Journal of Mechanics Engineering and Automation 5 (2015) 112-120
doi: 10.17265/2159-5275/2015.02.008

Self-configuring Two Types of Neural Networks by MPCA

Juliana A. Anochi[1], Haroldo F. Campos Velho[1], Helaine C.M. Furtado[2] and Eduardo F.P. Luz[3]

1. Computing and Applied Mathematics Laboratory, INPE (National Institute for Space Research), São José dos Campos 12227-010, Brazil

2. Institute of Engineering and Geology, UFOPA (Federal University of Western Pará), Santarém 68040-470, Brazil

3. CEMADEN (National Center for Monitoring and Early Warning of Natural Disasters), São José dos Campos 12247-016, Brazil

Received: December 02, 2014 / Accepted: December 24, 2014 / Published: February 25, 2015.

Abstract: ANN (artificial neural network) is a technique successfully employed in many applications on several research fields. An appropriate configuration for neural networks is a tedious task, and it often requires the knowledge of an expert on the application. In this paper, a technique for automatic configuration for two types of neural networks is presented. The multilayer perceptron and recurrent Elman are the neural networks used here. The determination of optimal parameters for the neural network is formulated as an optimization problem, solved with the use of meta-heuristic MPCA (multiple particle collision algorithm). The self-configuring networks are applied to perform data assimilation.

Key words: Neural networks, MPCA, data assimilation.

1. Introduction

ANNs (artificial neural networks) are computational techniques that present a mathematical model inspired by the neural structure of biological organisms, acquiring knowledge through experience which has been applied widely in several tasks such as control, prediction of weather and climate, data assimilation, optimization, image processing and others. ANNs have emerged as excellent tools for deriving data oriented models, due to their inherent characteristic of plasticity that permits the adaptation of the learning task when data are provided. In addition to plasticity, ANNs also present generalization and fault tolerance characteristics that are fundamental for systems that depend on observational data that may be incomplete and slightly different from the data used to derive the models.

The use of ANNs to solve real problems always includes the selecting of appropriated network topology. The later issue has been the subject of intensive research in recent years, since we are looking for an optimal topology to solve a specific problem. This is a complex task, and usually requires a great effort by an expert, identifying the best parameter set, and it is necessary a previous knowledge about the problem to be treated.

There are many algorithms in the literature for the ANN training aimed at the improvement of the ability of generalization and for the control of an adequate architecture specification, such as: (1) pruning [10] makes adjustment of neural network by modifying its structure, the training begins with an oversized architecture and the weights are eliminated until the capacity of generalization can be increased; (2) weight decay [10]: the algorithm is similar to the pruning, where the cost function and weight vector are modified; (3) early stopping [17]: the scheme performs the early interruption of training, without changing the ANN architecture, similarly; (4) cross validation: it is known

Corresponding author: Juliana A. Anochi, Ph.D. candidate, research field: applied artificial intelligence. E-mail: juliana.anochi@gmail.com.

to improve the generalization, where the data set is separated in two sets: training data and validation data [15]. However, all these algorithms still suffer from slow convergence. In addition, these are based on gradient techniques and can easily stick at a local minimum.

Teixeira et al. [16] present a new learning algorithm to improve the generalization of the model of multilayer perceptron. This algorithm uses the training techniques of multi-objective optimization, which proposes to control the complexity of NN (neural network) using simultaneous minimization of training error and norm of weight vector. Costa et al. [4] present a new constructive method and pruning approaches to control the design of MLP (multilayer perceptron) without loss in performance. Costa et al. [5] also have developed an optimization technique for multi-objective training of ANN which uses the control algorithm for sliding mode control. This algorithm controls the convergence of the system to the point of minimum. Carvalho et al. [3] propose an approach to configure the architecture of the neural network, using mono-objective optimization techniques. The authors used four metaheuristics: the generalized extremal optimization, the variable neighborhood search, simulated annealing and genetic algorithm.

