Research Article

Fuzzy Counter Propagation Neural Network Control for a Class of Nonlinear Dynamical Systems

Vandana Sakhre, 1 Sanjeev Jain, 1 Vilas S. Sapkal, 2 and Dev P. Agarwal 3

1 Madhav Institute of Technology & Science, Gwalior 474005, India
2 SGB Amravati University, Amravati 444062, India
3 IIT, Delhi 110001, India

Correspondence should be addressed to Vandana Sakhre; vssakhre@gmail.com

Received 3 May 2015; Revised 1 August 2015; Accepted 3 August 2015

Academic Editor: Ezequiel López-Rubio

Copyright © 2015 Vandana Sakhre et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Fuzzy Counter Propagation Neural Network (FCPN) controller design is developed, for a class of nonlinear dynamical systems. In this process, the weight connecting between the instar and outstar, that is, input-hidden and hidden-output layer, respectively, is adjusted by using Fuzzy Competitive Learning (FCL). FCL paradigm adopts the principle of learning, which is used to calculate Best Matched Node (BMN) which is proposed. This strategy offers a robust control of nonlinear dynamical systems. FCPN is compared with the existing network like Dynamic Network (DN) and Back Propagation Network (BPN) on the basis of Mean Absolute Error (MAE), Mean Square Error (MSE), Best Fit Rate (BFR), and so forth. It envisages that the proposed FCPN gives better results than DN and BPN. The effectiveness of the proposed FCPN algorithms is demonstrated through simulations of four nonlinear dynamical systems and multiple input and single output (MISO) and a single input and single output (SISO) gas furnace Box-Jenkins time series data.

1. Introduction

With the rapid advance of computers, the digitization has swept in all fields of science and technology with special emphasize on modeling and identification. Most of the physical processes are continuous in nature but they are generally modeled using discrete time with the use of derivative and integral as they are easily realizable. Taking these advantages, an enormous research has been focused on discrete time methods and many techniques based on linear and nonlinear discrete systems have been developed [1, 2]. Neural network has been widely used for nonlinear dynamical systems [3–5]. Various control systems like back stepping control [6, 7], sliding mode control [8, 9], and control using soft computing [10] are continuous source of interest. The nonlinear dynamical system is a generic problem which finds its application in every field of engineering. The technique based on soft computing, fuzzy logic, and neural network has also found its application in modeling of systems in various application domains. Feedforward Neural Network (FNN) is one of the most commonly used networks for this purpose; FNN with Back Propagation algorithm is another powerful network. Neural network is most popular approach due to its capability of modeling most of the nonlinear functions approximately to any arbitrary degree of accuracy [11]. Most of the systems are designed using feedforward network with gradient descent learning but the gradient descent method encounters the problem of slow convergence. This problem of convergence can be overcome by use of some Cauchy-Newton [12] Quasi-Newtonian method and Levenberg-Marquardt algorithms [13].

Optimal controllers for hybrid dynamical systems (HDS) developed using Hamilton-Jacobi-Bellman (HJB) solution method [14, 15]. Novel Lyapunov-Krasovskii functional (LKF) with triple integral time constructed for exponential synchronization of complex dynamical networks will control packet loss and additive time varying delay [16]. Control of discrete time varying system using dynamical model is difficult. To overcome this condition, new neural network approximation structure was developed to solve optimal tracking problem of nonlinear discrete time varying time system using reinforcement learning (RL) method [17]. To
enhance the performance of nonlinear multivariable system, fuzzy optimal controller based Takagi-Sugeno model was developed which was used to minimize a quadratic performance index [18]. An ultimate stable training algorithm inspired by adaptive observer for black box identification of nonlinear discrete systems developed using state space recurrent neural network. The algorithm developed was using only input and output measurement in discrete time [19]. Indirect adaptive controller that uses neural network was presented for the identification and control of experimental pilot distillation column. This system is multivariable with unknown dynamics and the neural network was trained using Levenberg-Marquardt algorithm [20]. Adaptive finite time stabilization of a class of switched nonlinear systems was investigated with unknown nonlinear terms using neural network. The finite law and adaptive law were constructed using radial basis function neural network (RBFNN) to approximate the unknown packaged function. Stability analysis of RBFNN was observed to evaluate the states of the closed loop systems [21]. An adaptive fuzzy output tracking control approach was proposed in [22] for single input single output (SISO) system which is uncertain and nonlinear under arbitrary switching. An iterative learning control scheme was proposed for a class of nonlinear dynamic systems which includes holonomic systems as its subsets with linear feedback mechanism and feedforward learning strategies [23].

In this proposed work we have used instar-outstar structure based CPN with Fuzzy Competitive Learning (FCL). We have designed Fuzzy Counter Propagation Network design to control some nonlinear dynamical systems. In the FCPN, CPN model is trained by FCL algorithms. The FCL learning is used for adjusting weights and update of Best Matched Node (BMN) in discrete time nonlinear dynamical system.

