

Nonlinear dynamic system identification using recurrent neural network with multi-segment piecewise-linear connection weight

Arit Thammano · Phongthep Ruxpakawong

Received: 15 July 2009 / Accepted: 19 April 2010 / Published online: 20 May 2010
© Springer-Verlag 2010

Abstract This paper introduces a new concept of the connection weight to the standard recurrent neural networks—Elman and Jordan networks. The architecture of the modified networks is the same as that of the original recurrent neural networks. However, unlike the original recurrent neural networks whose connection weight is a single real number, in the modified networks the weight of each connection is multi-valued, depending on the value of the input data involved. The backpropagation learning algorithm is also modified to suit the proposed concept. The modified networks have been benchmarked against the feedforward neural network and the original recurrent neural networks. The experimental results on twelve benchmark problems show that the modified networks are clearly superior to the other three methods.

Keywords Recurrent neural network · Nonlinear dynamic system · Identification

1 Introduction

Neural networks have long been applied successfully to the identification of complex nonlinear system. Even though neural networks have satisfactorily solved most of these problems, they generally require a large number of input neurons and necessitate a long computation time. From a structure

point of view, there are two main types of neural networks: feedforward and recurrent neural networks. A connection in a feedforward neural network (FNN) is unidirectional, which only allows the information to flow from input neurons to output neurons. In a recurrent neural network (RNN), however, there are both the feedforward and feedback connections, which allow the information to flow from input neurons to output neurons and vice versa.

Typically, the feedforward neural networks trained with backpropagation learning algorithm have been widely used for nonlinear system identification. However, since the feedforward neural networks lack the ability to retain information about the infinite past [1], their performance in identifying nonlinear dynamical system is not as good as those of the recurrent neural networks. Many previous research works, such as [2–4], show that RNNs are more suitable than FNNs for representing nonlinear dynamical system. Pham and Karaboga [5] further improve the performances of Elman and Jordan recurrent neural networks for dynamic system identification by training such networks using genetic algorithms (GAs). The results show that GAs are successful in training the Elman recurrent neural network but not the Jordan network. Yu et al. [6] state that the evolutionary algorithms (EAs), which are designed specifically for the stationary optimization problems, may no longer be efficient in dynamic environments. They, therefore, apply immigrants schemes to GA to maintain the diversity of the population throughout the run.

This paper introduces a new concept of the connection weight to the original recurrent neural networks. Typically, each connection weight of the recurrent neural network is a single real number. However, in this paper the weight of each feedforward connection is multi-valued, depending on the value of the input data involved. The performance of the proposed approach is evaluated against the feedforward

A. Thammano (✉) · P. Ruxpakawong
Computational Intelligence Laboratory,
Faculty of Information Technology,
King Mongkut's Institute of Technology Ladkrabang,
Bangkok 10520, Thailand
e-mail: arit@it.kmitl.ac.th

P. Ruxpakawong
e-mail: thep@psru.ac.th

neural network and the two most common recurrent neural networks: Elman and Jordan networks.

This paper is divided into five sections. Following this introduction, Sect. 2 briefly explains the background of recurrent neural networks. Section 3 presents the proposed approach and its learning algorithm. The experimental results are given in Sect. 4. Finally, Sect. 5 is the conclusion.

2 Recurrent neural network

Recurrent neural network (RNN) is a kind of neural network with feedback connections. The existence of these feedback connections gives RNN a big advantage over FNN when it comes to identifying and controlling dynamical systems. Two of the well-known RNNs are Elman and Jordan networks.

Elman network [7] is a neural network with three layers of neurons. The first layer is the input layer which consists of two groups of neurons: the internal input neurons and the external input neurons. The internal input neurons, also called the context units, take in the outputs of neurons in the hidden layer as shown in Fig. 1a. Both the outputs of the context units and the external input units are fed to the neurons in the hidden layer, and then the outputs of the hidden neurons are fed forward to activate the output neurons.

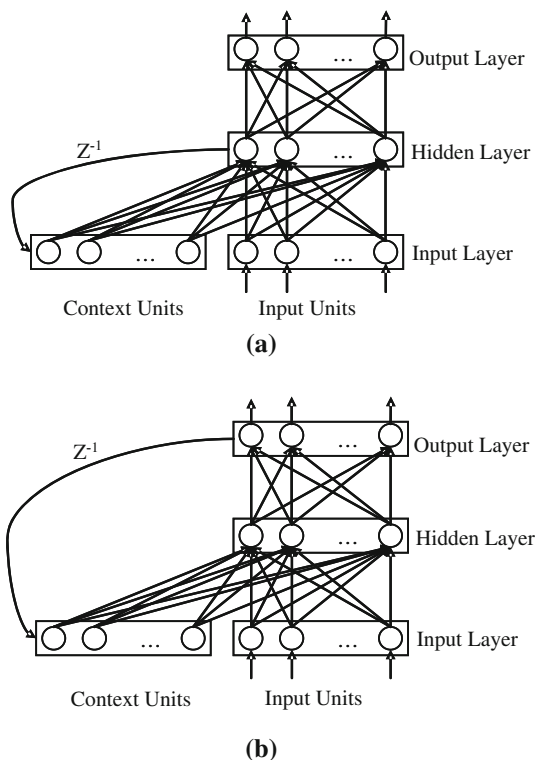


Fig. 1 Well-known recurrent neural networks. **a** Elman network, **b** Jordan network