Optimization problems have the goal of finding the best set within a variable set to maximize or minimize a function, defined as an objective function or cost function. Optimization problems can be classified as: continuous optimization (where the variable has real or continuous values); discrete optimization (where the variable has integer or discrete values) and mixed optimization (with integer and continuous values at the same time) [1].

This paper deals with self-configuration using a new meta-heuristic called the MPCA (multiple particle collision algorithm) [12] to compute the optimal architecture for an ANN. The cost function has two terms: a square difference between ANN output and the target data (for two data sets: learning process and the generalization), and a penalty term used to evaluate the complexity for the new network architecture at each iteration. The concept of network complexity is associated to the number of neurons and the number of iterations in the training phase. In this paper, two types of ANN with supervised learning are used, the multilayer perceptron and recurrent Elman. Here, the self-configuring networks are applied to perform data assimilation to emulate the KF (Kalman filter) which is carried out with linear 1D wave equation [2]. The paper is organized as follows: Section 2 addresses neural networks; Section 3 describes the MPCA; Section 4 explains the methodology in details; Section 5 shows the data assimilation; Section 6 presents the results and discussions; finally, Section 7 draws some conclusions.

2. ANN

ANNs are computational techniques inspired on the neural structure of biological organisms, acquiring knowledge through experience. The network has parallel and distributed processing units or neurons, which calculates certain mathematical functions, typically nonlinear. These neurons can be divided into one or more hidden layers interconnected by synaptic weights, which store the knowledge represented in the model, and serve to balance the input received by each network neuron [9].

The artificial neuron model proposed by McCullock and Pitts in 1943 interprets the functioning of the neuron as a simple binary circuit that combines multiple inputs and one output signal. The mathematical description resulted in a model with n input terminals representing the dendrites, and only one output simulating the axon. In the mathematical terms, a neuron k could be described considering the following pair of equations [9]:

$$v_k = \sum_{j=1}^{n} w_{kj} x_i \tag{1}$$

$$y_k = \varphi(v_k + b_k) \tag{2}$$

where, x_1, x_2, .., x_n are the input signals, w_{kj} are the synaptic weights of neuron k computed during the learning phase, v_k is the linear combination among the input signals, b_k is the bias, φ is the activation function, and y_k is the output signal of the neuron. And the use of bias b_k has the effect of applying an affine transformation to the output v_k of the linear combiner in the model.

The property of primary significance for a NN is the ability to learn from the environment, improving its performance through the learning. A neural network learns from its environment through an interactive process of adjustments applied to its synaptic weights and bias levels [9]. In the context of NN, the learning process can be defined as a set of well defined rules for solving a specific problem of learning.

The training algorithms are divided into two classes: supervised and unsupervised learning. For the supervised learning, the input pattern and the desired output are provided by an external supervisor. For this type of training, the output is compared with the desired response, and the weights of connections are adjusted by minimizing the error. In the training with unsupervised learning, only the input patterns are presented to the network: there is no external supervisor to indicate the desired output [9].

2.1 Multilayer Perceptron

The architectures of multilayer perceptron neural models are the most used and known. Typically, this architecture consists of a set of units forming an input layer, one or more intermediate (or hidden) layer(s) of computational units, and an output layer. It is a supervised neural network.

A very popular algorithm for training MLP-NN is called backpropagation of error. The backpropagation learning error consists of two steps through the different layers of the network: forward and backward steps. For the forward step, an activity pattern (input vector) is applied thought the nodes of the network, and its effect propagates on the entire the network, layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the backward step, the synaptic weights are all adjusted in accordance with an error correction rule. The response of the network is subtracted from a desired output to produce an error signal. The error signal is back propagated through the network. For minimizing the error, a correction is applied to the synaptic weight $\Delta w_{ji}(n)$, following the rule delta [9]:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{3}$$

where, η is the learning rate parameter. The use of the negative sign in Eq. (3) denotes the gradient descent direction so as to reduce the value of $\varepsilon(n)$.