The main contributions of this research are as follows:

1. This paper contributes the approximation for a class of nonlinear dynamical systems using Fuzzy Competitive Learning Based Counter Propagation Network (FCPN).

2. FCPN is employed to optimize the Mean Absolute Error, Mean Square Error, Best Fit Rate, and so forth.

3. Performance criteria of proposed FCPN for nonlinear dynamical systems are effectively improved by compensating reference signal and controller signal.

The paper is organized as follows. In Section 2, problem formulations for a class of nonlinear dynamical systems are given. Section 3 contains description of Feedforward Neural Network (FNN) with Back Propagation Network, Dynamic Network, and Fuzzy Learning Based Counter Propagation Network. Dynamical learning for CPN and optimal learning of dynamical system are presented in Section 4. The simulation results and comparison of different nonlinear dynamical models are presented in Section 5. Section 6 gives conclusion of the study.

2. Problem Formulation

Let us consider four models of discrete time nonlinear dynamical systems [24] for single input single output (SISO) and multiple input and single output (MISO) system considered in this paper and they are described by difference equations (1)–(4) and Box-Jenkins time series data [10]. Let \( f: \mathbb{R}^n \rightarrow \mathbb{R} \) and \( g: \mathbb{R}^m \rightarrow \mathbb{R} \) be the nonlinear continuous differentiable function of Model I–Model IV approximated by FCPN to the desired degree of accuracy. Once the system has been parameterized, the performance evaluation has been carried out by FCPN for (1) to (4).

Model I:

\[
y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + g[u(k), \ldots, u(k-m+1)].
\]

Model II:

\[
y(k+1) = f[y(k), \ldots, y(k-n+1)] + \sum_{i=0}^{m-1} \beta_i u(k-i).
\]

Model III:

\[
y(k+1) = f[y(k), \ldots, y(k-n+1)] + g[u(k), \ldots, u(k-m+1)].
\]

Model IV:

\[
y(k+1) = f[y(k), \ldots, y(k-n+1); u(k), \ldots, u(k-m+1)],
\]

where \([u(k), y(k)]\) represent the input-output pair of the system at time \(k\) and their order is represented by \((n,m)\). FCL is used to learn the system defined for (1) to (4) and the performance of the FCPN can be measured by error function given in

\[
E(k) = \frac{1}{2} \sum_{i=0}^{P} [y(k) - \overline{y}(k)]^2,
\]

where \(u(k)\) is the input, \(y(k)\) is the system output, \(\overline{y}(k)\) is neural network output, and (5) can be written for neural controller as

\[
\bar{E} = \frac{1}{P} \sum_{p=1}^{P} [y(k) - \overline{y}^r(k)]^2,
\]

where \(\overline{y}^r(k)\) is neural controller output and (6) is known as minimized error function and \(P\) is total number of input patterns.

3. Back Propagation Network (BPN)

A neural network is one of the most popular intelligent systems which has capability to approximate calculated and target value at an arbitrary degree of accuracy. BPN is one
of the most commonly used networks for this purpose. The BPN consists of input layer, hidden layer, and output layer and possesses weighted interconnections. The BPN is based on supervised learning. This method is used where error is propagated back to the hidden layer. The aim of this neural network is to train the net to achieve a balance between the net’s ability to respond (memorization) and its ability to give reasonable response to the input that is similar but not identical to the one that is used in training (generalization).

For a given set of training input-output pairs, this network provides a procedure for changing weights to classify the given input patterns correctly. The weight update algorithm is based on gradient descent method [25]. The network architecture is shown in Figure 1.

3.1. Dynamic Network. Neural network can be classified into two categories: (i) Static and (ii) Dynamic. In Static Neural Network, output can be calculated directly from input through feedforward network. But in Dynamic Network output depends on current or previous inputs, outputs, or states of network. Where the current output is function of current input and previous output, it is known as recurrent (feedback) network. Training process of Static and Dynamic Network can be differentiated by use of gradient or Jacobian which is computed. Dynamic Network contains delays and operates on a sequence of inputs. Dynamic Networks can have purely feedforward connections or they can have some recurrent connections. They can be trained using Dynamic Back Propagation Network [26].

