Soft Comput (2015) 19:3445–3454
DOI 10.1007/s00500-014-1344-3

FOCUS

A hybrid cascade neural network with an optimized pool in each cascade

Ye. Bodyanskiy · O. Tyshchenko · D. Kopaliani

Published online: 27 June 2014

Abstract This paper proposes a new architecture and learning algorithms for a hybrid cascade neural network with pool optimization in each cascade. The proposed system is different from existing cascade systems in its capability to operate in an online mode, which allows it to work with non-stationary and stochastic nonlinear chaotic signals with the required accuracy. Compared to conventional analogs, the proposed system provides computational simplicity and possesses both tracking and filtering capabilities.

Keywords Hybrid system · Learning method · Neo-fuzzy neuron · Cascade network

1 Introduction

Today artificial neural networks (ANNs) are successfully used in a wide range of data processing problems (when data can be presented either in the form of "object-property" tables or in the form of time series, often produced by non-stationary nonlinear stochastic or chaotic systems). The advantages ANNs have over other existing approaches derive from their universal approximating capabilities and learning capacities.

Communicated by V. Loia.

Y. Bodyanskiy · O. Tyshchenko (✉) · D. Kopaliani
Control Systems Research Laboratory, Kharkiv National University of Radioelectronics, Lenina av., 14, Kharkiv 61166, Ukraine
e-mail: lehatish@gmail.com

Y. Bodyanskiy
e-mail: bodya@kture.kharkov.ua

D. Kopaliani
e-mail: daria.kopaliani@gmail.com

Conventionally, "learning" is defined as a process of adjusting synaptic weights using an optimization procedure that involves searching for the extremum of the given learning criterion. The learning process quality can be improved by adjusting a network topology along with its synaptic weights (Haykin 1999; Cichocki and Unbehauen 1993). This idea is the foundation of evolving computational intelligence systems (Kasabov 2001, 2003, 2007; Kasabov and Song 2002; Kasabov et al. 2005; Lughofer 2011; Angelov and Filev 2004, 2005; Angelov and Kasabov 2005; Angelov and Lughofer 2008; Angelov and Zhou 2006, 2008; Angelov et al. 2004, 2005, 2006, 2007, 2008, 2010; Lughofer and Klement 2003, 2004; Lughofer and Bodenhofer 2006; Lughofer and Guardiola 2008a,b; Lughofer and Kindermann 2008, 2010; Lughofer and Angelov 2009, 2011; Lughofer 2006, 2008a,b,c, 2010a,b; Lughofer et al. 2003, 2004, 2005, 2007, 2009). Under this approach, the best known architectures are DENFIS by Kasabov and Song (2002), Angelov and Filev (2004) and FLEXFIS by Lughofer (2008c). These systems are actually five-layer Takagi-Sugeno networks and evolution is fulfilled in the fuzzification layer. These networks can process data in an online mode where a clusterization task is solved in the antecedent (unsupervised learning) and consequent parameter tuning is performed with supervised learning with the help of the exponentially weighted recurrent least-squares method. Though these systems are characterized by high approximating properties, they can also process non-stationary signals and they require big-volume samples to tune parameters. A clusterization procedure cannot be optimized in terms of speed like all the algorithms which are based on self-learning. A rather interesting class of computational intelligence systems where an architecture is evolving during a learning process is cascade-correlation neural networks (Fahlman and Lebiere 1990; Prechelt 1997; Schalkoff 1997; Avedjan et al. 1999) due to their high efficiency degree

and learning simplicity of both synaptic weights and a network topology. Such a network starts off with a simple architecture consisting of a pool (ensemble) of neurons which are trained independently (the first cascade). Each neuron in the pool can have a different activation function and/or a different learning algorithm. The neurons in the pool do not interact with each other while trained. After all the neurons in the pool of the first cascade have their weights adjusted, the best neuron with respect to a learning criterion forms the first cascade and its synaptic weights can no longer be adjusted. Then the second cascade is formed usually out of similar neurons in the training pool. The only difference is that the neurons which are trained in the pool of the second cascade have an additional input (and, therefore, an additional synaptic weight) which is an output of the first cascade. Similar to the first cascade, the second cascade will eliminate all but one neuron showing the best performance whose synaptic weights will thereafter be fixed. Neurons of the third cascade have two additional inputs, namely the outputs of the first and second cascades. The evolving network continues to add new cascades to its architecture until it reaches the desired quality of problem solving over the given training set.

Authors of the most popular cascade neural network, CasCorLA, S. E. Fahlman and C. Lebiere, used elementary Rosenblatt perceptrons with traditional sigmoidal activation functions and adjusted synaptic weights using the Quickprop-algorithm (a modification of the δ-learning rule). Since the output signal of such neurons is non-linearly dependent on its synaptic weights, the learning rate cannot be increased for such neurons. In order to avoid multi-epoch learning (Bodyanskiy et al. 2008, 2009, 2011a,b,c; Bodyanskiy and Viktorov 2009a,b; Bodyanskiy and Kolodyazhniy 2010), different types of neurons (with outputs that depend linearly on synaptic weights) should be used as network nodes. This would allow to use optimal learning algorithms in terms of speed and process data as it is an input to the network. However, if the network learns in an online mode, it is impossible to determine the best neuron in the pool. While working with non-stationary objects, one neuron of the training pool can be identified as the best for one part of the training set, but not for the others. Thus we suggest that all the neurons in the training pool should be retained and a certain optimization procedure (generated according to a general network quality criterion) should be used to determine an output of the cascade. In this paper, we try to create such a hybrid neural network with an optimized neuron pool in each cascade.

2 An optimized cascade neural network architecture

The architecture of the desired hybrid neural network with an optimized pool of neurons in each cascade is shown in Fig. 1.

An input of the network (the so-called "receptive layer") is a vector signal

$$x(k) = (x_1(k), x_2(k), \ldots, x_n(k))^T,$$

where $k = 1, 2, \ldots,$ is either the quantity of samples in the "object-property" table or the current discrete time. These