An ANN is composed of an input layer of neurons, one or more hidden layer(s) and an output layer. Each layer comprises multiple units fully connected (Fig. 1) with an independent weight attached to each connection.

In order to evaluate the performance of the models, the mean square error is used.

$$E_{gen} = \frac{1}{N}\sum_{k=1}^{N} (y_k - d_k)^2 \tag{4}$$

where, N is the number of patterns in the data set, y_k is the true observational value and d_k is the estimation computed by the neural model.

2.2 Recurrent Neural Network

The recurrent networks are neural networks with one or more feedback loops. Given a multilayer perceptron as the basic building block, the application of global feedback can take a variety of forms. We may have feedback from the output neurons to the input layer. Another possibility is the link among the hidden neurons with the input layer. When the MLP has two or

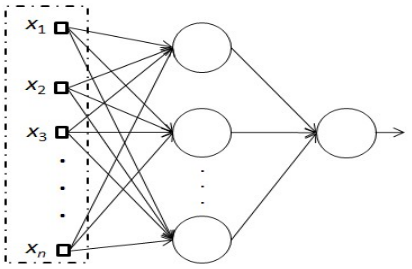


Fig. 1 Three layers multilayer perceptron architecture.

more hidden layers, the possible forms of global feedback expand even further [9].

The Elman network [7] is used in the current study and it has a similar architecture to the MLP. This network contains recurrent connections from the hidden neurons to a layer of context units, consisting of unit delays. These context units store the outputs of the hidden neurons for one time step, and then feed them back to the input layer. The hidden neurons have some record of their prior activations, which enables the network to perform learning tasks that extend over time.

The network Elman consists of four layers: an input layer, composed of storage neurons that receive an external signal to propagate it, without modification; a hidden layer where activation functions may be linear or non-linear; a context layer which is used to store the activation of neurons; the intermediate layer (it can be used with time delay), and an output layer, neurons whose outputs are linear sums of their respective input signals). Fig. 2 shows the architecture of a recurrent network inspired in a multilayer perceptron.

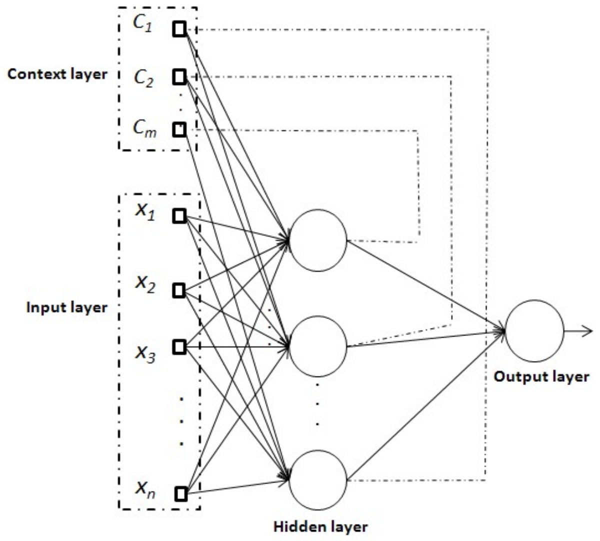


Fig. 2 Elman recurrent neural networks with an input layer, a hidden layer, a context layer and an output layer.

3. MPCA

The MPCA was developed in Ref. [12], inspired in the canonical PCA (particle collision algorithm) [14]. In this version of the algorithm, a new characteristic is introduced: the use of several particles, instead of only one particle, for moving in the search space. Both algorithms are greatly inspired on two physical behaviors inside of a nuclear reactor: absorption and scattering.

