Expert Systems with Applications 40 (2013) 7294–7304

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An online learning algorithm for adaptable topologies of neural networks

Beatriz Pérez-Sánchez*, Oscar Fontenla-Romero, Bertha Guijarro-Berdiñas, David Martínez-Rego

Department of Computer Science, Faculty of Informatics, University of A Coruña, Campus de Elviña s/n, 15071 A Coruña, Spain

A R T I C L E I N F O

Keywords:
Incremental learning
Sequential learning
Forgetting ability
Adaptive topology

A B S T R A C T

Many real scenarios in machine learning are of dynamic nature. Learning in these types of environments represents an important challenge for learning systems. In this context, the model used for learning should work in real time and have the ability to act and react by itself, adjusting its controlling parameters, even its structures, depending on the requirements of the process. In a previous work, the authors presented an *online* learning algorithm for two-layer feedforward neural networks that includes a factor that weights the errors committed in each of the samples. This method is effective in dynamic environments as well as in stationary contexts. As regards this method's incremental feature, we raise the possibility that the network topology is adapted according to the learning needs. In this paper, we demonstrate and justify the suitability of the *online* learning algorithm to work with adaptive structures without significantly degrading its performance. The theoretical basis for the method is given and its performance is illustrated by means of its application to different system identification problems. The results confirm that the proposed method is able to incorporate units to its hidden layer, during the learning process, without high performance degradation.

1. Introduction

In a significant number of real applications of machine learning such as adaptive sensory-motor control, prediction of customer behavior, fault detection, monitoring in biomedicine and industrial processes, web-mining, climate or financial data analysis, etc. information flows continuously and the new knowledge could affect a previously learned model (Elwell & Polikar, 2009; Wang, Fan, & Han, 2003). In such contexts, learning algorithms should be able to dynamically adjust their models whenever new data becomes available. Learning in a dynamic environment includes important topics such as sequential learning (Robins, 2004) or concept drift (Klinkenberg, 2004; Widmer & Kubat, 1996), both representing serious challenges for learning systems. Sequential learning is an adequate technique for situations in which the information arrives in separate batches over time. Concept drift happens when the target concept changes, in other words, when the distribution underlying the data evolves over time. Ideally, the representations developed by learning should be stable enough to preserve important information during new learning, but also plastic enough to incorporate recent knowledge when necessary. This condition constitutes the *stability-plasticity dilemma*: both stability and plasticity are desirable, but their requirements are in direct conflict (Grossberg, 1988; Robins, 2004). In this context, the system should not handle all the training instances in an equal fashion since recent examples are more relevant to the actual target concept (Kubat, Gamma, & Utgoff, 2004). Thus, the system should take into account the increase in the importance of current information in contrast with the past one. Therefore, adaptation plays an important role and the evolution of the process is taken into account by considering adaptive mechanisms able to track the system dynamics. In order to face different types of changes, systems must adjust their controlling parameters, or even their structures, as changes are detected. In other words, the system should be able to act and react by itself (Bouchachia, Gabrys, & Sahel, 2007).

Classical batch learning algorithms are not suitable for handling these types of situations, since they learn the concept from the beginning and therefore, whenever new samples are available, they need to start another learning session with all available instances whether they are new or old ones (Espósito, Ferilli, Fanizzi, Basile, & Mauro, 2004). This means, they discard the existing model and redesign a new one from scratch. This approach presents several problems, the waste of computational resources being the most significant. Therefore, in these situations an *online* or incremental learning technique would be a more appropriate approach since it assumes that the information available at any given moment is incomplete and therefore, any learned model is susceptible to modifications. These methods are able to adjust the model dynamically when new knowledge arrives.

* Corresponding author. Tel.: +34 981 167000x1359; fax: +34 981 167160.
E-mail address: bperezs@udc.es (B. Pérez-Sánchez).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.06.066

Incremental learning has been extensively studied in the machine learning community (Bottou, 2004; LeCunn, Bottou, Orr, & Müller, 1998; Moller, 1993; Rosenblatt, 1958) and many researchers have presented different learning methods. Fu, Hsu, and Principe (1996) proposed a method for pattern recognition, called an incremental backpropagation learning network, which employs bounded weight modifications and structural adaptation learning rules which apply initial knowledge to constrain the learning process. Bruzzon and Fernández (1999) proposed a novel classifier based on radial basis function neural networks for the classification of remote-sensing images. Hébert, Parizeau, and Ghazzali (1999) showed a method to combine an unsupervised self-organizing perceptron network for character detection. Yi, Wu, and Xu (2011) developed an improved incremental Support Vector Machine algorithm to deal with network intrusion detection whilst Hsiao and Chang (2008) proposed a new adaptive cluster-based classification method in order to improve the effectiveness of spam filtering. Moreover, it is worth mentioning the Online Support Vector Machine (Ma, Theiler, & Perkins, 2003; Parrella, 2007) and the Online Sequential Extreme Learning Machine (Huang, Zhu, & Siew, 2006; Liang & Huang, 2006). The former is a technique used to build support vector machines for regression problems with the possibility of adding or removing samples without retraining the machine from the beginning. The latter is another state of the art algorithm for multilayer feedforward neural networks characterized by its generalization power and its fast learning speed.

In Pérez-Sánchez, Fontenla-Romero, and Guijarro-Berdiñas (2010) we presented an *online* learning algorithm for two-layer feedforward neural networks. This method includes a factor that weights the errors committed in each one of the samples. Thanks to this, the method obtains good performance in environments that present an evolutionary behavior over a lifetime, whilst also maintaining correct behavior when the context is stationary and thus, solving the problem of concept drift. In addition to this capacity, it would be suitable for the network topology to be adjusted depending on the learning process' requirements, as it should correspond to the complexity of the analyzed data. When the network is too small, it is not able to learn the task properly; in contrast, if it is too large, it has tendency to overtrain (Kwok & Yeung, 1997). There are several reasons for searching for the optimal structure of a neural network, for instance improving and speeding up the prediction, achieving better generalization and saving computer resources, especially when working with large data sets (Reitermanová, 2008). In this work we propose a modification of our online learning algorithm to deal with changes in the network topology while maintaining, as far as possible, the knowledge previously acquired in the earlier stages of the learning without the need to retrain with all the samples. As a result, a new learning algorithm for adaptable topologies of neural networks, denoted as OANN, is presented.

The paper is structured as follows. In Section 2 the OANN method is described. In Section 3, its behavior is illustrated by its application to several time series in order to check its performance in different contexts. In Section 4 the results are discussed and some conclusions are given. Finally, in Section 5 we raise some future work lines.

2. Description of the proposed method

In a previous work (Pérez-Sánchez et al., 2010) we presented an incremental online learning algorithm for two-layer feedforward neural networks. This method is based in the introduction of a forgetting function which gives a monotonically crescent importance to new date. Due to this fact, the network forgets in presence of changes while maintaining a stable behavior when the context is stationary. In this current paper we propose a modification of the previous work and present an online incremental learning algorithm with adaptive network topology (OANN) which is described in what follows.

Before describing the method we will introduce some nomenclature and notations. Consider the two-layer feedforward neural network in Fig. 1 where the inputs are represented as the column vector $\mathbf{x}(s)$, the bias has been included by adding constant input $x_0 = 1$, and outputs are denoted as $\mathbf{y}(s), s = 1, 2, \ldots, S$ where S is the number of training samples. J and K are the number of outputs and hidden neurons, respectively. Functions $g_1, g_2, \ldots, g_K$ and $f_1, f_2, \ldots, f_J$ are the nonlinear activation functions of the hidden and output layer, respectively.

