

# Recurrent neural network based time series prediction: Particular design problems

Nouha Baccour<sup>1</sup>, Heni Kaaniche<sup>2</sup>, Mohamed Chtourou<sup>3</sup>, and Maher Ben Jemaa<sup>1</sup>

<sup>1</sup> Research unit: Research unit on development and control of distributed applications (ReDCAD), National school of engineers of Sfax.:B.P.W 3038, Sfax-Tunis, Tunisia.

e-mails: nouha.baccour@yahoo.fr (N. Baccour), maher.benjemmaa@enis.rnu.tn (M. Ben Jemaa)

<sup>2</sup> Research unit: Research center on network, image, system, structure and multimedia (CRISTAL)

e-mail: heni.kaaniche@enis.rnu.tn

<sup>3</sup> Research unit: Intelligent Control, design and Optimisation of complex System (ICOS)

e-mail: mohamed.chtourou@enis.rnu.tn

**Abstract**—In this paper, we propose a multi-layer and recurrent neural network using back propagation through time algorithm for training. This neural network is intended for multi-step prediction of time-series. In particular, it estimates the trajectory of an Ad Hoc mobile node, in order to improve routing performance in Ad Hoc wireless networks. We demonstrate the effectiveness of the proposed predictor by testing it on two time-series prediction problems. The selection of the number of inputs and the number of neurons in the hidden layer, which we call the neural network structure, is treated in detail.

**Key-words**—recurrent neural networks, multi-step time series prediction, trajectory prediction.

## I. INTRODUCTION

Time series prediction is an important research and application area. The ability to perform a reliable prediction is important in a wide range of disciplines in the physical sciences, engineering, medical research and economic studies [1-7].

Time series prediction techniques can be classified in two categories: statistical techniques and techniques based on advanced tools as neural networks and fuzzy systems.

Among the statistical prediction techniques, we find Autoregressive (AR), Moving Average (MA) and combined AR and MA (ARIMA) [8]. These techniques showed several limitations, such as their inefficiency for real world problems which are often complex and nonlinear. This is due to the fact that these techniques assume that a time series is generated by a linear process. Thus, they are called linear statistical predictors. The nonlinear statistical predictors such as threshold predictor, exponential predictor, polynomial predictor and bilinear predictor were proposed to add more precision to prediction [8]. However, the selection of the suitable nonlinear model, as well as the computation of its parameters is a difficult task for a practical problem without a priori knowledge about the time series under consideration. Moreover, it has been shown that the capability of the nonlinear model is limited because it is unable to provide a long-term prediction.

In recent years, artificial intelligence tools (e.g. neural networks, fuzzy systems, neuro-fuzzy systems, etc) have been extensively used also exploited for time series prediction [9-11]. In particular, artificial neural networks are frequently exploited for time series prediction problems.

In [9], the authors tested three solutions to predict a financial time series. The time series is divided into fixed size parts. In the first solution, they used a multi-layer neural network. Neural network training has been accomplished using each part of the time series, one after the other, using backpropagation algorithm. The aim of this strategy is to have a general picture on the entire time series, and then make prediction. Feedback connections have been added to the multi-layer neural network to constitute a recurrent neural network (feedback predictor), which has been used as second solution. These feedback connections added memory between the different parts of series, in training phase. The training is carried out with Real-Time Recurrent Learning (RTRL) algorithm [31]. In the third solution, a mixture of expert structures has been used. In this solution a number of statistics on the time series have been established, such as mean and variance, in order to calculate conditional probability values of each series part. These values will be used for prediction. The authors of [10] proposed a flexible neural tree (FNT) model for time series prediction. FNT model is generated initially as a flexible multi-layer feed-forward neural network trained by the backpropagation algorithm. Then, it evolves using a evolutionary procedure to form the most suitable model for prediction, in terms of structure (over-layer connections and different activation functions for different nodes) and corresponding parameters. We note also the work of [11] that introduces a self-organised neuro-fuzzy network which can be applied for time series prediction. This network uses a multi-layer neural network that integrates fuzzy rules like (IF-THEN), to describe the relation between input and output of the network.

The development of artificial neural networks (called also neural networks) was originally inspired by the study of biological neural systems, in particular, by the research on the human brain. The advantage of neural networks compared to other artificial intelligence tools is their ability to learn and then to generalize from their knowledge. A neural network is an information processing system that is capable of treating complex problems of pattern recognition, or of dynamic and nonlinear processes. In particular, it can be an efficient tool for prediction applications. In fact, a number of prior works applied neural network for prediction. However, it should be noted that the

majority of the published results focus on using neural network to perform short-term prediction (one-step prediction). In this study, we present a time series prediction technique based on recurrent neural network and providing long-term prediction (multi-step prediction). This technique can be applied to any time series, in particular location time series.

Location prediction called also mobility prediction is used in many fields like robotics and telecommunication systems. In this last, and particularly in wireless networks, location prediction is the determining of a mobile terminal's future location. The definition of 'location' depends on the nature of the wireless network:

- In *infrastructure networks*, location means the access point to which the mobile terminal is connected. Many location prediction methods are proposed: [12-18]. The main advantage of location prediction is to allocate, in advance, the convenient next access points before the mobile terminal leaves its current one, in order to reduce the time of interruption in communication between terminal mobiles.
- In *no infrastructure networks* called also Ad Hoc network, mobile's location means its geographic coordinates. Location prediction in Ad Hoc networks is a new topic. Its main advantage is to estimate link expiration time [19-20] in order to improve routing performances.

The paper is organized as follows: Section 2 introduces the location prediction as a particular problem of time series prediction. A recurrent neural network for multi-step time series prediction is proposed in Section 3. The choice of the network architecture as well as the training algorithm will be discussed in detail. Section 4 illustrates the effectiveness of the proposed predictor by testing it on two time-series prediction problems. The selection of the number of inputs and the number of neurons in the hidden layer is also treated. Finally, some concluding remarks are presented in Section 5.