The PCA starts with a selection of an initial solution (Old_Config). It is then modified by a stochastic perturbation (function Perturbation), leading to the construction of a new solution (New_Config). The new solution is compared (function Fitness), and the new solution can or cannot be accepted. If the new solution is not accepted, the particle can be sent to a different place in the search space, giving the algorithm the capability of escaping a local optimum. This approach is inspired on the scattering process (function Scattering). If a new solution is better than the previous one, this new solution is absorbed (function Absorption). The exploration around closer positions is applied by using the operators Perturbation{.} and Small_Perturbation{.} [12].

The implementation of the MPCA algorithm is similar to PCA, but it uses a set of *N* particles, where a mechanism for sharing the information among particles is employed. A blackboard strategy is adopted, where the Best_Fitness candidate is shared among all particles in the process. Results have shown that MPCA is able to computing good solutions, in which PCA cannot find good answers [12]. The MPCA is codified using MPI (message passing interface) libraries on a multiprocessing machine with distributed memory. The MPCA, as well as the PCA, applies a strategy presented by Metropolis et al. [13]. The MPCA belongs to the class of bio-inspired algorithms. However, the PCA is a trajectory based-method, while the MPCA is a population-based method.

4. Optimization of Neural Networks by MPCA

A neural network learns from its environment through an interactive process of adjustments of the synaptic weights and bias [9]. In the context of neural networks, the learning process can be defined as a set of well defined rules for solving a specific problem of learning.

There is no a general analytical method to compute ANN weights. Further, we not even know the correct network topology to be applied. In practice, this problem is usually solved by using empirical methods, based on repetitive trial and error approach: the configuration is determined performing tests with different topologies until satisfactory results. This empirical issue consists of an iterative process, where the neural network parameters are supplied by the user.

The problem to identify the best configuration of the supervised NN can be formulated as an optimization problem. The goal is to find the optimum value, representing the best combination of variables for the NN architecture, with the definition of the set of weights. This self-configured NN is determined by the minimization of cost function defined by Eq. (5).

The main advantage by using an automatic procedure to configure an ANN is the ability to define a near-optimal ANN architecture, without needing the help of an expert on the NN theory and/or on the application. Such approach avoids this time consuming and tiring process of trial and error to find the optimal neural network architecture.

Optimization problems have the goal of finding the best solution within a search space to maximize or minimize a function or a functional, defined as an objective function or cost function. They can be classified as: continuous optimization (real or complex number domain); discrete optimization (for integer or discrete values); and mixed optimization (when the unknown set has integer and continuous values) [1]. The ANN configuration treated here belongs to the last class.

The objective function used in this study is a weighted average of two terms of square difference between the target values and the ANN output and a penalty term. The cost function is given in Ref. [3].

$$f_{obj} = penalty \times \left(\frac{\rho 1 \times E_{train+} \rho 2 \times E_{gen}}{\rho 1 + \rho 2} \right) \tag{5}$$

where, $\rho 1 = 1$ and $\rho 2 = 0.1$, the same values proposed by Carvalho et al. [3]. These are adjustment factors for taking into account the relative relevance of training and generalization errors.

There is great flexibility in the evaluation of the objective function, because the training error is directly related to the network memory capacity, and generalization error refers to the ability of ANN to identify the patterns that are similar to ones contained in the validation set. The factor *penalty* is applied to compute the NN architecture with the lowest complexity.

The computational complexity of a supervised ANN architecture can be defined as a function of two factors: the total number of neurons, and the epochs number to reach the convergence. From this consideration, a penalty factor was developed to favor lightweight architectures, and it can be expressed by:

$$penalty = C_1 \left(\varepsilon^{neurons} \right)^2 \times C_2 (epochs) + 1 \tag{6}$$

where, $C_1 = 1$ and $C_2 = 0.1$ are adjustment parameters for finding a good balance for the two factors in the complexity measurement.

The MPCA is employed to estimate: the number of intermediate layers, the number of neurons in each intermediate (hidden) layer, the learning rate η, momentum constant α and the activation function. Allowed values for these parameters are shown in Table 1.