This network can be considered as the composition of two one-layer subnetworks. As the desired outputs for each hidden neuron k at the current learning epoch s, $z_k(s)$, are unknown, arbitrary values are employed. These values are obtained in base to a previous initialization of the weights using a standard method, for example Nguyen and Widrow (1990). The desired output of hidden nodes are not revised during the learning process and they are not influence by the desired output of the whole networks. Therefore, the training of the first subnetwork is avoided in the learning task. Regarding the second subnetwork, as $d_j(s)$ is the desired output for the j output neuron, that it is always available in a supervised learning, we can use $\bar{d}_j(s) = f_j^{-1}(d_j(s))$ to define the objective function for the j output of subnetwork 2 as the sum of squared errors *before* the nonlinear activation function f_j,

$$Q_j^{(2)}(s) = h_j(s)\left(f_j'(\bar{d}_j(s))\left(\mathbf{w}_j^{(2)^T}(s)\mathbf{z}(s) - \bar{d}_j(s)\right)\right)^2, \quad j = 1, \ldots, J \qquad (1)$$

where $\mathbf{w}_j^{(2)}(s)$ is the input vector of weights for output neuron j at the instant s and $f_j'(\bar{d}_j(s))$ is a scaling term which weights the errors. The inclusion of this scaling term is an alternative formulation proposed in Fontenla-Romero, Guijarro-Berdiñas, Pérez-Sánchez, and Alonso-Betanzos (2010). Moreover, the term $h_j(s)$ is included as forgetting function and it determines the importance of the error at the sth sample. This function is used to establish the form and the speed of the adaptation to the recent samples in a dynamic context

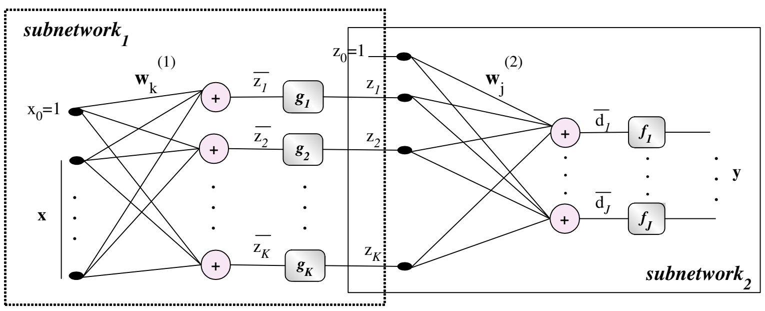


Fig. 1. Two-layer feedforward neural network.

(Martínez-Rego, Pérez-Sánchez, Fontenla-Romero, & Alonso-Betanzos, 2011). There exist several functions which can be used; for example, an exponential or linear function amongst others. In a stationary environment a constant function should be used in order to give the same weight to all the data points analyzed during the learning process. Whereas in a non-stationary context the function should be monotone increasing to take into account the increment in the importance of recent information in contrast with previous information.

The objective function presented in Eq. 1 is a convex function, whose global optimum can be easily obtained deriving it with respect to the parameters of the network and setting the derivative to zero (Fontenla-Romero et al., 2010). Therefore, we obtain the following system of linear equations,

$$\sum_{k=0}^{K} A_{qk}^{(2)}(s) w_{jk}^{(2)}(s) = b_{qj}^{(2)}(s), \quad q = 0, 1, \ldots, K;\ j = 1, \ldots, J, \tag{2}$$

where

$$A_{qk}^{(2)}(s) = A_{qk}^{(2)}(s-1) + h_j(s) z_k(s) z_q(s) f_j'^2(\bar{d}_j(s)), \tag{3}$$

$$b_{qj}^{(2)}(s) = b_{qj}^{(2)}(s-1) + h_j(s) \bar{d}_j(s) z_q(s) f_j'^2(\bar{d}_j(s)), \tag{4}$$

being $A^{(2)}(s-1)$ and $b^{(2)}(s-1)$, respectively, the matrices and the vectors that store the coefficients of the systems of linear equations employed to calculate the values of the weights of the second layer in previous iterations. In other words, the coefficients employed to calculate the weights in the actual epoch are used further to obtain the weights in the following iteration. Therefore, this permits handling the earlier knowledge and using it to incrementally approach the optimum value of the weights.

Eq. 2 can be rewritten using matrix notation as,

$$\mathbf{A}_j^{(2)}(s) \mathbf{w}_j^{(2)}(s) = \mathbf{b}_j^{(2)}(s), \quad j = 1, \ldots, J, \tag{5}$$

where

$$\mathbf{A}_j^{(2)}(s) = \mathbf{A}_j^{(2)}(s-1) + h_j(s) \mathbf{z}(s) \mathbf{z}^T(s) f_j'^2(\bar{d}_j(s)) \tag{6}$$

$$\mathbf{b}_j^{(2)}(s) = \mathbf{b}_j^{(2)}(s-1) + h_j(s) f_j^{-1}(d_j(s)) \mathbf{z}(s) f_j'^2(\bar{d}_j(s)). \tag{7}$$

Finally, from Eq. 5 the optimal weights for the second subnetwork can be obtained as:

$$\mathbf{w}_j^{(2)}(s) = \mathbf{A}_j^{(2)^{-1}}(s) \mathbf{b}_j^{(2)}(s), \forall j. \tag{8}$$

As regards the incremental property of the learning algorithm, the network structure could be adapted depending on the requirements of the learning process. The number of hidden neurons can be increased by adequately redimensionning the number of weights. Thus, in this paper, we described an algorithm which allows incorporating new units in the hidden layer according to the needs of the learning process. Therefore, several modifications have to be carried out in order to convert the current topology of the network to the new one.

The fact of increasing the number of hidden neurons implies modifications in both layers of the network. As it can be observed in Fig. 2 the increment of hidden neurons affects not only the first subnetwork (its number of output units increases) but also the second subnetwork (the number of its inputs also grow). As it was mentioned previously the training of the first subnetwork is avoided in the training task, therefore we only comment the modifications corresponding to the second subnetwork. Thus, in Fig. 2 it can be observed as the number of its inputs grows and consequently, all matrices $\mathbf{A}_j^{(2)}$ and the vectors $\mathbf{b}_j^{(2)}$ $(j = 1, \ldots, J)$, computed previously modify their size. Therefore in order to adapt them, each matrix $\mathbf{A}_j^{(2)}$ is enlarged by including a new row and a new column of zero values, in order to continue the learning process from this point (zero is the null element for the addition). At the same time, each vector $\mathbf{b}_j^{(2)}$ incorporates a new element of zero value. The rest of elements of the matrices and vectors are maintained without variation, this fact allows us to incorporate the knowledge acquired previously with the earlier topology.

After these modifications, the latter matrices and vectors of coefficients established allow us to obtain the new set of weights for the current topology of the network. Algorithm 1 details the proposed incremental *online* learning method employing the concepts earlier described.

Algorithm 1

Finally, the complexity of the algorithm is determined by the complexity of solving several linear equations system for each layer of the network. Several computationally efficient methods can be used to solve this kind of system (except for ill-conditioned matrices) with a complexity from $O(n^2)$ to $O(n^3)$, n being the number of weights of the network (Bojanczyk, 1984; Carayannis, Kalouptsidis, & Manolakis, 1982). In the case of ill-conditioned matrices, the problem can be solved by applying the Moore–Penrose pseudoinverse (Penrose, 1955).

