2 \) (where \( k = 1, 2, ..., p \) and \( j = k, k+1, ..., p \))

by doing so, the p dimensional input space is transformed into a \( r = \frac{p(p+3)}{2} \) dimensional quadratic space. Consequently:

\[
\hat{y}_i = w_i^T \bar{u} \quad \text{where} \quad \bar{u} = \begin{bmatrix} u_1 \\ 1 \end{bmatrix}
\]

\[ \hat{y} = \sum_{i=1}^{M} \hat{y}_i \phi_i(u) \quad (2) \]

where \( u = [u_1, u_2, ..., u_p]^T \) is the input of the model, M is the number of local quadratic models (neurons), and \( w_i \) denotes the consequent parameters of the ith LQM. Weighting (validity) functions are in the form of normalized orthogonal Gaussians. This normalization is necessary for a proper interpretation of validity functions.

\[
\phi_i(u) = \frac{\mu_i(u)}{\sum_{j=1}^{M} \mu_j(u)} \quad (3)
\]

\[
\mu_i(u) = \exp \left\{ -\frac{1}{2} \left( \frac{(u_1-c_{1i})^2}{\sigma^2_1} + ... + \frac{(u_p-c_{pi})^2}{\sigma^2_p} \right) \right\} \]

\[
= \exp \left\{ -\frac{1}{2} \left( \frac{(u_1-c_{1i})^2}{\sigma^2_1} \right) \right\} \times ... \times \exp \left\{ -\frac{1}{2} \left( \frac{(u_p-c_{pi})^2}{\sigma^2_p} \right) \right\} \quad (4)
\]

Figure 1: Network structure of the LQFNM model with M neurons.
It can be seen from (4) that every Gaussian weighting function has two sets of nonlinear parameters (which are calculated in the first stage): $c_{ij}$ centers and $\sigma_{ij}$ standard deviations along the jth coordination and ith LQM.

Assuming that nonlinear parameters of membership functions have been already determined, consequent parameters for each LQM are estimated separately (locally) by weighted least squares method: [11]

$$\mathbf{w}_j = (\mathbf{U}^T \mathbf{Q}_j \mathbf{U})^{-1} \mathbf{U}^T \mathbf{Q}_j \mathbf{Y}$$

(5)

In equation (5), $\mathbf{U}$ is data matrix with each row representing one measurement of input data at time instance $k$:

$$\mathbf{U}_k = [\mathbf{u}_k^T, 1]$$

(a bias term “1” is added to each row), $\mathbf{Y}$ is the desired output vector and $\mathbf{Q}_j$ is the diagonal weighting matrix with each diagonal element corresponding to data $\mathbf{u}_k^T$:

$$\mathbf{Q}_j = \text{diag}(\{\mu_i(\mathbf{u}_k) \}^N_{k=1})$$

(7)

where N is total number of data points. In the local parameter estimation in (5) we ignore the overlap of weighting errors (because the number of neurons cannot be very large) and bias errors. However, local estimation approach is much more computationally efficient than global estimation, because the global estimation generates variance errors [17]. Local estimation is a very fast and robust algorithm. In this method, instead of calculating the parameters of all of the LQMs simultaneously, consequent parameters of only one LQM are calculated in each iteration.

Calculation of nonlinear parameters of membership functions is performed in three fundamental iterative steps:

1) Among all of the LQMs, the one having the biggest sum of weighted squared errors is selected to be partitioned:

$$E_i = e^T \mathbf{Q}_i e$$

$$i^{\text{w}} = \text{arg}(\text{max}(E_i)) \quad (i = 1, 2, \ldots, M)$$

(9)

where $e$ is the error vector between network output and the desired target and $i^{\text{w}}$ denotes the worst neuron.