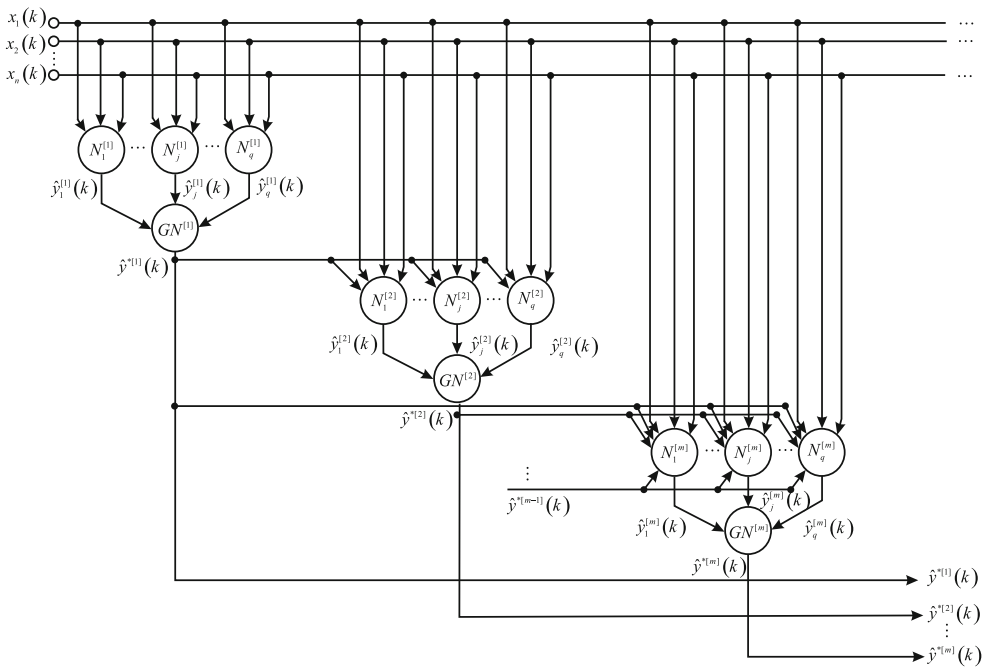

Fig. 1 The optimized cascade neural network architecture

signals are fed to the inputs of each neuron in the network $N_j^{[m]}$ ($j = 1, 2, \ldots, q$ is the quantity of neurons in the training pool, $m = 1, 2, \ldots$ is the number of the cascade), that produces outputs $\hat{y}_j^{[m]}(k)$. These outputs are then combined with a generalizing neuron $GN^{[m]}$, which generates an optimal output $\hat{y}^{*[m]}(k)$ of the m-th cascade. While the input of the neurons in the first cascade is $x(k)$ (which may also contain a threshold value $x_0(k) \equiv 1$), neurons in the second cascade have an additional input for the generated signal $\hat{y}^{*[1]}(k)$, neurons in the third cascade have two additional inputs $\hat{y}^{*[1]}(k)$, $\hat{y}^{*[2]}(k)$, and neurons in the m-th cascade have $(m-1)$ additional inputs $\hat{y}^{*[1]}(k)$, $\hat{y}^{*[2]}(k), \ldots, \hat{y}^{*[m-1]}(k)$. The new cascades become a part of the network during a training process when it becomes clear that the current cascades do not provide the desired quality.

3 Training elementary Rosenblatt perceptrons in a cascade neural network

For now, let us assume that the j-th node in the m-th cascade is an elementary Rosenblatt perceptron with the activation function

$$0 < \sigma_j^{[m]}(\gamma_j{}^{[m]} u_j^{[m]}) = \frac{1}{1 + e^{-\gamma_j^{[m]} u_j^{[m]}}} < 1,$$

where $u_j^{[m]}$ is an internal activation signal of the j-th neuron in the m-th cascade, and $\gamma_j^{[m]}$ is a gain parameter. In such a case, the neurons in the pool of the first cascade will have the following outputs:

$$\hat{y}_j^{[1]} = \sigma_j^{[1]}\left(\gamma_j^{[1]} \sum_{i=0}^{n} w_{ji}^{[1]} x_i\right) = \sigma_j^{[1]}\left(\gamma_j^{[1]} w_j^{[1]T} x\right),$$

where $w_{ji}^{[1]}$ is the i-th synaptic weight for the j-th neuron in the first cascade. Outputs of the second cascade

$$\hat{y}_j^{[2]} = \sigma_j^{[2]}\left(\gamma_j^{[2]}\left(\sum_{i=0}^{n} w_{ji}^{[2]} x_i + w_{j,n+1}^{[2]} \hat{y}^{*[1]}\right)\right),$$

outputs of the m-th cascade

$$\begin{aligned}\hat{y}_j^{[m]} &= \sigma_j^{[m]}\left(\gamma_j^{[m]}\left(\sum_{i=0}^{n} w_{ji}^{[m]} x_i + w_{j,n+1}^{[m]} \hat{y}^{*[1]}\right.\right.\\ &\quad \left.\left. + w_{j,n+2}^{[m]} \hat{y}^{*[2]} + \cdots + w_{j,n+m-1}^{[m]} \hat{y}^{*[m-1]}\right)\right)\\ &= \sigma_j^{[m]}\left(\gamma_j^{[m]} \sum_{i=0}^{n+m-1} w_{ji}^{[m]} x_j^{[m]}\right) = \sigma_j^{[m]}\left(w_j^{[m]T} x^{[m]}\right),\end{aligned}$$

where $x^{[m]} = \left(x^T, \hat{y}^{*[1]}, \ldots, \hat{y}^{*[m-1]}\right)^T$.

Thus the cascade network, using Rosenblatt perceptrons as nodes and containing m cascades, is dependent on $(m(n+2) + \sum_{p=1}^{m-1} p)$ parameters including the gain parameters $\gamma_j^{[p]}$, $p = 1, 2, \ldots, m$. We use a conventional quadratic function as a learning criterion

$$\begin{aligned}E_j^{[m]} &= \frac{1}{2}\left(e_j^{[m]}(k)\right)^2 = \frac{1}{2}\left(y(k) - \hat{y}_j^{[m]}(k)\right)^2\\ &= \frac{1}{2}\left(y(k) - \sigma_j^{[m]}\left(\gamma_j^{[m]} w_j^{[m]T} x^{[m]}(k)\right)\right)^2, \qquad (1)\end{aligned}$$

where $y(k)$ is a reference signal. The gradient optimization of the criterion (1) with respect to $w_j^{[m]}$ is

$$\begin{aligned}w_j^{[m]}(k+1) &= w_j^{[m]}(k) + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1)\gamma_j^{[m]}\\ &\quad \times \hat{y}_j^{[m]}(k+1)\left(1 - \hat{y}_j^{[m]}(k+1)\right) x^{[m]}(k+1)\\ &= w_j^{[m]}(k) + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1)\\ &\quad \times \gamma_j^{[m]} J_j^{[m]}(k+1), \qquad (2)\end{aligned}$$

(here $\eta_j^{[m](k+1)}$ is a learning rate parameter), and minimization of (1) with respect to $\gamma_j^{[m]}$ can be performed using the Kruschke–Movellan algorithm (Kruschke and Movellan 1991):

$$\begin{aligned}\gamma_j^{[m]}(k+1) &= \gamma_j^{[m]}(k) + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1)\hat{y}_j^{[m]}(k+1)\\ &\quad \times \left(1 - \hat{y}_j^{[m]}(k+1)\right) u_j^{[m]}(k+1). \qquad (3)\end{aligned}$$