$$O_r^t = O_j^{t-1} \quad (1)$$

$$\text{net}_j^t = \sum_{i=1}^N x_i^t w_{ij} + \sum_{r=1}^R O_r^t w_{rj} \quad (2)$$

$$O_j^t = f(\text{net}_j^t) \quad (3)$$

$$\text{net}_k^t = \sum_{j=1}^M O_j^t w_{jk} \quad (4)$$

$$O_k^t = f(\text{net}_k^t) \quad (5)$$

where O_r^t is the output of the context node r at time t . O_j^t and O_j^{t-1} are the outputs of the hidden node j at time t and $t - 1$, respectively. O_k^t is the output of the output node k at time t . w_{ij} is the weight of the connection that connects the external input node i to the hidden node j . w_{rj} is the weight of the connection that connects the context node r to the hidden node j . w_{jk} is the weight of the connection that connects node j in the hidden layer to node k in the output layer. N is the total number of external input nodes. R is the total number of context nodes. M is the total number of nodes in the hidden layer. $f(\bullet)$ is the activation function.

Jordan network [8] also has three layers. The main difference from Elman network is that the context units take inputs from the outputs of neurons in the output layer as shown in Fig. 1b.

$$O_r^t = O_k^{t-1} \quad (6)$$

Basically, both Elman and Jordan networks can be trained in the same way as feedforward neural networks by means of the backpropagation learning algorithm. During the training phase, the error is propagated backward from the output layer to adjust the weights of all feedforward connections. However, the weights of the feedback connections must be kept constant, and are not subject to any adjustment.

Cheng et al. [9] suggested that Elman network inherently has only integral (I) dynamic properties, which will result in slow dynamic response and poor convergence speed. Various models that improve on the original Elman and Jordan networks have been proposed. Gao et al. [10] improve the dynamic characteristics and convergence speed of the original Elman network by adding feedforward connections between the context nodes and the output nodes (Fig. 2). This modification provides the modified Elman neural network with the proportional (P) and integral (I) properties, which make this modified Elman network to have a better dynamic response than the original Elman network. Chen and Soo [11] proposed a recurrent neural network which has two kinds of context units as shown in Fig. 3. The first group of context units stores the feedback from the outputs of the hidden layer while the second group stores the feedback of the output layer. Ji and Qi [12] proposed Proportional–Integral–Derivative (PID) Elman neural network (Fig. 4).

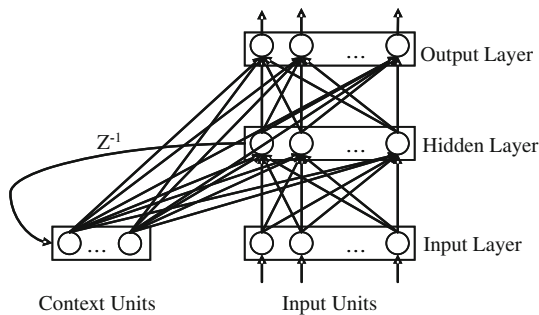


Fig. 2 Gao, Gao, and Ovaska's recurrent neural network

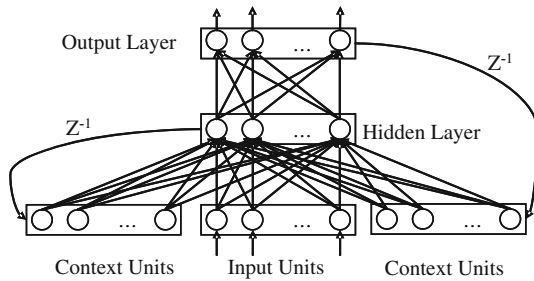


Fig. 3 Chen and Soo's recurrent neural network

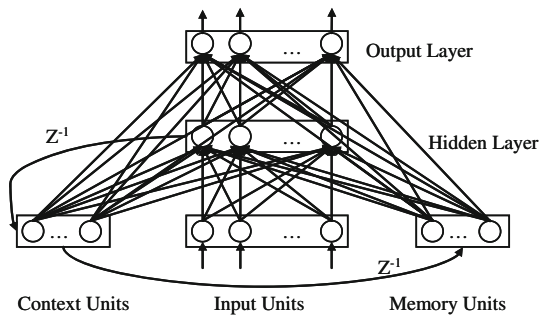


Fig. 4 Ji and Qi's recurrent neural network

In comparison to PI Elman network proposed by Gao, Gao, and Ovaska, this PID Elman network has an additional group of context units called memory layer. The outputs of the context units are fed to the memory units through the unit delays. Based on the experimental results, the proposed PID Elman network performs better than PI Elman network both in terms of the identification accuracy and the convergence speed.

3 The proposed approach

A new concept of the connection weight proposed in this paper is applied to both the original Elman and Jordan recurrent neural networks. As mentioned in Sect. 2, the recurrent neural network consists of three layers. The first layer is the input layer which consists of two groups of neurons: the internal input neurons and the external input neurons. The number of internal input neurons is equal to the number of neurons in the hidden layer in the case of Elman network and equal

to the number of neurons in the output layer in the case of Jordan network. The number of external input neurons is equal to the number of feature components of the input data. The second layer is the hidden layer. The nodes in the hidden layer are fully connected to the nodes in the input layer and the output layer. The third layer is the output layer.

