Sains Malaysiana 43(11)(2014): 1781–1790

Multivariate Relationship Modeling using Nested Fuzzy Cognitive Map

(Model Hubungan Multivariasi Menggunakan Peta Kognitif Kabur Tersarang)

O. MOTLAGH*, E.I. PAPAGEORGIOU, S.H. TANG & ZAMBERI JAMALUDIN

ABSTRACT

Soft computing is an alternative to hard and classic math models especially when it comes to uncertain and incomplete data. This includes regression and relationship modeling of highly interrelated variables with applications in curve fitting, interpolation, classification, supervised learning, generalization, unsupervised learning and forecast. Fuzzy cognitive map (FCM) is a recurrent neural structure that encompasses all possible connections including relationships among inputs, inputs to outputs and feedbacks. This article examines a new methods for nonlinear multivariate regression using fuzzy cognitive map. The main contribution is the application of nested FCM structure to define edge weights in form of meaningful functions rather than crisp values. There are example cases in this article which serve as a platform to modelling even more complex engineering systems. The obtained results, analysis and comparison with similar techniques are included to show the robustness and accuracy of the developed method in multivariate regression, along with future lines of research.

Keywords: Nested fuzzy cognitive map; neural activation; regression

ABSTRAK

Pengiraan lembut adalah alternatif kepada model matematik klasik dan sukar terutama apabila ia melibatkan data yang tidak menentu dan tidak lengkap. Ini termasuk regresi dan pemodelan hubungan pemboleh ubah yang sangat berkait dengan aplikasi dalam penyesuaian lengkung, interpolasi, pengelasan, pembelajaran yang diselia, generalisasi, pembelajaran tanpa penyeliaan dan ramalan. Peta kognitif kabur (FCM) merupakan struktur neural berulang yang merangkumi semua kemungkinan sambungan termasuk hubungan antara input, input kepada output dan maklum balas. Artikel ini mengkaji kaedah baru untuk regresi multivariasi tak linear menggunakan peta kognitif kabur. Penyumbang utama adalah penggunaan struktur FCM bersarang untuk menentukan kelebihan pemberat dalam bentuk fungsi bermakna dan bukannya nilai-nilai bersih. Terdapat kes-kes contoh dalam artikel ini yang berfungsi sebagai satu platform untuk pemodelan sistem kejuruteraan yang lebih kompleks. Keputusan yang diperoleh, analisis dan perbandingan dengan teknik yang sama disertakan untuk menunjukkan keberkesanan dan ketepatan kaedah yang dibangunkan dalam regresi multivariasi bersama-sama dengan hala tuju untuk penyelidikan yang akan datang.

Kata kunci: pengaktifan neural; peta kognitif kabur tersarang; regresi

INTRODUCTION

NEURAL REGRESSION USING FUZZY COGNITIVE MAP (FCM)

Thorough understanding about a system's behaviours is fundamental to reliable modelling and forecast. The notion of regression or in broader scope relationship modelling, is concerned about discovering rules which govern relationships among system variables. However, in most cases one has to consider all variables concurrently which lies in multi-dimensional or multivariate regression. On the other hand, multivariate regression is barely performed using hard mathematics as level of complexity exponentially increases with number of variables. Alternatively, the state of the art in modelling and forecast is based on utilization of artificial neural networks (ANN) (Darbellay & Slama 2000), genetic algorithm (Pai & Hong 2005), fuzzy and hybrid methods (Abraham & Nath 2001; Bartkiewicz 2000; Chang et al. 2011) and other soft approaches.

Smaller and mainly linear problems could be modelled using single-layer perceptron (Motlagh et al. 2013). Single hidden layer feed-forward networks are also used for modelling and forecasting mainly using standard or enhanced back propagation rule, e.g. using conjugate gradient algorithm (GCA) (Abraham & Nath 2001). However, recurrent models such as Hopfield (Khan & Ondrůšek 2001) and fuzzy cognitive map (FCM) (Kosko 1996; Motlagh et al. 2012a) became more popular as more relationship configurations could be obtained, a task math models barely accomplish. Overall, numerical and neural models provide easier implementation and more information especially about relationships among variables themselves. Another advantage of neural models is based on efficient utilization of principal component analysis (PCA) (Taylor et al. 2006), and exploratory factor analysis (EFA) (Santin 2011), both on the assumption of linearity of observed variables and other variable reduction techniques.

Fuzzy cognitive map (FCM) is a powerful inference mechanism employed in two distinct domains: Control systems (Motlagh et al. 2012) and system modelling (Motlagh et al. 2013). In modelling, FCM is employed to extract regression models or relationships among known, partially known and even unknown factors contributing to a single complex system. FCM is advantageous in implementation of causal relationships within inputs, outputs or a combination of them, a feature ANN does not fully provide. An n-node FCM is a recurrent neural network where each node (so called concept) C_j, $C_j \in \{C_1 \dots C_n\}$, is the output of some or all other nodes C_i, $C_i \in \{C_1 \dots C_n\}$, i≠j, i.e., n interrelated single-layer (n-1)-node perceptron networks. The goal is to train the matrix of edge weights (or FCM events) in such a way to hold the FCM at convergence (1). In other words, if a neutral activation function is employed (i.e. linear function with unit slope where each net weight is directly assigned to output without squashing), a product of the 1×n state vector of nodes $C=\{C_1 \dots C_n\}$ at cycle τ ($C^{(\tau)}$) and the $n \times n$ matrix of weights W, should show the same values for the next cycle ($C^{(\tau+1)}=C^{(\tau)}$). To obtain W as a non-identity matrix, initial weights must start from zero while the perceptron training rule ((2) with learning rate α) applies to all $w_{i,j} \in W$, provided i≠j. The matrix W is called the adjacency model which describes all possible relationships among variables, including input-output relationship as well as relationships among inputs themselves and feedback relationships.

$$C^{(\tau+1)} = C^{(\tau)} = C^{(\tau)} \times W^{(\tau)}. \quad (1)$$

$$w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \alpha\, C_i^{(\tau)} \left(C^{(\tau)} \times W_j^{(\tau)} - C_j^{(\tau)} \right). \quad (2)$$

$$C_j = C \times W_j, \quad w_{jj} = 0. \quad (3)$$

Upon sufficient training, matrix W simply provides an (n-1)-dimensional linear regression model for all variables assigned to FCM nodes, i.e. each variable (node C_j) is a product of other variables (nodes C_i, i=1…n, i≠j) which influence on C_j through respective links' weights $w_{i,j}$ being the j^{th} column of W (W_j) as shown in (3). W is also called the relationship model of all variables (Motlagh et al. 2013a) with wide range of applications especially when it comes to multivariate systems. However, an important issue is that although variables do not need to be linearly independent as opposed to other state space models, perceptron learning is limited to single-trend or single-behaviour systems, i.e. extents of impacts of variables on a specific output must be linearly independent. Therefore, when it comes to non-linear and multi-behaviour systems, utilization of nonlinear rules such as Hebbian rule is more viable.