Combining (2) and (3), we obtain a general learning algorithm for the j-th neuron in the m-th cascade:

$$\begin{aligned}\begin{pmatrix} w_j^{[m]}(k+1) \\ \gamma_j^{[m]}(k+1) \end{pmatrix} &= \begin{pmatrix} w_j^{[m]}(k) \\ \gamma_j^{[m]}(k) \end{pmatrix}\\ &\quad + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1)\hat{y}_j^{[m]}(k+1)\\ &\quad \times (1 - \hat{y}_j^{[m]}(k+1))\begin{pmatrix} \gamma_j^{[m]} x^{[m]}(k+1) \\ u_j^{[m]}(k+1) \end{pmatrix},\end{aligned}$$

or, introducing new variables, in a more compact form:

$$\begin{aligned}&\tilde{w}_j^{[m]}(k+1)\\ &= \tilde{w}_j^{[m]}(k) + \eta_j^{[m]}(k+1)\\ &\quad \times e_j^{[m]}(k+1)\hat{y}_j^{[m]}(k+1)(1 - \hat{y}_j^{[m]}(k+1))\tilde{x}^{[m]}(k+1)\\ &= \tilde{w}_j^{[m]}(k) + \eta_j^{[m]}(k+1) e_j^{[m]}(k+1)\tilde{J}_j^{[m]}(k+1).\end{aligned}$$

Weight adjustment can be improved by introducing a momentum term to the learning process (Chan and Fallside 1987; Almeida and Silva 1990; Holmes and Veitch 1991), so that instead of the learning criterion (1) we use the function

$$\begin{aligned}E_j^{[m]}(k) &= \tfrac{\eta}{2}(e_j^{[m]}(k))^2\\ &\quad + \tfrac{1-\eta}{2}\|\tilde{w}_j^{[m]}(k) - \tilde{w}_j^{[m]}(k-1)\|^2, 0 < \eta \leqslant 1 \qquad (4)\end{aligned}$$

and the algorithm is

$$\begin{aligned}\tilde{w}_j^{[m]}(k+1) &= \tilde{w}_j^{[m]}(k) \\ &+ \eta_j^{[m]}(k+1)(\eta e_j^{[m]}(k+1)\tilde{J}_j^{[m]}(k+1) \\ &+ (1-\eta)(\tilde{w}_j^{[m]}(k) - \tilde{w}_j^{[m]}(k-1))), \end{aligned} \quad (5)$$

which is a modification of the Silva–Almeida procedure (Almeida and Silva 1990).

Using the approach suggested in (Bodyanskiy et al. 2001b, 2003b), we can introduce tracking and filtering properties. So a final version of the algorithm is

$$\begin{cases} \tilde{w}_j^{[m]}(k+1) = \tilde{w}_j^{[m]}(k) \\ \quad + \dfrac{\eta e_j^{[m]}(k+1)\tilde{J}_j^{[m]}(k+1)}{r_j^{[m]}(k+1)} \\ \quad + \dfrac{(1-\eta)(\tilde{w}_j^{[m]}(k) - \tilde{w}_j^{[m]}(k-1))}{r_j^{[m]}(k+1)}, \\ r_j^{[m]}(k+1) = r_j^{[m]}(k) + ||\tilde{J}_j^{[m]}(k+1)||^2 \\ \quad - ||\tilde{J}_j^{[m]}(k-s)||^2 \end{cases} \quad (6)$$

where s is a sliding window size.

It is interesting that when $s = 1$, $\eta = 1$, we get a non-linear version of the well-known Kaczmarz–Widrow–Hoff algorithm (Kaczmarz 1937, 1993; Hoff and Widrow 1960):

$$\tilde{w}_j^{[m]}(k+1) = \tilde{w}_j^{[m]}(k) + \frac{e_j^{[m]}(k+1)\tilde{J}_j^{[m]}(k+1)}{\|\tilde{J}_j^{[m]}(k+1)\|^2},$$

which is widely used in artificial neural networks' learning and characterized by high convergence rate.

4 Training neo-fuzzy neurons in a cascade neural network

A low learning rate of Rosenblatt perceptrons along with difficulty in interpreting results (inherent to all ANNs in general) encourages us to search for alternative approaches to the synthesis of evolving systems in general and cascade neural networks in particular. High interpretability and transparency along with good approximation capabilities and ability to learn are the main features of the neuro-fuzzy systems (Jang et al. 1997), which are the foundation of hybrid artificial intelligence systems.

In Bodyanskiy and Viktorov (2009a,b), Bodyanskiy and Kolodyazhniy (2010) hybrid cascade systems were introduced which used neo-fuzzy neurons (Kusanagi et al. 1992; Uchino and Yamakawa 1997; Miki and Yamakawa 1999) as network nodes, allowing them to significantly increase a rate of synaptic weight adjustment. A neo-fuzzy neuron (NFN) is a non-linear system providing the following mapping:

$$\hat{y} = \sum_{i=1}^{n} f_i(x_i),$$

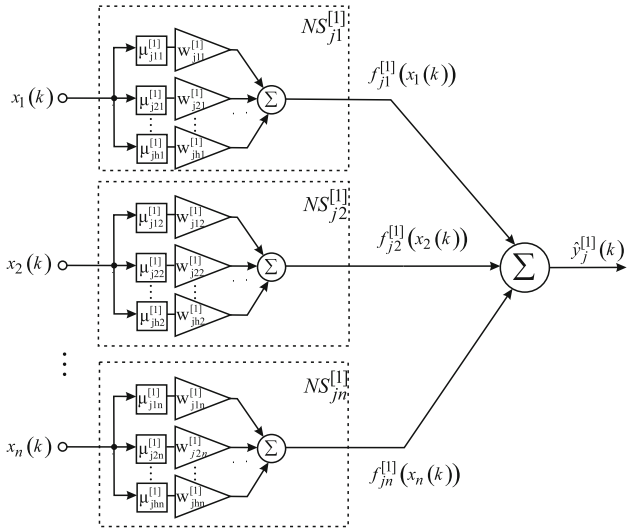

Fig. 2 A neo-fuzzy neuron of the first cascade

where x_i is the ith input ($i = 1, 2, \ldots, n$), $\hat{y}$ is an output of the neo-fuzzy neuron. Structural units of the neo-fuzzy neuron are non-linear synapses NS_i which transform input signals in the following way:

$$f_i(x_i) = \sum_{l=1}^{h} w_{li}\mu_{li}(x_i),$$

where w_{li} is the lth synaptic weight of the ith non-linear synapse, $l = 1, 2, \ldots, h$ is the total quantity of synaptic weights and, therefore, membership functions $\mu_{li}(x_i)$ in the synapse. So NS_i implements fuzzy inference in the form

IF x_i IS X_{li} THEN THE OUTPUT IS w_{li},

where X_{li} is a fuzzy set with a membership function μ_{li}, w_{li} is a singleton (a synaptic weight in a consequent). It can be seen that, in fact, the non-linear synapse implements the zero-order Takagi-Sugeno fuzzy inference.