Once the network receives the input and its associated target output (X, Y), the training process will begin. In this paper, the input vector is denoted by $X = (x_1^t, x_2^t, \dots, x_N^t, \dots, x_N^t)$, where N is the number of features in X ; the target output is denoted by $Y = (y_1^t, y_2^t, \dots, y_k^t, \dots, y_C^t)$, where C is the number of output nodes. The detail procedure of the training process is as follows:

- (1) Normalize the input data to a value between $[0, 1]$.
- (2) Define the number of nodes in the hidden layer (M) and the maximum number of segments into which the input data range can be divided (MAXSEGMENTS).
- (3) Initialize the weights of all feedforward connections— w_{ij} , w_{rj} , and w_{jk} . In this research, the weight of each feedforward connection is multi-valued, depending on the value of the input data involved. Therefore, the values of the weights w_{ij} , w_{rj} , and w_{jk} depend on the values of x_i^t , O_r^t , and O_j^t , respectively. To initialize the weight of each feedforward connection (w_{ij} , w_{rj} , and w_{jk}), randomly pick 2 numbers between -1 and 1 and let them be the weights at the opposite ends of the input data range (0 and 1).

Considering the ij connection, for example, the weight initialization is performed at the following two points: $w_{ij}(x_i^t = 0)$ and $w_{ij}(x_i^t = 1)$. For the specific ij connection in Fig. 5a, $w_{ij}(x_i^t = 0) = 0.6$ and $w_{ij}(x_i^t = 1) = -0.7$. In Fig. 5b, 0.35 and 0.85 are randomly picked and assigned to be the weights at the points where $O_r^t = 0$ and $O_r^t = 1$, respectively. In Fig. 5c, -0.25 and 0.75 are randomly picked and assigned to be the weights at the points where $O_j^t = 0$ and $O_j^t = 1$, respectively. It is also important to note that at this initialization step all the connection weights consist of only one segment.

- (4) Make a copy of the output from the last time step to form the group of context units. For Elman network, the group of context units is a copy of the hidden layer's previous activations. For Jordan network, the group of context units is a copy of the output layer. Therefore, the number of context units of Elman network is equal to the number of nodes in the hidden layer. For Jordan network, the number of context units is equal to the number of nodes in the output layer.

$$O_r^t = \begin{cases} O_j^{t-1} & \text{for Elman network} \\ O_k^{t-1} & \text{for Jordan network} \end{cases} \quad (7)$$

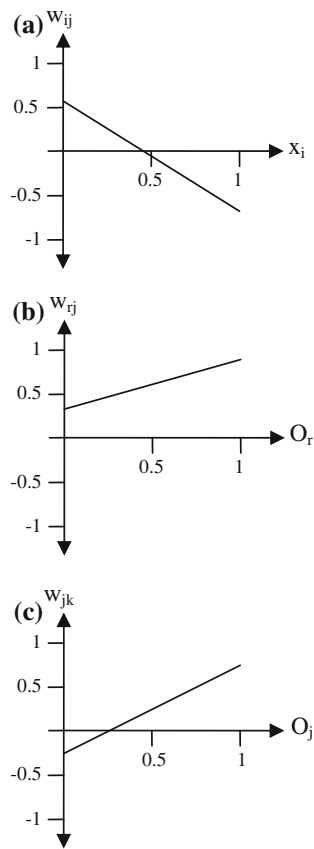


Fig. 5 Example of the weight initialization

where $r = 1, 2, 3, \dots, R$. R is the total number of context nodes. $j = 1, 2, 3, \dots, M$. M is the total number of nodes in the hidden layer. $k = 1, 2, 3, \dots, C$. C is the total number of nodes in the output layer.

- (5) Calculate the output of each hidden node (O_j^t) as follows:

$$O_j^t = f(\text{net}_j^t) \quad (8)$$

$$\text{net}_j^t = \sum_{i=1}^N x_i^t w_{ij}(x_i^t) + \sum_{r=1}^R O_r^t w_{rj}(O_r^t) \quad (9)$$

$$w_{ij}(x_i^t) = w_{ij}(L_s) + \frac{x_i^t - L_s}{U_s - L_s} \times (w_{ij}(U_s) - w_{ij}(L_s)) \quad (10)$$

$$w_{rj}(O_r^t) = w_{rj}(L_s) + \frac{O_r^t - L_s}{U_s - L_s} \times (w_{rj}(U_s) - w_{rj}(L_s)) \quad (11)$$

where $f(\bullet)$ is the activation function. In this research, the sigmoid function is used as the activation function in both the hidden layer and the output layer. N is the total number of external input nodes. R is the total number of context nodes. s is the segment that

the input x_i^t or O_r^t falls into. L_s is a lower bound of the s th segment. U_s is an upper bound of the s th segment. $w_{ij}(L_s)$ is the weight of the ij connection at the lower bound of the s th segment that x_i^t falls into. $w_{ij}(U_s)$ is the weight of the ij connection at the upper bound of the s th segment that x_i^t falls into. $w_{rj}(L_s)$ is the weight of the rj connection at the lower bound of the s th segment that O_r^t falls into. $w_{rj}(U_s)$ is the weight of the rj connection at the upper bound of the s th segment that O_r^t falls into.

- (6) Calculate the output of each output node (O_k^t) as follows:

$$O_k^t = f(\text{net}_k^t) \quad (12)$$

$$\text{net}_k^t = \sum_{j=1}^M O_j^t w_{jk}(O_j^t) \quad (13)$$

$$w_{jk}(O_j^t) = w_{jk}(L_s) + \frac{O_j^t - L_s}{U_s - L_s} \times (w_{jk}(U_s) - w_{jk}(L_s)) \quad (14)$$

where M is the total number of nodes in the hidden layer. $w_{jk}(L_s)$ is the weight of the jk connection at the lower bound of the s th segment that O_j^t falls into. $w_{jk}(U_s)$ is the weight of the jk connection at the upper bound of the s th segment that O_j^t falls into.