Algorithm 1. OANN algorithm for two-layer feedforward neural networks

```
Inputs:
  $\mathbf{x}_s = (x_{1s}, x_{2s}, \ldots x_{Is});\quad \mathbf{d}_s = (d_{1s}, d_{2s}, \ldots d_{Js});\quad s = 1, \ldots, S.$
Initialization Phase
    $\mathbf{A}_j^{(2)}(0) = \mathbf{0}_{(K+1)\times(K+1)},\quad \mathbf{b}_j^{(2)}(0) = \mathbf{0}_{(K+1)},\quad \forall j = 1, \ldots, J.$
    Calculate the initial weights, $\mathbf{w}_k^{(1)}(0)$, by means of an
  initialization method.
For every new sample $s$ $(s = 1, 2, \ldots, S)$ and $\forall k = 1, \ldots, K$
    $z_k(s) = g(\mathbf{w}_k^{(1)}(0), \mathbf{x}(s))$
    For each output $j$ of the subnetwork 2 $(j = 1, \ldots, J)$,
      $\mathbf{A}_j^{(2)}(s) = \mathbf{A}_j^{(2)}(s-1) + h_j(s)\mathbf{z}(s)\mathbf{z}^T(s) f_j'^2(\bar{d}_j(s))$ (see Eq. 6),
      $\mathbf{b}_j^{(2)}(s) = \mathbf{b}_j^{(2)}(s-1) + h_j(s) f_j^{-1}(d_j(s))\mathbf{z}(s) f_j'^2(\bar{d}_j(s))$
  (see Eq. 7),
      Calculate $\mathbf{w}_j^{(2)}(s)$ solving the system of linear equations
  using Eq. 8.
    end of For
    If the user decides to changes the network topology then,
      For each output $j$ of the subnetwork 2 $(j = 1, \ldots, J)$
        $\mathbf{A}_j^{(2)}(s) = \left(\begin{array}{ccccc} & & & (m) & \\ & \mathbf{A}_j^{(2)}(s-1) & 0 & \ldots & 0 \\ & 0 & 0 & \ldots & 0 \\ (m) & \vdots & \vdots & \vdots\vdots\vdots & \vdots \\ & 0 & 0 & \ldots & 0 \end{array}\right), \mathbf{b}_j^{(2)}(s) = \left(\begin{array}{cc} & \mathbf{b}_j^{(2)}(s-1) \\ & 0 \\ (m) & \vdots \\ & 0 \end{array}\right),$
        $m$ being the number of hidden units to include, this
  increment can be carried out both one-by-one and group by
  group each time
      end of For
      For each new connection,
        calculate the weights, $\mathbf{w}_m^{(1)}(0)$, by means of some
  initialization method
      end of For
      $K = K + m$
    end of If,
end of For
```

3. Experimental results

The goal of this section is to evaluate the performance of the algorithm with regards to the mean squared error that it ob-

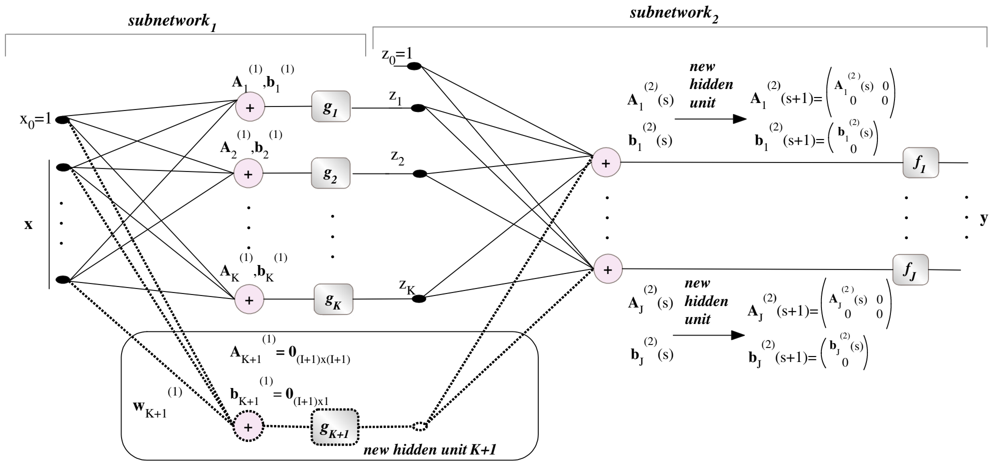


Fig. 2. Incremental topology.

tains for a given problem. As we already mentioned, the algorithm proposed in this paper, OANN, is based on a previous method, as described in Section 2. This basis *online* learning method, unable to adapt network topology, was compared in Pérez-Sánchez et al. (2010) with other *online* methods, specifically with the Online Support Vector Machine (Ma et al., 2003; Parrella, 2007) and the Online Sequential Extreme Learning Machine (Huang et al., 2006; Liang & Huang, 2006). In that experimental study, it was demonstrated that in dynamic environments, this method is able to obtain better results than the others while in stationary environments, it works properly and obtains similar results. Therefore, taking these results into account, in this experimental study the new proposed algorithm, OANN, will only be compared to its previous version with the following objectives:

- To check if the OANN is able to incorporate hidden units without significant performance degradation with respect to the basis method.
- To prove that the performance of the OANN at the end of the learning process, (when it reaches a final and stable network topology), is similar to the performance that would be obtained by employing this final topology from the beginning to the end of learning process.
- To confirm that the forgetting ability of the OANN is also useful when the topology changes.

For this experimental study, we employed six different system identification problems, four of them are real and the other two are artificial data sets. Moreover, the behavior of the learning algorithm will be checked when it operates in both stationary and dynamic environments. All experiments shared the following conditions:

- The neural functions. In all cases, the logistic sigmoid function was employed for hidden neurons, while for output units a linear function was applied as recommended for regression problems (Bishop, 1995).
- The network topology. In order to demonstrate the ability of the OANN to work with adaptive structures the modification of the topology is done manually, starting with a small network and forcing the addition of one hidden unit every certain number of iterations, at regular intervals depending on the number of samples, until the final structure is obtained. In the case of stationary contexts, 5 and 10 hidden units were employed as initial and final topologies respectively. Whereas for dynamic environments the structure starts from 10 hidden units and it grows until 15 neurons are obtained.
- The input data set was normalized, with mean = 0 and standard deviation = 1.
- In order to obtain significant results, five simulations were carried out. Therefore, mean results will be presented in this section. Moreover, in the case of stationary context 10-fold cross validation was applied.
- In all cases an exponential forgetting function, defined as,

$$h(s) = e^{\mu s}, \quad s = 1, \dots, S, \tag{9}$$

was employed, where μ is a positive real parameter that controls the growth of the function and thus the response of the network to changes in the environment. When $\mu = 0$ we obtain a constant function, and therefore all errors have the same weight and the forgetting function has no effect. The value of the μ factor for the forgetting ability was set to 0.01 except for the *Exchange rate Dollars against Pounds* data set for which a value of 0.05 resulted more appropriate.

3.1. Stationary contexts

In this section, we consider three stationary time series: Lorenz, K.U. Leuven and exchange rate US Dollars against UK Pounds. Next, a brief description of each selected data is given.

- *Lorenz Time Series Prediction.*[1] A Lorenz system Lorenz (1963) is described by the solution of three simultaneous differential equations. We have obtained a data set with a total of 5,000 patterns from this chaotic series generated by code. The goal of the network is to predict the current sample based on the eight previous ones.
- *K. U. Leuven Competition Data.*[2] The K.U. Leuven time series prediction competition data was generated from a computer simulated 5-scroll attractor, resulting from a generalized Chua's circuit Suykens and Vandewalle (1998). The aim of the neural network is to predict the current sample using only four previous data points. We employed 1800 data points for training.

[1] Available at http://www.cse.ogi.edu/~ericwan/data.html.
[2] Available at http://research.stlouisfed.org/fred2.