Figure 2 shows the jth neo-fuzzy neuron of the first cascade (according to the network topology shown in Fig. 1).

$$\begin{cases} \hat{y}_j^{[1]}(k) = \sum_{i=1}^{n} f_{ji}^{[1]}(x_i(k)) \\ \quad = \sum_{i=1}^{n}\sum_{l=1}^{h} w_{jli}^{[1]}\mu_{jli}^{[1]}(x_i(k)), \\ \text{IF } x_i(k) \text{ IS } X_{jli} \text{ THEN} \\ \text{THE OUTPUT IS } w_{jli}^{[1]}. \end{cases} \quad (7)$$

Authors of the neo-fuzzy neuron (Kusanagi et al. 1992; Uchino and Yamakawa 1997; Miki and Yamakawa 1999) used a traditional triangular structure meeting the conditions of Ruspini partitioning (unity partitioning) as membership functions:

$$\mu_{jli}^{[1]}(x_i)=\begin{cases}\frac{x_i-c_{j,l-1,i}^{[1]}}{c_{jli}^{[1]}-c_{j,l-1,i}^{[1]}}, & \text{if } x_i\in[c_{j,l-1,i}^{[1]},c_{jli}^{[1]}],\\ \frac{c_{j,l+1,i}^{[1]}-x_i}{c_{j,l+1,i}^{[1]}-c_{jli}^{[1]}}, & \text{if } x_i\in[c_{jli}^{[1]},c_{j,l+1,i}^{[1]}]\\ 0, & \text{otherwise,}\end{cases}\tag{8}$$

where $c_{jli}^{[1]}$ are center parameters of membership functions over the interval [0,1] which are relatively arbitrarily chosen (usually evenly distributed) where, naturally, $0\le x_i\le 1$. This choice ensures that the input signal x_i activates only two neighboring membership functions, and their sum is always equal to 1, which means that

$$\mu_{jli}^{[1]}(x_i)+\mu_{j,l+1,i}^{[1]}(x_i)=1$$

and

$$f_{ji}^{[1]}(x_i)=w_{jli}^{[1]}\mu_{jli}^{[1]}(x_i)+w_{j,l+1,i}^{[1]}\mu_{j,l+1,i}^{[1]}(x_i).$$

Approximating capabilities can be improved using cubic splines (Bodyanskiy and Viktorov 2009b) instead of triangular membership functions:

$$\mu_{jli}^{[1]}(x_i)=\begin{cases}\frac{1}{4}\left(2+3\frac{2x_i-c_{jli}^{[1]}-c_{j,l-1,i}^{[1]}}{c_{jli}^{[1]}-c_{j,l-1,i}^{[1]}}\right.\\ \left.-\left(\frac{2x_i-c_{jli}^{[1]}-c_{j,l-1,i}^{[1]}}{c_{jli}^{[1]}-c_{j,l-1,i}^{[1]}}\right)^3\right),\\ \text{if } x_i\in[c_{j,l-1,i}^{[1]},c_{jli}^{[1]}],\\ \frac{1}{4}\left(2-3\frac{2x_i-c_{j,l+1,i}^{[1]}-c_{jli}^{[1]}}{c_{j,l+1,i}^{[1]}-c_{jli}^{[1]}}\right.\\ \left.+\left(\frac{2x_i-c_{j,l+1,i}^{[1]}-c_{jli}^{[1]}}{c_{j,l+1,i}^{[1]}-c_{jli}^{[1]}}\right)^3\right),\\ \text{if } x_i\in[c_{jli}^{[1]},c_{j,l+1,i}^{[1]}],\\ 0, \text{otherwise}\end{cases}\tag{9}$$

or B-splines (Bodyanskiy and Kolodyazhniy 2010):

$$\mu_{jli}^{g[1]}(x_i)=\begin{cases}\left.\begin{array}{l}1, \text{ if } x_i\in[c_{jli}^{[1]},c_{j,l+1,i}^{[1]}]\\ 0, \text{ otherwise,}\end{array}\right\} & \text{for } g=1,\\ \frac{x_i-c_{jli}^{[1]}}{c_{j,l+g-1,i}^{[1]}-c_{jli}^{[1]}}\mu_{jli}^{g-1,[1]}(x_i)\\ \quad+\frac{c_{j,l+g,i}^{[1]}-x_i}{c_{j,l+g,i}^{[1]}-c_{j,l+1,i}^{[1]}}\\ \quad\times\mu_{j,l+1,i}^{g-1,[1]}(x_i)\Big\} & \text{for } g>1,\end{cases}\tag{10}$$

where $\mu_{jli}^{g[1]}(x_i)$ is the l-th spline of the g-th order. It can be seen that when $g=2$ we obtain triangular membership functions in (8). B-splines also ensure unity partitioning, but in a general case they can activate an arbitrary number of membership functions, beyond the interval [0,1], which might be useful for subsequent cascades (i.e., those following the first).

It is clear that other structures such as polynomial harmonic functions, wavelets, orthogonal functions, etc. can be used as membership functions for non-linear synapses.

It is still unclear which of the functions can provide the best results, which is why the idea of using not a single neuron, but a pool of neurons with different membership and activation functions seems promising.

Similar to (7) we can determine outputs for the remaining cascades: outputs of the neurons in the second cascade

$$\hat{y}_j^{[2]}=\sum_{i=1}^{n}\sum_{l=1}^{h}w_{jli}^{[2]}\mu_{jli}^{[2]}(x_i)+\sum_{l=1}^{h}w_{jl,n+1}^{[2]}\mu_{jl,n+1}^{[2]}(\hat{y}^{*[1]}),$$

outputs of the mth cascade

$$\hat{y}_j^{[m]}=\sum_{i=1}^{n}\sum_{l=1}^{h}w_{jli}^{[m]}\mu_{jli}^{[m]}(x_i)+\sum_{p=n+1}^{n+m-1}\sum_{l=1}^{h}w_{jlp}^{[m]}\mu_{jlp}^{[m]}(\hat{y}^{*[p-n]}).$$

Thus, the cascade network formed with the neo-fuzzy neurons, consisting of m cascades, contains $h\left(n+\sum_{p=1}^{m-1}p\right)$ parameters. Introducing a vector of membership functions for the j-th neo-fuzzy neuron in the m-th cascade,

$$\mu_j^{[m]}(k)=(\mu_{j11}^{[m]}(x_1(k)),\ldots,\mu_{jh1}^{[m]}(x_1(k)),\mu_{j12}^{[m]}(x_2(k)),\ldots,\mu_{jh2}^{[m]}(x_2(k)),\ldots,\mu_{jli}^{[m]}(x_i(k)),\ldots,\mu_{jhn}^{[m]}(x_n(k)),\mu_{j1,n+1}^{[m]}(\hat{y}^{*[1]}(k)),\ldots,\mu_{jh,n+m-1}^{[m]}(\hat{y}^{*[m-1]}(k)))^T$$

and a corresponding vector of synaptic weights,

$$w_j^{[m]}=(w_{j11}^{[m]},\ldots,w_{jh1}^{[m]},w_{j12}^{[m]},\ldots,w_{jh2}^{[m]},\ldots,w_{jli}^{[m]},\ldots,w_{jhn}^{[m]},w_{j1,n+1}^{[m]},\ldots,w_{jh,n+m-1}^{[m]})^T,$$