- (7) Compare the desired output (y_k^t) with the actual output (O_k^t) and determine the error at the output layer by

$$E = \frac{1}{2} \sum_{k=1}^C (y_k^t - O_k^t)^2 \quad (15)$$

where C is the total number of nodes in the output layer.

- (8) Update the connection weights between the hidden layer and the output layer by using the equations

$$w_{jk}^{\text{new}}(L_s) = w_{jk}^{\text{old}}(L_s) + \eta \Delta_{jk} L_s \quad (16)$$

$$w_{jk}^{\text{new}}(U_s) = w_{jk}^{\text{old}}(U_s) + \eta \Delta_{jk} U_s \quad (17)$$

$$\Delta_{jk} L_s = \left[\frac{\partial E}{\partial w_{jk}(O_j^t)} \right] \left[\frac{U_s - O_j^t}{U_s - L_s} \right] \quad (18)$$

$$\Delta_{jk} U_s = \left[\frac{\partial E}{\partial w_{jk}(O_j^t)} \right] \left[\frac{O_j^t - L_s}{U_s - L_s} \right] \quad (19)$$

$$\frac{\partial E}{\partial w_{jk}(O_j^t)} = (y_k^t - O_k^t)(O_k^t(1 - O_k^t))O_j^t \quad (20)$$

where η is the learning rate, which has a value between 0 and 1.

- (9) Update the connection weights between the group of external input units and the hidden layer by using the equations

$$w_{ij}^{\text{new}}(L_s) = w_{ij}^{\text{old}}(L_s) + \eta \Delta_{ij} L_s \quad (21)$$

$$w_{ij}^{\text{new}}(U_s) = w_{ij}^{\text{old}}(U_s) + \eta \Delta_{ij} U_s \quad (22)$$

$$\Delta_{ij} L_s = \left[\frac{\partial E}{\partial w_{ij}(x_i^t)} \right] \left[\frac{U_s - x_i^t}{U_s - L_s} \right] \quad (23)$$

$$\Delta_{ij} U_s = \left[\frac{\partial E}{\partial w_{ij}(x_i^t)} \right] \left[\frac{x_i^t - L_s}{U_s - L_s} \right] \quad (24)$$

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}(x_i^t)} &= (O_j^t(1 - O_j^t))x_i^t \\ &\times \sum_{k=1}^C (y_k^t - O_k^t)(O_k^t(1 - O_k^t))w_{jk} \end{aligned} \quad (25)$$

- (10) Update the connection weights between the group of context units and the hidden layer by using the equations

$$w_{rj}^{\text{new}}(L_s) = w_{rj}^{\text{old}}(L_s) + \eta \Delta_{rj} L_s \quad (26)$$

$$w_{rj}^{\text{new}}(U_s) = w_{rj}^{\text{old}}(U_s) + \eta \Delta_{rj} U_s \quad (27)$$

$$\Delta_{rj} L_s = \left[\frac{\partial E}{\partial w_{rj}(O_r^t)} \right] \left[\frac{U_s - O_r^t}{U_s - L_s} \right] \quad (28)$$

$$\Delta_{rj} U_s = \left[\frac{\partial E}{\partial w_{rj}(O_r^t)} \right] \left[\frac{O_r^t - L_s}{U_s - L_s} \right] \quad (29)$$

$$\begin{aligned} \frac{\partial E}{\partial w_{rj}(O_r^t)} &= (O_j^t(1 - O_j^t))O_r^t \\ &\times \sum_{k=1}^C (y_k^t - O_k^t)(O_k^t(1 - O_k^t))w_{jk} \end{aligned} \quad (30)$$

- (11) Repeat steps 4 through 10 for all remaining input patterns in the training data set.
- (12) Evaluate the weight updates occurred at the lower bound and the upper bound of each s th segment in the last training iteration whether the lower bound and the upper bound were updated in the same direction. If they were updated in different directions, the s th segment must be split into two separate segments in order to allow both end of the segment to be entirely independent of one another. For example, in Fig. 6 the lower bound of the s th segment was updated in the positive direction, while the upper bound was updated

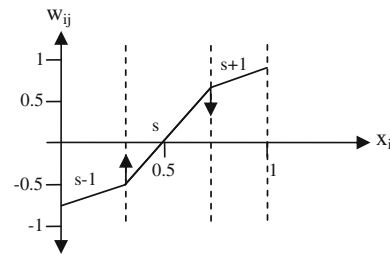


Fig. 6 An example of the case that the lower bound and the upper bound of the s th segment were updated in different directions

in the negative direction. The number of segments into which the input data range may be split cannot exceed MAXSEGMENTS. Therefore, if the number of segments into which the input data range is split is already equal to MAXSEGMENTS, skip this step and go to step 13. However if the number of segments is less than MAXSEGMENTS, perform the splitting procedure as follows:

- a. For each segment of the ij , rj , and jk connections, combine all positive $\Delta_{\bullet} L_s$ together, all negative $\Delta_{\bullet} L_s$ together, all positive $\Delta_{\bullet} U_s$ together, and all negative $\Delta_{\bullet} U_s$ together.