5. Application: Data Assimilation

In this work, two models of ANN are used: a MLP-NN and

Table 1 Parameters to define a network topology.

Parameter	Value
Neuron in the hidden layer	(1, 32)
Learning rate η	(0.0, 1.0)
Momentum constant α	(0.1, 0.9)
Activation function	Tanh, Logistic, Gauss

the Elman-NN (Elman recurrent neural network). Both ANNs are trained to emulate the KF applied to assimilate data to the linear wave equation of first order. The issue is how to specify a quasi-optimal topology for two types of ANNs trained to emulate the data assimilation KF process, generating good results.

Mathematical models describing physical phenomena are imprecise, with incomplete understanding of the process. Therefore, the prediction of a certain model is inexact. The forecast is detached from reality every time step during the prediction process. For mitigating such disagreement, information is inserted into the model from the observation system. The result of a balanced combination between model and observation data is called analysis, the initial condition for the prediction system. This characterizes a cycle of data assimilation and forecasting [6, 11].

However, if we consider hypothetically that the models are perfect, observation measures contain errors. For a chaotic regime, any small changes in initial conditions alter the dynamics, making the data assimilation a necessary process for an operational forecasting.

The more accurate the estimate of the initial condition is, the better the quality of the forecast will be. For this, it is necessary to use tools of data assimilation to initialize the numerical forecast models. Mathematically, data assimilation is a two-step process:

(1) Forecast step

$$\eta_t^f = M\,(\eta_{t-1}^a) \qquad (7)$$

(2) Analysis step

$$\eta_t^a = \eta_t^f + \rho \qquad (8)$$

where, η_t^a is the vector of model state variables, the superscripts f and a represent the forecast and analysis steps, M represents the numerical model, ρ is the increment of the analysis (or innovation), determined according to the technique assimilation used, η_t^a is the analysis data or initial condition.

5.1 Forward Model: First Order Wave Equation

The dynamical system used in our tests is one of models employed by Bennett [2] for evaluation on data assimilation schemes. The mathematical model is the first order linear 1D wave equation:

$$\frac{\partial \eta}{\partial t} + c\frac{\partial \eta}{\partial t} = F(x,1) \qquad (9)$$

where, $\eta(x,\ t)$ is the unknown variable to be estimated by data assimilation, c is the constant phase speed, $F(x,\ t)$ is the external forcing, t is time, and x is space. Periodic boundary conditions are assumed.

The wave model was integrated with centered finite difference method, for space variable, and with the Crank-Nicholson scheme for the time. The initial condition is given by the following expression:

$$\eta_F(x,0) = \eta_0 \frac{1}{\cos h^2[(x-v)/\Delta]} \qquad 0 \le x \le L_x \qquad (10)$$

where, η_0 = -60, $v = c + (\alpha\eta_0)/3$, $\alpha = -1.62 \times 10^{-2}$, c = 2.42, and Δ = 1,340, the same values employed by Bennett [2].

5.2 Kalmam Filter

The result of the data fusion from observations and from a mathematical model is called analysis. Among several probabilistic techniques to make this merger is the KF. It is an estimation procedure for taking into account the statistical model of Gaussian type. For complete description of a Gaussian model, only the mean and the standard deviation need to be known. For a stochastic process, the distribution (or the statistical properties) will go to change with time. Thus, if the time variation of the distribution is known, the stochastic process will be well determined. There are two components for the KF: (1) forecasting step, where the updated state and the estimation of uncertainties are propagated forward in time and (2) up-dating step, in which the estimated state and the uncertainties are adjusted, preparing to the next observation set. In the KF, the prediction is made from a new analysis obtained when observations are available. The filter propagates the analysis by the forecasting model:

$$\eta_{n+1}^f = M\,(\eta_n^a) \qquad (11)$$

where, n and f denote the discrete time and the forecasting values, respectively. After this step, the modeling error is estimated by calculating the matrix covariance of the prediction error:

$$P_{n+1}^{f} = M_n P_n^a M_n^T + Q_n \tag{12}$$

where, Q_n is the covariance matrix of the modeling error. The analysis is calculated as a weighted average:

$$\eta_{n+1}^{a} = \eta_{n+1}^{f} + K_{n+1}^{f}\left[\eta_{n+1}^{0} - H_{n+1}\left(\eta_{n+1}^{f}\right)\right]^{-1} \tag{13}$$

where, η_{n+1}^{0} is the observation vector at time $n + 1$, H_{n+1} is the observation operator, and K_{n+1} is the Kalman gain. The matrix K is computed as following:

$$k_{n+1} = p_{n+1}^{f} + H_{n+1}^{T}\left[R_{n+1} + H_{n+1}P_{n+1}^{f}H_{n+1}^{T}\right]^{-1} \tag{14}$$

where, R_{n+1} is the co-variance matrix for observations errors. The KF also calculates the co-variance error for the analysis:

$$P_{n+1}^{a} = \left[I - K_{n+1}H_{n+1}^{T}\right] P_{n+1}^{f} \tag{15}$$

being I the identify matrix.

6. Results for Data Assimilation

As mentioned earlier, the goal is to identify neural network for emulating the assimilation process produced by the KF. The results below show the network automatically configured with MPCA, and a comparison with previous results obtained with ANN defined by an expert [8].

Parameters used in the MPCA are six particles multi-processing machine, one particle per processor and 10 iterations. The stopping criterion is the maximum number of objective function evaluations. The numerical experiment was performed with synthetic observational data. The MPCA was applied to optimize the parameters of ANN: number of neurons in the hidden layer, activation function, learning rate, and momentum.

The KF deals with the following parameters: $Q_n = 0.1I$, $R_n = 0.5I$, $H = I$, corresponding to the covariance matrix of modelling error, covariance matrix of the observation error, H representing the observation system, respectively, and I is the identity matrix. The observations are synthetically generated, and they are assimilated every 10 steps. Observations were generated from the forward model, adding a random Gaussian increment with 5% of noise level. For the proposed data assimilation method using neural networks, the equation below describes the analysis determination:

$$\eta_n^a = F_{NN}(\eta_n^f\, y_n^0) \tag{16}$$

where, F_{NN} is the neural networks used in the assimilation process. These ANNs are supervised networks. The desired output for the neural networks is an estimation obtained with KF. The ANN training phase consists to determine the best weights for connecting the internal neurons.

The numerical experiment was performed by automatic configuration of two types of ANN for data assimilation. The first experiment is carried out with the feedforward Elman neural network, and the second one is the MLP-NN. For both experiments, weights and bias were initialized with random values.

The best architectures found by MPCA are shown in Table 2, where MPCA-ANN1 is the Elman recurrent network, MPCA-ANN2 is the MLP network, and the Expert-ANN is the network defined by Furtado et al. [8].

The forward model was executed considering $t_{\max} = N_t \times \Delta t = 3{,}000 \times 10 = 30{,}000$ s, and $L_x = N_x \times \Delta x = 128 \times 150 = 19{,}200$ m.

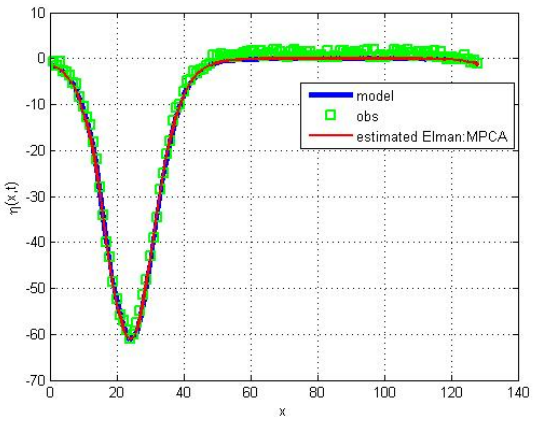
Fig. 3a shows the wave profile at the time step 3,000, performing 300 cycles of assimilation. The blue curve corresponds to the true state, the red one is the wave profile obtained from the analysis computed by

Table 2 Neural networks architectures.