The general equation of the net input \( n^m(t) \) for \( m \)th layer is given by

\[
n^m(t) = \sum_{l \in L_m} \sum_{d \in DI_{m,l}} IW^{m,l}(d) a^l(t - d) + \sum_{l \in L_m} \sum_{d \in DI_{m,l}} IW^{m,l}(d) b^l(t - d) + b^m,
\]

where \( b^l(t) \) is the \( l \)th input vector at time \( t \), \( IW^{m,l} \) is the input weight between \( l \)th and \( m \)th layer, \( LW^{m,l} \) is the layer weight between \( l \)th and \( m \)th layer, and \( b^m \) is the bias vector for layer \( m \). \( DL_{m,l} \) is the set of all delays in the Tapped Delay Line (TDL) between \( l \)th and \( m \)th layer and \( DI_{m,l} \) is the set of all delays in the TDL between input \( l \) and \( m \)th layer, \( I_m \) is the set of indices of input vectors that connect to layer \( m \), and \( L_f^m \) is the set of indices of layers that connect to layer \( m \). The output of \( m \)th layer at time \( t \) is computed as

\[
a^m(t) = f^m(n^m(t)).
\]

Network has a TDL on the input with \( DI_{1,1} = \{0, 1, 2\} \) and its output is represented as

\[
a(t) = f(n(t)) = \sum_d IW(d) * p(t - d),
\]

\[
a(t) = f[\sum_d IW_{1,1}(d) p(t - d) + IW_{1,1}(0) p(t) + IW_{1,1}(1) p(t - 1) + IW_{1,1}(2) p(t - 2)].
\]

One simple architecture with delay \( DI_{1,1} = \{0, 1, 2\} \) for feedforward Dynamic Network is shown in Figure 2 known as Dynamic Adaptive Linear Neuron.

3.2. Counter Propagation Network (CPN). It is multilayer feedforward network based on the combination of the input, competitive, and output layers. Model of CPN is instar-outstar. It is three-layer neural network that performs input-output data mapping, that is, producing output in the response to an input vector on the basis of Fuzzy Competitive Learning. In CPN the connection between input layer and competitive layer is instar structure and connection between competitive and output layer is outstar structure. Counter Propagation Network involves two-stage training process. In first stage input vector is clustered on the basis of Euclidean...
distance and the performance of network is improved using linear topology. Using Euclidean distance between input and weight vector, we evaluated BMN. In phase II, the desired response is obtained by adapting the weights from competitive layer to output layer [27]. Let $x = [x_1 x_2 \cdots x_n]$ and $y = [y_1^p y_2^p \cdots y_n^p]$ be input and desired vector, respectively, let $v_{ij}$ be weight of input layer and competitive layer, where $1 \leq i \leq n$ and $1 \leq j \leq p$, and let $w_{kj}$ be the weight between competitive layer and output layer, where $1 \leq k \leq m$. Euclidean distance between input vector $x$ and weight vector $v_{ij}$ is

$$D_j = \sum_{i=1}^{n} (x_i - v_{ij})^2, \text{ where } j = 1, 2, \ldots, p. \quad (10)$$

The architecture of CPN is shown in Figure 3.

### 3.3. Fuzzy Competitive Learning (FCL)

FCL is used for adjusting instar-outstar weights and update of BMN for discrete time nonlinear dynamical systems. Learning of CPN is divided into two phases: Fuzzy Competitive Learning phase (unsupervised) and Grossberg Learning phase (supervised) [28–31]. CPN is efficient network for mapping the input vector of size $n$ placed in clusters of size $m$. Like traditional self-organizing map, the weight vector for every node is considered as a sample vector to the input pattern. The process is to determine which weight vector (say $J$) is more similar to the input pattern chosen as BMN. First phase is based on closeness between weight vector and input vector and second phase is weight update between competitive and output layer for desired response.

Let $v_{ij}$ be weight of input node $i$ to neuron $j$ and let $w_{kj}$ be the weight between competitive node $j$ and neuron $k$. We propose a new learning rate calculation method for use of weight update:

$$\alpha(J) = \frac{\sum_{j=1}^{n} (v_{ij} - x_i)^2}{\sum_{j=1}^{n} \sum_{k=1}^{m} (v_{ij} - x_i)^2}, \quad (11)$$

where $J$ denotes the index of BMN and $\alpha$ is known as Fuzzy Competitive Learning rate.

Fuzzy Competitive Learning phase is described as follows.

#### 3.3.1. Training Algorithm of FCL

**Phase I (for determination of BMN).** There are steps for training of CPN as follows.

**Step 1.** Initialize instar-outstar weights.

**Step 2.** Perform Steps 3–8 until stopping criteria for Phase I training fail. The stopping condition may be fixed number of epochs or learning rate has reduced to negligible value.

**Step 3.** Perform Steps 4–6 for all input training vector $X$.

**Step 4.** Set the $X$-input layer activation to vector $X$.

**Step 5.** Compute the Best Matched Node (BMN) ($J$) using Euclidean distance; find the node $Z_j$ whose distance from the input pattern is minimum. Euclidean distance is calculated as follows:

$$D_j = \sum_{i=1}^{n} (x_i - v_{ij})^2, \text{ where } j = 1, 2, \ldots, p. \quad (12)$$