Nonlinear Hebbian rule facilitates learning of multi-trend behaviours. Of the fundamental ideas in biological learning, Hebb law has become an established learning method in artificial neural networks (NN) when it comes to distinct patterns of inputs, used in classification, memory and retrieval systems. The idea is to strengthen the connection between neurons i and j, if neuron i is near enough to excite neuron j making it more sensitive to such stimuli. Therefore, a learning rate α is multiplied by the values of the input and output neurons to be added to the current weight of the link connecting input to the output. On the other hand, to prevent weights from indefinite growth, a decay factor ϕ is multiplied by the connecting weight and the value of the memorized output to mimic the phenomenon of forgetting in biological brain. Equation 4 (Negnevitsky 2005) shows the concept of Hebbian rule for weight update from iteration τ to τ+1 in a typical feed forward neural network.

$$\Delta w_{ij}^{(\tau)} = \alpha\, C_i^{(\tau)} C_j^{(\tau)} - \varphi\, C_j^{(\tau)}\, w_{ij}^{(\tau)} \;\; and \;\; w_{ij}^{(\tau)} + \Delta w_{ij}^{(\tau)}. \quad (4)$$

$$\Delta w_{ij}^{(\tau)} = \alpha\, \Delta C_i^{(\tau)} \Delta C_j^{(\tau)} \;\; and \;\; w_{ij}^{(\tau+1)} = w_{ij}^{(\tau)} + \Delta w_{ij}^{(\tau)}. \quad (5)$$

The basic rule in (4) needs modifications in recurrent models such as FCM depend on several criteria such as sequence of activation of neurons and selection of active nodes. Kosko proposed the initial model known as differential Hebbian (DH) model (Dickerson & Kosko 1994) in (5). DH was then improved into balanced differential algorithm (BDA) (Vazquez 2002). In BDA, weight update depends on values of selected FCM nodes which are acting at the same time. However, major advancements were related to non-linear Hebbian learning (NHL) (Papageorgiou et al. 2003) and active Hebbian learning (AHL) (Papageorgiou et al. 2004). NHL is based on the premise that FCM graph has to be updated synchronously whereby all neurons (state vector of nodes or concepts) are updated at the same time. In contrast, AHL involves the sequence of active concepts and updates new weights of all concepts as influenced by the active concept at any time. A summary of the approaches to FCM training is given in Motlagh et al. (2013) along with example cases.

NATURAL INFERENCE METHODS

A critical design element in FCM-based regression and generalization is about type of employed neural activation function such as sigmoid-based and hard limiters. As described earlier, FCM inference is an iterative process of aggregation of net weights and then activation. Let τ be the current iteration, C_j be the output concept node, C_i and $i \in \{1 \dots n\}$ be the inputs, w_{ij} and $i \in \{1 \dots n\}$ be respective links' weights and *f* be the employed function (here symmetric sigmoid with steepness λ), then C_j at iteration (τ +1) could be obtained from (6) (Kosko 1996). An advanced activation model is also presented in Motlagh et al. (2012a) as given in (7). The improvement made in the function *f* (7) leads to smoother convergence due to squashing net weights about own nodal values rather than a fixed point (as compared against *f* in (6)).

Also as described, apart from inference, FCM requires weight training to adapt weights of graph edges. There are various supervised and unsupervised training techniques such as genetic-based (Stach et al. 2005) and Hebbian-

based (Papageorgiou & Salmeron 2012), respectively. Likewise, the two fundamental and yet very practical learning models are based on supervised perceptron (2) and unsupervised differential Hebbian rule (5). However, experiments have proven that each type of activation function leads to a different regression model (due to solving for a different adjacency matrix W). In other words, the linear regression model (3) finds different solutions depend on the employed activation function. The problem is addressed as the lack of a unique (natural) activation function that leads to a realistic regression model for each output or a realistic relationship model of all concepts within an entire FCM network.

$$C_j^{(\tau+1)} = f_{(T)} = f\left(\sum_{i=1}^{n} C_i^{(\tau)} \cdot w_{ij}\right),\ j \in \{1 \ldots n\} \quad \text{where}$$

$$f_{(T)} = \frac{1}{1+e^{-\lambda T}}. \tag{6}$$

$$C_j^{(\tau+1)} = f\left(\sum_{i=1}^{n} C_i^{(\tau)}\right) \text{ where } f_{(T)} = \frac{\gamma C_j^{(\tau)} e^{\lambda T}}{\gamma C_j^{(\tau)}\left(e^{\lambda T}-1\right)+1} \text{ and}$$

$$\gamma \in (0, 1). \tag{7}$$

Accordingly, *natural* activation is proposed (Motlagh et al. 2013a) to replace the many choices of activation functions with unique functions which come from the natural behavior of each variable within a given system. The philosophy is to remove activation function from output (a property incorporated into all types of ANN structures) and instead add unique activation functions to each individual graph edge. The edge functions, such as polynomial and trigonometric are therefore to be uniquely obtained regardless of the employed training technique. To introduce the idea, let us first consider a conventional neural structure as shown in Figure 1(a) and (8) where aggregated net weight is squashed and then assigned to the output. By applying a reverse order, activation could be performed prior to aggregation of net weight (Figure 1(b) and (9)) and since there is a one-to-one relationship between the output and a specific input, the respective edge function would be unique, provided the net weight is directly assigned to the output (w_j=1) or in other words, neural activation function in Figure 1(a) to be removed.