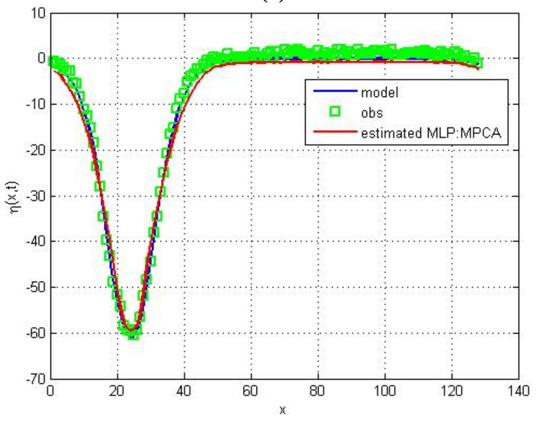
Parameter	MPCA-ANN1	MPCA-ANN2	MLP ANN
Hidden layer	1	1	1
Hidden layer neuron	8	23	3
Learning rate η	0.2	0.4	0.9
Momentum α	0.9	0.6	0.3
Activation function	Tanh	Tanh	Tanh
Quadratic error	0.051	0.158	0.421

ANN (emulating the KF), and observations are represented by green square points.

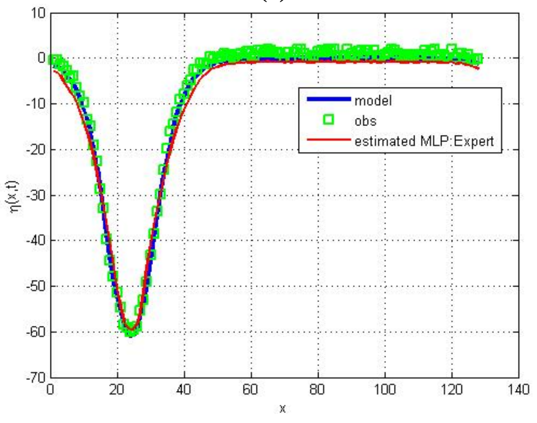
The results using both ANN models determined by MPCA (Figs. 3a and 3b) obtained better performance than ANN defined by an expert. Indeed, both ANNs computed by MPCA show a high fidelity with real dynamics (Fig. 3c).



(a)



(b)



(c)

Fig. 3 (a) MPCA-ANN1: Elman network; (b) MPCA-ANN2: MLP network and (c) Empirical ANN.

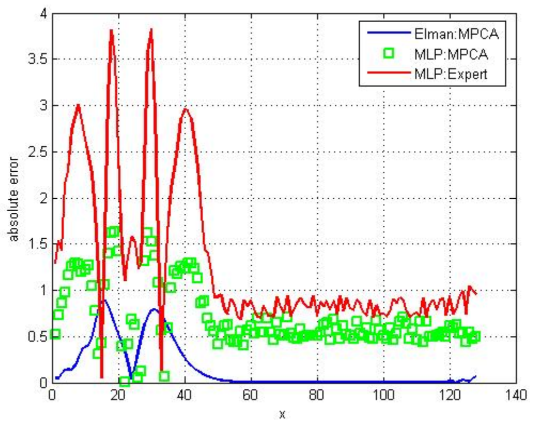


Fig. 4 Difference between observed and the output of neural networks.

Fig. 4 shows the error between the observation and the assimilation executed by ANN. In this figure, the red curve corresponds to the result with the assimilation obtained by ANN-Expert, the green and blue curves are results obtained by the MLP-NN and Elman-NN configured by MPCA, respectively.