Minimum net input:

$$z_{wj} = \sum_{i=1}^{n} x_i v_{ij}. \quad (13)$$

**Step 6.** Calculate Fuzzy Competitive Learning (14) rate for weight update using BMN.

**Step 7.** We update weight for unit $Z_j$:

$$v_{ij} \text{ (new)} = v_{ij} \text{ (old)} + \alpha h(J; i, j) (x_i - v_{ij}), \quad (14)$$

where $i = 1, 2, 3, \ldots, n$ and $h(\cdot; \cdot)$ is neighborhood function around BMN. Consider

$$h(J; i, j) = \exp \left( -\frac{\|v_{ij} - v_{ij}\|^2}{2\sigma^2(t)} \right), \quad (15)$$

where $h(J; i, j) \in [0, 1],$

$$\sigma(t) = \sigma_0 \exp \left( -\frac{t}{T} \right). \quad (16)$$

**Step 8.** Test the stopping criteria.

**Phase II (to obtain desired response)**

**Step 1.** Set activation function to $(x, y)$ input and output layer, respectively.

**Step 2.** Update the BMN $(J)$ (Step 5 from Phase I). Also update the weights into unit $Z_j$ as

$$v_{ij} \text{ (new)} = v_{ij} \text{ (old)} + \alpha h(J; i, j) [x_i - v_{ij} \text{ (old)}]. \quad (17)$$
Step 3. Update the weights from node $Z_j$ to the output unit as

$$ w_{jk} \text{ (new)} = w_{jk} \text{ (old)} + \beta h(J; j, k) \left[ y_j - w_{jk} \text{ (old)} \right] $$

for $1 \leq k \leq m$.

Step 4. Learning rate $\beta$ is calculated as

$$ \beta(J) = \frac{\sum_{k=1}^{m} (y_k - w_{jk})^2}{\sum_{j=1}^{p} \sum_{k=1}^{m} (y_k - w_{jk})^2}. $$

Step 5. Decrease the rate of learning s.t. $\beta = \beta - \epsilon$, where $\epsilon$ is a small positive number.

Step 6. Test the stopping condition for Phase II (i.e., fixed number of epochs or its learning rate has reduced to negligible value).

3.3.2. Testing Phase (to Test FCPN)

Step 1. Set the initial weights, that is, the weights obtained during training.

Step 2. Apply FCPN to the input vector $X$.

Step 3. Find unit $J$ that is closest to vector $X$.

Step 4. Set activations of output units.

Step 5. Apply activation function at $y_k$, where $y_k = \sum_j z_j w_{jk}$.

4. Dynamical Learning for CPN

Learning stabilities are fundamental issues for CPN; however, there are few studies on the learning issue of FNN. BPN for learning is not always successful because of its sensitivity to learning parameters. Optimal learning rate always changes during the training process. Dynamical learning of CPN is carried out using Lemmas 1, 2, and 3.

Assumption. Let us assume $\phi(x)$ is a sigmoid function if it is bounded continuous and increasing function. Since input to the neural network in this model is bounded, we consider Lemmas 1 and 2 given below.

Lemma 1. Let $\phi(x)$ be a sigmoid function and let $\Omega$ be a compact set in $\mathbb{R}^n$, and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ on $\Omega$ is a continuous function and, for arbitrary $\epsilon > 0$, $\exists$ integer $N$ and real constants $c_j$, $\theta_i$, and $w_{ij}$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, n$, such that

$$ \tilde{f}(x_1, x_2, \ldots, x_N) = \sum_{i=1}^{N} c_i \theta \left( \sum_{j=1}^{n} w_{ij} x_j - \theta_i \right) $$

satisfies

$$ \max_{x \in \Omega} \left\| f(x_1, x_2, \ldots, x_n) - \tilde{f}(x_1, x_2, \ldots, x_n) \right\| < \epsilon. $$

Using Lemma 1, dynamical learning for a three-layer CPN can be formulated where the hidden layer transfer functions are $\phi(x)$ and transfer function at output layer is linear.

Let all the vectors be column vectors and superscript $d_k^p$ refers to specific output vectors component.
Let $X = [x_1, x_2, \ldots, x_p] \in \mathbb{R}^{L \times P}$, $Y = [y_1, y_2, \ldots, y_p] \in \mathbb{R}^{H \times P}$, $O = [o_1, o_2, \ldots, o_p] \in \mathbb{R}^{K \times P}$, $D = [d_1, d_2, \ldots, d_p] \in \mathbb{R}^{K \times P}$ be the input, hidden, output, and desired vector, respectively, where $L$, $H$, and $K$ denote the number of input, hidden, and output layer neurons. Let $V$ and $W$ represent the input-hidden and hidden-output layer weight matrix, respectively. The objectives of the network training minimize error function $J$, where $J$ is

$$J = \frac{1}{2PK} \sum_{p=1}^{P} \sum_{k=1}^{K} (o_{pk}^k - d_{pk}^k)^2. \quad (22)$$