## II. TIME SERIES PREDICTION AND LOCATION PREDICTION

Time series is a set of observations from past until present, denoted by  $s(t-i)_{\{i=0..P\}}$ , where  $P$  is the number of observations (patterns). Time series prediction is to estimate future observations, let's  $s(t+i)_{\{i=1..N\}}$ , where  $N$  is the size of prediction window. Location prediction is a particular case of time series prediction. Indeed, let's consider a mobile node, which moves according to a trajectory. This trajectory is defined by mobile locations  $(x(t-i), y(t-i), z(t-i))_{\{i=0..P\}}$ , which represent three time series:  $x(t-i)_{\{i=0..P\}}$ ,  $y(t-i)_{\{i=0..P\}}$  and  $z(t-i)_{\{i=0..P\}}$  (see Fig1). So, location prediction problem can be resolved by the prediction of these three time series i.e.  $x(t+i)_{\{i=0..N\}}$ ,  $y(t+i)_{\{i=0..N\}}$  and  $z(t+i)_{\{i=0..N\}}$ .

## III. RECURRENT NEURAL NETWORK BASED PREDICTION

Time series prediction requires neural network taking into account temporal dimension. This taking into account is only possible by dynamic neural network architectures. Static neural networks are unable to do this kind of treatment. In a dynamic neural network, time can be represented either as an external mechanism, or as an internal mechanism. When time is represented as an external mechanism, prediction window (number of steps or predicted observations) can not exceed one: one step prediction. Whereas, dynamic neural network, when time is represented as an internal mechanism can be used, in order to exploit long-term prediction: multi-step prediction (prediction window is more than one). Among this kind of neural network, we find recurrent neural network, which are the most suitable for the multi-step prediction [9], [6], [21]. Indeed, recurrent neural networks include internal dynamic memory, created from its structure containing cycles. This feature grants a dynamic mapping relating input to output sequences. Recurrent neural networks propose several architectures and training algorithms. The efficiency of the neuronal prediction technique depends on the choice of the architecture and the training algorithm.

### A. Recurrent neural Network architecture

A neural network's performance is highly dependent on its architecture. A neural network architecture is not unique for a given problem. Depending on the problem, it may be appropriate to have more than one hidden layer, feedforward or feedback connections, or in some cases, direct connections between input and output layer [10]. Neural architecture considered in this investigation is a recurrent and multi-layer neural network (see Fig2). This architecture is composed of three layers arranged in a feedforward fashion: The first layer is the input layer which represents the dynamic memory of network. This memory is originated by a feedback between the output layer neuron and the input layer neuron; as well as feedbacks between neurons themselves from input layer. The network input layer can contain external inputs coming from time series observations, as well as estimated input coming from previous network output. In fact, this architecture is used  $N$  times ( $N$ -step prediction) to estimate  $N$  observations of time series. In each prediction step  $k$  ( $k \neq N$ ), network output  $\hat{s}(t+k)$  is stored in the input layer to be able to estimate the following value  $\hat{s}(t+(k+1))$ , that corresponds to prediction step  $(k+1)$ . This storage is due to the feedback connection between the output neuron and a neuron in the input layer. The second layer is called hidden layer. The role of this layer is to store the characteristics of the learned patterns, to increase the freedom degree in the problem resolution and to improve generalization ability of the network, just like the traditional multi-layer networks. The third layer, called the output one, constitutes the output of the network. This output, i.e.  $\hat{s}(t+k)$ , corresponds to the

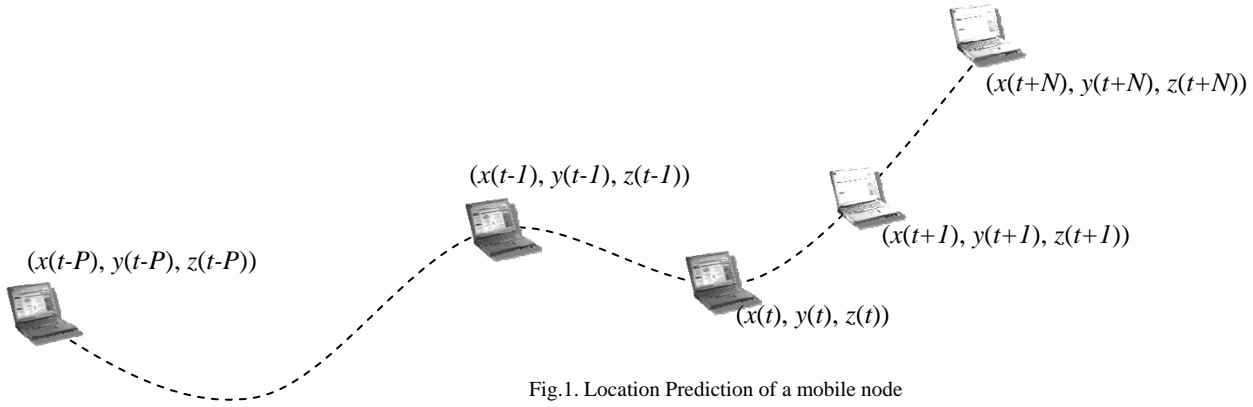


Fig.1. Location Prediction of a mobile node

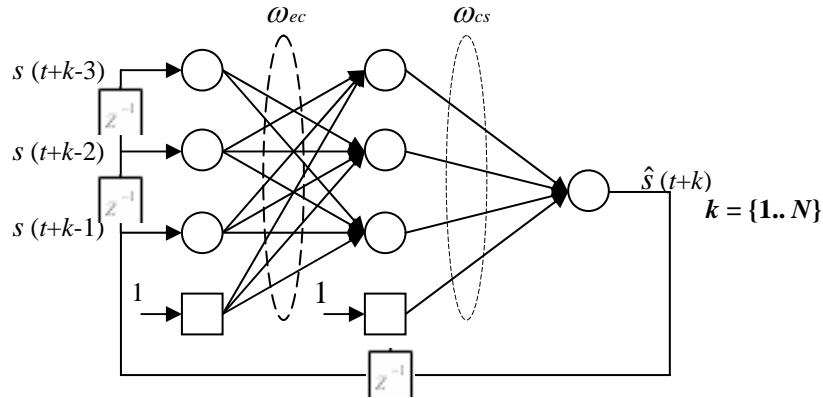


Fig.2. Neural architecture for the prediction

predicted value of  $y$  according to the step  $k$  of prediction, at the moment  $t+k$ .