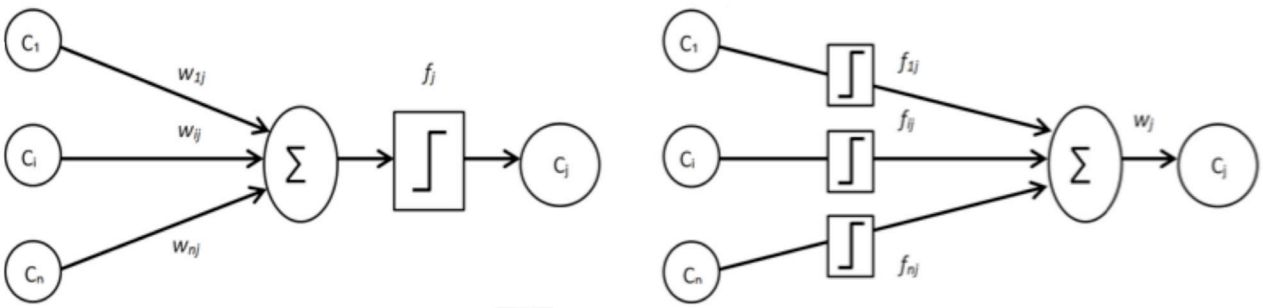
$$C_j = f_j\left(\sum_i^n C_i w_{ij}\right),\ \ j = 1 \ldots n. \tag{8}$$

$$C_j = w_j\left(\sum_i^n f_{ij}(C_i)\right),\ \ j = 1 \ldots n. \tag{9}$$

Let function f_{ij} be a linear polynomial function connecting node C_i to C_j. Accordingly, many classic regression or heuristic search algorithms could be suggested to obtain functions $f_{1j} \ldots f_{nj}$ by solving for polynomial parameters with any desired degree. For example, as one of the fundamental training techniques, (10) represents a parallel perceptron rule with learning rates α, β, σ to tune parameters a, b and c, of a quadratic polynomial connecting C_i to C_j, that is: $C_j = f_{ij}(C_i) = a_{ij} + b_{ij} C_i + c_{ij} C_i^2$. The degree of the polynomial (quadratic, cubic, quartic) is directly proporonal to the desired accuracy while it is inversely related to the computational cost of training. However, error-based training such as perceptron rule is less effective when it comes to higher degree polynomials. This is due to the fact that error comes from different sources associated with polynomial power series which may result into ambiguity.

$$\begin{bmatrix} \Delta a_{ij} \\ \Delta b_{ij} \\ \Delta c_{ij} \end{bmatrix} = -\left(\Delta C_j\right) C_i \begin{bmatrix} \alpha \\ \beta \\ \sigma \end{bmatrix}. \tag{10}$$

In contrast to local search, global techniques such as the genetic algorithm (GA) ensure complete search, and therefore have been traditionally used to fit polynomial curves (Gulsen et al. 1995; Karr et al. 1995). A GA-based formulation of the problem is presented in Figure 2 while (11) and (12) represent the k^{th} GA chromosome ch_k of a population N and the respective cost function. It must be noted that the GA representations are related to the example of quadratic functions where the objective is to tune polynomial parameters A, B, C, for all n^2 possible graph links in an n-node FCM. Accordingly, ch_k includes $n^2 \times 3$



(a) (b)

FIGURE 1. (a) Neural activation vs (b) natural activation

encoded parameters as also shown in Figure 2 to represent edges as optimized quadratic polynomials. Here GA aims to hold FCM at convergence. Accordingly, (12) is used as the cost function, where C_j equal to desired $C_{j(desired)}$ serves as the fitness measure. The algorithm then continues for generations until global solution is sought. GA warrants a global solution with relatively acceptable accuracy but high computational cost which makes it not suitable for real-time applications.

$$ch_k = \{a_{11}, b_{11}, c_{11} \ldots a_{nn}, b_{nn}, c_{nn}\}_k, \quad ch_k \in \{N\}. \tag{11}$$

$$\varphi(ch_k) = C_j - C_{j(desired)} = \sum_i^n f_{ij}(C_i) - C_j, \quad j = 1 \ldots n. \tag{12}$$

$$where, f_{ij}(C_i) = A_{ij} + B_{ij} C_i + C_{ij} C_i^2$$

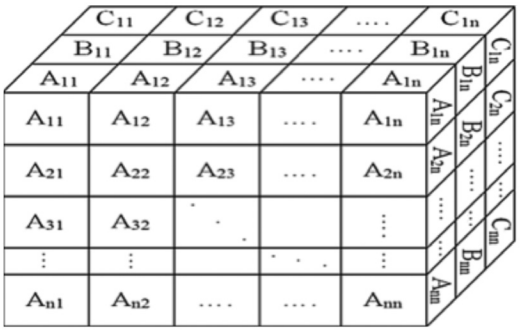


FIGURE 2. Solution for an n-node FCM consists of $3n^2$ encoded values

$$\begin{bmatrix} \sum y \\ \sum xy \\ \sum x^2 y \end{bmatrix} = \begin{bmatrix} d & \sum x & \sum x^2 \\ \sum x & \sum x^2 & \sum x^3 \\ \sum x^2 & \sum x^3 & \sum x^4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}. \tag{13}$$

There are other classic regression models such as spline method (Bianco 2009), Lagrangian, least square and maximum likelihood (Espey 2004), smooth transition regression (STR), threshold regression (TR), and switching regression (SR). Probabilistic such as Bayesian (Cottet & Smith 2003) and grey methods (Yao et al. 2003) have been also employed. A common regression modeling is based on utilization of least square error principle (13). However, it is considerably bulky when it comes to multivariate systems and or higher polynomial degrees (from quadratic (13) to cubic). In the next section, we introduce practical applications of natural inference using a new strategy with minimal computation cost known as nested structure along with the obtained results, related analysis and comparison.

NESTED-FCM STRUCTURE

FCM provides a relatively fast yet reliable linear regression model as given in (3), where each state C_j could be the expressed as a linear function of other variables (entire state vector C). The relationship model called the adjacency model is derived in form of an adjacency matrix W which holds FCM at convergence. It was also discussed that by retaining w_{ij}=0 for i=j, the main diagonal of adjacency matrix could be kept to zero for W to be non-identity. The idea of natural activation suggests application of neutral neural activation at output while FCM links themselves could be tuned in form of functions, such as polynomial. Accordingly, W would not be a matrix of crisp values but a matrix of functions which would fully describe natural relationships among system variables. Accordingly, in contrast to conventional NN-based regression which only suggests linear models, i.e., (3), here, relationship between every pair of nodes could be expressed in the form of non-linear functions of any desired type, such as polynomials. In addition, each node still can be defined as a function of more than one other node, that is in common with classic neural models in the literature. To materialize the notion of natural inference, in this section, nested-FCM structure is described as an effective alternative to other implementation techniques as described in the previous section.