we obtain an output

$$\hat{y}_j^{[m]}(k)=w_j^{[m]T}\mu_j^{[m]}(k).$$

The learning criterion (1) for this case will be

$$E_j^{[m]}(k)=\frac{1}{2}(e_j^{[m]}(k))^2=\frac{1}{2}(y(k)-w_j^{[m]T}\mu_j^{[m]}(k))^2\tag{11}$$

and its minimization can be reached by using a "sliding window" modification of the procedure (Bodyanskiy et al. 1986):

$$\begin{cases}w_j^{[m]}(k+1)=w_j^{[m]}(k)\\ \quad+\frac{e_j^{[m]}(k+1)\mu_j^{[m]}(k+1)}{r_j^{[m]}(k+1)}\\ r_j^{[m]}(k+1)=r_j^{[m]}(k)+||\mu_j^{[m]}(k+1)||^2\\ \quad-\|\mu_j^{[m]}(k-s)\|^2\end{cases}\tag{12}$$

or when $s=1$ (Bodyanskiy et al. 2003a):

$$w_j^{[m]}(k+1)=w_j^{[m]}(k)+\frac{e_j^{[m]}(k+1)\mu_j^{[m]}(k+1)}{\|\mu_j^{[m]}(k+1)\|^2},$$

which reduces to the one-step-optimal Kaczmarz–Widrow–Hoff algorithm. Its clear that one could use other algorithms instead of (12), for example, the exponentially weighted recurrent least squares method (EWRLSM) that is used in DENFIS (Kasabov and Song 2002), eTS (Angelov and Filev 2004) and FLEXFIS (Angelov et al. 2005; Lughofer 2008c). But one should remember that EWRLSM may be unstable when a forgetting factor is rather low. When using the criterion with the momentum term (3) instead of (11) we obtain a final learning algorithm for the neo-fuzzy neuron:

$$\begin{cases} w_j^{[m]}(k+1) = w_j^{[m]}(k) \\ \quad + \left(\frac{\eta e_j^{[m]}(k+1)\mu_j^{[m]}(k+1)}{r_j^{[m]}(k+1)} \right. \\ \quad \left. + \frac{(1-\eta)(w_j^{[m]}(k)-w_j^{[m]}(k-1))}{r_j^{[m]}(k+1)} \right), \\ r_j^{[m]}(k+1) = r_j^{[m]}(k) + ||\mu_j^{[m]}(k+1)||^2 \\ \quad -\|\mu_j^{[m]}(k-s)\|^2. \end{cases} \tag{13}$$

It should be kept in mind that, since a NFN output is linearly dependent on its synaptic weights, we can use any adaptive linear identification algorithm (Ljung 1999) (second-order recursive least square methods, robust, ignoring outdated information, etc.), which allows us to process non-stationary signals in an online mode.

5 Optimization of the pool output

Outputs generated by neurons in each pool are combined with the corresponding neuron $GN^{[m]}$, the output accuracy of which $\hat{y}^{*[m]}(k)$ must be higher than the accuracy of any output $\hat{y}_j^{[m]}(k)$. This task can be solved with the help of the neural networks ensembles approach. Although the well-known algorithms are not designated for working in an online mode, the adaptive generalizing forecasting could be used in this case (Bodyanskiy et al. 1983, 1989, 1999, 2001a; Bodyanskiy and Pliss 1990; Bodyanskiy and Vorobyov 2000).

Let us introduce a vector of pool inputs for the m-th cascade:

$$\hat{y}^{[m]}(k) = (\hat{y}_1^{[m]}(k), \hat{y}_2^{[m]}(k), \ldots, \hat{y}_q^{[m]}(k))^T;$$

then an optimal output of the neuron $GN^{[m]}$, which is in essence an adaptive linear associator (Cichocki and Unbehauen 1993; Haykin 1999), can be defined as

$$\hat{y}^{*[m]}(k) = \sum_{j=1}^{q} c_j^{[m]} \hat{y}_j^{[m]}(k) = c^{[m]T} \hat{y}^{[m]}(k)$$

with additional restrictions on unbiasedness

$$\sum_{j=1}^{q} c_j^{[m]} = E^T c^{[m]} = 1, \tag{14}$$

where $c^{[m]} = (c_1^{[m]}, c_2^{[m]}, \ldots, c_q^{[m]})^T$, $E = (1, 1, \ldots, 1)^T$ are $(q \times 1)$-vectors.

Introducing a learning criterion on a sliding window

$$E^{[m]}(k) = \frac{1}{2} \sum_{\tau=k-s+1}^{k} (y(\tau) - \hat{y}^{*[m]}(\tau))^2$$
$$= \frac{1}{2} \sum_{\tau=k-s+1}^{k} (y(\tau) - c^{[m]T} \hat{y}^{[m]}(\tau))^2,$$

taking into account the constraints (14), the Lagrangian function will be

$$L^{[m]}(k) = E^{[m]}(k) + \lambda(1 - E^T c^{[m]}) \tag{15}$$

where λ is an undetermined Lagrange multiplier.

Direct minimization of (15) with respect to $c^{[m]}$ gives

$$\begin{cases} \hat{y}^{*[m]}(k+1) = \frac{\hat{y}^{[m]T}(k+1)P^{[m]}(k+1)E}{E^T P^{[m]}(k+1)E}, \\ P^{[m]}(k+1) = \left(\sum_{\tau=k-s+2}^{k+1} \hat{y}^{[m]}(\tau)\hat{y}^{[m]T}(\tau) \right)^{-1} \end{cases} \tag{16}$$

or in a recurrent form:

$$\begin{cases} \tilde{P}^{[m]}(k+1) = P^{[m]}(k) \\ \quad - \frac{P^{[m]}(k)\hat{y}^{[m]}(k+1)\hat{y}^{[m]T}(k+1)P^{[m]}(k)}{1+\hat{y}^{[m]T}(k+1)P^{[m]}(k)\hat{y}^{[m]}(k+1)}, \\ P^{[m]}(k+1) = \tilde{P}^{[m]}(k+1) \\ \quad + \frac{\tilde{P}^{[m]}(k+1)\hat{y}^{[m]}(k-s+1)\hat{y}^{[m]T}(k-s+1)\tilde{P}^{[m]}(k+1)}{1-\hat{y}^{[m]T}(k-s+1)\tilde{P}^{[m]}(k+1)\hat{y}^{[m]}(k-s+1)}, \\ \hat{y}^{*[m]}(k+1) = \frac{\hat{y}^{[m]T}(k+1)P^{[m]}(k+1)E}{E^T P^{[m]}(k+1)E}. \end{cases} \tag{17}$$

when $s = 1$, ratios (16) and (17) take on an extremely simple form:

$$\hat{y}^{*[m]}(k+1) = \frac{\hat{y}^{[m]T}(k+1)\hat{y}^{[m]}(k+1)}{E^T \hat{y}^{[m]}(k+1)}$$
$$= \frac{||\hat{y}^{[m]}(k+1)||^2}{E^T \hat{y}^{[m]}(k+1)} = \frac{\sum_{j=1}^{q} (\hat{y}_j^{[m]}(k+1))^2}{\sum_{j=1}^{q} \hat{y}_j^{[m]}(k+1)}. \tag{18}$$

It is important to note that training both neo-fuzzy neurons and neuron-generalizers can be organized in an online adaptive mode. In this way, the neurons' weights of all previous cascades are not frozen, but they are constantly adjusted, and the number of cascades can both increase and decrease in real time, which distinguishes the proposed neural network from other well-known cascade systems.