$$\Delta_{\bullet} L_s^+ = \sum_{\forall \Delta_{\bullet} L_s: \Delta_{\bullet} L_s > 0} (\Delta_{\bullet} L_s) \quad (31)$$

$$\Delta_{\bullet} L_s^- = \sum_{\forall \Delta_{\bullet} L_s: \Delta_{\bullet} L_s < 0} |\Delta_{\bullet} L_s| \quad (32)$$

$$\Delta_{\bullet} U_s^+ = \sum_{\forall \Delta_{\bullet} U_s: \Delta_{\bullet} U_s > 0} (\Delta_{\bullet} U_s) \quad (33)$$

$$\Delta_{\bullet} U_s^- = \sum_{\forall \Delta_{\bullet} U_s: \Delta_{\bullet} U_s < 0} |\Delta_{\bullet} U_s| \quad (34)$$

- b. Since the lower bound of the s th segment and the upper bound of the $(s - 1)$ th segment are the same point, combine the weight update value at the lower bound of the s th segment with the weight update value at the upper bound of the $(s - 1)$ th segment by using the Eqs. (35) and (36). Similarly, combine the weight update value at the upper bound of the s th segment with the weight update value at the lower bound of the $(s + 1)$ th segment by using the Eqs. (37) and (38).

$$P_s^{L^+} = \Delta_{\bullet} L_s^+ + \Delta_{\bullet} U_{s-1}^+ \quad (35)$$

$$P_s^{L^-} = \Delta_{\bullet} L_s^- + \Delta_{\bullet} U_{s-1}^- \quad (36)$$

$$P_s^{U^+} = \Delta_{\bullet} U_s^+ + \Delta_{\bullet} L_{s+1}^+ \quad (37)$$

$$P_s^{U^-} = \Delta_{\bullet} U_s^- + \Delta_{\bullet} L_{s+1}^- \quad (38)$$

- c. Compare the direction of the weight updates occurred at the lower bound with that occurred at

the upper bound of each s th segment.

$$P_s^L = \frac{P_s^{L^+} - P_s^{L^-}}{\max(P_s^{L^+}, P_s^{L^-})} \quad (39)$$

$$P_s^U = \frac{P_s^{U^+} - P_s^{U^-}}{\max(P_s^{U^+}, P_s^{U^-})} \quad (40)$$

$$P_s = P_s^L \cdot P_s^U \quad (41)$$

“ $P_s > 0$ ” means that both the lower bound and the upper bound of the s th segment were updated in the same direction. On the other hand, “ $P_s < 0$ ” means that they were updated in different directions.

Therefore, if P_s is greater than 0, skip this step and go to step 13. However, if P_s is less than 0, the s th segment will be split into two separate segments.

- d. Determine the splitting point (sp) for the s th segment whose P_s value is less than 0.

$$\bar{I}_s^+ = \frac{\sum \forall I: (L_s \leq I \leq U_s) \wedge (\Delta w_{\bullet}(I) > 0) I}{\sum \forall I: (L_s \leq I \leq U_s) \wedge (\Delta w_{\bullet}(I) > 0) 1} \quad (42)$$

$$\bar{I}_s^- = \frac{\sum \forall I: (L_s \leq I \leq U_s) \wedge (\Delta w_{\bullet}(I) < 0) I}{\sum \forall I: (L_s \leq I \leq U_s) \wedge (\Delta w_{\bullet}(I) < 0) 1} \quad (43)$$

$$sp = \frac{\bar{I}_s^+ + \bar{I}_s^-}{2} \quad (44)$$

where $\Delta w_{\bullet}(I) = \frac{\partial E}{\partial w_{\bullet}(I)}$

The variable I represents the input x_i^t , O_r^t , or O_j^t . Equations (42), (43), and (44) are used to determine the splitting point for the s th segment of the ij , rj , and jk connections. For the s th segment of the ij connection, $I = x_i^t$ and $\bullet = ij$. For the s th segment of the rj connection, $I = O_r^t$ and $\bullet = rj$. For the s th segment of the jk connection, $I = O_j^t$ and $\bullet = jk$.

- (13) Merge the unused segments. Any two segments will be joined together if (1) they are adjacent to each other and (2) they were not used in the last training iteration.
- (14) Repeat steps 4 through 13 until a predetermined number of iteration is reached or the error is within the acceptable range. Figure 7 is an example of a connection weight before and after the training process.

4 Experimental results

In this paper, the proposed approach is applied to both the original Elman and Jordan networks. The performance of the modified networks in identifying chaotic dynamical systems is evaluated and compared to the feedforward neural network and the original versions of the Elman and Jordan networks. To test the performance, all five networks were used to model 12 chaotic dynamical systems [13], which consist of two one-dimensional, two two-dimensional, seven three-dimensional,

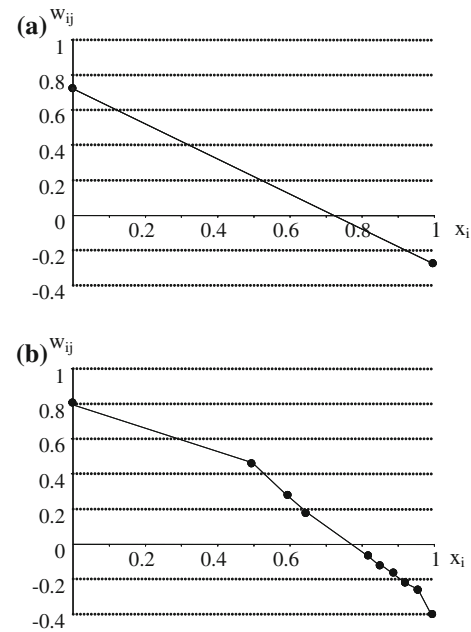


Fig. 7 Example of a connection weight **a** before and **b** after the training process

and one four-dimensional dynamical systems. The differential equations of the twelve chaotic systems used in this paper are as follows:

- (1) One-dimensional chaotic systems

- a. Mackey–Glass function

$$\frac{dx(t)}{dt} = \frac{ax(t - \tau)}{1 + x^{10}(t - \tau)} - bx \quad (45)$$

where $a = 2$, $b = 0.1$, and $\tau = 17$.