7. Conclusions

A challenge for the ANN application is to define an appropriated configuration. Here, the self-configuring neural network is formulated as optimization problem. A new meta-heuristic MPCA is applied for automatic configuration for two types of ANN: MLP-NN and Elman-NN. The ANNs were employed into an important application: data assimilation. The assimilation with the Elman-NN presented the best results (Fig. 4).

The automatic method to identify the best configuration for the ANN does not require the help of an expert. This special feature allows the application of ANN approach to a larger community. As mentioned earlier, the goal is to identify neural network for emulating the assimilation process produced by the KF.

Acknowledgments

The authors would like to thank CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), Brazilian agencies for research support.

References

[1] Becceneri, J. 2009. "Fundamentals of Optimization and Artificial Intelligence." *Technical Intelligence Computational Inspired by Nature Application in Inverse Problems in Radiative Transfer* 41: 35-42.

[2] Bennett, A. F. 2002. *Inverse Modeling of the Ocean and Atmosphere.* UK: Cambridge University Press.

[3] Carvalho, A., Ramos, F. M., and Chaves, A. C. 2011. "Metaheuristics for the Feedforward ANN (Artificial Neural Network) Architecture Optimization Problem." *Neural Comput Applic* 20: 1273-84.

[4] Costa, M. A., Braga, A. P., and Menezes, B. R. 2003. "Improving Neural Networks Generalization with New Constructive and Pruning Methods." *Intelligent and Fuzzy Systems* 13: 75-83.

[5] Costa, M. A., Braga, A. P., Menezes, B. R., Teixeira, R. A., and Parma, G. G. 2003. "Training Neural Networks with a Multi-objective Sliding Mode Control Algorithm." *Neurocomputing* 51: 467-73.

[6] Daley, R., 1993. *Atmospheric Data Analysis.* UK: Cambridge University Press.

[7] Elman, J. L. 1990. "Finding Structure in Time." *Cognitive Science* 14: 179-211.

[8] Furtado, H. C. M., Velho, H. F. C., and Macau, E. E. N. 2011. "Data Assimilation with Artificial Neural Networks in Differential Equations." In *Proceedings of the Brazilian Conference on Dynamics,* 595-8.

[9] Haykin, S. 1998. *Neural Networks Principles and Practices.* New Jersey: Prentice Hall.

[10] Hinton, G. E. 1989. "Connectionist Learning Procedures." *Artificial Intelligence* 40: 185-234.

[11] Kalnay, E. 2003. *Atmospheric Modeling, Data Assimilation, and Predictability.* UK: Cambridge University Press.

[12] Luz, E. F. P., Becceneri, J. C., and Campos, H. F. V. 2008. "A New Multi-particle Collision Algorithm for Optimization in a High-Performance Environment." *Journal of Computational Interdisciplinary Sciences* 1: 1-7.

[13] Metropolis, N., Resenbluth, A., Rosenbluth, M., and Teller, A. H. T. E. 1953. "Equation of State Calculations by Fast Computing Machines." *The Journal of Chemical Physics* 21 (6): 1087-92.

[14] Sacco, M., Lapa, C., Pereira, C., and Filho, H. 2006. "Two Stochastic Optimization Algorithms Applied to Nuclear Reactor Core Design." *Progress in Nuclear Energy* 48 (6): 525-39.

[15] Stone, M. 1978. "Cross-Validation: A Review." *Math. Operations Forsch. Statist.* 9 (1): 127-39.

[16] Teixeira, A., Braga, A. P., Takahashi, R. H., and Saldanha, R. R. 2000. "Improving Generalization of MLP with Multi-objective Optimization." *Neurocomputing* 35: 189-94.

[17] Weigend, A., and Huberman, B. 1990. "Predicting the Future: A Connectionist Approach." *International Journal of Neural Systems* 1 (3): 193-209.