### B. Notation

To cope with the structural complexity of neural networks, the following notational convention is employed. The input, hidden and output layers contain  $Ne$ ,  $Nc$  and one neurons, respectively.

Let  $\omega_{ec}$  denotes the weight of the link connecting neuron  $e$  in the input layer and neuron  $c$  in the hidden layer and  $\omega_{cs}$  denotes the weight of the link connecting neuron  $c$  in the hidden layer and neuron  $s$  in the output layer,  $e = \{1..Ne\}$ ,  $c = \{1..Nc\}$  and  $s=1$ . Bias terms are included, by adding a bias neuron in both input and hidden layers, whose input have constant value equal to one (see Fig.2).

Let  $v$  denotes a neuron belonging to one of treatment layer (hidden and output i.e.  $v = c$  or  $s$ ). Each neuron  $v$  computes a weighted sum of the inputs denoted by  $I_v$ . The quantity  $I_v$  computed at each neuron becomes then argument of a sigmoid activation function, which resides in the neuron itself. Hence, the value returned by the activation function of neuron  $v$  is  $O_v$  (see Fig.3 and Eqs 1).

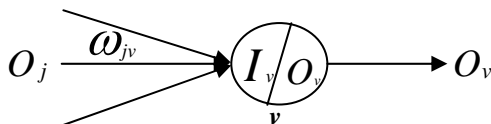


Fig.3. Notation of the neuron's input/output

$$I_v = \sum_j \omega_{jv} \cdot O_j, \quad O_v = f(I_v),$$

$$f: x \mapsto \frac{1}{1+e^{-x}}, \quad f'(x) = f(x) \cdot (1-f(x)) \quad (1)$$

### C. Learning algorithm

In prediction step  $k$ , real observation  $r(t+k)$  is forecasted as  $\hat{s}(t+k)$ , that correspond to the network output. The quantity  $J_k$ , represents the square error function in prediction step  $k$ :

$$J_k = \frac{1}{2} [\hat{s}(t+k) - r(t+k)]^2 \quad (2)$$

The total error function in prediction step  $N$  is the sum of the error functions  $J_k$  that corresponds to prediction step  $k$ :

$$J = \sum_{k=1}^N J_k = \sum_{k=1}^N \frac{1}{2} [\hat{s}(t+k) - r(t+k)]^2 \quad (3)$$

Fig.4 illustrates an example on the error function calculation, for  $Ne = 3$  and  $Nc = 3$ .

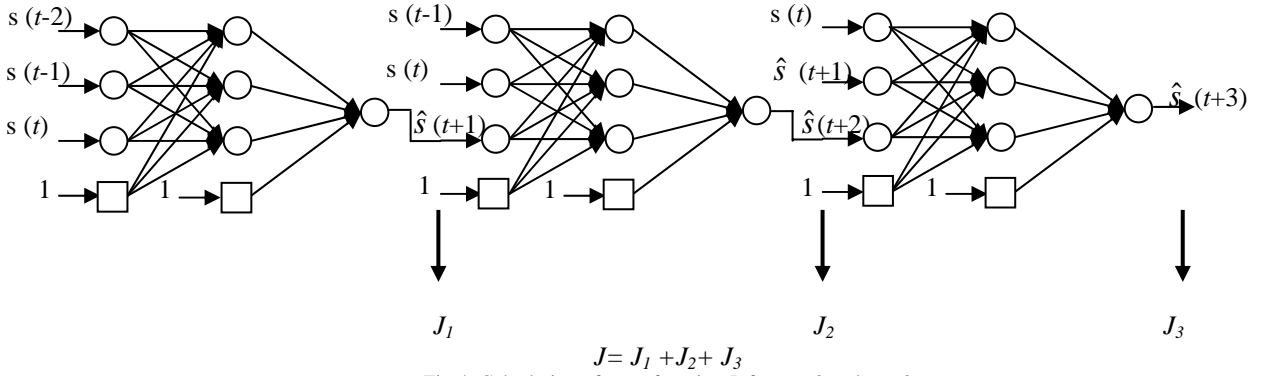


Fig.4. Calculation of error function J, for  $N_e=3$  and  $N_c=3$

The network is trained using BPTT (*Back Propagation Through Time*) [22] algorithm by minimizing the error function  $J$ . The weights can be adapted according to two methods. In the first method, the weights are adapted at every time  $(t+k)$ , that correspond to  $k^{\text{th}}$  prediction step and using the error function  $J_k$  (4).

In the second method, the weights are adapted only at the time  $(t+N)$  using the sum of errors according to times  $\{(t+1), \dots, (t+k), \dots, (t+N)\}$  i.e. the function error  $J$  that correspond to  $N^{\text{th}}$  prediction step (5).

$$\Delta \omega_{uv} = -\varepsilon \frac{\partial J_k}{\partial \omega_{uv}} \quad (4)$$

$$\Delta \omega_{uv} = -\varepsilon \frac{\partial J}{\partial \omega_{uv}} = -\varepsilon \sum_{k=1}^N \frac{\partial J_k}{\partial \omega_{uv}} \quad (5)$$

We adopted the second approach. The weight adjustment is accomplished according to following equations:

$$\begin{aligned} \Delta \omega_{uv} &= -\varepsilon \sum_{k=1}^N \frac{\partial J_k}{\partial \omega_{uv}} \\ &= -\varepsilon \sum_{k=1}^N \frac{d J_k}{d \hat{s}(t+k)} \cdot \frac{d \hat{s}(t+k)}{d \omega_{uv}} \\ &= -\varepsilon \sum_{k=1}^N (\hat{s}(t+k) - r(t+k)) \cdot \\ &\quad \left( \sum_{j=1}^n \frac{\partial \hat{s}(t+k)}{\partial \hat{s}(t+k-j)} \cdot \frac{d \hat{s}(t+k-j)}{d \omega_{uv}} + \frac{\partial \hat{s}(t+k)}{\partial \omega_{uv}} \right) \end{aligned} \quad (6)$$

$n$  denotes the number of forecasted inputs, (because the network output is reinjected in the input layer).