2) The Selected LQM can be divided into two new LQMs along all r possible directions (input dimensions). A very fast heuristic technique is proposed to find the best division direction. This technique is based on normalized cross correlation between regressors vectors ($\mathbf{X}_i = \mathbf{U}(\cdot, i) \quad (i = 1, 2, \ldots, r)$) and output vector ($\mathbf{Y}$). This criterion is defined as:

$$C_i = \frac{\mathbf{X}_i^T \mathbf{Y}}{||\mathbf{X}_i|| \cdot ||\mathbf{Y}||} \quad i = 1, 2, \ldots, r$$

(10)

The cross correlation shows the linear correlation between the regressors vector and the output vector. The heuristic idea suggests that the regressors vector with small correlation to the output vector may have more nonlinearity than the ones which have large correlations. Therefore, to have more reduction in nonlinearity of the system at a selected LQM, it is divided in a direction so as to the corresponding regressors vector has the least cross correlation with $\mathbf{Y}$.

$$i^{b} = \text{arg}(\text{min}(C_i)) \quad (i = 1, 2, \ldots, r)$$

(11)

Where $i^{b}$ denotes the best division. An example which illustrates that how a LQM is partitioned into two new LQMs when $r=2$ is illustrated in figure 2. the validation regions for each LQM can be considered as a hyper-rectangle due to using normalized Gaussian functions.

![Figure 2: Worst LQM ($i^{\text{w}}$) has been partitioned through best division into two new LQMs ($i^{b1}$, $i^{b2}$).](image)

3) The algorithm finishes if the stop criteria are satisfied, otherwise, it returns to step 1. The stop criteria can be set to indicate the situation in which the model generates less validation error.

Nonlinear parameters are easily found: $c_{ij}$ s are centers of hyper-rectangles and $\sigma_{ij}$ s are chosen proportional to the dimensions of hyper-rectangles by the factor $1/\alpha$:

$$\sigma_{ij} = L_{ij} / \alpha$$

(12)

Parameter $1/\alpha$ is called “Smoothing Factor” and determines the amount of overlapping between neighbor LQMs. Actually, there is no technique to assign the value of $\alpha$ prior to simulations but $\alpha$ is typically set to be 3 [17].

The LQFNM algorithm can be summarized as follows:

1- Define an initial hyper-rectangle, which fits to all available data points in input space. Estimate the parameters of the first LQM (M=1) through (4).
2- From the available M LQMs, choose the worst LQM regarding (8).
3- Find the best division cut for the worst LQM through (10) and (11). Define two new LQMs instead of the worst LQM as explained (M=M+1).
4- Go to step 2 until the stop condition is satisfied.

3 Case Studies
Four case studies are conducted to demonstrate the capability of the LQFNM method. The case studies are presented in the following subsections.

3-1 Case Study 1: Forecasting of Chaotic Mackey-Glass Time series
The Mackey–Glass system has been introduced as a model of white blood cell production [19]. This time series is produced by a time-delay difference system of the form of:

\[
\frac{dx(t)}{dt} = \beta x(t) + \frac{\alpha x(t-\tau)}{1+x^{10}(t-\tau)}
\]  

(13)

where \( x(t) \) is the value of the time series at time \( t \). This system is chaotic for \( \tau > 16.8 \). In this case study the time series was constructed with the following parameter values \( \alpha = 0.2, \beta = -0.1, \tau = 17 \) and \( x_0 = 1.2 \).

From the beginning of the series, 500 samples were selected for training and the next 700 samples were used for testing for long-term prediction.

Figure 3 shows the forecasting results for the first 200 samples in the test period using the proposed LQFNM method.

In order to investigate the proposed method’s performance, a comparison is presented in table 1. The methods named in this table are developed considering the same set of parameters for the Mackey-Glass time series and the same train and test scopes.

In this comparison:

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}
\]

(14)
in which \( y_i \) and \( \hat{y}_i \) are observed output and predicted output respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Learning Algorithm</th>
<th>Neurons</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>BP</td>
<td>43</td>
<td>0.0016</td>
</tr>
<tr>
<td>T-S</td>
<td>ANFIS</td>
<td>16</td>
<td>0.0015</td>
</tr>
<tr>
<td>RBF</td>
<td>OLS</td>
<td>42</td>
<td>0.00102</td>
</tr>
<tr>
<td>LLNF</td>
<td>LOLIMOT</td>
<td>47</td>
<td>0.000961</td>
</tr>
<tr>
<td>LQFNM</td>
<td>Local Estimation</td>
<td>45</td>
<td>0.000543</td>
</tr>
</tbody>
</table>