### 4.1. Optimal Learning of Dynamical System

Consider a three-layer CPN; the network error matrix is defined by the error between differences of desired and FCPN output at any iteration and given as

$$E_t = O_t - D = WX_tY_t - D = WX_tX - D. \quad (23)$$

The objective of the network for minimization of error given in (24) is defined as follows:

$$\text{s.t. } J = \frac{1}{2PK} \text{Tr} \left( E_t E_t^T \right), \quad (24)$$

where $\text{Tr}$ represent the trace of matrix. Using gradient descent method updated weight is given by

$$W_{t+1} = W_t - \beta \frac{\partial J}{\partial W_t},$$

$$V_{t+1} = V_t - \beta \frac{\partial J}{\partial V_t}, \quad \text{(25)}$$

or

$$W_{t+1} = W_t - \frac{\beta}{PK} E_t Y_t^T,$$

$$V_{t+1} = V_t - \frac{\beta}{PK} W_t^T E_t X^T. \quad \text{(26)}$$

Using (23)–(26), we have

$$E_{t+1} E_{t+1}^T = (W_t X_t + D - D) (W_t X_t + D) = (W_t X_t - D) (W_t X_t - D)^T. \quad (27)$$

To obtain minimum error for multi-layer FNN in above equation (28) after simplification, we have

$$J_{t+1} - J_t = \frac{1}{2PK} \text{Tr} \left( (W_t - \frac{\beta}{PK} E_t Y_t^T) (V_t - \frac{\beta}{PK} W_t^T E_t X^T) X \right.$$

$$- D \left( [E_t - \frac{\beta}{PK} (W_t W_t^T E_t X^T X + E_t Y_t^T V_t X)] + \frac{\beta^2}{(PK)^2} E_t Y_t^T W_t^T E_t X^T X \right) + \frac{\beta^3}{(PK)^3} W_t W_t^T E_t X^T X E_t^T X$

$$+ E_t Y_t^T W_t^T E_t X^T X E_t^T X$$

$$- \frac{\beta}{PK} [E_t (W_t W_t^T E_t X^T X) + E_t (E_t Y_t^T V_t X)]$$

$$+ \left( W_t W_t^T E_t X^T X E_t^T X \right) (E_t Y_t^T V_t X)$$

$$- \frac{\beta^2}{(PK)^3} E_t \left( \frac{\beta}{PK} W_t E_t X^T X \right) E_t Y_t^T W_t^T E_t X^T X \right)^T$$

$$+ E_t Y_t^T W_t^T E_t X^T X E_t^T X$$

$$+ \frac{\beta^3}{(PK)^3} [E_t (W_t W_t^T E_t X^T X) + E_t (E_t Y_t^T V_t X)]$$

$$+ \left( W_t W_t^T E_t X^T X E_t^T X \right) (E_t Y_t^T V_t X)$$

$$- \frac{\beta^2}{(PK)^3} E_t \left( \frac{\beta}{PK} W_t E_t X^T X \right) E_t Y_t^T W_t^T E_t X^T X \right)^T.$$

(28)

For simplification, omit subscript $t$, we have; that is,

$$J_{t+1} - J_t = \frac{1}{2PK} \left( A\beta^4 + B\beta^3 + C\beta^2 + M\beta \right), \quad \text{(29)}$$

where

$$A = \frac{1}{(PK)^4} \text{Tr} \left( E Y_t^T W E X^T X \right)^T E Y_t^T W E X^T X$$

$$B = \frac{1}{(PK)^3} \text{Tr} \left( W W^T E X^T X \right)^T E Y_t^T W E X^T X$$

$$C = \frac{1}{(PK)^2} \text{Tr} \left( W W^T E X^T X \right)^T E Y_t^T W E X^T X$$

$$M = \frac{1}{(PK)} \text{Tr} \left( W W^T E X^T X \right)^T E Y_t^T W E X^T X.$$
+ \mathbf{E} \mathbf{Y}^{T} \mathbf{V} \mathbf{X} \left( \mathbf{E} \mathbf{Y}^{T} \mathbf{W}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \right)^{T} \\
+ \mathbf{E} \mathbf{Y}^{T} \mathbf{W}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \left( \mathbf{W} \mathbf{W}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \right)^{T} \\
+ \mathbf{E} \mathbf{Y}^{T} \mathbf{W}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \left( \mathbf{E} \mathbf{Y}^{T} \mathbf{V} \mathbf{X} \right) \\
C = \frac{1}{(PK)^{2}} \mathbf{T}_{r} \left[ \left( \mathbf{E} \mathbf{Y}^{T} \mathbf{W}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \right)^{T} \\
+ \mathbf{E} \mathbf{Y}^{T} \mathbf{W}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \left( \mathbf{W} \mathbf{W}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \right)^{T} \\
+ \mathbf{E} \mathbf{Y}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \left( \mathbf{E} \mathbf{Y}^{T} \mathbf{V} \mathbf{X} \right) \right], \\
M = \frac{1}{PK} \mathbf{T}_{r} \left[ \mathbf{E} \left( \mathbf{W} \mathbf{W}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \right)^{T} + \mathbf{E} \left( \mathbf{E} \mathbf{Y}^{T} \mathbf{V} \mathbf{X} \right)^{T} \\
+ \mathbf{W} \mathbf{W}^{T} \mathbf{E} \mathbf{X}^{T} \mathbf{X} + \mathbf{E} \mathbf{Y}^{T} \mathbf{V} \mathbf{X} \mathbf{E} \mathbf{X}^{T} \mathbf{X} \right], \\
(30)
\]