The quantities  $\frac{\partial \hat{s}(t+k)}{\partial \hat{s}(t+k-j)}$  and  $\frac{\partial \hat{s}(t+k)}{\partial \omega_{uv}}$  are given

by:

$$\frac{\partial \hat{s}(t+k)}{\partial \hat{s}(t+k-j)}$$

$$\begin{aligned} &= \sum_{c=1}^{N_c} \frac{\partial \hat{s}(t+k)}{\partial I_c} \cdot \frac{\partial I_c}{\partial \hat{s}(t+k-j)} \\ &= \sum_{c=1}^{N_c} \frac{\partial \hat{s}(t+k)}{\partial O_c} \cdot \frac{\partial O_c}{\partial I_c} \cdot \frac{\partial I_c}{\partial \hat{s}(t+k-j)} \\ &= \sum_{c=1}^{N_c} \frac{\partial \hat{s}(t+k)}{\partial I_s} \cdot \frac{\partial I_s}{\partial O_c} \cdot \frac{\partial O_c}{\partial I_c} \cdot \frac{\partial I_c}{\partial \hat{s}(t+k-j)} \\ &= f'(I_s) \cdot \sum_{c=1}^{N_c} \omega_{cs} \cdot f'(I_c) \cdot \omega_{ec} \end{aligned} \quad (7)$$

The weight  $\omega_{ec}$  denotes the weight of the link connecting the neuron  $e$  in the input layer, whose input is equal to  $s(t+k-j)$  and a neuron  $c$  in the hidden layer.

$$\frac{\partial \hat{s}(t+k)}{\partial \omega_{uv}} = \frac{\partial \hat{s}(t+k)}{\partial I_v} \cdot \frac{\partial I_v}{d \omega_{uv}} \quad (8)$$

Two cases must be considered:

First case:  $\omega_{uv} = \omega_{ec}$  (i.e.  $u$  belongs to input layer and  $v$  belongs to hidden layer)

$$\begin{aligned} \frac{\partial \hat{s}(t+k)}{\partial \omega_{ec}} &= \frac{\partial \hat{s}(t+k)}{\partial I_c} \cdot \frac{\partial I_c}{d \omega_{ec}} \\ &= \frac{\partial \hat{s}(t+k)}{\partial I_s} \cdot \frac{\partial I_s}{\partial O_c} \cdot \frac{\partial O_c}{\partial I_c} \cdot \frac{\partial I_c}{d \omega_{ec}} \\ &= f'(I_s) \omega_{cs} \cdot f'(I_c) \cdot O_e \end{aligned} \quad (9)$$

Second case:  $\omega_{uv} = \omega_{cs}$  (i.e.  $u$  belongs to hidden layer and  $v$  belongs to output layer)

$$\begin{aligned} \frac{\partial \hat{s}(t+k)}{\partial \omega_{cs}} &= \frac{\partial \hat{s}(t+k)}{\partial I_s} \cdot \frac{\partial I_s}{d \omega_{cs}} \\ &= f'(I_s) \cdot O_c \end{aligned} \quad (10)$$

From equations (5) - (10), equation (6) is given by:

$$\frac{\partial J}{\partial \omega_{ec}} = \sum_{k=1}^N (\hat{s}(t+k) - r(t+k)) \cdot \left\{ \sum_{j=1}^n [f'(I_s) \cdot \sum_{c=1}^{Nc} (\omega_{cs} \cdot f'(I_c) \cdot \omega_{jc}) \cdot g_{k-j}] + g_k \right\} \quad (11)$$

Where :

-  $g_k = f'(I_s) \cdot \omega_{cs} \cdot f'(I_c) \cdot O_e$  , when the output is  $\hat{s}(t+k)$   
i.e.  $f'(I_s) =$

$$\hat{s}(t+k-j) \cdot [1 - \hat{s}(t+k-j)]$$

-  $g_{k-j} = f'(I_s) \cdot \omega_{cs} \cdot f'(I_c) \cdot O_e$  , when the output is  $\hat{s}(t+(k-j))$ , i.e.  $f'(I_s) =$

$$\hat{s}(t+k) \cdot [1 - \hat{s}(t+k)]$$

$$\frac{\partial J}{\partial \omega_{cs}} = \sum_{k=1}^N (\hat{s}(t+k) - r(t+k)) \cdot \left\{ \sum_{j=1}^n [f'(I_s) \cdot \sum_{c=1}^{Nc} (\omega_{cs} \cdot f'(I_c) \cdot \omega_{jc}) \cdot g_{k-j}] + g_k \right\} \quad (12)$$

Where:

- $g_k = f'$  when the output is  $\hat{s}(t+k)$
- $g_{k-j} = f'(I_s) \cdot O_c$  , when the output is  $\hat{s}(t+(k-j))$

The following section deals with two case studies where we put the accent in some particular design problems which are the input and hidden layers selection.

#### IV. CASE STUDIES

In this section, two case studies are performed in order to test the developed recursive predictor. The first test case is the « Mackey-Glass » time series. The second one is time series describing locations of an Ad hoc mobile node.