6 Experimental results

Considering a comparison with the evolving approaches eTS (Angelov and Filev 2004), Simp_eTS (Angelov and Filev 2005), SAFIS (Rong et al. 2006), MRAN (Yingwei et al. 1997), RANEKF (Kadirkamanathan and Niranjan 1993), and

Table 1 FLEXFIS, eTS, Simp_eTS, SAFIS, MRAN, RANEKF and Cascade NN

Method	$RMSE_{test}$	# of rules/neurons
eTS	0.212	49
Simp_eTS	0.0225	22
SAFIS	0.0221	17
MRAN	0.0271	22
RANEKF	0.0297	35
FLEXFIS (A)	0.0176	5
FLEXFIS (B)	0.0171	8
Cascade NN	0.0104	8

A nonlinear dynamic system identification problem

FLEXFIS (Angelov et al. 2005; Lughofer 2008c), the following nonlinear dynamic system identification example was applied (the results of these methods are reported in (Angelov and Zhou 2006)):

$$y(n+1) = \frac{y(n)y(n-1)(y(n)-0.5)}{1+y^2(n)+y^2(n-1)} + u(n), \tag{19}$$

where $u(n) = \sin(2\pi/25)$, $y(0) = 0$, and $y(1) = 0$. For the incremental and evolving training procedures, 5,000 samples were created starting with $y(0) = 0$, and further, 200 test samples were created for eliciting the root mean-squared error (RMSE) on these samples as a reliable estimator for the generalized prediction error.

Regarding a comparison with the evolving fuzzy modelling approaches DENFIS (Kasabov and Song 2002) and eTS (Angelov and Filev 2004) and its extension exTS (Angelov and Zhou 2006) for the minimum-input-minimum-output Takagi–Sugeno model case, FLEXFIS was used for predicting the Mackey–Glass chaotic time series, given by (the results on the first method are reported in Kasabov and Song (2002), while for the other two are reported in Angelov and Zhou (2006)):

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t), \tag{20}$$

where $x(0) = 1.2$, $\tau = 17$, and $x(t) = 0$ for $t < 0$. The task is to predict $x(t+85)$ from the input vectors $[x(t-18)x(t-12)x(t-6)x(t)]$ for any value of t. 3,000 training samples were collected for t in the interval [201, 3,200]. 500 test samples in the interval [5,001, 5,500] were collected to elicit the NDEI on unseen samples, which is the RMSE divided by the standard deviation of the target series (Table 1).

It should be mentioned that the proposed cascade NN works in an online mode (unlike the other systems in Table 2).

The electric load data were provided by a local supplier from Kharkiv, Ukraine. The data describe hourly electric load in that region in 2007 (8,760 samples). Sampling time of these data is one hour. 6,132 data points were used for training, and

Table 2 FLEXFIS, eTS, Simp_eTS, SAFIS, MRAN, RANEKF and Cascade NN

Method	$NDEI_{test}$	# of rules/neurons
eTS	0.372	9
exTS	0.361	9
DENFIS	0.276	58
FLEXFIS (A)	0.206	69
FLEXFIS (B)	0.157	89
Cascade NN	0.018	8

Chaotic time series prediction

2,628 data points were used for testing. The forecast made by the cascade neural network was one hour ahead.

Input variables to the cascade neural network were the load a week ago and yesterday at the same hour as predicted, the load an hour ago, the current load, the change in load from the previous to the current hour, and the number of the current hour within the current year (6 inputs altogether).

The cascade neural network used for prediction contained 4 cascades (3 neo-fuzzy neurons in each cascade with 5 membership functions per input in each neo-fuzzy neuron). After training the network gave 0.02165 mean squared error (MSE) of prediction. In Fig. 3, the forecast for 3 weeks at the end of March and the beginning of April 2007 is shown (Table 3).

This period includes Easter holidays and the change-over from the winter to summer time, so it is characterized by less regularity of electricity consumption than most other weeks throughout a year. For this period alone, the cascade network provided a prediction with MSE = 0.02886. For comparison, for July 2007 MSE = 0.0223, because there were no holidays during this month. These results are about 40 % more accurate than those previously obtained for the same time series by RBFN and MLP (Table 4).

The proposed cascade NN was also used for the Narendra time series prediction (Narendra and Parthasarathy 1990).

$$\begin{aligned} &y(k+1) = \frac{y(k)}{1+y^2(k)} + f(u(k)),\\ &f(u(k)) = u^3(k),\\ &u(k) = \sin(\pi k/250), \quad \text{if } k < 500\\ &u(k) = 0.8\sin(\pi k/250) + 0.2\sin(\pi k/25),\\ &\text{if } k \geq 500. \end{aligned} \tag{21}$$

The number of inputs is 6, the number of cascades is 5, and the number of neurons in each cascade is 3 (Table 5).

7 Summary

This paper proposes a new architecture and learning algorithms for a hybrid cascade neural network with pool optimization in each cascade. The proposed system is different from existing cascade systems in its capability to operate in an

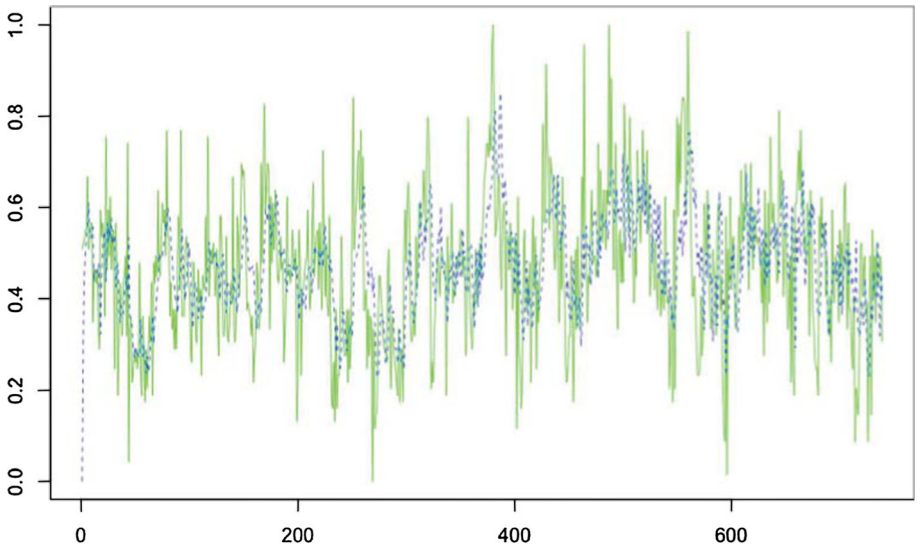

Fig. 3 The electric energy forecast

Table 3 Cascades' forecasting accuracy

Neuron	Casc. 1	Casc. 2	Casc. 3	Casc. 4
Neuron 1	0.02707	0.02421	0.02238	0.02248
Neuron 2	0.02892	0.02538	0.02315	0.02249
Neuron 3	0.02989	0.02946	0.02886	0.02851