where
\[
g(\beta) = \left( A\beta^{3} + B\beta^{2} + C\beta + M\beta \right), \\
given (31)
\]

\[
\frac{\partial g}{\partial \beta} = 4A \left( \beta^{3} + a\beta^{2} + b\beta + c \right), \\
(32)
\]

where \(a = 3B/4A, b = 2C/4A,\) and \(c = M/4A.\)

**Lemma 2.** For solution of general real cubic equation, one uses the following lemma:

\[
f(x) = x^{3} + ax^{2} + bx + c,
\]

\(L = -27c^{2} + 18abc + a^{2}b^{2} - 4a^{3}b^{3} - 4b^{3},\)

where \(D\) is discriminant of \(f(x).\)

Then we have the following:

1. If \(D < 0, f(x)\) has one real root.
2. If \(D \geq 0, f(x)\) has three real roots:
   1. \(D > 0; f(x)\) has three different real roots,
   2. \(D = 0, 6b - 2a^{2} \neq 0; f(x)\) has one single root and one multiple root,
   3. \(D = 0, 6b - 2a^{2} = 0; f(x)\) has one root of three multiplicities.

**Lemma 3.** For given polynomial \(g(\beta)\) given (31) if optimum \(\beta = \{\beta_{i} | g(\beta_{i}) = \min(g(\beta_{1}), g(\beta_{2}), g(\beta_{3})), i \in \{1, 2, 3\}\},\)

where \(\beta_{i}\) is real root of \(\partial g/\partial \beta,\) the optimum \((\beta)\) is the optimal learning rate and this learning process is stable.

**Proof.** To find stable learning range of \(\beta,\) consider that Lyapunov function is

\[
\mathcal{V}_{t} = I_{t}^{2},
\]

\[
\Delta \mathcal{V}_{t} = I_{t+1}^{2} - I_{t}^{2} \quad \text{if} \ \Delta \mathcal{V}_{t} < 0,
\]

and then dynamical system is guaranteed to be stable if \(\Delta \mathcal{V}_{t} < 0;\) that is, \(I_{t+1} - I_{t} = (1/2PK)(A\beta^{3} + B\beta^{2} + C\beta^{2} + M\beta) < 0 (29),\)

where \(\beta\) is learning rate, since in the training process input matrix remains the same during the whole training process. To find the range of \(\beta,\) which satisfy \(g(\beta) = (A\beta^{3} + B\beta^{2} + C\beta^{2} + M\beta) < 0.\) Since \(\partial g/\partial \beta,\) where has at least one real root (Lemma 2) and one of them must give optimum \(g(\beta).\)

Obviously minimum value of \(g(\beta)\) gives the largest reduction in error at two successive times and minimum value is obtained by differentiating (31) with respect to \(\beta\) we have from (32):

\[
\frac{\partial g}{\partial \beta} = 4A \left( \beta^{3} + a\beta^{2} + b\beta + c \right),
\]

(35)

where \(a = 3B/4A, b = 2C/4A,\) and \(c = M/4A.\)

Solving \(\partial g/\partial \beta = 0\) gives \(\beta\) which minimizes error in (5).

5. Simulation Results

To demonstrate the effectiveness and merit of proposed FCPN, simulation results are presented and discussed. Four different nonlinear dynamical systems and one general benchmark problem known as Box-Jenkins model with time series data are considered.

5.1. Performance Criteria. For accessing the performance criteria of FCPN and comparing with Dynamic and Back Propagation Network, we have evaluated various errors as given below. In case of four dynamical models, it is recommended to access criterion such as subset of the following.

Given \(N\) pair of data points \((y(k), \bar{y}(k)),\) where \(y(k)\) is output of system and \(\bar{y}(k)\) is output of controller, the Maximum Absolute Error (MAE) is

\[
J_{\text{MAE}} = J_{\text{max}} = \max_{1 \leq k \leq N} \left| y(k) - \bar{y}(k) \right|,
\]

(36)

the Sum of Squared Errors (SSE) is

\[
J_{\text{SSE}} = \sum_{k=1}^{N} (y(k) - \bar{y}(k))^{2},
\]

(37)

the Mean Squared Error (MSE) is

\[
J_{\text{MSE}} = \frac{1}{N} \sum_{k=1}^{N} (y(k) - \bar{y}(k))^{2},
\]

(38)
Figure 4: (a) Mean Square Error of system (Example 1) using FCPN. (b) Performance of the controller using FCPN algorithm for Example 1.