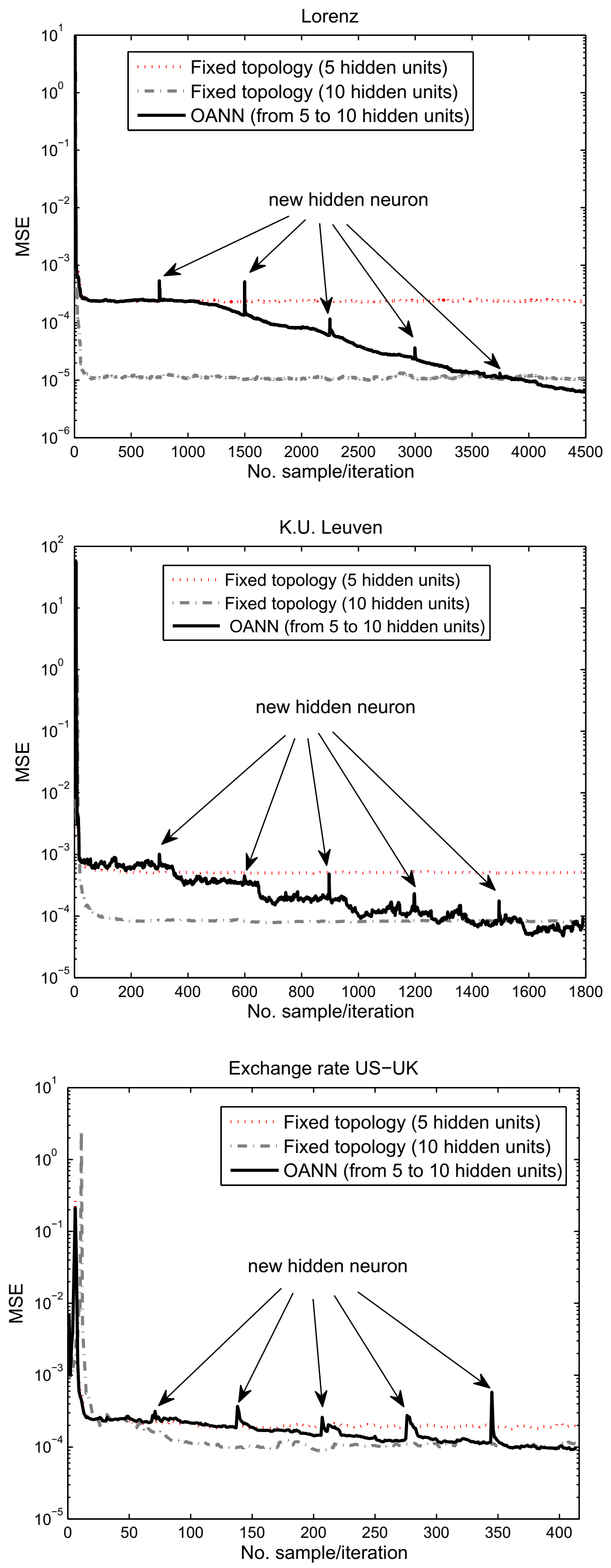


Fig. 3. Test MSE curves for stationary contexts. Comparison between the proposed method OANN (incremental structure) and its previous version (fixed topology).

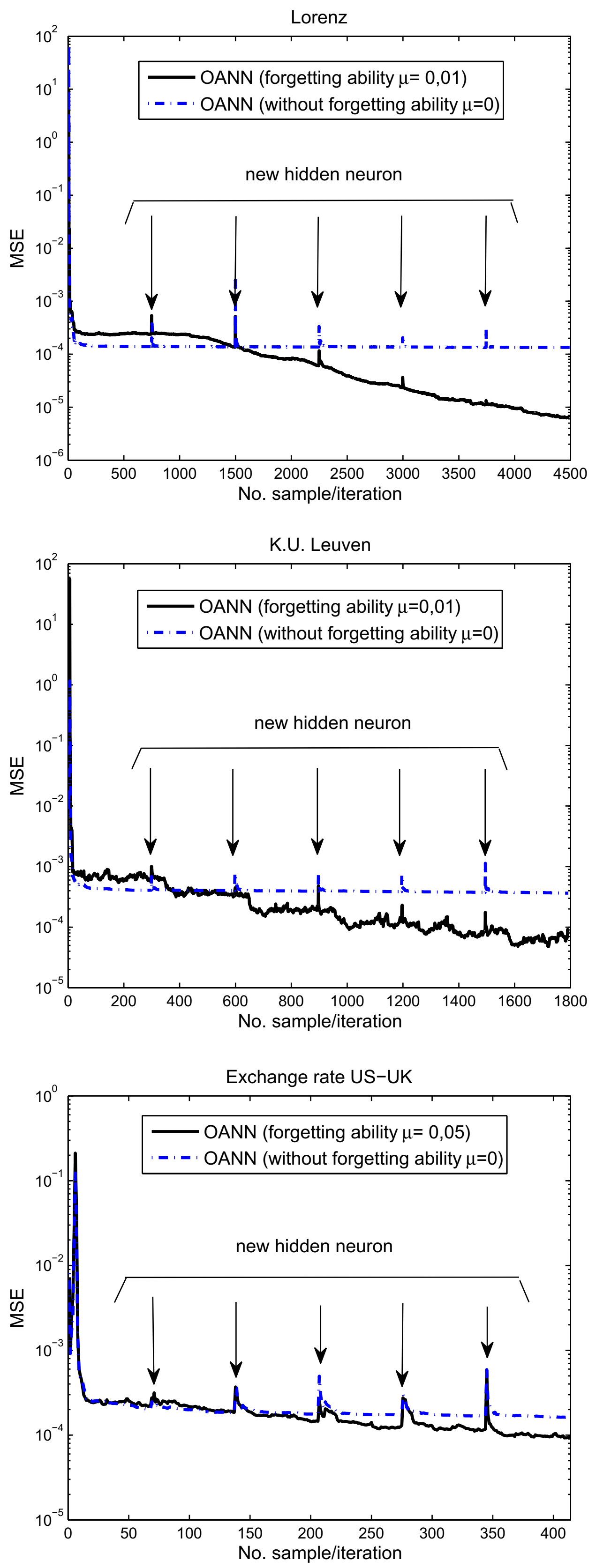
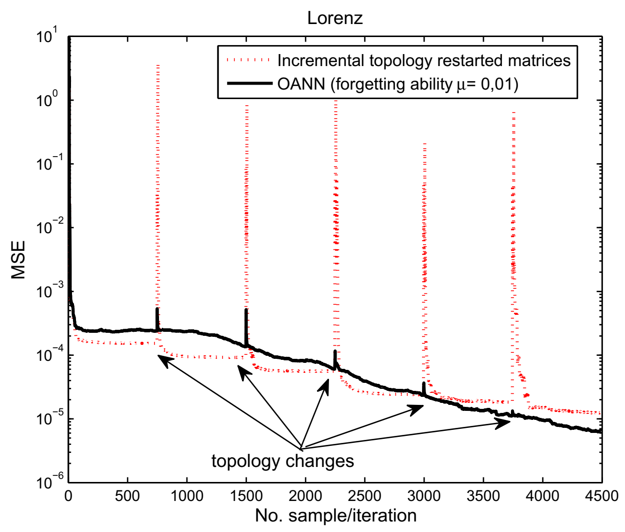
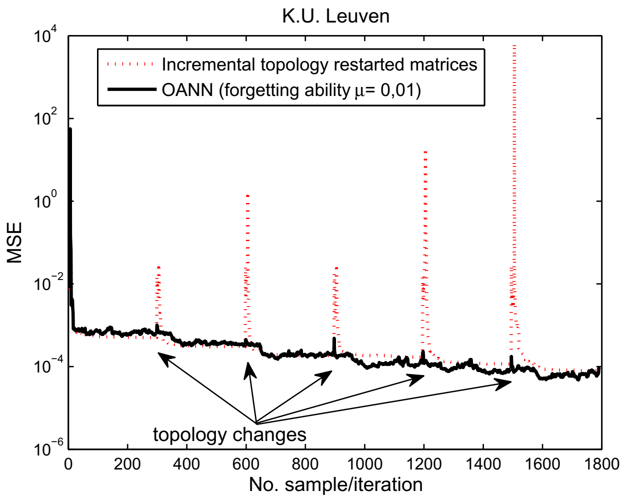


Fig. 4. Test MSE curves for stationary contexts. Checking the influence of the forgetting ability in the behavior of the OANN method when the topology of the network varies.





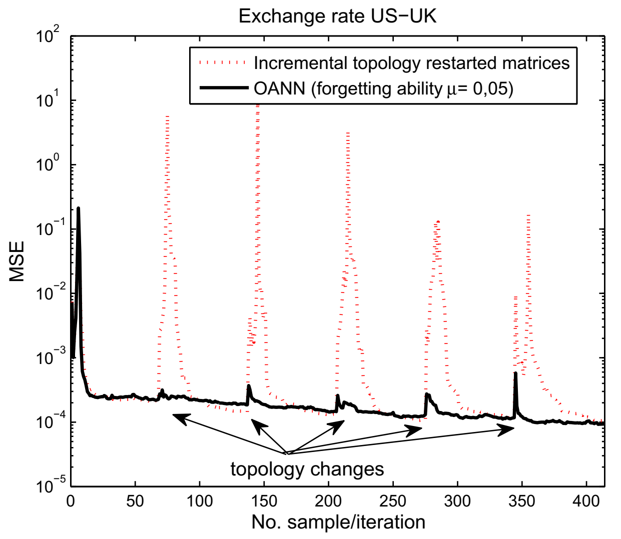


Fig. 5. Test MSE curves for stationary contexts. Influence of restarting/storing previously acquired knowledge when the topology of the network varies.