To begin, Figure 3 provides a clear implementation of a 4-node FCM with natural activation (Figure 3(a)) and in form of its equivalent nested-FCM (Figure 3(b)). A link f_{ij} (such as f_{12}) which in this case needs to be obtained in form of a cubic polynomial function, is broken down into four links (or a larger set of nested branches if higher degrees were desired) with scalar weights a, b, c and d. Accordingly, each node (x_i) is replaced with a set of nested nodes which follow the power series of that node 1, x_i, x_i^2 and x_i^3. It must be noted that, for the sake of brevity, the causal relationships between x_1 and x_4, and between x_2 and x_3, are removed in the nested model. It must be also noted that concept nodes (C_1 to C_4) are mapped to system variables (x_1 to x_4) if one attempts to review (1) through (12). Nested-FCM is advantageous as it can be used along with simple conventional weight training techniques, such as the standard perceptron rule (2) and differential Hebbian rule (5) for supervised and unsupervised scenarios. Training is relatively fast and accurate without involving complex (such as least square error) and bulky (such as GA) algorithms.

An important advantage is adaptability to principal component analysis (PCA) (Taylor et al. 2006), and exploratory factor analysis (EFA) (Santin 2011) and other variable reduction techniques, to reduce the size of nests without sacrificing higher degrees. For example, nests can start with a higher degree (six-node nests 1, x_i, x_i^2, x_i^3, $x_i^4$and x_i^5, for a quantic representation) while a simple covariance filter can be used to remove nested nodes which are less influential. Integration of more complex and heterogeneous functions is also allowed. A series of trigonometric (sin x), logarithmic (log x) or exponential (e^x) power series assigned to nested nodes will lead to the respective functions along connecting edges.

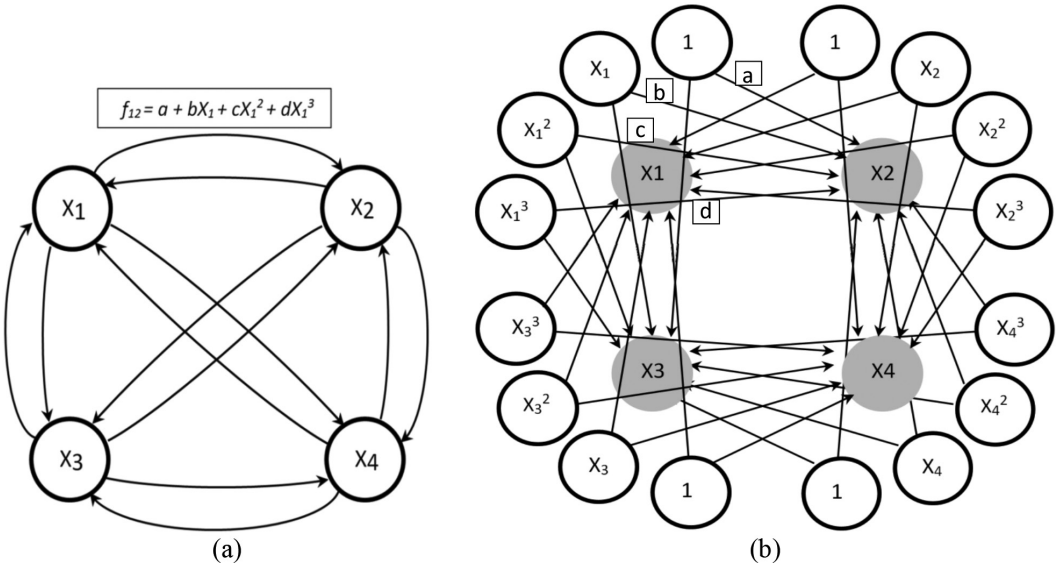


FIGURE 3. (a) FCM structure with natural activation and (b) nested-FCM with natural activation

EXAMPLE CASE OF A KINEMATIC MODEL

A kinematic model of a projectile with negligible size is concerned with initial velocity of 2 *m/s* at -20° with horizon, on planet Mars with no air resistance and gravity of g_m=3.711 m/s^2. The classic (14) through (19) calculate position, velocity, and acceleration of the object as functions of time (t) along the two dimensions X (forward) and Y (downward). This example case is adapted from (Motlagh et al. 2013a) to describe the strategy for deriving quadratic polynomials along with additional analysis for comparison of results with other regression tools.

$$x(t) = v_{x_0}\, t = 1.879\, t. \tag{14}$$

$$v_x(t) = v_{x_0} = 1.879. \tag{15}$$

$$a_x(t) = 0. \tag{16}$$

$$y(t) = v_{y_0}\, t + \tfrac{g_m}{2}\, t^2 = 0.684\, t + 1.856\, t^2. \tag{17}$$

$$v_y(t) = g_m\, t = 3.711\, t. \tag{18}$$

$$a_y(t) = g_m = 3.711. \tag{19}$$

To train respective links of a nested time concept (1, t, t^2), a time series dataset (P) is obtained using the above math model as given in Table 1, where each record contains values of {t, x, v_x, a_x, y, v_y, a_y} which represent sampling time, position, velocity, acceleration along X-axis and position, velocity and acceleration along Y-axis, respectively. It must be noted that there is some noise purposely injected into the data by rounding all values up or down to three decimal points. Upon perceptron training, with minimal computation cost, the respective weights matrix (w_{train}) is obtained which connects the nested concept time (t) to the kinematical variables in Table 2.

Equations 20 to 25 showed the obtained results for the polynomial functions of the kinematical variables

TABLE 1. Time series data of the kinematic parameters

	t (sec)	x (m)	vx (m/s)	ax (m/s2)	y (m)	vy (m/s)	ay (m/s2)
P:	0.500	0.940	1.879	0	0.806	1.856	3.711
	1.000	1.879	1.879	0	2.540	3.711	3.711
	1.500	2.819	1.879	0	5.201	5.567	3.711
	2.000	3.758	1.879	0	8.790	7.422	3.711
	2.500	4.698	1.879	0	13.307	9.278	3.711

TABLE 2. Soft kinematical model using nested FCM technique

w_{train}=

Nested Nodes of t							
1	0.0000	0.0008	1.8793	-0.0000	0.0004	0.0006	3.7108
t	0.0000	1.8781	0.0000	0.0000	0.6841	3.7101	0.0000
t^2	0.0000	0.0003	-0.0000	-0.0003	1.8554	0.0003	-0.0002