Table 4 Forecasting results

MSE	Cascade network	RBFN	MLP
MSE_{trn}	0.02165	0.03172	0.04311
MSE_{chk}	0.02474	0.03422	0.04564

Table 5 The Narendra time series prediction

Neur.	Casc. 1	Casc. 2	Casc. 3	Casc. 4	Casc. 5
1st-*n*	0.00027	0.00029	0.00031	0.00032	0.00033
2nd-*n*	0.00047	0.00049	0.00051	0.00053	0.00054
3rd-*n*	0.00053	0.00058	0.00062	0.00065	0.00069
Gener.	0.00039	0.00042	0.00045	0.00047	0.00048

MSE Mean squared error

online mode, which allows it to work with non-stationary and stochastic nonlinear chaotic signals with the required accuracy. Compared to the well-known evolving neuro-fuzzy systems based on Takagi–Sugeno fuzzy reasoning, the proposed system provides computational simplicity and possesses both tracking and filtering capabilities.

References

Almeida L, Silva F (1990) Speeding up backpropagation. Adv Neural Comput, pp 151–158

Angelov P, Filev D (2004) An approach to online identification of Takagi–Sugeno fuzzy models. IEEE Trans Syst Man Cybern Part B Cybern 34(1):484–498

Angelov P, Filev D (2005) Simpl e TS: a simplified method for learning evolving Takagi–Sugeno fuzzy models. In: Proceedings of FUZZ-IEEE, pp 1068–1073

Angelov P, Kasabov N (2005) Evolving computational intelligence systems. In: Proceedings of the 1st international workshop on genetic fuzzy systems, pp 76–82

Angelov P, Lughofer E (2008) Data-driven evolving fuzzy systems using e TS and FLEXFIS: comparative analysis. Int J Gen Syst 37(1):45–67

Angelov P, Zhou X (2006) Evolving fuzzy systems from data streams in real-time. In: Proceedings of 2006 international symposium on evolving fuzzy systems, pp 29–35

Angelov P, Zhou X (2008) Evolving fuzzy-rule-based classifiers from data streams. IEEE Trans Fuzzy Syst 16(6):1462–1475

Angelov P, Xydeas C, Filev D (2004) Online identification of MIMO evolving Takagi–Sugeno fuzzy models. In: Proceedings of IJCNN-FUZZ-IEEE, pp 55–60

Angelov P, Lughofer E, Klement E (2005) Two approaches to data-driven design of evolving fuzzy systems: e TS and FLEXFIS. In: Proceedings of NAFIPS, pp 31–35

Angelov P, Giglio V, Guardiola C, Lughofer E, Lujan J (2006) An approach to modelbased fault detection in industrial measurement systems with application to engine test benches. Meas Sci Technol 17(7):1809–1818

Angelov P, Zhou X, Filev D, Lughofer E (2007) Architectures for evolving fuzzy rulebased classifiers. In: Proceedings of SMC, pp 2050–2055

Angelov P, Lughofer E, Zhou X (2008) Evolving fuzzy classifiers using different model architectures. Fuzzy Sets Syst 159(23):3160–3182

Angelov P, Filev D, Kasabov N (2010) Evolving intelligent systems: methodology and applications. Wiley, New York

Avedjan E, Barkan G, Levin I (1999) Cascade neural networks. Avtomatika i telemekhanika 3:38–55

Bodyanskiy Y, Kolodyazhniy V (2010) Cascaded multi-resolution spline-based fuzzy neural network. In: Proceedings of international symposium on evolving intelligent systems, pp 26–29

Bodyanskiy Y, Pliss I (1990) Adaptive generalized forecasting of multivariate stochastic signals. In: Proceedings Latvian signal proceedings of international conference, vol 2, pp 80–83