- b. Logistic map

$$\frac{dx(t)}{dt} = ax(1 - x) \quad (46)$$

where $a = 4$.

- (2) Two-dimensional chaotic systems

- c. Hénon map

$$\frac{dx(t)}{dt} = 1 - ax^2 + y \quad (47)$$

$$\frac{dy(t)}{dt} = bx \quad (48)$$

where $a = 1.4$, $b = 0.3$.

- d. Ikeda map

$$\frac{dx(t)}{dt} = 1 + \mu(x \cos(t) - y \sin(t)) \quad (49)$$

$$\frac{dy(t)}{dt} = \mu(x \sin(t) + y \cos(t)) \quad (50)$$

where $\mu = 0.9$.

(3) Three-dimensional chaotic systems

e. Rössler attractor

$$\frac{dx(t)}{dt} = -y - z \quad (51)$$

$$\frac{dy(t)}{dt} = x + ay \quad (52)$$

$$\frac{dz(t)}{dt} = b + z(x - c) \quad (53)$$

where $a = 0.2$, $b = 0.4$, and $c = 5.7$.

f. Chua's autonomous circuit

$$\frac{dx(t)}{dt} = \alpha(-x + h(x)) \quad (54)$$

$$\frac{dy(t)}{dt} = x - y + z \quad (55)$$

$$\frac{dz(t)}{dt} = -\beta y \quad (56)$$

where $h(x) = bx + 0.5(a - b)(|x + 1| - |x - 1|)$, $\alpha = 10$, $\beta = 15$, $a = -1.3$, and $b = -0.7$.

g. Chua's circuit with a cubic nonlinearity

$$\frac{dx(t)}{dt} = \alpha(-x + h(x)) \quad (57)$$

$$\frac{dy(t)}{dt} = x - y + z \quad (58)$$

$$\frac{dz(t)}{dt} = -\beta y \quad (59)$$

where $h(x) = ax + bx^3$, $\alpha = 8.85$, $\beta = 15$, $a = -1.3$, and $b = 0.07$.

h. Lorenz chaotic system

$$\frac{dx(t)}{dt} = \sigma(y - x) \quad (60)$$

$$\frac{dy(t)}{dt} = rx - y - xz \quad (61)$$

$$\frac{dz(t)}{dt} = xy - bz \quad (62)$$

where $\sigma = 16$, $r = 45.92$, and $b = 4$.

i. Colpitts oscillator

$$\frac{dx(t)}{dt} = z - \beta \cdot f(y) \quad (63)$$

$$\frac{dy(t)}{dt} = \alpha(-r(y + \varphi) - z - f(y)) \quad (64)$$

$$\frac{dz(t)}{dt} = \delta(-x + y + \varepsilon) - \rho z \quad (65)$$

where

$$f(y) = \begin{cases} 0; & y \leq 1 \\ y - 1; & \text{otherwise} \end{cases}$$

$\alpha = 1$, $\beta = 200$, $\varphi = -20/3$, $\delta = 5.5$, $\varepsilon = 20/3$, $\rho = 2$, and $r = 0.25$.

j. RC chaotic oscillator

$$\frac{dx(t)}{dt} = k \cdot m(x - z) - x \quad (66)$$

$$\frac{dy(t)}{dt} = k \cdot n \cdot f(y, z) \quad (67)$$

$$\frac{dz(t)}{dt} = k \cdot m(x - z) - k \cdot n \cdot f(y, z) \quad (68)$$

where

$$f(y, z) = \begin{cases} z - y; & z - y \leq 1 \\ 1; & \text{otherwise} \end{cases}$$

$k = 2$, $m = 11$, and $n = 0.9$.

k. Third order RC ladder phase shift oscillator

$$\frac{dx(t)}{dt} = -2x + y + f(z) \quad (69)$$

$$\frac{dy(t)}{dt} = x - 2y + z \quad (70)$$

$$\frac{dz(t)}{dt} = y - z \quad (71)$$

where

$$f(z) = \begin{cases} m_b \cdot z + m_b - m_a; & z < -1 \\ m_a \cdot z; & -1 \leq z \leq 1 \\ m_b \cdot z - m_b + m_a; & \text{otherwise} \end{cases}$$

$m_a = -33.03$, $m_b = 250$, $x(t = 0) = 3$, $y(t = 0) = 0$, and $z(t = 0) = -1$.

(4) Four-dimensional chaotic system

1. Four-dimensional autonomous chaotic circuit

$$\frac{dx(t)}{dt} = \delta * (x + \gamma * y) - z - f_{d1}(x) \quad (72)$$

$$\frac{dy(t)}{dt} = \beta * (\delta * (x + \gamma * y) - w - \gamma * f_{d2}(y)) \quad (73)$$

$$\frac{dz(t)}{dt} = x \quad (74)$$

$$\frac{dw(t)}{dt} = \alpha * y \quad (75)$$

where

$$f_{dk}(x) = \frac{0.5 * (|x - 1| + x - 1)}{\varepsilon_k}; \quad k = 1, 2$$

$\alpha = 2$, $\beta = 0.17$, $\gamma = 3$, $\delta = 0.27$, $\varepsilon_1 = 0.8$, $\varepsilon_2 = 0.2$, $x(t = 0) = 0.1$, $y(t = 0) = 0$, $z(t = 0) = 0$, and $w(t = 0) = 0$.