##### A. Mackey-Glass time series

The Mackey-Glass chaotic time series [23] is frequently used to test a time series predictor [24], [25], [10], [26]. This series (see Fig.5) is defined by the following nonlinear differential equation in time:

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-\Delta)}{1+x(t-\Delta)^{10}} \quad (13)$$

The technique of Runge-Kutta of 4<sup>th</sup> order is used to resolve the system. Two values of the parameter  $\Delta$ , lead to a quasi-periodical chaotic behavior:  $\Delta = 17$  and  $\Delta = 30$ . The data base which we will use in this part is obtained with the following conditions:  $\Delta = 17$ ,  $x(0) = 1.2$  and  $x(t-\Delta) = 0$  when  $0 \leq t \leq \Delta$ , with a sampling time equals to one second. Thus, the data base contains 1200 observations of

Mackey-Glass time series: the first 500 observations starting from the 118<sup>th</sup> observation are used as training set, the rest of observations (starting from the 618<sup>th</sup> observation) are used as testing set. The training as well as the test of the multi-step neural predictor will be done on a horizon of six steps.

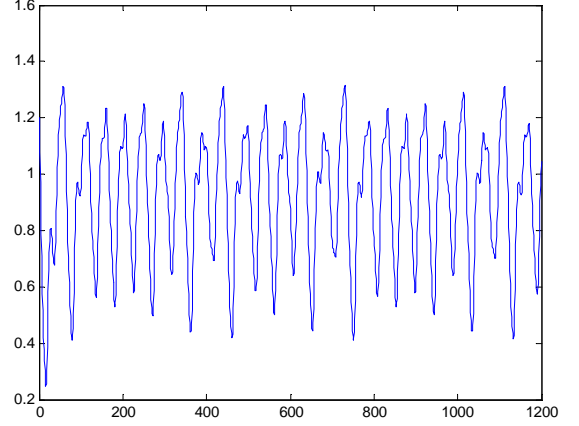


Fig.5. Mackey-Glass time series

##### A.1. Selection of the recurrent neural network structure

The number of inputs  $Ne$ , as well as the number of neurons in the hidden layer  $Nc$ , are both important parameters which need to be appropriately selected. In fact, their values have significant impact on prediction accuracy. An inappropriate choice of  $Ne$  and  $Nc$  can lead to an underlearning or overfitting of the training data. For example for  $Ne=8$  and  $Nc=3$  prediction results, as shown by Fig.6, are not accurate.

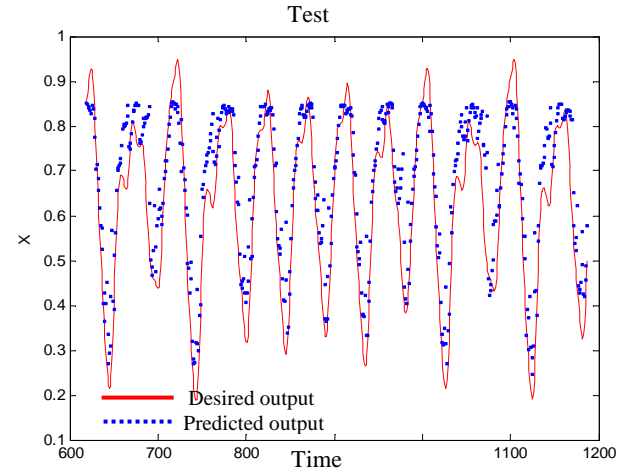


Fig.6. Prediction of Mackey-Glass series with  $Ne=8$ ,  $Nc=3$

To determine the values of  $Ne$  and  $Nc$  which carry out towards the best prediction, we realized two procedures:

The number  $Nc$  depends on the length of subpatterns provided for training, i.e.  $Ne$ , and the number  $Ne$  depends only on the training data [27] [28]. So, in the first procedure which is a heuristic method, we fix  $Nc=3$  and vary  $Ne$ , in order to find the value of  $Ne$  that provides the least

generalization error (i.e. best generalization capability). This error is defined by the following formula.

$$E_{gen} = \sum_{\text{sample}} \sum_{k=1}^6 \frac{1}{2} [\hat{s}(t+k) - r(t+k)]^2 \quad (14)$$

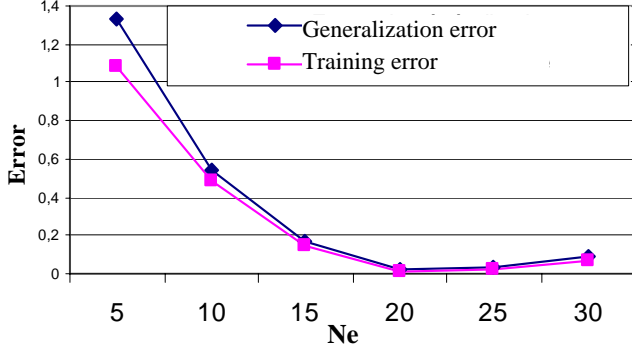


Fig.7. Evolution of the generalization and the training error according to  $N_e$

Fig.7 illustrates the evolution of the generalization error and the training errors according to the number  $N_e$ . This figure shows that the least generalization error is obtained when  $N_e=20$ . When  $N_e$  is higher than 20, the generalization error increases. This is due to an overfitting because the training error also increases.