TABLE 3. Forecasting kinematical variables at time t = 3 s

Kinematic parameter:	x	v_x	a_x	y	v_y	a_y
Hard model:	5.6370	1.8790	0	18.7560	11.1330	3.7110
Soft model:	5.6354	1.8793	0.0027	18.7513	11.1318	3.7107

which closely match with the hard equations ((1) to (19)). The error is negligible with small computation cost of 618 *sec* on a Quad-Core 2.3 *GHz* machine with *32-b OS* and *2 GB* RAM. The developed model is also capable of forecasting future time series data. To examine forecasting and generalization capability, the nested power series of concept time are removed and replaced back with its original single node representing concept of time (*t*). A seven-node FCM is obtained including the concept node (*t*) connected to all other six nodes via links which carry not crisp or fuzzy weights but meaningful functions ((20) to (25)). The model is used to guess on the future kinematical characteristic of the projectile at time *t*=3 *sec* with negligible RMS error of 0.0024 compared with the precise math model (Table 3).

$$x(t) = 0.0008 + 1.8781\ t + 0.0003 \simeq 1.879\ (t). \quad (20)$$

$$v_x(t) = 1.8793 \simeq 1.879. \quad (21)$$

$$a_x(t) = -0.0003\ t^2\ \simeq 0. \quad (22)$$

$$y(t) = 0.0004 + 0.6841\ t + 1.8554\ t^2 \simeq 0.684\ t + 1.856\ t^2. \quad (23)$$

$$v_y(t) = 0.0006 + 3.7101\ t + 0.0003\ t^2 \simeq 3.711\ t. \quad (24)$$

$$a_y(t) = 3.7108 - 0.0002\ t^2 \simeq 3.711. \quad (25)$$

The results were examined against the original math model as well as polynomial curve fitting in MATLAB. Functions, such as polyfit and polyval, or more conveniently MATLAB's least square curve fitting tool on figure window (under tools menu), can be used to draw up to 10th degree polynomial as well as spline fits. Figure 4(a) shows the position along Y-axis (y) as a function of time (t) using same data points in Table 1 in MATLAB. The resulted equation from our nested-FCM technique (23) and the MATLAB-generated equation (Figure 4) are compared against the benchmark equation from the original kinematic model (17). In terms of forecasting the position at *t*=3 *sec*, MATLAB equation resulted in +0.193% error (18.7922 *m*), while nested-FCM resulted in -0.025% error (18.7513 *m*) in ratio with the exact benchmark position (18.7560 *m*), which shows around 0.168% improvement in absolute forecasting error.

MODELING STOCHASTIC DATA POINTS

Dynamic and irregular system behaviors are often due to stochastic variables, which they themselves often depend on many other variables. Electricity consumption (Thatcher 2007) at individual households or a power network at broader scope is a good example of random variables. Metrological variables, temperature, precipitation, wind speed, wind direction, humidity, pressure and solar radiation, are only one type of variables influencing on level of electricity demand, while on the other hand, there are tens of socio-cultural, geographical and demographic factors, e.g. gender, age and income, which concurrently influence on electricity demand.

There is abundance of research on modeling electricity demand mainly as a baseline for forecasting future data, from half hour lead time till one day ahead, using various regression techniques (Bianco et al. 2009), as well as neural regression (Abraham & Nath 2001). While in most of the literature, focus goes merely on regression between electricity demand and influencing multivariate, e.g., temperature (Thatcher 2007), there are other attempts on modeling demand as a function of time. The stochastic nature of demand as a function of time has been modelled using time series analysis, such as autoregressive moving average (ARMA) (Pappas et al. 2010) autoregressive

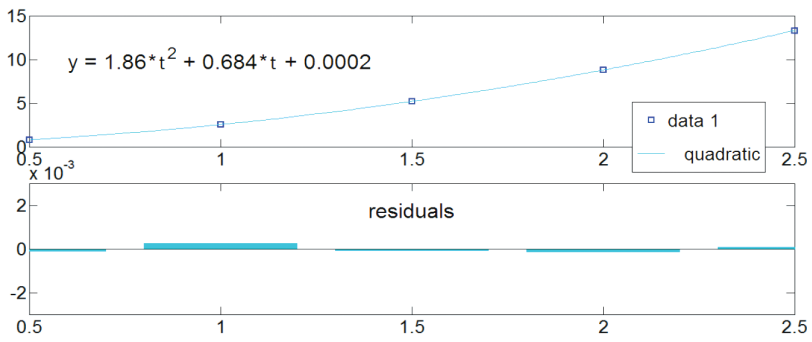


FIGURE 4. MATLAB curve fitting tool applied to the kinematic model

integrated moving average (ARIMA) (Erdogdu 2007), classic Kalman filter (Connor 1996) and cyclic time horizon such as daily, weekly (Taylor et al. 2006) and monthly (Abraham & Nath 2001) forecasting.

On an attempt using nested technique of this article, we applied a perceptron rule for regression between half-hourly demand levels and the time. The dataset is obtained from a typical summer day in eastern Australia which is publically available on Australian energy market operator (AEMO) website (Historical 2013). An interesting capability of the developed technique, as showed through experiments, is the fact that dataset could be extremely heterogeneous which by itself signifies robustness of the developed nested-FCM methodology. In this experiment, half-hourly observations of demand levels (scaled down within (0, 1) times 10^4 MW) and time dimension (scaled down to 0, 0.47) equivalent from (00:00 to 23:30 PM), are two heterogeneous variables both in range and type. Yet, as shown in Figure 5, the nested technique is able to draw a valid regression model, e.g. a quartic polynomial in the given experiment. MATLAB curve fitting tool is also applied to the same data which qualitatively compares with the obtained result from the nested-FCM, however with slightly less error. Both models show similar functions for relating 48 data points of half-hourly demand levels in one day. However, the nested technique has the advantage for switching from polynomial mode to other functions with minor computational effort and adjustment of training parameters.

APPLICATION IN MULTIVARIATE REGRESSION

There is limited literature about multivariate regression. This section elaborates the application of the introduced nested-FCM technique in accurate yet robust data interpolation. A random dataset of imaginary points in a 5-dimentional space (Points) is generated as given in Table 4. It must be noted here that the data is assumed to be certain and the interest is in obtaining exact curve rather than approximated fit. There are five state variables A, B, C, D and E, with the given dataset. The goal is to model each variable as a function of the others, e.g., C= F(A,B,D,E). It must be also noted that we are interested in finding a single polynomial function to attain such relationship which is more challenging compared to interpolation using piecewise splines. The experiment is repeated twice using 4-node and 5-node nests, to obtain cubic and quartic polynomials, respectively.