- *Exchange rate Dollars against Pounds (US–UK).*2 The data set corresponds to the exchange rate of UK pounds against US dollars. The historical data has monthly frequency and covers the period 1971–2009 with a total of 456 observations.

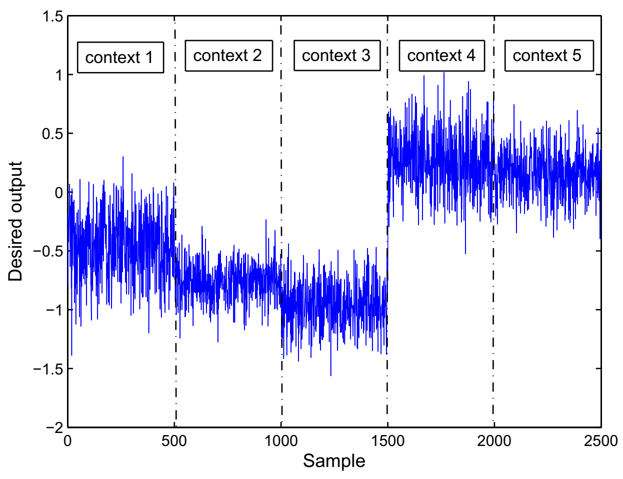


Fig. 6. Artificial data set 1. Example of the desired output for the training set.

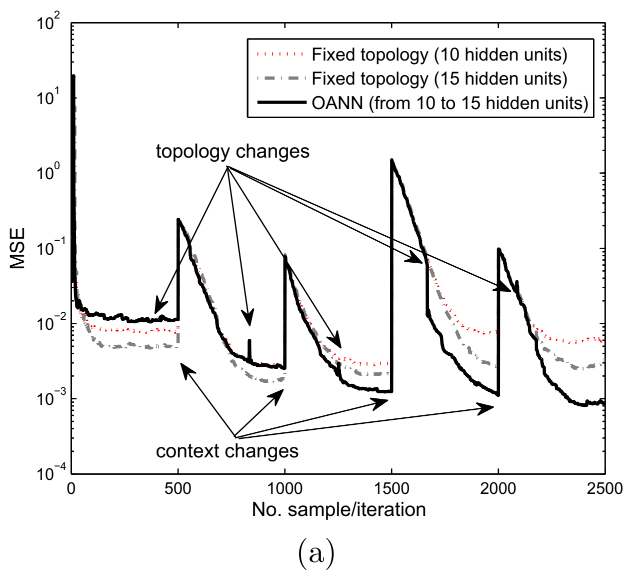
In the experimental study and for each data set, the figures show different results. In the first (see Fig. 3), we present the test MSE curves obtained by the two methods being compared: (1) the OANN, where the structure starts from an initial number of hidden units and grows until the desired final topology is reached, and (2) the previous method with fixed topology, for which Fig. 3 shows the curves corresponding to the initial and final topologies used for adaptive case. It can be observed how the OANN is able to add new neurons to its topology without significant performance degradation. Only when the topology of the network varies there is performance degradation, so several peaks can be observed in the MSE curves of the proposed method. This fact is due to the incorporation of new unit generated perturbations (new elements of zero value in the matrices). However, these degradations are corrected speedily. Apart from that, the OANN gets similar results than those obtained when the final fixed topology is trained from the initial epoch.

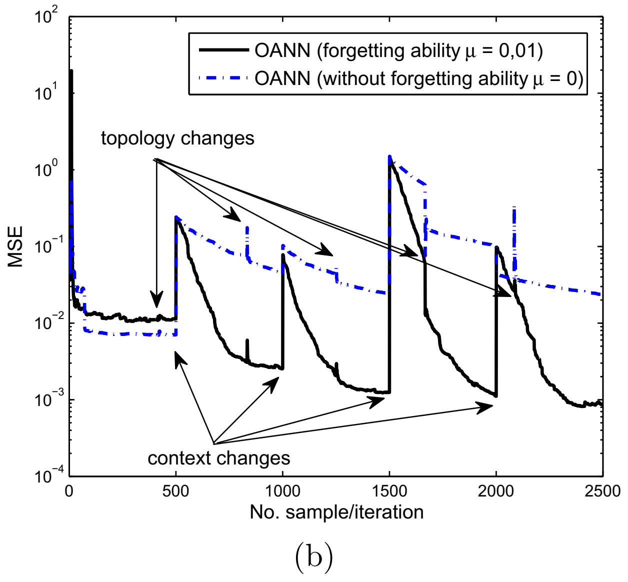
Moreover, we want to check if the forgetting ability is useful when the topology changes. Therefore, Fig. 4 shows the test MSE curves obtained by the OANN method with adaptive structure both with and without forgetting ability ($\mu = 0$). Despite being a stationary context, the without forgetting ability method obtains worse performance since it is not able to improve the committed error after a topology change. However, if the forgetting skill is considered, recent knowledge weights more than the one previously acquired and, consequently, learning advances satisfactorily while reusing the earlier learned model.

As commented previously, each time a new unit is added the method incorporates a row and/or column of zero elements in the matrix and vectors of coefficients. However, the rest of their elements are maintained in order to consider the knowledge acquired previously with the earlier topology. To check the validity of this approach we compare its results with that obtained when all the elements of matrices **A** and vectors **b** are reinitialized to zero when the topology changes. Fig. 5 shows the MSE curves obtained by both approaches. It can be observed how restarting the matrices and vectors of coefficients when the topology varies, is not a successful approach as the network forgets all the previous knowledge and consequently it has to start to learn from scratch.

3.2. Dynamic environments

In dynamic environments, the distribution of the data could change over time. This situation often occurs when dealing with





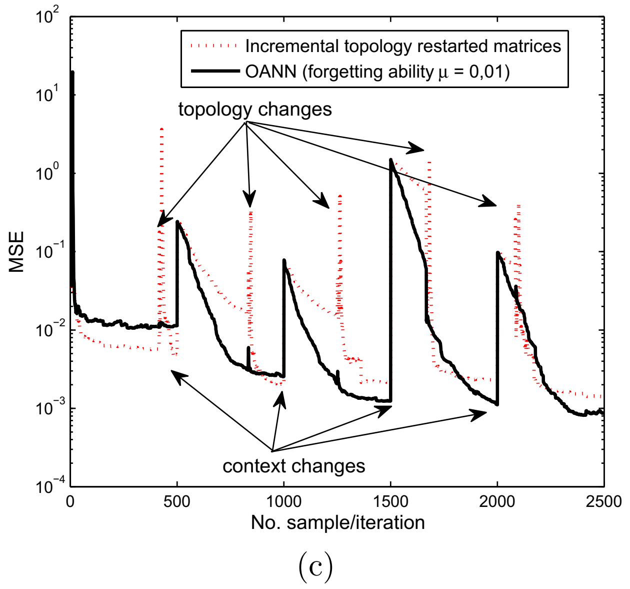


Fig. 7. Test MSE curves for dynamic environments. Artificial data set 1. (a) Comparison between the OANN (incremental structure) and its previous version (fixed topology), (b) Checking the influence of the forgetting ability in the behavior of the method when the topology of the network varies, (c) Influence of restarting/storing previously acquired knowledge when the topology of the network varies.

signals in real-life situations (Wegman & Smith, 1984). Important applications such as adaptive sensory-motor control systems (Franklin & Wolpert, 2011), economic data analysis, brain control interfaces, natural speech, robotics and, in general, applications dealing with biological or physiological signals (Mukhopadhyay & Sircar, 1997), have to manage the non-stationarity of the implied signals. Moreover, in these types of environment, changes between contexts can be *abrupt* when the distribution changes quickly or *gradual* if there is a smooth transition between distributions (Gama, Medas, Castillo, & Rodrigues, 2004). In order to check the performance of the proposed method in different situations we have considered both types of changing environments. The aim of these experiments is to check that the proposed method is able to obtain appropriate behavior when it works in dynamic environments.