\[ J_{RMSE} = \sqrt{J_{MSE}}. \]  

(39)

The measure Variance Accounting Factor (VAF) is

\[ J_{VAF} = \left(1 - \frac{\text{Var}(y(k) - \overline{y}(k))}{\text{Var}(y(k))}\right)100\%. \]

(40)

A related measure is the Normalized Mean Squared Error (NMSE):

\[ J_{NMSE} = \frac{\sum_{k=1}^{N} (y(k) - \overline{y}(k))^2}{\sum_{k=1}^{N} (y(k) - \bar{y}(k))^2}, \]

and the Best Fit Rate (BFR) is

\[ J_{BFR} = \left(1 - \frac{\sum_{k=1}^{N} (y(k) - \overline{y}(k))^2}{\sum_{k=1}^{N} (y(k) - \bar{y}(k))^2}\right)100\%. \]

(42)

**Example 1.** The nonlinear dynamical system [24] is governed by the following difference equation:

\[ y(k + 1) = 0.3y(k) + 0.6y(k - 1) + f[u(k)]. \]

(43)

The nonlinear function in the system in this case is \( f(u) = u^3 + 0.3u^2 - 0.4u \), where \( y \) and \( u \) are uniformly distributed in \([-2, 2]\). The objective of Example 1 is to control the system to track reference output given as 250 sample data points. The model has two inputs \( y(k) \) and \( u(k) \) and single output \( y(k+1) \) and the system identification was initially performed with the system input being uniformly distributed over \([-2, 2]\). Training and testing samples contain 100 sample data points. The FCPN controller uses two inputs \( y(k) \) and \( u(k) \) to output \( y(k + 1) \). FCPN model governed by the difference equation \( \overline{y}(k + 1) = 0.3y(k) + 0.6y(k - 1) + N[u(k)] \) was used.

Figures 4(a) and 4(b) show the error \( e(k + 1) = y(k + 1) - \overline{y}(k + 1) \) and outputs of the dynamical system and the FCPN.

**Example 2.** The nonlinear dynamical system [24] is governed by the following difference equation:

\[ y(k + 1) = f[y_p(k), y_p(k - 1)] + u(k), \]

(44)

where

\[ f[y_p(k), y_p(k - 1)] = \left[ y_p(k) y_p(k - 1) \frac{y_p(k) + 2.5}{1 + y_p^2(k) + y_p^2(k - 1)} \right]. \]

(45)

The objective of (44) is to control the system to track reference output given 100 sample data points. The model has two inputs \( y(k) \) and \( u(k) \) and single output \( y(k+1) \) and the system identification was initially performed with the system input being uniformly distributed over \([-2, 2]\). Training and testing samples contain 100 sample data points. The FCPN controller uses two inputs \( y(k) \) and \( u(k) \) to output \( \overline{y}(k + 1) \). FCPN

<table>
<thead>
<tr>
<th>Table 1: Calculated various errors for Example 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN model</td>
</tr>
<tr>
<td>FCPN</td>
</tr>
<tr>
<td>Dynamic</td>
</tr>
<tr>
<td>BPN</td>
</tr>
</tbody>
</table>

As can be seen from the figure, the identification error is small even when the input is changed to a sum of two sinusoids \( u(k) = \sin(2\pi k/250) + \sin(2\pi k/25) \) at \( k = 250 \).

Table 1 includes various calculated errors of different NN models for Example 1. It can be seen from the table that various errors calculated for FCPN are minimum as compared to the DN and BPN.
network discussed earlier is used to identify the system from input-output data and is described by the equation
\[ \tilde{y}(k+1) = N [ y(k) , y(k-1) ] + u(k). \] (46)

For FCPN, the identification process involves the adjustment of the weights of \( N \) using FCL. FCPN identifier needs some prior information concerning the input-output behavior of the system before identification can be undertaken. FCL is used to adjust the weights of the neural network so that the error \( e(k+1) = y(k+1) - \tilde{y}(k+1) \) is minimized as shown in Figure 5(a). The behavior of the FCPN model for (16) is shown in Figure 5(b). The input \( u(k) \) was assumed to be a random signal uniformly distributed in the interval \([-2,2]\).

Table 2 includes various calculated errors of different NN models for Example 2. It can be seen from the table that various errors calculated for FCPN are minimum as compared to the DN and BPN.

**Example 3.** The nonlinear dynamical system [23] is governed by the following difference equation:
\[ y(k+1) = \frac{y(k)}{1 + y(k)^2} + u^3(k), \] (47)

which corresponds to \( f[y(k)] = y(k)/(1 + y(k)^2) \) and \( g[u(k)] = u^3(k) \). FCPN network equation (47) is used to identify the system from input-output data. Weights in the neural networks were adjusted for every instant of discrete time steps.