The results from the two experiments on the same machine as used for the other experiments are given in Table 4. The result from the first experiment with 4-node nests (at left) showed a set of cubic functions which rebuild the original dataset almost precisely (Points →(or mapped to) Points_Learnt), i.e. error is closed to zero upon 1 M iterations in 24 min. A mask is put on the matrix of weights which serves as a perceptron guide to accumulate all constant values at a single node (i.e. the first node of each nest). For example, without the mask, variable C initially would be obtained in the form of (26) as a function of other variables as well as its own first degree (in brackets). However, when the mask is applied, the term in brackets is automatically moved to the left side of the equation whereby a magnifying coefficient will be created to be multiplied by all terms at the right side of the equation (27). The masking technique is particularly useful in order to maintain the range of the polynomial parameters without need for normalizing net weights which is essential in classic activation techniques. Table 4 also shows the results from the quartic nests (at right). However, prior to explanation about the results, a modification technique is described as made to the learning rule in (28) and (29).

$$C = 0.0366 + 0.0451\,A - 0.0071\,A^2 - 0.0018\,A^3 + 0.0407\,B - 0.0124\,B^2 + 0.0005\,B^3 + (0.9727\,C + 0.0457\,D - 0.0138\,D^2 + 0.0006\,D^3 + 0.2407\,E + 0.0354\,E^2 - 0.0061\,E^3 \quad (26)$$

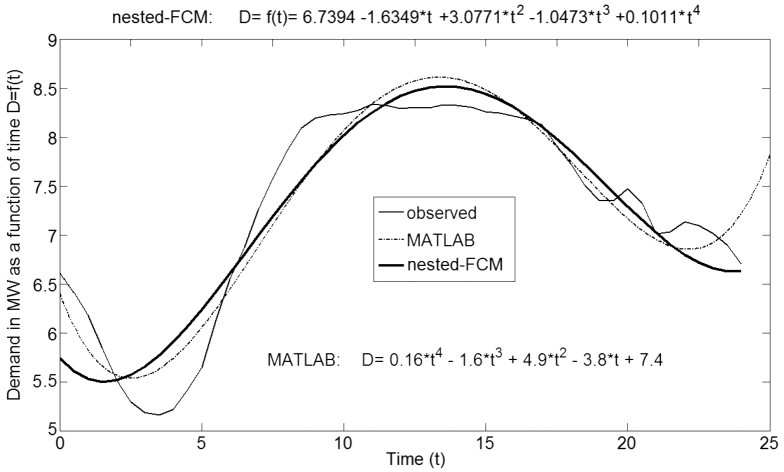


FIGURE 5. Nested-FCM and MATLAB curves (set to 4th degree polynomial) for modeling electricity demand as a function of time

TABLE 4. Experimental work for obtaining relationship model of the five variables

Experiment 1: using Cubic nested-FCM

```
Points =
           A         B         C         D         E
     -7.1623    5.8441   -9.2858    1.0747   -1.1555
     -1.5648    9.1898    6.9826    5.1548    2.1096
      8.3147   -3.1148    8.6799    4.8626   -6.5763
     -0.9353    7.0745   -8.8637   -7.7042    5.5125
     -8.5050   -0.6420   -7.1413   -5.2787    1.0554
      3.2668   -0.8304   -6.5716   -4.2184    1.0938
      4.0730    6.1221    2.5169   -6.5449    4.6129
      8.3790    6.4953   -9.4097   -3.5259    5.4725
      5.2014   -6.1913   -0.5534    6.0222    8.0169
     -3.8021   -9.4865    3.5689   -3.0073   -7.2367

w_Trained =                       C
                                  =
1     0.1175         0         0         0         0
A     0.9816    0.0523    0.0451   -0.1847   -0.1078
A2         0    0.0036   -0.0071   -0.0027    0.0173
A3         0    0.0003   -0.0018    0.0036    0.0045

1          0    0.1172         0         0         0
B     0.0796    0.7247    0.0407    0.0979   -0.1736
B2    0.0184         0   -0.0124    0.0083   -0.0027
B3   -0.0050         0    0.0005   -0.0064    0.0078

1          0         0    0.0366         0         0
C    -0.0115    0.4243    0.9717    0.2456   -0.0842
C2    0.0472    0.0098         0    0.0467   -0.0882
C3    0.0053   -0.0062         0    0.0013   -0.0067

1          0         0         0    0.0376         0
D     0.0169    0.1786    0.0457    0.8236   -0.0927
D2   -0.0678    0.0229   -0.0138         0    0.1285
D3   -0.0073   -0.0036    0.0006         0    0.0095

1          0         0         0         0    0.0618
E     0.1300    0.1797    0.2407   -0.1347    0.2610
E2   -0.0373   -0.0578    0.0354   -0.0769         0
E3    0.0075   -0.0017   -0.0061    0.0126         0

Points_Learnt =
           A         B         C         D         E
     -7.1623    5.8441   -9.2858    1.0747   -1.1555
     -1.5648    9.1898    6.9826    5.1548    2.1096
      8.3147   -3.1148    8.6799    4.8626   -6.5763
     -0.9353    7.0745   -8.8637   -7.7042    5.5125
     -8.5050   -0.6420   -7.1413   -5.2787    1.0554
      3.2668   -0.8304   -6.5716   -4.2184    1.0938
      4.0730    6.1221    2.5169   -6.5449    4.6129
      8.3790    6.4953   -9.4097   -3.5259    5.4725
      5.2014   -6.1913   -0.5534    6.0222    8.0169
     -3.8021   -9.4865    3.5689   -3.0073   -7.2367

RMSE = 5.0437e-005
Run_Time = 1463.5 sec (for 1000000 iterations)
```

```
mask=[  1    0    0    0    0     %  Nested A
        1    1    1    1    1
        0    1    1    1    1
        0    1    1    1    1
        0    1    0    0    0     %  Nested B
        1    1    1    1    1
        1    0    1    1    1
        1    0    1    1    1
        0    0    1    0    0     %  Nested C
        1    1    1    1    1
        1    1    0    1    1
        1    1    0    1    1
        0    0    0    1    0     %  Nested D
        1    1    1    1    1
        1    1    1    0    1
        1    1    1    0    1
        0    0    0    0    1     %  Nested E
        1    1    1    1    1
        1    1    1    1    0
        1    1    1    1    0  ]  20x5
```

Experiment 2: using Quartic nested-FCM

```
Points =
           A         B         C         D         E

     -7.1623    5.8441   -9.2858    1.0747   -1.1555
     -1.5648    9.1898    6.9826    5.1548    2.1096
      8.3147   -3.1148    8.6799    4.8626   -6.5763
     -0.9353    7.0745   -8.8637   -7.7042    5.5125
     -8.5050   -0.6420   -7.1413   -5.2787    1.0554
      3.2668   -0.8304   -6.5716   -4.2184    1.0938
      4.0730    6.1221    2.5169   -6.5449    4.6129
      8.3790    6.4953   -9.4097   -3.5259    5.4725
      5.2014   -6.1913   -0.5534    6.0222    8.0169
     -3.8021   -9.4865    3.5689   -3.0073   -7.2367


w_Trained =
                                  C
                                  =
1     0.1981         0         0         0         0
A     0.8206    0.0080    0.1550    0.0320    0.0715
A2         0    0.1491   -0.1010    0.0173    0.0470
A3         0    0.0054    0.0002   -0.0045    0.0035
A4         0   -0.0012    0.0016   -0.0011   -0.0001

1          0    0.0606         0         0         0
B     0.1050    0.4055    0.3016    0.0087    0.0208
B2   -0.1314         0    0.1328    0.1064   -0.0691
B3    0.0474         0   -0.0231   -0.0852    0.0059
B4   -0.0012         0   -0.0010    0.0029    0.0007

1          0         0   -0.0249         0         0
C     0.0433    0.1909    0.8665    0.1434    0.0623
C2    0.1149   -0.1124         0   -0.0665   -0.0455
C3   -0.0297   -0.0194         0    0.0595   -0.0110
C4   -0.0053   -0.0005         0    0.0089   -0.0008

1          0         0         0    0.0084         0
D     0.1011    0.0357    0.2077    0.1750    0.0910
D2   -0.1200    0.0291    0.0865         0    0.0713
D3    0.0034    0.0600    0.0359         0    0.0101
D4    0.0000    0.0097    0.0046         0    0.0018

1          0         0         0         0    0.1428
E     0.0095    0.0534    0.0570    0.1594    0.1614
E2    0.1723    0.2938    0.1721    0.0942         0
E3   -0.0663   -0.0221    0.0039    0.1329         0
E4    0.0110   -0.0103   -0.0093   -0.0238         0


Points_Learned =

           A         B         C         D         E

     -7.1623    5.8441   -9.2858    1.0747   -1.1555
     -1.5648    9.1898    6.9826    5.1548    2.1096
      8.3147   -3.1148    8.6799    4.8626   -6.5763
     -0.9353    7.0745   -8.8637   -7.7042    5.5125
     -8.5050   -0.6420   -7.1413   -5.2787    1.0554
      3.2668   -0.8304   -6.5716   -4.2184    1.0938
      4.0730    6.1221    2.5169   -6.5449    4.6129
      8.3790    6.4953   -9.4097   -3.5259    5.4725
      5.2014   -6.1913   -0.5534    6.0222    8.0169
     -3.8021   -9.4865    3.5689   -3.0073   -7.2367


RMSE =
      1.5270e-009

Run_Time =

      189.2 sec (for 100000 iterations)
```

```
mask=[  similar mask been applied
        with modified dimensions: 25x5   ]
```

$$C = F(A, B, D, E) = 1.2933 + 1.5936\,A - 0.2509\,A^2 - 0.0636\,A^3 + 1.4382\,B - 0.4382\,B^2 + 0.0177\,B^3 + 1.6148\,D - 0.4876\,D^2 + 0.0212\,D^3 + 8.5053\,E + 1.2509\,E^2 - 0.2155\,E^3 \quad (27)$$

Let us consider a node x_i being the forth node from a nest A with an exponential power, $x_i=A^3$. The classic perceptron rule as employed in the first experiment suggests that weight of link l_{ij} at training iteration (k+1) is corrected according to level of error e_j at output y_j at the previous training iteration (k). Since the perceptron rate α is constant, it has to be limited to extremely small values to avoid chaotic behavior of nodes with higher degrees. Adversely, very small learning rate leads to prolonged training with considerably high computation cost.

$$l_{ij}^{(k+1)} = l_{ij}^{(k)} + \alpha . e_j^{(k)} . x_i^{(k)}, \; and \; e_j^{(k)} = y_j^{(k)} - x_j^{(k)} \quad (28)$$

$$\alpha^{(k)} = \frac{\alpha_0}{k10^{d_i}}, \; and \; d_i : degree\ of\ node\ x_i \quad (29)$$

$$C = -0.1865 + 1.1610\,A - 0.7566\,A^2 + 0.0015\,A^3 + 0.0120\,A^4 + 2.2592\,B + 0.9948\,B^2 + 0.1730\,B^3 - 0.0075\,B^4 + 1.5558\,D + 0.6479\,D^2 + 0.2689\,D^3 + 0.0345\,D^4 + 0.4270\,E + 1.2891\,E^2 + 0.0292\,E^3 - 0.0697\,E^4 \quad (30)$$

A modification is made through the application of variable learning rate, i.e. a function of the source node's degree and the elapsed time, instead of constant rate (29). Accordingly, agile training is made possible with low error and computation cost and without the risk of FCM chaotic behavior. The quartic functions, as compared against the cubic functions, were obtained nearly eight times faster using the above technique, which were also twice more accurate as indicated in Table 4. Thanks to the high precision, the functions can now be used to generate new points. Overall, it must also be noted that the length of the dataset, e.g. 10 records in this case and the number of variables to correlate, both exponentially increase the cost of computation if the same precision level is desired.

Conclusion

Neural regression is commonly used to draw relationship models of linear systems without involving complexity of mathematics. In this article, we discussed the application of nested FCM to materialize the notion of natural inference, whereby relationship model of nonlinear and multivariate systems could be expressed in the form of a set of functions, such as polynomial. Each state variable could be modeled as a function of one or more other variables. The developed strategy was examined in three situations, namely a kinematic problem, a stochastic example, and a multivariate closed loop system. The obtained results were compared against that of math equivalents and MATLAB standard curve fitting tool, which showed satisfactory accuracy and computational cost. This research is in progress at both aspects of theatrical methods and validation through practical applications. For example, the model appeared to be a viable alternative to stepwise and classic regression in modeling stochastic electricity demand. This signifies possible application in learning and forecasting of electricity demand being one of the challenging engineering problems. Another ambitious research is on defining universal functions which automatically adapt to the nature of problem. At last, the application of principle component analysis (PCA) is within our future scope to remove useless nodes for reduced nested-FCM size.

Acknowledgements

This research is supported by the Ministry of Education (MOE), Malaysia, under the fundamental research grant scheme (FRGS/2013/FKP/ICT02/02/3/F00158) at Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia, in collaboration with Universiti Putra Malaysia (UPM), Selangor, Malaysia.