Two thousand data points were sampled from each chaotic system. The first 1,000 data points of the series were used as a training data, while the remaining 1,000 were used as a testing data.

Since the efficiency of the artificial neural networks depends highly on the user's ability to specify appropriate

Table 1 Experimental results: RMSE of the tested models

Chaotic function	Feedforward network		Elman network		Jordan network		Modified Elman network			Modified Jordan network		
	RMSE	<i>M</i>	RMSE	<i>M</i>	RMSE	<i>M</i>	RMSE	<i>M</i>	<i>S</i>	RMSE	<i>M</i>	<i>S</i>
Mackey–Glass	0.0357	8	0.0163	8	0.0307	3	0.0116	41	1–15	0.0157	3	20–20
Logistic	0.0415	38	0.0251	30	0.0367	30	0.0166	3	1–10	0.0112	49	1–15
Hénon	0.0272	50	0.0193	48	0.0223	48	0.0160	34	1–20	0.0098	50	1–20
Ikeda	0.0797	37	0.0225	42	0.0559	19	0.0193	26	1–1	0.0162	49	1–15
Rössler	0.0210	14	0.0189	12	0.0208	12	0.0114	36	1–15	0.0158	43	1–20
Chua's autonomous circuit	0.0216	7	0.0246	5	0.0213	8	0.0115	49	1–20	0.0143	49	1–15
Chua's circuit with a cubic nonlinearity	0.0227	18	0.0209	6	0.0220	12	0.0153	47	1–20	0.0162	48	1–15
Lorenz	0.0213	7	0.0157	18	0.0166	9	0.0136	8	1–2	0.0144	14	1–1
Colpitts oscillator	0.0186	37	0.0169	8	0.0186	36	0.0135	45	1–20	0.0139	49	1–15
RC chaotic oscillator	0.0206	10	0.0128	7	0.0196	6	0.0114	50	1–15	0.0123	45	1–20
Third order RC ladder phase shift oscillator	0.0205	13	0.0182	9	0.0207	34	0.0204	42	1–20	0.0187	50	1–15
Four dimensional autonomous chaotic circuit	0.0237	14	0.0232	20	0.0235	14	0.0202	6	1–1	0.0199	11	1–1

M total number of nodes in the hidden layer, *S* number of segments into which the input data range is divided

neural network parameters, a very important parameter of all five models, which is the number of hidden neurons, is varied from 1 to 50 hidden nodes in order to get the best out of the five models. The training process is terminated when either of the following two conditions holds: (1) the root mean square error (RMSE) is less than 0.02, or (2) the number of training epochs reaches 1,000.

Results of the experiments are shown in Tables 1 and 2. Table 1 illustrates the best results (minimum RMSE) of the feedforward neural network, Elman network, Jordan network, the modified Elman network, and the modified Jordan network. In the modified networks, each weight is divided into different number of segments. Some have many segments while others have only a few. The variable *S* in Table 1 is in the format of two numbers separated by dash. The first number is the minimum number of segments. The second number is the maximum number of segments. The following summarizes the percentage of improvement in RMSE of the modified networks over both their original counterparts and the feedforward neural network:

- The performance of the modified Elman network is better than the performance of the feedforward neural network by 75.78, 67.51, 60.00, 46.76, 45.71, 44.66, 41.18, 36.15, 32.60, 27.42, 14.77, and 0.49% for Ikeda, Mackey–Glass, Logistic, Chua's autonomous circuit, Rössler, RC chaotic oscillator, Hénon, Lorenz, Chua's circuit with a cubic nonlinearity, Colpitts oscillator, Four dimensional autonomous chaotic circuit, and Third order RC ladder phase shift oscillator, respectively.

- The performance of the modified Elman network is better than the performance of the Elman network by 53.25, 39.68, 33.86, 28.83, 26.79, 20.12, 17.10, 14.22, 13.38, 12.93, 10.94, and –12.09% for Chua's autonomous circuit, Rössler, Logistic, Mackey–Glass, Chua's circuit with a cubic nonlinearity, Colpitts oscillator, Hénon, Ikeda, Lorenz, Four dimensional autonomous chaotic circuit, RC chaotic oscillator, and Third order RC ladder phase shift oscillator, respectively.
- The performance of the modified Jordan network is better than the performance of the feedforward neural network by 79.67, 73.01, 63.97, 56.02, 40.29, 33.80, 32.39, 28.63, 25.27, 24.76, 16.03, and 8.78% for Ikeda, Logistic, Hénon, Mackey–Glass, RC chaotic oscillator, Chua's autonomous circuit, Lorenz, Chua's circuit with a cubic nonlinearity, Colpitts oscillator, Rössler, Four dimensional autonomous chaotic circuit, and Third order RC ladder phase shift oscillator, respectively.
- The performance of the modified Jordan network is better than the performance of the Jordan network by 71.02, 69.48, 56.05, 48.86, 37.24, 32.86, 26.36, 25.27, 24.04, 15.32, 13.25, and 9.66% for Ikeda, Logistic, Hénon, Mackey–Glass, RC chaotic oscillator, Chua's autonomous circuit, Chua's circuit with a cubic nonlinearity, Colpitts oscillator, Rössler, Four dimensional autonomous chaotic circuit, Lorenz, and Third order RC ladder phase shift oscillator, respectively.