The objective of (47) is to control the system to track reference output given 100 sample data points. The model has two inputs \( y(k) \) and \( u(k) \) and single output \( y(k+1) \) and the system identification was initially performed with the system input being uniformly distributed over \([-2,2]\]. Training and testing samples contain 250 sample data points. The FCPN controller uses two inputs \( y(k) \) and \( u(k) \) to output \( \tilde{y}(k+1) \). The input data is generated using uniform distribution in interval \([-2,2]\). The function \( f \) obtained by FCPN is used to adjust the weights of the neural network so that the error \( e(k+1) = y(k+1) - \tilde{y}(k+1) \) is minimized as shown in Figure 6(a). In Figure 6(b), the desired outputs of the system as well as the FCPN model are shown and are seen to be indistinguishable.

Table 3 includes various calculated errors of different NN models for Example 3. It can be seen from the table that various errors calculated for FCPN are minimum as compared to the DN and BPN.

**Example 4.** The nonlinear dynamical system [24] is governed by the difference equation (48). Generalized form of this equation is Model IV. In this example, the system is assumed to be of the form
\[ y_p(k+1) = f \left[ y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1) \right], \] (48)
where the unknown function $f$ has the form

$$f[x_1, x_2, x_3, x_4, x_5] = \frac{x_3 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_3^2 + x_2^2}.$$  \hspace{1cm} (49)

In the identification model, which is approximated by FCPN, Figure 7(b) shows the output of the system and the model when the identification procedure carried random input signal uniformly distributed in the interval $[-1, 1]$. The performance of the model is studied and the error $e(k + 1) = N[y_p(k + 1)]$ is minimized as shown in Figure 7(a). In Figure 7(b), the outputs of the system as well as output of FCPN model are shown. Input to the system and the identified model is given by $u(k) = \sin(2nk/250)$ for $k \leq 500$ and $u(k) = 0.8 \sin(2nk/250) + 0.2 \sin(2nk/25)$ for $k > 500$.

Table 4 includes various calculated errors of different NN models for Example 4. It can be seen from the table that various errors calculated for FCPN are minimum as compared to the DN and BPN.

<table>
<thead>
<tr>
<th>NN models</th>
<th>MAE</th>
<th>SSE</th>
<th>MSE</th>
<th>RMSE</th>
<th>NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCPN</td>
<td>30.4348</td>
<td>2.1931</td>
<td>0.0027</td>
<td>0.0524</td>
<td>0.0186</td>
</tr>
<tr>
<td>Dynamic</td>
<td>32.8952</td>
<td>6.4059</td>
<td>0.0080</td>
<td>0.0895</td>
<td>0.0276</td>
</tr>
<tr>
<td>BP</td>
<td>53.1988</td>
<td>13.4174</td>
<td>0.0168</td>
<td>0.1295</td>
<td>0.0844</td>
</tr>
</tbody>
</table>

**Example 5.** In this example we have used Box-Jenkin time series [32], of 296 pairs of data measured from a gas furnace system with single input $u(t)$ being gas flow rate and single output $y(t)$ being CO$_2$ concentration in outlet gas. Training samples and testing samples contained 196 and 100 data points, respectively. The FCPN uses the two inputs, the current state $y(k)$ and the desired state $y^d(k)$, to produce an output which is $\hat{y}(k)$ [33].

Figure 8(a) shows the mean squared control errors of FCPN methods. Figure 8(b) shows the performance of the controller with FCPN algorithm. Result shows that FCPN algorithm enable us to appropriate the approximation using...
fuzzy learning as given in equation (11) of Box Jenkins time series data, based on calculation of BMN.

Table 5 shows BFR (%) various NN models for all Examples 1–5, respectively. It can be observed from Table 5 that the Best Fit Rate found for FCPN is maximum as compared to DN and BPN which shows better performance of the FCPN network for nonlinear system.

6. Conclusions

FCPN is a neural network control method which was developed and presented. It is based on the concept of combining FCL algorithm and CPN. The key ideas explored are the use of the FCL algorithm for training the weights of the instar-outstar. The performances of the FCPN controller based training are tested using four nonlinear dynamical systems and on time series (Box-Jenkins) data and compared with the Dynamic Network and standard Back Propagation algorithm. The comparative performances of the FCPN algorithm and Dynamic Network, such as the number of iterations and performance functions like MAE, MSE, SSE, NMSE, BFR, and so forth, of FCPN and Dynamic Network error are summarized in Tables 1–4. It can be seen that the FCPN algorithm gives minimum errors as compared to the Dynamic Network and standard Back Propagation Network for all four models of nonlinear dynamical systems and Box-Jenkins time series data. Results obtained from FCPN were compared for various errors and it is clearly shown that FCPN works much better for the control of nonlinear dynamical systems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors gratefully acknowledged the financial assistance provided by the All India Council of Technical Education (AICTE) in the form of Research Promotion Scheme (RPS) project in 2012.

References


Submit your manuscripts at http://www.hindawi.com