The Multi Layer Perceptron (MLP) Network, Radial Basis Functions (RBF) Network, and LLNF network are developed in [20] and use Back-Propagation (BP), Orthogonal Least Square (OLS), and Locally Linear Model Trees (LOLIMOT) Learning algorithms respectively. The Takagi-Sugeno (T-S) Fuzzy system approach is proposed in [21] and uses ANFIS learning algorithm. The number of necessary neurons for each method is also included in the table 1. The comparison shows satisfactory accuracy of the LQFNM method and the reasonable number of required Neurons to achieve this performance.

3-2 Case Study 1: Predicting sunspot number time series
In this case study, prediction of monthly smoothed sunspot number time series (as a well studied index of space weather phenomena) is considered. Sunspot number prediction is known as a very important practical example. Some space weather risks and hazards on satellites, power systems, communication systems, and several other space borne and earth borne systems can be related to this factor. These risks necessitate the need to design reliable warning systems which use forecasting as their core element.

The monthly smoothed sunspot number time series is available in SIDC (World Data Center for sunspot Index) [22]. In order to compare the results with some of the previous studies, the LQFNM algorithm is tested in the similar order of train and test data conditions. The training phase starts by the first 1000 samples of the database, from Jan. 1900 until Jan. 1999.
The trained LQFNFM is then used for prediction of the 23rd solar cycle from Jan. 1999 to May 2001. The normalized MSE index (NMSE) defined as (15) is used to compare the performance of the models:

\[
NMSE = \frac{\sum_{i=1}^{Q} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{Q} y_i - \bar{y}}
\]  

(15)

where

\[
\bar{y} = \frac{\sum_{i=1}^{Q} y_i}{Q}
\]  

(16)

Figure 4 shows the forecasted and original sunspot time series. Table 2 summarizes the results of LQFNFM algorithm and some other available approaches developed in recent years.

<table>
<thead>
<tr>
<th>Methods</th>
<th>NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sello [23]</td>
<td>0.34</td>
</tr>
<tr>
<td>McNish-Linkon [23]</td>
<td>0.08</td>
</tr>
<tr>
<td>LLNF [20]</td>
<td>0.046</td>
</tr>
<tr>
<td>RBF_OLS [20]</td>
<td>0.032</td>
</tr>
<tr>
<td>LQFNFM</td>
<td>0.016</td>
</tr>
</tbody>
</table>

3-3 Case Study 3: Exchange rate Forecasting: Financial systems are characterized by high uncertainty, nonlinearity and time varying behavior. The proposed method is applied in the third case study in order to forecast currency exchange rate between United States Dollar and Mexican Peso (MXNUSD). The time series of this Index contains the data of Jan. 2 1997 to Jun. 17 2006 as the training and validation data set. This dataset is used to forecast the MXNUSD index for the test period of Jun. 18 2006 to Oct. 12 2006. The data is available in [24].

Figure 5 shows the results of the LQFNFM method for the test period in comparison with the actual observations. In order to forecast the same time series, a Fuzzy system is developed in [25] having four delays in its input. The comparison presented in table 3 shows the better performance of the proposed LQFNFM method.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuzzy</td>
<td>0.020</td>
</tr>
<tr>
<td>LQFNFM</td>
<td>0.013</td>
</tr>
</tbody>
</table>

3-4 Case Study 4: Electricity Price Forecasting: Reconstruction of electricity markets has become widespread all over the world in recent years. The market mechanism has replaced the traditional monopolistic approaches in many countries; a process that has bred a developing tendency in other countries to apply this system in their power markets, making a competitive environment in which a proper understanding of market’s behavior is essential to make profit.