References

Abraham, A. & Nath, B. 2001. A neuro-fuzzy approach for modelling electricity demand in Victoria. *Applied Soft Computing* 1(2): 127-138.

Bartkiewicz, W. 2000. Neuro-fuzzy approaches to short-term electrical load forecasting. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*. 6: 229-234.

Bianco, V., Manca, O. & Nardini, S. 2009. Electricity consumption forecasting in Italy using linear regression models. *Energy* 34(9): 1413-1421.

Chang, P-C., Fan, C-Y. & Lin, J-J. 2011. Monthly electricity demand forecasting based on a weighted evolving fuzzy neural network approach. *International Journal of Electrical Power and Energy Systems* 33(1): 17-27.

Connor, J.T. 1996. A robust neural network filter for electricity demand prediction. *Journal of Forecasting* 15(6): 437-458.

Cottet, R. & Smith, M. 2003. Bayesian modeling and forecasting of intraday electricity load. *Journal of the American Statistical Association* 98(464): 839-849.

Darbellay, G.A. & Slama, M. 2000. Forecasting the short-term demand for electricity: Do neural networks stand a better chance. *International Journal of Forecasting* 16(1): 71-83.

Dickerson, J.A. & Kosko, B. 1994. Virtual worlds as fuzzy cognitive maps. *Presence* 3(2): 173-189.

Erdogdu, E. 2007. Electricity demand analysis using cointegration and ARIMA modelling: A case study of Turkey. *Energy Policy* 35(2): 1129-1146.

Espey, J.A. & Espey, M. 2004. Turning on the lights: A meta-analysis of residential electricity demand elasticities. *Journal of Agricultural and Applied Economics* 36(1): 65-81.

Gulsen, M., Smith, A.E. & Tate, D.M. 1995. A genetic algorithm approach to curve fitting. *International Journal of Production Research* 33(7): 1911-1923.

Historical demand data. 2013. Historical demand data from 2004 to date. http://www.nemweb.com.au/REPORTS/Archive/HistDemand/. Accessed on 10 November 2013.

Karr, C.L., Weck, B., Massart, D.L. & Vankeerberghen, P. 1995. Least median squares curve fitting using a genetic algorithm. *Engineering Applications of Artificial Intelligence* 8(2): 177-189.

Khan, M.R. & Ondrůšek, Č. 2001. The Hopfield model for short-term load prediction. *2nd Spring International Power Engineering Conference*, UVEE, FEI, Brno University of Technology, Czech Republic. pp. 81-85.

Kosko, B. 1996. *Fuzzy Engineering*. Upper Saddle River, NJ: Prentice-Hall Inc.

Motlagh, O., Jamaludin, Z., Tang, S.H. & Khaksar, W. 2013. An agile FCM for real-time modeling of dynamic and real-life systems, evolving systems: Special issue on temporal aspects in fuzzy cognitive maps. DOI: 10.1007/s12530-013-9077-6.

Motlagh, O., Tang, S.H., Maslan, M.N., Jafar, F.A. & Maslita, A.A. 2013a. A novel graph computation technique for multi-dimensional curve fitting. *Connection Science* 25(2-3): 129-138.

Motlagh, O., Tang, S.H., Ismail, N. & Ramli, A.R. 2012. An expert fuzzy cognitive map for reactive navigation of mobile robots. *Fuzzy Sets and Systems* 201: 105-121.

Motlagh, O., Tang, S.H., Khaksar, W. & Ismail, N. 2012a. An alternative approach to FCM activation for modeling dynamic systems. *Applied Artificial Intelligence* 26(8): 733-742.

Negnevitsky, M. 2005. *Artificial Intelligence: A Guide to Intelligent Systems*. England: Pearson Education Ltd.

Pai, P-F. & Hong, W-C. 2005. Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms. *Electric Power Systems Research* 74(3): 417-425.

Papageorgiou, E.I. & Salmeron, J.L. 2012. Learning fuzzy grey cognitive maps using non-linear Hebbian. *International Journal of Approximate Reasoning* 53(1): 54-65.

Pappas, S.Sp., Ekonomou, L., Karampelas, P., Karamousantas, D.C., Katsikas, S.K., Chatzarakis, G.E. & Skafidas, P.D. 2010. Electricity demand load forecasting of the Hellenic power system using an ARMA model. *Electric Power Systems Research* 80(3): 256-264.

Papageorgiou, E.I., Stylios, C.D. & Groumpos, P.P. 2004. Active Hebbian learning algorithm to train fuzzy cognitive maps. *Int. Journal of Approximate Reasoning* 37(3): 219-247.

Papageorgiou, E.I., Stylios, C.D. & Groumpos, P.P. 2003. Fuzzy cognitive map learning based on nonlinear Hebbian rule. *Australian Conf. on Artificial Intelligence*. pp. 256-268.

Santin, O.G. 2011. Behavioural patterns and user profiles related to energy consumption for heating. *Energy and Buildings* 43(10): 2662-2672.

Stach, W., Kurgan, L., Pedrycz, W. & Reformat, M. 2005. Genetic learning of fuzzy cognitive maps. *Fuzzy Sets and Systems* 153: 371-401.

Taylor, J.W., de Menezes, L.M. & McSharry, P.E. 2006. A comparison of univariate methods for forecasting electricity demand up to a day ahead. *International Journal of Forecasting* 22(1): 1-16.

Thatcher, M.J. 2007. Modelling changes to electricity demand load duration curves as a consequence of predicted climate change for Australia. *Energy* 32: 1647-1659.

Vazquez, A. 2002. A balanced differential learning algorithm in fuzzy cognitive maps, Technical Report, Departament de Lenguatges I Sistemes Informatics, Universitat Politecnica de Catalunya (UPC).

Yao, A.W.L., Chi, S.C. & Chen, J.H. 2003. An improved Grey-based approach for electricity demand forecasting. *Electric Power Systems Research* 67(3): 217-224.

O. Motlagh* & Zamberi Jamaludin
Robotics and Automation
Universiti Teknikal Malaysia Melaka
76100 Durian Tunggal, Melaka
Malaysia

E.I. Papageorgiou
Informatics and Computer Tech.
Technological Education Institute (TEI)
Lamia, 654 04 Kavala
Greece

S.H. Tang
Mechanical and Manufacturing Department
Universiti Putra Malaysia
43400 Serdang, Selangor
Malaysia

*Corresponding author; email: omid@utem.edu.my

Received: 7 June 2013
Accepted: 11 March 2014