3.2.1. Artificial data set 1

The first data set is formed by 2500 samples of 4 input random variables which contain values drawn from a normal distribution

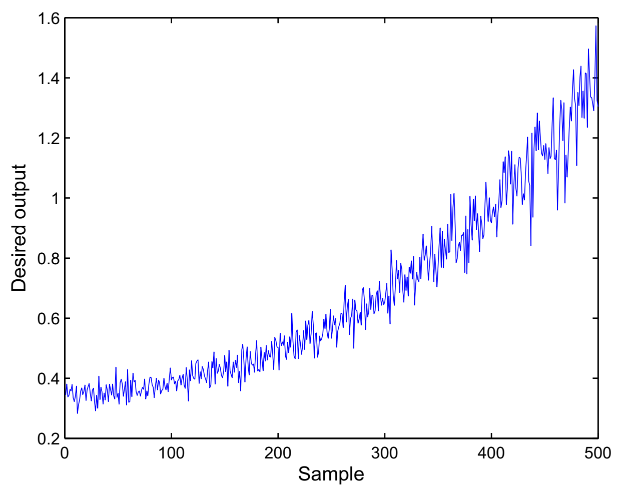


Fig. 8. Artificial data set 2. Example of the desired output for the training set.

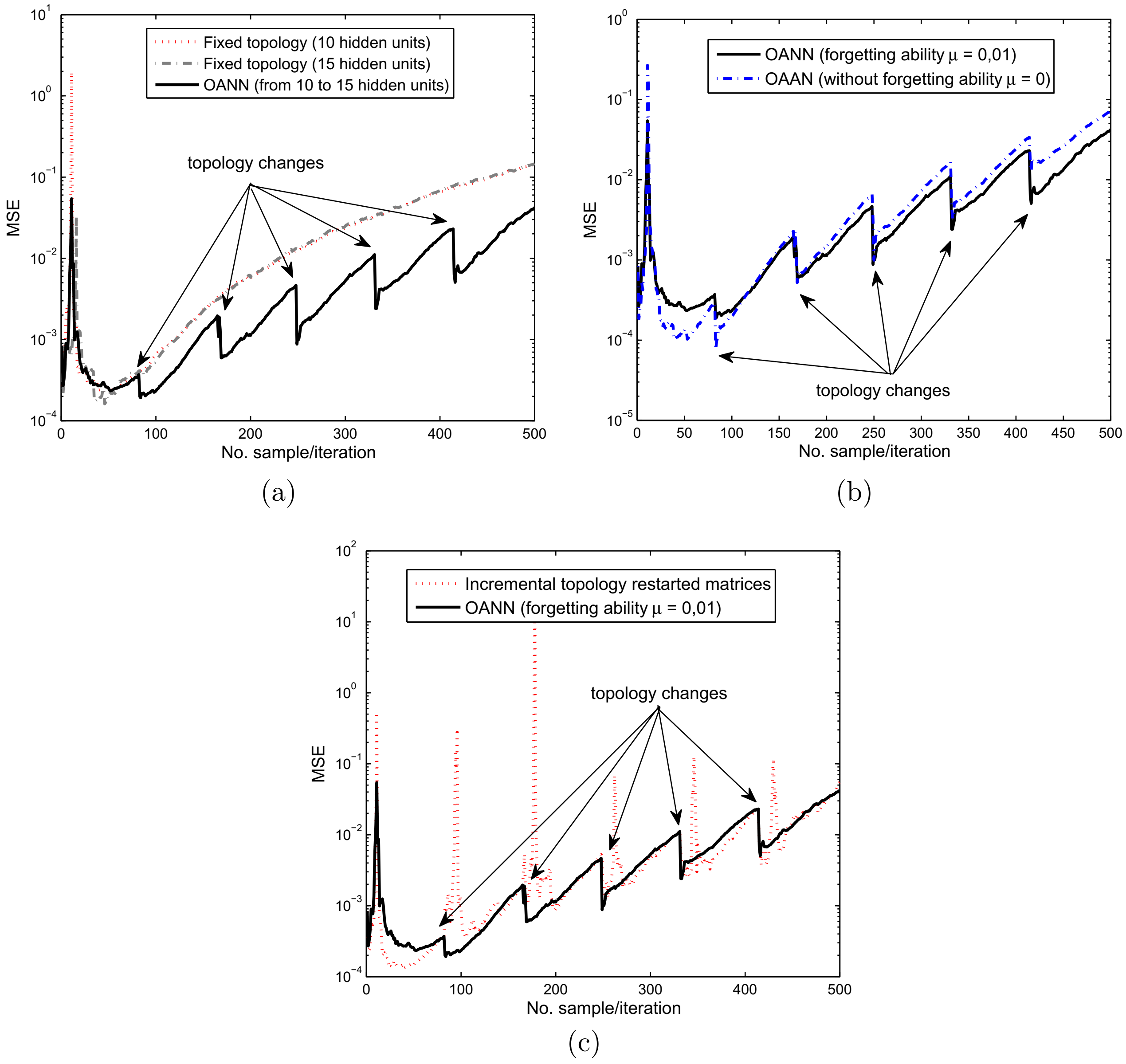


Fig. 9. Test MSE curves for dynamic environments. Artificial data set 2. (a) Comparison between the OANN (incremental structure) and its previous version (fixed topology), (b) Checking the influence of the forgetting ability in the behavior of the method when the topology of the network varies, (c) Influence of restarting/storing previously acquired knowledge when the topology of the network varies.

with zero mean and standard deviation equal to 0.1. To obtain the desired output, first, every input is transformed by a nonlinear function. Specifically, hyperbolic tangent sigmoid, exponential, sine and logarithmic sigmoid functions were applied, respectively, to the first, second, third and fourth output. Finally, the desired output is obtained by a linear mixture of the transformed inputs.

$$M = \begin{pmatrix} 2.1 & 1.3 & -1.1 & 1.3 \\ -1.1 & -0.3 & 0.7 & -1.1 \\ -0.1 & 0.2 & 1.8 & -2.3 \\ -3.1 & -1.5 & 0.4 & 1.8 \\ 1.5 & -0.9 & 1.2 & 0.6 \end{pmatrix}. \qquad (10)$$

Fig. 6 contains the signal employed as desired target during the training process. As can be seen, the signal evolves over time and 5 context changes are generated. Also we created different test sets for each context, so every training sample has associated the test set that represents the context to which it belongs. Thus, for this case we obtain 5 test sets, one for each of the changes in the linear mixture of the process. Once again, the training was carried out for fixed and incremental topologies. Fig. 7(a) shows the test error curves obtained by the method with fixed and incremental topologies. The error shown for each point of the signal is the mean value obtained in the corresponding test set for the current training sample. It can be seen that the proposed method is able to obtain better results than the fixed topology even when 15 hidden units are employed from the beginning. In this case, the spikes observed in the curves are due to two different causes. The changes between contexts, each 500 samples, cause the large peaks while in the case of incremental topology, the small spikes are a result of the variations in the topology. Also results for the adaptive structure with and without forgetting ability ($\mu = 0$) are presented in Fig. 7(b).

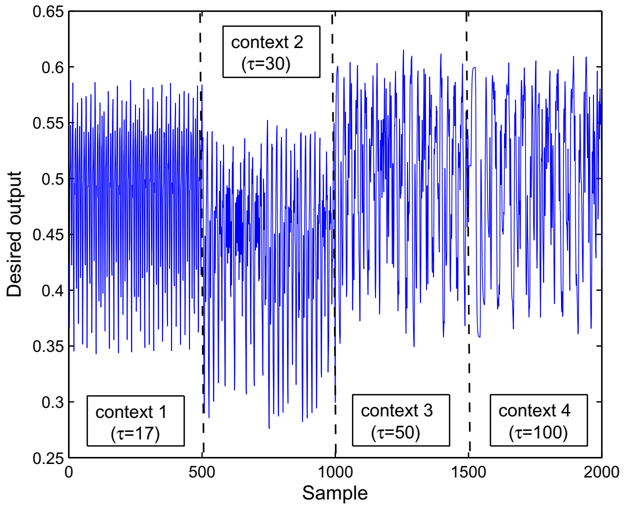


Fig. 10. Mackey–Glass time series. Example of the desired output for the training set.