Table 2 illustrates the number of iterations and the training time each model requires to reach the stopping criteria. With

Table 2 Experimental results: the number of iterations the tested models require to reach the stopping criteria

Chaotic function	Number of training epochs				
	Feedforward network	Elman network	Jordan network	Modified Elman network	Modified Jordan network
Mackey–Glass	1,000 (00-05-37)	785 (00-03-57)	1,000 (00-05-09)	6 (00-03-17)	564 (00-03-12)
Logistic	1,000 (00-12-28)	1,000 (02-24-29)	1,000 (00-16-38)	48 (00-00-12)	11 (00-00-16)
Hénon	1,000 (00-28-59)	309 (01-53-10)	1,000 (00-43-31)	12 (00-02-53)	10 (00-00-39)
Ikeda	1,000 (00-23-50)	1,000 (04-47-37)	1,000 (00-19-21)	904 (01-30-36)	12 (00-00-41)
Rössler	927 (00-12-45)	384 (00-07-41)	820 (00-12-22)	11 (00-02-51)	12 (00-00-52)
Chua's autonomous circuit	328 (00-02-31)	104 (00-00-28)	199 (00-02-23)	6 (00-05-13)	8 (00-00-54)
Chua's circuit with a cubic nonlinearity	1,000 (00-18-14)	628 (00-07-58)	971 (00-17-36)	6 (00-05-01)	8 (00-00-55)
Lorenz	282 (00-02-24)	119 (00-04-04)	838 (00-05-39)	137 (00-01-42)	288 (00-03-57)
Colpitts oscillator	978 (00-33-22)	922 (00-17-11)	872 (00-44-38)	8 (00-04-30)	9 (00-00-51)
RC chaotic oscillator	270 (00-02-33)	481 (00-04-20)	301 (00-03-09)	10 (00-05-13)	12 (00-00-57)
Third order RC ladder phase shift oscillator	783 (00-06-42)	314 (00-04-08)	882 (00-23-41)	10 (00-03-57)	8 (00-00-47)
Four dimensional autonomous chaotic circuit	1,000 (00-19-33)	219 (00-19-22)	1,000 (00-24-10)	459 (00-06-57)	410 (00-10-13)

The figures in brackets, which are in the format of HH-MM-SS, indicate the time used in training the models

The machines used in the experiments are Pentium IV 3.0 GHz PCs with 512 MB of RAM, running Window XP

the modified networks, the training is significantly faster than their original counterparts. Even though the training time per epoch is longer for the modified networks, the total training time of the modified networks is lower due to their ability to learn every dynamical system within a small number of training iterations.

In summary, experimental results from Tables 1 and 2 show that both modified networks, the ones with multi-segment piecewise-linear connection weights, are far superior to both their original counterparts and the feedforward neural network.

5 Conclusion

This paper applies a new concept of the connection weight to the standard Elman and Jordan recurrent neural networks. Unlike the connection weights of the original recurrent neural networks, the weight of each connection in this proposed approach is multi-valued, depending on the value of the input data involved. The performance of the modified networks is evaluated against the feedforward neural network and the original Elman and Jordan recurrent neural networks. All the compared models were used to approximate 12 chaotic functions. The experimental results show that the modified networks outperform the other three networks by a wide margin.

References

1. Rao SS, Kumthekar B (1995) A composite neural architecture and algorithm for nonlinear system identification. In: Proceedings of the 1995 artificial neural networks in engineering conference (ANNIE' 95), pp 77–84
2. Lee H, Park Y (1991) Nonlinear system identification using recurrent networks. In: Proceedings of the 1991 IEEE international joint conference on neural networks, pp 2410–2415
3. Karaboga D, Kalinli A (1997) Training recurrent neural networks for dynamic system identification using parallel tabu search algorithm. In: Proceedings of the 12th IEEE international symposium on intelligent control, pp 113–117
4. Yu W (2004) Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms. *Inf Sci* 158:131–147
5. Pham DT, Karaboga D (1999) Training Elman and Jordan networks for system identification using genetic algorithms. *Artificial Intelligence in Engineering* 13, no 2, Elsevier, Amsterdam, pp 107–117
6. Yu X, Tang K, Chen T, Yao X (2009) Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. *Memetic Computing*, vol 1, no 1, Springer, Berlin, pp 3–24
7. Elman JL (1990) Finding structure in time. *Cognitive Science* 14:179–211
8. Jordan MI (1986) Attractor dynamics and parallelism in a connectionist sequential machine. In: Proceedings of the eighth annual conference of the cognitive science society, pp 531–546
9. Cheng YC, Qi WM, Cai WY (2002) Dynamic properties of Elman and modified Elman neural network. In: Proceedings of the first international conference on machine learning and cybernetics, pp 637–640
10. Gao XZ, Gao XM, Ovaska SJ (1996) A modified Elman neural network model with application to dynamical systems identification.

- In: Proceedings of the IEEE international conference on systems, man, and cybernetics, pp 1376–1381
11. Chen TB, Soo VW (1996) A comparative study of recurrent neural network architectures on learning temporal sequences. In: Proceedings of the IEEE international conference on neural networks, pp 1945–1950
 12. Ji QL, Qi WM (2007) The property of PID Elman neural network and its application in identification of hydraulic unit. In: Proceedings of the 2007 IEEE international conference on control and automation, pp 1795–1798
 13. Bogdan C (Online) Chaotic Generators Demo. Available at <http://www.mathworks.com/matlabcentral/fileexchange/8382>