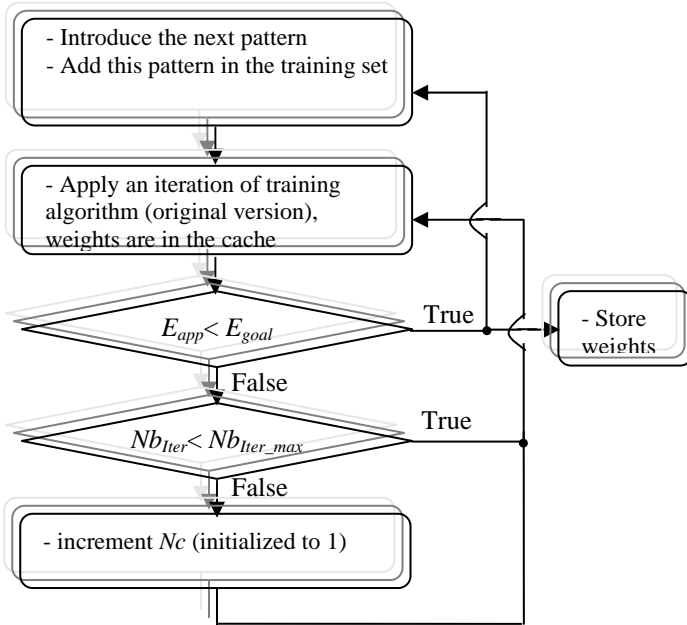


Fig.8. Diagram of the modified training algorithm

$Nb_{Iter}$  : Current number of iterations  
 $Nb_{Iter\_max}$  : Maximum number of iterations  
 $E_{ap}$  : Training error  
 $E_{goal}$  : Training error goal

The second procedure consists in determining the number  $N_c$  ( $N_e$  is fixed to 20). With this intention, we applied the algorithm presented by [29]. In this paper, an incremental

training for multi-layer networks has been proposed. The aim of this algorithm is to force training procedure to converge towards a prefixed error that we note  $E_{goal}$ , with the minimum number of hidden neurons. The diagram of Fig.8 explains the resulting training algorithm.

After applying this training algorithm for a recurrent neural network having 20 input neurons, we obtained the following results: The minimal number  $N_c$  to reach a training error equal to 0.15 is 6 ( $N_c=6$ ). The disadvantage of this algorithm is that it is very slow in its execution.

#### A.2. Testing the selected predictor architecture

After selecting  $N_e=20$  and  $N_c=6$ , another training phase has been considered again. Predictions of the Mackey-Glass series are shown by Fig.9.

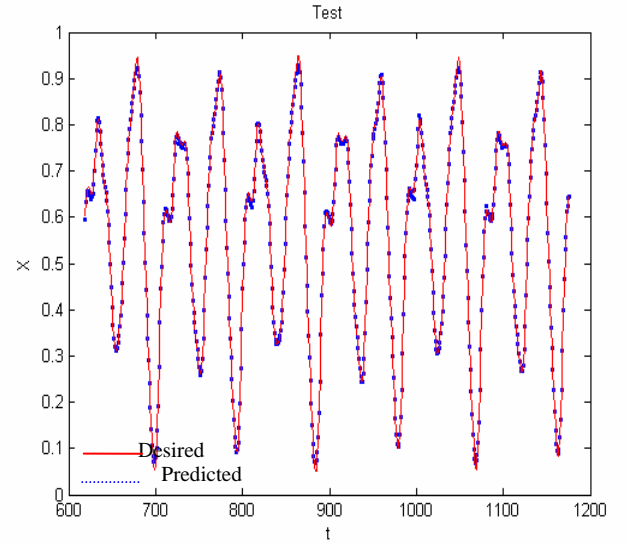


Fig.9. Prediction of Mackey-Glass series with  $N_e=20$  and  $N_c=6$

#### B. Location time series

The movement of an Ad hoc mobile node can be described by three location time series:  $x(t-i)_{\{i=0..P\}}$ ,  $y(t-i)_{\{i=0..P\}}$  and  $z(t-i)_{\{i=0..P\}}$ . The particularity of such a series is that it follows a specific mobility model. In fact, to simulate the movement of an Ad hoc mobile node, the literature offers a set of mobility models [30]: RWM (Random Walk Mobility), RWM (Random Waypoint Mobility), ABSAM (A Boundless Simulation Area Mobility), GM (Gauss-Markov), CSM (City Section Mobility). The majority of the searchers use RWM model (Random Waypoint Mobility) in their Ad hoc networks simulations because it is flexible and simulates in a realistic way the movement of people [30].

We chosen RWM model to construct location time series. In this model [31], a mobile node starts from its current position and chooses randomly a destination position in its mobility territory. Then, it moves straight towards this destination with a constant speed, chosen randomly in an interval of speeds. When it reaches the destination, it remains immobile during a certain period of time then restarts the same movement process (see Fig.10).

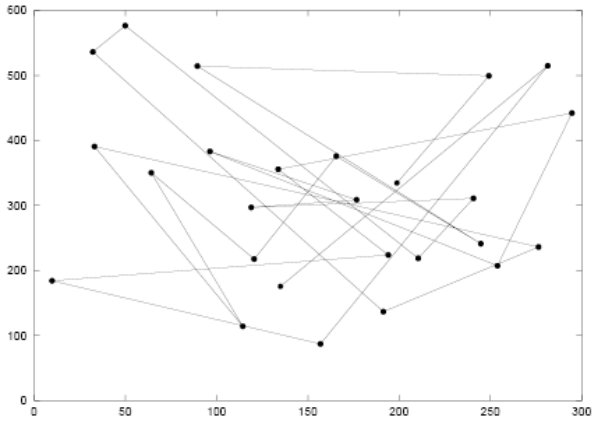


Fig.10. Example of movement of a mobile according to the RWM model

### B.1. Selection the recurrent neural network structure

The selection of the number of input neurons ( $N_e$ ) and of the number of neurons in the hidden layer ( $N_c$ ), can be done by heuristic methods. The method that we adopted is as follows: We consider thirty location time series based on RWM model, containing 400 patterns each one: the first 200 patterns are used for training, the rest of patterns for the test (generalization). Each series is applied to the network while varying  $N_e$  and  $N_c$  in the interval  $[1..30]$ . For each combination of the couple  $(N_e, N_c)$ , we calculate the training error and the generalization error (formula 15). The following observations were then retained:

- Not like  $N_e$ , a small variation of  $N_c$  can affect the prediction accuracy.
- A great value of  $N_e$  can lead to an underlearning and a very small value of  $N_e$  can lead to an overfitting.
- A very good training, i.e. training with a very small error can affect the generalization ability of the neural network, this is why, we chose the generalization error as criterion in the selection of the  $N_e$  and  $N_c$  parameters and not the training error.
- The number  $N_c$  depends on the length of subpatterns provided for training, i.e.  $N_e$ , and the number  $N_e$  depends only on the training data [27] [28]. So we fixed  $N_c$  to determine optimal value of  $N_e$ , i.e. that provides the smallest generalization error. We found that 70% of location time series have optimal  $N_e$  between 5 and 10. For the rest of series (30 %), we note that the generalization error obtained by theirs optimal  $N_e$  is not far (about 0.01) from that of the firsts series. Thus, we chose  $N_e$  equal to 8, being sure that we will not penalize some particular series.
- The variation of  $N_c$  don't affect greatly the prediction accuracy, starting from  $N_c$  equal to 5. The choice of an elevated value of  $N_c$  will increase the number of parameters to be estimated and training delay, without improving considerably the generalization ability. This is why, we chose  $N_c$  equal to 5.

$$E_{train/gener} = \sum_{\text{sample}} \sum_{k=1}^3 \frac{1}{2} [\hat{s}(t+k) - s(t+k)]^2 \quad (15)$$

### B.2. Testing the selected predictor architecture

Let us consider an Ad hoc mobile node which moves according to RWM mobility model with a speed varying in  $[0..20]$ . Its coordinates are recorded each 10s, starting from 0s (initial time), until 4000s. So we obtain two location time series  $x(t-i)_{\{i=0..400\}}$  and  $y(t-i)_{\{i=0..400\}}$ . Now, we will test the predictor on his two location time series in order to forecast the mobile node movement (trajectory). The first 200 co-ordinates are used for training and the rest for generalization. Training as well as generalization (test) was done on a horizon of three steps. The parameters  $N_e$  and  $N_c$  are fixed respectively at 8 and 5. Fig.11 illustrates the test of the neuronal predictor on the two series  $x$  and  $y$ . The forecasted trajectory of the mobile is deduced from predicting  $x$  and  $y$  (Fig.12).

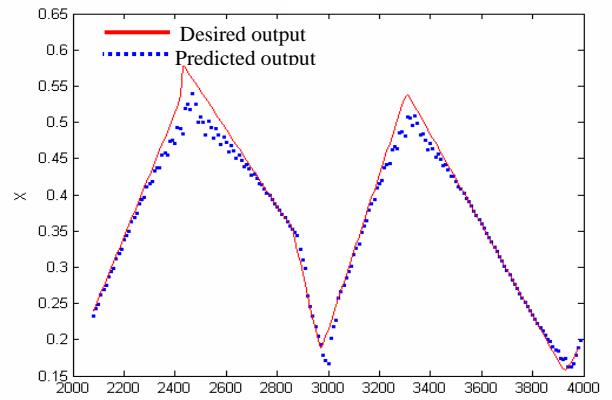


Fig.11.a. Test of time series x

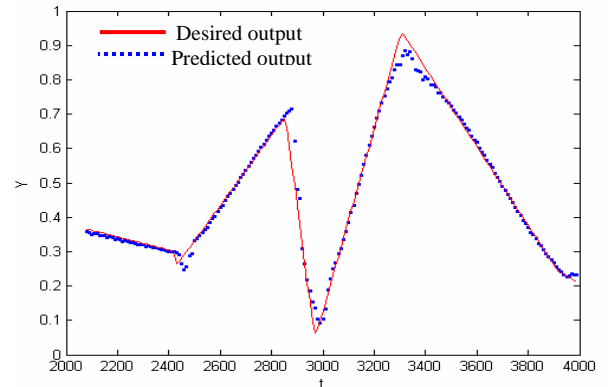


Fig.11.b. Test of time series y

Fig11. Prediction of tow location time series

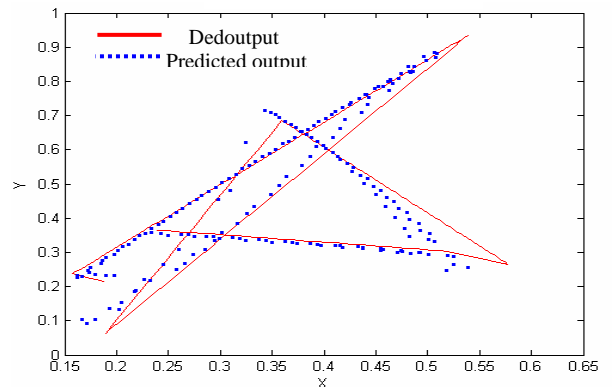


Fig.12. Prediction of the mobile trajectory

## V. CONCLUSION

In this study, a recurrent neural network for multi-step prediction has been proposed. The architecture of this neural predictor is a three-layer network with feedback connections. Back propagation through Time algorithm has been used to train the recurrent neural network. We have demonstrated the effectiveness of the proposed predictor by testing it on two time-series prediction problems. The choice of the recurrent neural network structure has been discussed in detail.

In this study, we have shown also that our neural predictor can estimate an Ad Hoc mobile node trajectory (mobility) by predicting three location time-series, representing its geographic coordinates. In fact mobility prediction is a very important matter insofar as it can improve significantly routing performance in wireless Ad Hoc networks [31], [19], [17].

In a future work, we plan to develop a routing protocol for Ad Hoc network that use mobility prediction to estimate the link expiration time between two neighbored nodes.