The without forgetting capacity method is not able to improve its performance after a context change. Finally, again in Fig. 7(c) it can be confirmed that restarting the matrices of coefficients when a new hidden neuron is added is not an adequate alternative and a more unstable behavior is obtained.

3.2.2. Artificial data set 2

In this case, we generated another artificial data set that presents a *gradual* evolution in each sample of training. The data set is formed by 4 input variables (generated by a normal distribution with zero mean and standard deviation equal to 0.1) and the output is obtained by means of a linear mixture of these. In order to construct the time series we fixed the initial coefficients of the linear mixture vector as

$$a(0) = [\,0.5 \quad 0.2 \quad 0.7 \quad 0.8\,] \tag{11}$$

and subsequently these values evolve over time according to the following equation,

$$a_j(s) = a_j(s-1) + \frac{s}{10^{4.7}} logsig(a_j(s-1)), \quad s = 1, \ldots, S$$

where j indicates the component of the mixture vector. Finally, we obtained a set with $S = 500$ training samples. The signal employed as desired output during the training process can be observed in Fig. 8. As the distribution presents constant changes we have a different context for each sample thus, for this example, we have a different test data set for each one.Fig. 9(a) shows the test error curves obtained by the method with fixed and incremental topologies. As the input signal suffers continuous gradual changes, the error grows constantly. In spite of this, we can observe that the OANN overcomes the results obtained by the fixed topologies. Moreover, it is worth pointing out that the incorporation of a new unit implies a punctual performance improvement due to the perturbations included in the matrices that store the earlier information. This is not very relevant in this case as the process changes continuously. However, as can be observed in 9(c), a total reset of the previous knowledge is even less appropriate since it involves severe deteriorations when the topology changes, as the learning process has to start again from scratch. Finally, the influence of forgetting ability on the proposed method can be observed in Fig. 9(b) as the differences are not relevant since previous knowledge is less important.

3.2.3. Mackey–Glass chaotic time series

Finally, the proposed approach has been tested on the chaotic Mackey–Glass time series, which was originally proposed as a physiological control model for the production of white blood cells (Mackey & Glass, 1977). The time series is based on the following differential delay equation,

$$\frac{dy(t)}{dt} = \frac{0.2y(t-\tau)}{1 + y^{10}(t-\tau)} - \rho y(t). \tag{12}$$

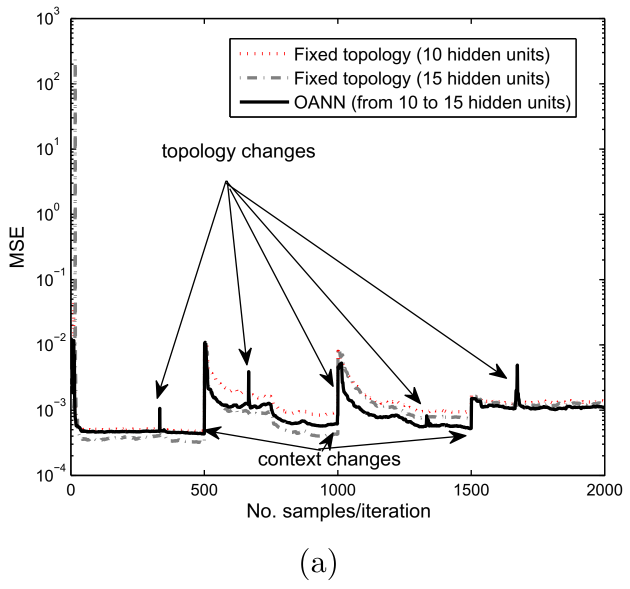
Considering this equation and using ρ=0.1, 2,000 observations are generated by means of the fourth order Runge–Kutta method Butcher (2008). It is a well-known fact that the chaotic behavior of the Mackey–Glass system increases as the delay coefficient τ is increased. Therefore, taking this idea we generated 4 different contexts of 500 samples each, which correspond to different values of parameter τ (17,30,50 and 100). The goal of the network is to predict the current sample based on the eight previous ones. Fig. 10 contains the signal employed as the desired target during the training process.

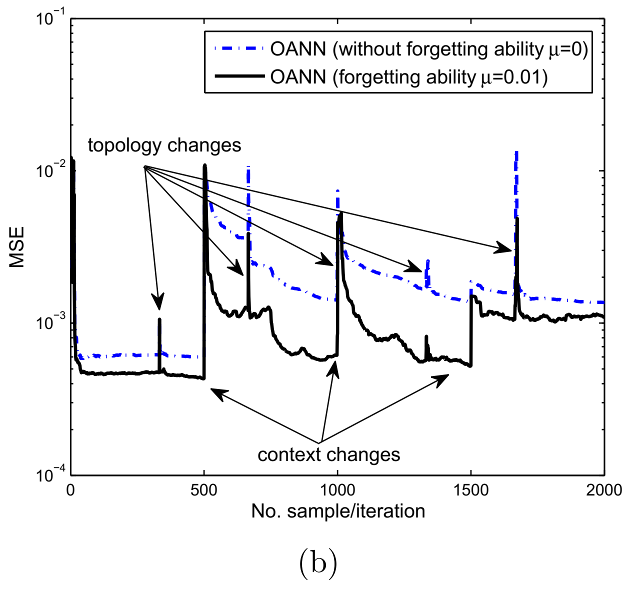
In Fig. 11(a) it can be seen that the proposed method can obtain similar results to the fixed topology even when 15 hidden units are employed from the beginning. Again, the spikes observed in curves are due to two different causes. The small peaks are a result of the variations in the topology while the changes between contexts (every 500 samples) cause the large spikes. Also in Fig. 11(b), we can check as the without forgetting capacity method ($\mu = 0$) cannot enhance its performance after a context change. Finally, Fig. 11(c) shows restarting the matrices of coefficients (when a new hidden neuron is added) implies unstable behavior.

4. Discussion

In view of the experiments made and the results presented in Section 3, there are some features of the method that stand out:

- Thanks to the incremental learning capacity, the proposed method is an appropriate approach to work in real time-manner and deal with environments that present an evolutionary behavior over a lifetime. The network can start with little knowledge and later on achieve a satisfactory result.
- As regards network topology, the capability of the method to hold up an incremental adaption of the topology was demonstrated. The increment of hidden units can be carried out both one-by-one and group by group each time. This skill has an important advantage, saving time and computer resources without degradation of the final performance.
- The proposed method can deal with stationary and non stationary environments, since it dynamically adapts its forgetting capabilities whilst maintaining adequate equilibrium between learning new information while retaining relevant old knowledge (*stability – plasticity* equilibrium). Moreover, the proposed method can tackle both gradual and abrupt changes in the environment.
- Finally, the method can deal with large data sets since it permits optimal employment of available computer resources. I, K and J being the number of inputs, hidden units and outputs of the network it only has to store, for the first layer, and update K matrices **A** and vectors **b** of sizes $(I+1)\times(I+1)$ and $(I+1)$ respectively, while for the second layer it has to store and update J matrices **A** and vectors **b** of sizes $(K+1)\times(K+1)$ and $(K+1)$, respectively. Thus, the method does not need to store and reprocess old training instances, and only the current training sample is kept.





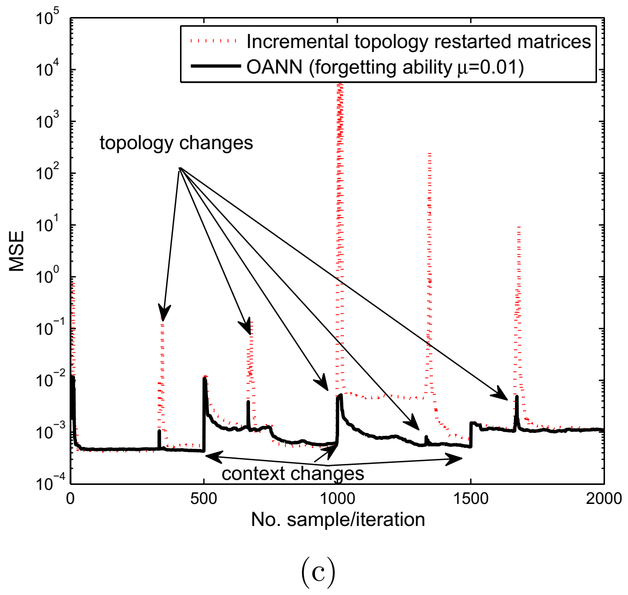


Fig. 11. Test MSE curves for dynamic environments. Mackey–Glass time series. (a) Comparison between the OANN (incremental structure) and its previous version (fixed topology), (b) Checking the influence of the forgetting ability in the behavior of the method when the topology of the network varies, (c) Influence of restarting/storing previously acquired knowledge when the topology of the network varies.

5. Conclusions and future work

In this work we have presented a new *online* and incremental learning algorithm with forgetting capability and incremental topology for two-layer feedforward neural networks. The main conclusions that can be obtained from the study reported in this paper are,

1. The proposed method, OANN, is able to dynamically adapt its forgetting capabilities, attaining appropriate behavior in environments that present an evolutionary behavior over a lifetime.
2. The OANN has the capability to maintain the obtained information in previous phases and adapt its topology without forgetting this previous knowledge.

In spite of these favorable characteristics, there are some aspects that need an in-depth study and will be addressed as future work. In this paper, we have demonstrated and justified the suitability of our *online* learning algorithm to work with adaptive structures without significantly degrading its performance. However, the modification of the topology is manually forced every fixed number of iterations. It will be interesting in future work, (the learning method will be completed with an automatic mechanism) to control whether new hidden units should be added, or even removed, depending on the requirements of the *online* learning process.

Acknowledgements

The authors would like to acknowledge support for this work from the Xunta de Galicia (Grant codes CN2011/007 and CN2012/211), the Secretaría de Estado de Investigación of the Spanish Government (Grant code TIN2012-37954), all partially supported by the European Union ERDF. Also, David Martínez-Rego would like to thank the Spanish Ministry of Education for the FPU supporting grant.

References

Bishop, C. M. (1995). *Neural networks for pattern recognition*. New York: Oxford University Press.

Bojanczyk, A. (1984). Complexity of solving linear systems in different models of computation. *SIAM Journal on Numerical Analysis, 21*(3), 591–603.

Bottou, L. (2004). Stochastic learning. In O. Bousquet & U. von Luxburg (Eds.), *Advanced lectures on machine learning, lecture notes in artificial intelligence, LNAI 3176* (pp. 146–168). Berlin: Springer Verlag.

Bouchachia, A., Gabrys, B., & Sahel, Z. (2007). Overview of some incremental learning algorithms. *FUZZ-IEEE 2007: Proceedings of the IEEE international conference on fuzzy systems, 2007*, 1–6.

Bruzzon, L., & Fernández, P. (1999). An incremental-learning neural network for the classification of remote-sensing images. *Pattern Recognition Letters, 20*(11–13), 1241–1248.

Butcher, J. (2008). *Numerical methods for ordinary differential equations*. John Willey & Sons.

Carayannis, G., Kalouptsidis, N., & Manolakis, D. (1982). Fast recursive algorithms for a class of linear equations. *IEEE Transactions on Acoustics, Speech and Signal Processing, 30*(2), 227–239.

Elwell, R., & Polikar, R. (2009). Incremental learning in nonstationary environments with controlled forgetting. In *IJCNN'09: Proceedings of the 2009 international joint conference on neural networks* (pp. 1388–1395). Piscataway, NJ, USA: IEEE Press.

Espósito, F., Ferilli, S., Fanizzi, N., Basile, T., & Mauro, M. D. (2004). Incremental learning and concept drift in inthelex. *Intelligent Data Analysis, 8*(3), 213–237.

Fontenla-Romero, O., Guijarro-Berdiñas, B., Pérez-Sánchez, B., & Alonso-Betanzos, A. (2010). A new convex objective function for the supervised learning of single-layer neural networks. *Pattern Recognition, 43*(5), 1984–1992.

Franklin, D., & Wolpert, D. (2011). Computational mechanisms of sensorimotor control. *Neuron, 72*(3), 425–442.

Fu, L., Hsu, H.-H., & Principe, J. (1996). Incremental backpropagation learning networks. *IEEE Transactions on Neural Networks, 7*(3), 757–761.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. *Intelligent Data Analysis, 8*, 213–237.

Grossberg, S. (1988). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 243–283.

Hébert, J.-F., Parizeau, M., & Ghazzali, N. (1999). Cursive character detection using incremental learning. In *ICDAR'99: Proceedings of the fifth international conference on document analysis and recognition* (pp. 808–811). Washington, DC, USA: IEEE Computer Society.

Hsiao, W.-F., & Chang, T.-M. (2008). An incremental cluster-based approach to spam filtering. *Expert Systems with Applications, 34*(3), 1599–1608.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing, 70*(1–3), 489–501.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis, 8*, 281–300.

Kubat, M., Gamma, J., & Utgoff, P. (2004). Incremental learning and concept drift: Editor's introduction: Guest-editorial. *Intelligent Data Analysis, 8*(3), 211–212.

Kwok, T.-Y., & Yeung, D.-Y. (1997). Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Transactions on Neural Networks, 8*(3), 630–645.

LeCunn, Y., Bottou, L., Orr, G., & Müller, K.-R. (1998). Efficient backprop. In G. Orr & K.-R. Müller (Eds.). *Neural networks: Tricks of the trade* (Vol. 1524, pp. 9–50). Springer.

Liang, N.-Y., & Huang, G.-B. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks, 17*(6), 1411–1423.

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences, 20*, 130–141.

Mackey, M., & Glass, L. (1977). Oscillation and chaos in physiological control sytems. *Science, 197*(4300), 287–289.

Martínez-Rego, D., Pérez-Sánchez, B., Fontenla-Romero, O., & Alonso-Betanzos, A. (2011). A robust incremental learning method for non-stationary environments. *NeuroComputing, 74*(11), 1800–1808.

Ma, J., Theiler, J., & Perkins, S. (2003). Accurate on-line support vector regression. *Neural Computation, 15*(11), 2686–2703.

Moller, M. (1993). Supervised learning on large redundant training sets. *International Journal of Neural Systems, 4*(1), 15–25.

Mukhopadhyay, S., & Sircar, P. (1997). Parametric modelling of non-stationary signals: A unified approach. *Signal Process, 60*(2), 135–152.

Nguyen, D., & Widrow, B. (1990). Improving the learning speed of 2-layer neural networks choosing initial values of the adaptive weights. *Proccedings of the international joint conference on neural networks, (IJCNN 1990)* (Vol. 3, pp. 21–26). .

Parrella, F. (2007). Online support vector regression. Ph.D. Thesis, Department of Information Science, University of Genova.

Penrose, R. (1955). A generalized inverse for matrices. *Proceedings of the Cambridge philosophical society* (Vol. 51, pp. 406–413). .

Pérez-Sánchez, B., Fontenla-Romero, & O., Guijarro-Berdiñas, B. (2010). An incremental learning method for neural networks in adaptive environments. In *Proceedings of the international conference on neural networks (IJCNN 2010)* (pp. 815–822).

Reitermanová, Z. (2008). Feedforward neural networks – architecture optimization and knowledge extraction. In *Proccedings of week of doctoral students, (WDS 2008), Part I* (pp. 159–164).

Robins, A. (2004). Sequential learning in neural networks: A review and a discussion of pseudorehearsal based methods. *Intelligent Data Analysis, 8*(3), 301–322.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*(6), 386–408.

Suykens, J. A. K., & Vandewalle, J. (1998). *Nonlinear modeling: Advanced black-box techniques*. Boston: Kluwer Academic Publishers.

Wang, H., Fan, W., & Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *KDD'03: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 226–235). New York, NY, USA: ACM.

Wegman, E., & Smith, J. (1984). *Statistical signal processing*. New York: Marcel Dekker Inc.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning, 23*, 69–101.

Yi, Y., Wu, J., & Xu, W. (2011). Incremental SVM based on reserved set for network intrusion detection. *Expert Systems with Applications, 38*(6), 7698–7707.