Bodyanskiy Y, Viktorov Y (2009a) The cascaded neo-fuzzy architecture and its on-line learning algorithm. Intell Process 9:110–116
Bodyanskiy Y, Viktorov Y (2009b) The cascaded neo-fuzzy architecture using cubic-spline activation functions. Inf Theor Appl 16(3):245–259
Bodyanskiy Y, Vorobyov S (2000) Recurrent neural network detecting changes in the properties of nonlinear stochastic sequences. Autom Remote Control 61(7):1113–1124
Bodyanskiy Y, Madjarov N, Pliss I (1983) Adaptive forecasting of non-stationary processes. Avtomatika I Izchislitelna Tekhnika 6:5–12
Bodyanskiy Y, Pliss I, Solovyova T (1986) Multistep optimal predictors of multidimensional non-stationary stochastic processes. Doklady AN USSR A(12):47–49
Bodyanskiy Y, Pliss I, Solovyova T (1989) Adaptive generalized forecasting of multidimensional stochastic sequences. Doklady AN USSR A(9):73–75
Bodyanskiy Y, Stephan A, Vorobyov S (1999) Algorithm for adaptive identification of dynamical parametrically nonstationary objects. J Comp Syst Sci Int 38(1):14–38
Bodyanskiy Y, Cichocki A, Vorobyov S (2001a) An adaptive noise cancellation for multisensory signals. Fluct Noise Lett 1(1):13–24
Bodyanskiy Y, Kolodyazhniy V, Stephan A (2001b) An adaptive learning algorithm for a neuro-fuzzy network, pp 68–75
Bodyanskiy Y, Kokshenev I, Kolodyazhniy V (2003a) An adaptive learning algorithm for a neo-fuzzy neuron. In: Proceedings of international conference of European Union Society for fuzzy logic and technology, pp 375–379
Bodyanskiy Y, Kolodyazhniy V, Otto P (2003b) A new learning algorithm for a forecasting neuro-fuzzy network. Integr Comput Aided Eng 10(4):399–409
Bodyanskiy Y, Dolotov A, Pliss I, Viktorov Y (2008) The cascaded orthogonal neural network. Inf Sci Comput 2:13–20
Bodyanskiy Y, Viktorov Y, Pliss I (2009) The cascade growing neural network using quadratic neurons and its learning algorithms for on-line information processing. Intell Inf Eng Syst 13:27–34
Bodyanskiy Y, Grimm P, Teslenko N (2011a) Evolving cascaded neural network based on multidimensional Epanechnikov's kernels and its learning algorithm. Inf Technol Knowl 5(1):25–30
Bodyanskiy Y, Kharchenko O, Vynokurova O (2011b) Hybrid cascaded neural network based on wavelet-neuron. Inf Theor Appl 18(4):335–343
Bodyanskiy Y, Teslenko N, Vynokurova O (2011c) Cascaded GMDH-wavelet-neuro-fuzzy network. In: international workshop on inductive modelling, pp 22–30
Chan L, Fallside F (1987) An adaptive learning algorithm for backpropagation networks. Comput Speech Lang 2:205–218
Cichocki A, Unbehauen R (1993) Neural Netw Optim Signal Process. Teubner, Stuttgart
Fahlman S, Lebiere C (1990) The cascade-correlation learning architecture. Adv Neural Inf Process Syst 2:524–532
Haykin S (1999) Neural networks: a comprehensive foundation. Prentice Hall, NJ
Hoff M, Widrow B (1960) Adaptive switching circuits, pp 96–104
Holmes G, Veitch A (1991) A modified quickprop algorithm. Neural Comput 3:310–311
Jang JSR, Sun CT, Muzutani E (1997) Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice Hall, Upper Saddle River
Kaczmarz S (1937) Angenäeherte Auflösung von Systemen linearer Gleichungen. Bull Acad Polon Sci Lett A 35:355–357
Kaczmarz S (1993) Approximate solution of systems of linear equations. Int J Control 53:1269–1271
Kadirkamanathan V, Niranjan M (1993) A function estimation approach to sequential learning with neural networks. Neural Comput 5:954–975
Kasabov N (2001) Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. IEEE Trans Syst Man Cybern Part B Cybern 31(6):902–918
Kasabov N (2003) Evolving connectionist systems. Springer, London
Kasabov N (2007) Evolving connectionist systems: the knowledge engineering approach. Springer, London
Kasabov N, Song Q (2002) DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Trans Fuzzy Syst 10(2):144–154
Kasabov N, Zhang D, Pang P (2005) Incremental learning in autonomous systems: evolving connectionist systems for on-line image and speech recognition. In: Proceedings of IEEE workshop on advanced robotics and its social impacts, pp 120–125
Kruschke J, Movellan J (1991) Benefits of gain: speed learning and minimum layers backpropagation networks. IEEE Trans Syst Man Cybern 21:273–280
Kusanagi H, Miki T, Uchino E, Yamakawa T (1992) A neo-fuzzy neuron and its applications to system identification and prediction of the system behavior. In: Proceedings of international conference on fuzzy logic and neural networks, pp 477–483
Ljung L (1999) System identification: theory for the user. Prentice Hall, NJ
Lughofer E (2006) Process safety enhancements for data-driven evolving fuzzy models. In: Proceedings of the 2nd symposium on evolving fuzzy systems, pp 42–48
Lughofer E (2008a) Evolving vector quantization for classification of on-line data streams. In: Proceedings of conference on computational intelligence for modelling, control and automation, pp 780–786
Lughofer E (2008b) Extensions of vector quantization for incremental clustering. Pattern Recognit 41(3):995–1011
Lughofer E (2008c) FLEXFIS: a robust incremental learning approach for evolving ts fuzzy models. IEEE Trans Fuzzy Syst 16(6):1393–1410
Lughofer E (2010a) On dynamic selection of the most informative samples in classification problems. In: Proceedings of the 9th international conference in machine learning and applications, pp 120–125
Lughofer E (2010b) On-line evolving image classifiers and their application to surface inspection. Image Vis Comput 28(7):1063–1172
Lughofer E (2011) Evolving fuzzy systems and methodologies: advanced concepts and applications. Springer, Heidelberg
Lughofer E, Angelov P (2009) Detecting and reacting on drifts and shifts in on-line data streams with evolving fuzzy systems. In: Proceedings of the IFSA/EUSFLAT 2009 conference, pp 931–937
Lughofer E, Angelov P (2011) Handling drifts and shifts in on-line data streams with evolving fuzzy systems. Appl Soft Comput 11(2):2057–2068
Lughofer E, Bodenhofer U (2006) Incremental learning of fuzzy basis function networks with a modified version of vector quantization. IPMU 2006:56–63
Lughofer E, Guardiola C (2008a) Applying evolving fuzzy models with adaptive local error bars to on-line fault detection. In: Proceedings of genetic and evolving fuzzy systems 2008, pp 35–40
Lughofer E, Guardiola C (2008b) On-line fault detection with data-driven evolving fuzzy models. J Control Intell Syst 36(4):307–317
Lughofer E, Kindermann S (2008) Improving the robustness of data-driven fuzzy systems with regularization. In: Proceedings of the IEEE world congress on computational intelligence 2008, pp 703–709
Lughofer E, Kindermann S (2010) Sparse FIS: data-driven learning of fuzzy systems with sparsity constraints. IEEE Trans Fuzzy Syst 18(2):396–411
Lughofer E, Klement E (2003) Online adaptation of Takagi–Sugeno fuzzy inference systems. In: Proceedings of CES-IMACS multiconference

Lughofer E, Klement E (2004) Premise parameter estimation and adaptation in fuzzy systems with open-loop clustering methods. In: Proceedings of FUZZ-IEEE 2004

Lughofer E, Efendic H, Re L, Klement E (2003) Filtering of dynamic measurements in intelligent sensors for fault detection based on data-driven models. In: Proceedings of the IEEE CDC conference, pp 463–468

Lughofer E, Klement E, Lujan J, Guardiola C (2004) Model-based fault detection in multi-sensor measurement systems. In: Proceedings of IEEE IS 2004, pp 184–189

Lughofer E, Huellermeier E, Klement E (2005) Improving the interpretability of data-driven evolving fuzzy systems. In: Proceedings of EUSFLAT 2005, pp 28–33

Lughofer E, Angelov P, Zhou X (2007) Evolving single- and multi-model fuzzy classifiers with FLEXFIS- class. In: Proceedings of FUZZ-IEEE 2007, pp 363–368

Lughofer E, Smith J, Caleb-Solly P, Tahir M, Eitzinger C, Sannen D, Nuttin M (2009) On human-machine interaction during on-line image classifier training. IEEE Trans Syst Man Cybern Part A Syst Hum 39(5):960–971

Miki T, Yamakawa T (1999) Analog implementation of neo-fuzzy neuron and its on-board learning. In: Computational intelligence and applications. WSES Press, Piraeus, pp 144–149

Narendra K, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1(1):4–27

Prechelt L (1997) Investigation of the Cas Cor family of learning algorithms. Neural Netw 10:885–896

Rong NJ, Huang GB, Saratchandran P (2006) Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. Fuzzy Sets Syst 157(9):1260–1275

Schalkoff R (1997) Artificial neural networks. The McGraw-Hill Comp., New York

Uchino E, Yamakawa T (1997) Soft computing based signal prediction, restoration and filtering. Fuzzy logic, neural networks and genetic algorithms. In: Intelligent Hybrid Systems. Kluwer Academic Publishers, Boston, pp 331–349

Yingwei L, Sundararajan N, Saratchandran P (1997) A sequential learning scheme for function approximation using minimal radial basis function (RBF) neural networks. Neural Comput 9:461–478