## VI. REFERENCES

- [1] N.Lopes, B. Ribeiro, "Part Quality Prediction in an Injection Moulding Process Using Neural Networks", in proceedings of WMC, ISM,1999.
- [2] F.J. Chang, J.M. Liang, Y.C. Chen, "Flood Forecasting Using Radial Basis Function Neural Network", IEEE Transactions on Systems, Man and Cybernetics – Part C : Applications and Reviews, Vol. 31, N° 4, November 2001.
- [3] N. Freitas, I.M. Macleod and J.S. Maltz., "Neural networks for pneumatic actuator fault Detection", Transactions of the SAIEE, vol. 90, n° 1, 1999, p. 28-34.
- [4] J.F. Legrand, et P.Garda, "Prédiction de Trafic GSM par Méthodes Connexionnistes", XIèmes Journées Neurosciences et Science pour l'Ingénieur NSI'2002.
- [5] M. C. Mozer, "Neural network architectures for temporal pattern processing", In A. S. Weigend & N. A. Gershenfeld (Eds.), Time series prediction: Forecasting the future and understanding the past (pp. 243-264). Redwood City, CA: Sante Fe Institute Studies in the Sciences of Complexity, Proceedings Volume XVII, Addison-Wesley Publishing, 1993
- [6] T.G. Barbounis and J.B. Theocharis, "Locally recurrent neural networks for long-term wind speed and power prediction", Science, Neurocomputing pp 466–496, 2005.
- [7] G. Corani, "Air quality prediction in Milan: feed-forward neural networks, pruned neural networks and lazy learning", Science, Ecological Modelling pp 513–529, 2005.
- [8] O. Voitcu, Y. Wong, "On the construction of a nonlinear recursive predictor", Science B.V., Journal of Computational and Applied Mathematics, 2004.
- [9] S. Ymlu, F. Gurgen, N.Okay, "A comparison of global, recurrent and smoothed-piecewise neural models for Istanbul stock exchange (ISE) prediction", science B.V., Proc Pattern Recognition Letters 26 2093–2103, 2005.
- [10] Y. Chen B. Yang J. Dong A. Abraham, "Time-series forecasting using flexible neural tree model", Science, Information Sciences pp 219–235, 2004.
- [11] C.J. Lin and Y.J. Xu, "A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications", Science, Fuzzy Sets and Systems, 2 September 2005.
- [12] S. Tabbane, "An alternative strategy for location tracking", IEEE J.Selected Areas in Comm. Vol.13, no. 5, pp. 880–892, June 1995.
- [13] G. Y. Liu and G. Q. Maguire Jr., "A predictive mobility management algorithm for Wireless mobile computation and communication", Proc IEEE Int. Conf. Universal Personal Commun., 1995.
- [14] T. Liu, Paramvir Bahl, "Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks", IEE Journal on Selected Areas In Communications, Vol. 16, No. 6, August 1998.
- [15] M.M. Islam, M. Murshed, L.S. Dooley, "New mobility based call admission control with on-demand borrowing scheme for QoS provisioning", in: IEEE International Conference on Information Technology: Coding and Computing'2003 (ITCC'2003), Las Vegas, Nevada, USA, pp. 263–267, April 2003.
- [16] A. Bhattacharya and S.K. Das, "Lezi-Update: An Information-Theoretic Approach to Track Mobile Users in PCS Networks", Mobile Computing and Networking, pp. 1-12, 1999.
- [17] S.AlaaEldin, K.Anup, "Handoff management in wireless data networks using topography-aware mobility prediction", Science, J. Parallel Distrib. Comput. 65 pp 963 – 982, (2005).
- [18] V. Kumar and P. Venkataram, "A Prediction Based Location Management Using Multi-Layer Neural Networks", J. Indian Inst. of Science, vol. 82, no. 1, pp. 7-21, 2002.
- [19] W. Su, S.J. Lee, G. Mario, "Mobility prediction and routing in Ad Hoc wireless networks", Proc. IEEE MILCOM, 2000.
- [20] M.Gerharz, D.de.Waal and P.Martini from Institute of Computer Science IV, University of Bonn et P.James from Nokia Research Center, "Strategies for Finding Stable Paths in Mobile Wireless Ad Hoc" Networks, IEEE, 2003.
- [21] W. Liu, L.L. Yang and L. Hanzo,"Recurrent Neural Network Based Narrowband Channel Prediction", School of ECS, University of Southampton, SO17 1BJ, UK, 2003, <http://www-mobile.ecs.soton.ac.uk>.
- [22] P. Werbos, "Backpropagation Trough time : What it does and how to do it", Proceedings IEEE, Vol. 78, 1990.
- [23] M.Mackey et L.Glass, «Oscillations and Chaos in Physiological Control System », Science, pp. 197-287, 1977.
- [24] Y.Chena , B.Yanga, J.Donga, "Time-series prediction using a local linear wavelet neural network", Science, Neurocomputing 69, pp 449–465, 2006.
- [25] J.Mao, J.Zhang, Y.Yue, and H.Ding, "Adaptive-Tree-Structure-Based Fuzzy Inference System", IEEE, Transactions On Fuzzy Systems, Vol. 13, NO 1, Fefrier 2005.
- [26] M. R. Zemouri, "Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques" : Application à la e-maintenance, THESE présentée à L'UFR des Sciences et Techniques de l'Université de Franche-Comté, 28 Novembre 2003.
- [27] B. P. Vijay Kumar and P. Venkataram, "Prediction-based location management using multilayer neural networks",J. Indian Institute of Science , 82, pp7-21, 2002.
- [28] G. Peter Zhang, B. Eddy Patuwo and M. Y. Hu, "A simulation study of artificial neural networks for nonlinear time series forecasting", Computers Op. Res., 28, pp381-396,2001.
- [29] D.Liu, T.S.Chang, and Y.Z.hang, "A Constructive Algorithm for Feedforward Neural Networks With Incremental Training", IEEE Transactions on Circuit And System—I:Fundamental Theory And Applications, Vol. 49, No. 12,pp 1876-1879, December 2002.
- [30] T.Camp J.Boleng V.Davies, "A Survey of Mobility Models for Ad Hoc Network Research", Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking:Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002.
- [31] R.J. Williams, D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks , Neural Computation", vol.1, juin 1989, p. 270-280.
- [32] S. William, G. Mario, "IPv6 flow handoff in Ad Hoc wireless networks using mobility prediction", Proc. IEEE Global Telecommunications Conference (GLOBECOM), pp. 271–275, December, 1999.