As the 4th case study, the proposed LQFNFM method is used to forecast the Market Clearing Price in a competitive power market. The Mainland Spain Electricity market is considered as the target market. Time series of the historical prices for this market is available in [26]. The MCP in this market is forecasted for a winter week period. Precisely, the MCP of Feb. 18 to Feb. 24, 2002 is forecasted using the data of 42 previous days, Jan. 7 to Feb. 17, 2002. An appropriate input selection for the model can have a crucial positive effect on the forecasting performance. Common appropriate inputs, used in [27] and [28], aiming to forecast the MCP in hour \( h \), \( P_h \), are as follows:

\[
P_{h-1}, P_{h-2}, P_{h-3}, P_{h-24}, P_{h-25}, P_{h-48}, P_{h-49}, P_{h-72}, P_{h-73}, P_{h-96}, P_{h-97}, P_{h-120}, P_{h-121}, P_{h-144}, P_{h-145}, P_{h-168}, P_{h-169}, P_{h-192}, P_{h-193}
\]

where \( P_{h-i} \) is the price in \( i \) hours ago. We performed an autocorrelation analysis to investigate the appropriateness of the stated inputs. The autocorrelation analysis verifies that the selected inputs are reasonable for forecasting models. Regarding the 19 selected inputs, a total of 815 training and
validation observes and 168 testing observes are used in the forecasting process. Figure 6 illustrates the forecasted and actual prices, confirming the desirable performance of the proposed LQFNM method.

![Figure 6: Electricity Price Forecasting](image)

Additionally, a comparison is presented to evaluate the proposed method’s performance. Works that attempted to forecast Market Clearing Prices in the Spanish power market in the aforementioned time periods are incorporated in comparisons. Table 4 shows the comparison results with two error criteria, MAPE and VPE, for the test period in winter season of the year in the Spanish market.

These error criteria are defined as follows:

\[
MAPE = \frac{100}{N} \sum_{h=1}^{N} \frac{|\hat{p}_h - p_h|}{\bar{p}}
\]  
(17)

Where

\[
\bar{p} = \frac{1}{N} \sum_{h=1}^{N} p_h
\]  
(18)

\[
VPE = \frac{1}{N} \sum_{h=1}^{N} \left( \frac{\hat{p}_h - p_h}{\bar{p}} - \frac{MAPE}{100} \right)^2
\]  
(19)

Naïve, ARIMA and ARIMA-WAVELET models are discussed in [27]. In the ARIMA-WAVELET model, having better results compared to the other two methods, the MCP series has been decomposed into four constitutive series through Wavelet transform. Then an ARIMA model is used for each constitutive series. In the stand alone ARIMA model, a single ARIMA model is allocated to the whole MCP series.

Fuzzy Neural Networks (FNN) approach is proposed in [28] consisting of a three inter-layer and feed-forward architecture with a hyper-cubic training mechanism. The Proposed LQFNM model yields remarkable better performances for both error criteria.

### 4 Conclusion

This paper presents a Locally Quadratic Fuzzy Neural Model (LQFNM) to forecast the behavior of highly nonlinear and chaotic time series. A fast heuristic learning algorithm is developed to derive the structure as well as the parameters of Locally Quadratic Models.

The method was used in four different time series forecasting problems namely Mackey-Glass chaotic time series, monthly smoothed sunspot number time series, forecasting the daily exchange rate between Mexican Peso and US Dollar, and electricity price forecasting in a competitive power market. Efficiency and accuracy of the method was demonstrated through comparisons with other available techniques developed in recent years. The LQFNM is a competent approach to be used in real world applications dealing with highly nonlinear and chaotic time series forecasting, as well as a variety of other forecasting, prediction and estimation applications.

### Table 4: Comparison results for electricity price forecasting

<table>
<thead>
<tr>
<th>Performance Criteria</th>
<th>Naïve</th>
<th>ARIMA</th>
<th>ARIMA-WAVELET</th>
<th>FNN</th>
<th>LQFNM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAPE</td>
<td>7.68%</td>
<td>6.32%</td>
<td>4.78%</td>
<td>4.60%</td>
<td>4.12%</td>
</tr>
<tr>
<td>VPE</td>
<td>0.0043</td>
<td>0.0034</td>
<td>0.0019</td>
<td>0.0018</td>
<td>0.0015</td>
</tr>
</tbody>
</table>

### References:


