Journal of Computer Science 3 (9): 747-753, 2007
ISSN 1549-3636

Data-Driven Forecasting Schemes: Evaluation and Applications

Josip Vrbanek Jr. and Wilson Wang
Mechanical Engineering, Lakehead University, Thunder Bay, Ontario, P7B 5E1, Canada

Abstract: A reliable multi-step predictor is very useful to a wide array of applications to forecast the behavior of dynamic systems. The objective of this paper is to develop a more robust data-driven predictor for time series forecasting. Based on simulation analysis, it is found that multi-step-ahead forecasting schemes based on step inputs perform better than those based on sequential inputs. It is also realized that recurrent neural fuzzy predictor is superior to both recurrent neural networks and feedforward networks. In order to enhance the forecasting convergence, a hybrid training technique is proposed base on the real-time recurrent training and weighted least squares estimate. The developed predictor is also implemented for real-time applications in material property testing. The investigation results show that the developed adaptive predictor is a reliable forecasting tool. It can capture the system's dynamic behavior quickly and track the system's characteristics accurately. Its performance is superior to other classical data-driven forecasting schemes.

Key words: Recurrent neural fuzzy paradigm, multi-step prediction, hybrid learning, material property testing

INTRODUCTION

A reliable multi-step forecasting tool is very useful to a wide array of real-world applications to predict the future states of a dynamic system. In industrial applications, for example, forecast information can be used to schedule repairs and maintenance for important fabrication facilities so as to prevent production performance degradation, malfunction, or even catastrophic failures. A robust predictor is also critically needed in applications such as time consuming operations (e.g., material property testing), structure remaining life estimation, and earthquake forecasting. System state forecasting utilizes some available observations to predict the future states of a dynamic system. The representative observations can be derived from different information carriers, such as vibration, temperature, and acoustic measurement data, by using appropriate signal processing techniques.

Several techniques have been proposed in the literature for time series prediction [1]. The classical approaches are the use of stochastic (or the extended) models [2, 3]; However, an accurate analytical model is usually difficult to derive for a complex dynamic system, especially under noisy and uncertain environment such as in real-world industrial applications. Since the last decade, more interests in time series forecasting have focused on the use of knowledge-based data-driven paradigms, such as various neural networks (NNs) [4-6] and neural fuzzy (NF) paradigms [7-9]. Even though the data-driven predictors have demonstrated their superior potential in many applications over the classical numerical models, as a matter of fact, these currently available predictors are mainly for one-step-ahead prediction in real applications. Advanced research needs to be done in several aspects before a multi-step-ahead predictor can be effectively applied to real-time industrial applications. Correspondingly, the objectives of this work are to 1) evaluate the input patterns on the performance of multi-step prediction paradigms, 2) improve convergence for a recurrent NF predictor by adopting a hybrid training technique, and 3) implement the proposed recurrent NF predictor for real-world applications.

MATERIALS AND METHODS

The knowledge-based data-driven forecasting tools interested in this work include feedforward NN, recurrent NN, as well as a weighted recurrent NF scheme that is an extension to that as proposed in [8]. A brief description is given in this section for these related predictors, as well as the suggested hybrid training technique.

Corresponding Author: Wilson Wang, Dept. of Mechanical Engineering, Lakehead University, Thunder Bay, Ontario, Canada, P7B 5E1

Recurrent NN Predictor: System state forecasting is to predict the future states of a dynamic system based on its current and some previous states. Since the performance of a knowledge-based data drive forecasting scheme is directly related to its reasoning structure (or the number of network parameters), to facilitate the comparison study among different predictors, the number of the network parameters will be kept comparable.

The network architecture of the recurrent NN predictor is as shown in Fig. 1. Consider (n+1) input state variables $\{x_0 \ x_{-s} \ \cdots \ x_{-ns}\}$, which represent the current (x_0) and the previous states of a dynamic system with a time step s. If ten inputs are employed in the input layer, that is, n = 9, ten nodes will be used in the recurrent context layer. The s-step-ahead state, x_{+s}, will be given from the output node. In total, there are 151 parameters to be updated in this predictor: 100 input weights (w_{ij}^1), 10 output weights (w_{ij}^2), 10 recurrent weights (w_{ij}^3), and 31 biases.

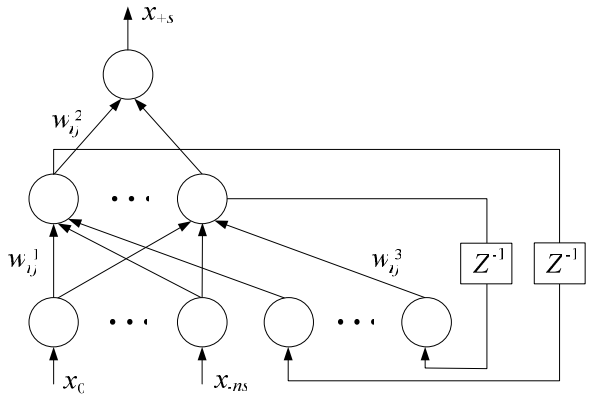

Fig. 1: Network architecture of the recurrent NN predictor

NN-based Predictor: Different from the recurrent NNs, the feedforward NN has no feedback links, or without context layer as illustrated in Fig. 1. To make the networks comparable, the feedforward NN predictor also has 10 input nodes, which has a 10-12-1 three layer network architecture. The total number of parameters to be updated is 155, that is, 120 input weights, 12 output weights, and 23 biases.

Weighted Recurrent NF Predictor: An NF reasoning scheme applies a set of fuzzy IF-THEN rules to describe the input/output data mapping and forecast the future states of a dynamic system. The fuzzy system parameters are optimized by an adaptive training algorithm. According to some advanced investigation, it is found that the first order TS fuzzy paradigm is more flexible in modeling higher order nonlinear systems [8], and will be adopted in this work. To simplify forecasting reasoning, two membership functions (MFs), *small* and *large*, are assigned to each input state variable. The s-step-ahead state of the dynamic system x_{+s} can be formulated by:

$$\Re_j: \text{IF } (x_0 \text{ is } A_1^j) \text{ and } (x_{-s} \text{ is } A_2^j) \ \cdots \ (x_{-ns} \text{ is } A_m^j)$$
$$\text{then } (x_{+s} = c_0^j x_0 + c_1^j x_{-s} + \cdots + c_n^j x_{-ns} + c_{n+1}^j) \qquad (1)$$

where $\Re_j$ denotes the jth fuzzy rule, j = 1, 2, $\cdots$, M, M is the total number of fuzzy rules; A_k^j is the jth fuzzy set for x_{-is}, i = 0, 1, $\cdots$, n, k = 1, 2, $\cdots$, m, where $m = 2n$ in this case. To make the network comparable with the aforementioned NN predictors, four input state variables are applied in this case, that is, i = 0, 1, $\cdots$, 3, $M = 16$, and $m = 2n = 8$, and the number of parameters to be updated is 104.

The network architecture of this weighted recurrent NF predictor is schematically shown in Fig. 2. It is a 5-layer network in which each node performs a particular activation function on the incoming signals. The links have unity weights unless specified. Layer 1 is the input layer. Each node in layer 2 acts as an MF. Different from the general NF schemes [7] and the predictor as proposed in [8], this recurrent NF system has a weighted feedback link to each node in layer 2. The recurrent context units copy the activations of output nodes from the previous time steps, and allow the network to remember cues from the past.

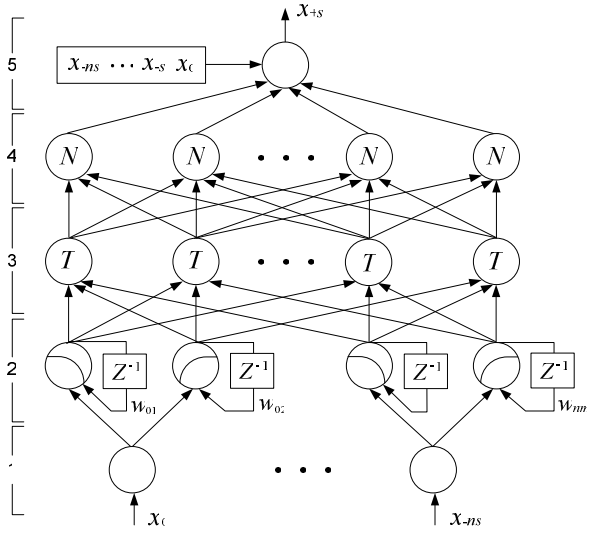

Fig. 2: Network architecture of the weighted recurrent NF predictor

Given an MF, the node actual input at the *t*th time instant is

$$X_{-is}^{(t)} = x_{-is}^{(t)} + w_{ik} A_k^j\left(x_{-is}^{(t-1)}\right) \quad (2)$$

where $x_{-is}^{(t)}$ and $x_{-is}^{(t-1)}$ are, respectively, the input x_{-is} at the *t*th and (*t*-1)th time instants, respectively; $A_k^j\left(x_{-is}^{(t-1)}\right)$ is the node output (membership grade) in the last time step, and w_{ik} is the weight of the feedback link; $i = 0, 1, \ldots, n$; $k = 1, 2, \ldots, m$; and $t = 0, 1, \ldots, P$-1, where P is the total number of time instants (or training data sets) of interest.

Each node in layer 3 performs a fuzzy *T*-norm operation. If a max-product operator is applied in this case, the rule firing strength is

$$\mu_j = \prod_{i=0}^{n} A_k^j\left(X_{-is}^{(t)}\right) \quad (3)$$

where , $j = 1, 2, \ldots, M$; $i = 0, 1, \ldots, n$.

All the rule firing strengths are normalized in layer 4. After a linear combination of the input variables in layer 5, the predicted output x_{+s}, after s steps, is computed by using the centroid defuzzification:

$$x_{+s} = \sum_{j=1}^{M} \overline{\mu}_j (c_0^j x_0 + c_1^j x_{-s} + \cdots + c_n^j x_{-ns} + c_{n+1}^j) \quad (4)$$

where $\overline{\mu}_j = \dfrac{\mu_j}{\sum_{j=1}^{M} \mu_j}$ is the normalized firing strength of the *j*th rule. The fuzzy system parameters w_{ik} and c_i^j and A_k^j are optimized by the related training algorithms as discussed as follows.

A Hybrid Training Technique: Once the NF predictor structure is created, a forecasting paradigm should be properly trained to optimize the input/output mapping [10-12]. Since the suggested recurrent NF scheme in (1) contains both linear consequent weights and nonlinear premise parameters, to improve the training convergence, a hybrid training technique is suggested in this paper: the nonlinear fuzzy parameters in the recurrent context layer are optimized by using a real-time recurrent learning (RTRL), whereas the consequent linear parameters are fine-tuned by employing a weighted least squares estimate (LSE). A hybrid training technique is superior to classical single training algorithms, because it possesses randomness that may help to escape from local minima and it is also necessary to accommodate different characteristics in time-varying systems.

RTRL Algorithm for Recurrent Parameters: The RTRL is an approach to training a recurrent network by adjusting weights along the error gradient while the network continues to perform its signal processing function, rather than at the end of the presented sequences. Details of this algorithm can be found in [13, 14].

At time instant *t*, the output to the activation function of each node in the recurrent context layer of the NF scheme is described by

$$y_k(t) = A_k^j\left(X_{-is}^{(t)}\right) \quad (5)$$

where $A_k^j(\circ)$ is the activation function, which is an MF in the recurrent layer; $X_{-is}^{(t)}$ is the net input to node *k* in the context layer at time instant *t*, as determined by (2). If there is no such input for that node at that time instant, then $x_{-is}^{(t)}$ is set to 0. The total error function to be minimized is

$$E_1 = \sum_{t=0}^{P-1} E_1(t) = \frac{1}{2} \sum_{t=0}^{P-1} \sum_k E_{1k}^2(t) \quad (6)$$

where $E_{1k}(t) = b_k(t) - y_k(t)$ if node k has a desired output $b_k(t)$ at time *t*; otherwise $E_{1k}(t) = 0$.

Equation (6) is the error measure for node *k* at time instant *t*. Since the gradient of E_1 separates in time, by taking the full change in the $w_{ik}(t)$ as the time summation of $\Delta w_{ik}(t)$, the gradient decent method can be performed by defining

$$\Delta w_{ik}(t) = -\eta \frac{\partial E_1(t)}{\partial w_{ik}} = \eta \sum_t E_{1k}(t) \frac{\partial y_k(t)}{\partial w_{ik}} \quad (7)$$

where η is the learning rate.

The derivative $\dfrac{\partial y_k(t)}{\partial w_{ik}}$ in (7) can now be found by differentiating the dynamical rule (5), based on the assumption that the initial state of the network has no functional dependence on the weights, that is, $\dfrac{\partial y_k(0)}{\partial w_{ik}} = 0$. $\dfrac{\partial y_k(t)}{\partial w_{ik}}$ needs to be calculated recursively at each time instant. In general, for the error propagation at time instant *t*, $\dfrac{\partial y_k(t-1)}{\partial w_{ik}}$ is already available from the calculation at the previous time instant *t*-1. Therefore, by trying to minimize each individual E_{1k}, we can recurrently find the gradient $\dfrac{\partial E_{1k}}{\partial A_k^j}$ at each time instant. There is no need to wait until the end of the presented sequence. Since it is an

approximation of the gradient method, the learning rate should be kept small (as a result, the learning may be a little slower than a pure gradient decent algorithm).

LSE for Consequent Linear Parameters: Consider the tth input data pair $\{x_0^{(t)}\ x_{-s}^{(t)}\ \cdots\ x_{-ns}^{(t)}\}$, t = 0, 1, …, P-1, P is the total number of training data pairs. The forecast state after s steps, $x_{+s}^{(t)}$, is computed by (4). If $d^{(t)}$ denotes the desired output state, which represents the actually observed output value for the (t+1)th data set, that is, $d^{(t)} = x_0^{(t+1)}$, the forecasting error is defined as

$$E_2 = \sum_{t=0}^{P-1} E_{2t} = \frac{1}{2}\sum_{t=0}^{P-1} \lambda (x_{+s}^{(t)} - d^{(t)})^2 \quad (8)$$

where $\lambda \in [0.95\ 1]$ is a weight factor to highlight the contribution of the recent data sets from a time-variant system.

Given the values of the MF parameters and P training data pairs to the adaptive predictor, $\{x_0^{(t)}\ x_{-s}^{(t)}\ \cdots\ x_{-ns}^{(t)}\ d^{(t)}\}$, k = 0, 1, 2, . . ., P-1, P linear equations in terms of the consequent parameters $\boldsymbol{\theta}$ can be formed as:

$$\boldsymbol{R}\boldsymbol{\theta} = \boldsymbol{d}, \quad (9)$$

where $\boldsymbol{R}$ is the resulting matrix from the inference operation of the NF predictor:

$$\boldsymbol{R} = \begin{bmatrix} \bar{\mu}_1^{(1)} x_0^{(1)} & \bar{\mu}_1^{(1)} x_{-s}^{(1)} & \cdots & \bar{\mu}_M^{(1)} x_{-ns}^{(1)} & \bar{\mu}_M^{(1)} \\ \bar{\mu}_1^{(2)} x_0^{(2)} & \bar{\mu}_1^{(2)} x_{-s}^{(2)} & \cdots & \bar{\mu}_M^{(2)} x_{-ns}^{(2)} & \bar{\mu}_M^{(2)} \\ & & \vdots & & \\ \bar{\mu}_1^{(P)} x_0^{(P)} & \bar{\mu}_1^{(P)} x_{-s}^{(P)} & \cdots & \bar{\mu}_M^{(P)} x_{-ns}^{(P)} & \bar{\mu}_M^{(P)} \end{bmatrix}; \quad (10)$$

$\boldsymbol{\theta}$ is the consequent parameter set whose elements are to be updated:

$$\boldsymbol{\theta} = \begin{bmatrix} c_0^1\ c_1^1\ c_2^1\ \cdots\ c_n^M\ c_{n+1}^M \end{bmatrix}^T; \quad (11)$$

$\boldsymbol{d}$ represents the vector of the desired system states:

$$\boldsymbol{d} = \begin{bmatrix} d^{(1)}\ d^{(2)}\ \cdots\ d^{(P)} \end{bmatrix}^T. \quad (12)$$

Because the row vectors in $\boldsymbol{R}$ and the associated elements in $\boldsymbol{d}$ are obtained sequentially, $\boldsymbol{\theta}$ in (9) can be computed recursively. For the objective function with respect to the adjustable parameters $\boldsymbol{\theta}_t$ at the current time instant t,

$$E_2(\boldsymbol{\theta}_t) = \frac{1}{2}\sum_{t=0}^{P-1} \lambda \left[x_{+s}(\boldsymbol{\theta}_t) - d^{(t)} \right]^2 \quad (13)$$

where $x_{+s}(\boldsymbol{\theta}_t)$ is determined by (4); $\boldsymbol{R}_t$ in (10) is the resulting matrix from the corresponding fuzzy inference operation at time t. The LSE is computed by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{S}_{t+1}\boldsymbol{R}_t(d_t - \boldsymbol{R}_t^T\boldsymbol{\theta}_t) \quad (14)$$

$$\boldsymbol{S}_{t+1} = \frac{1}{\lambda}\left[\boldsymbol{S}_t - \frac{\boldsymbol{S}_t\boldsymbol{R}_t\boldsymbol{R}_t^T\boldsymbol{S}_t}{\lambda + \boldsymbol{R}_t^T\boldsymbol{S}_t\boldsymbol{R}_t} \right]. \quad (15)$$

where t = 0, 1, . . ., P-1. $\lambda \in [0.95\ 1]$ is the weight factor. The optimal estimate of $\boldsymbol{\theta}$ is $\boldsymbol{\theta}_P$, whereas $\boldsymbol{\theta}_0 = \boldsymbol{0}$. The initial conditions for the covariance matrix $\boldsymbol{S}_t$ is $\boldsymbol{S}_0 = \alpha\boldsymbol{I}$, $\alpha \in [10^2\ \ 10^6]$, and $\boldsymbol{I}$ is an identity matrix.

RESULTS AND DISCUSSION

Performance Comparison in terms of Step Properties: The first objective of this comparison study is to investigate the effects of the input patterns on the forecasting performance, that is, with the sequential inputs and the step inputs, as long as the same number of inputs is used. The comparison is performed by a series of simulation tests based on data sets from the Mackey-Glass equation [15],

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t) \quad (16)$$

The data sets from (16) have specific natures such as chaotic, non-periodic, and non-convergence, and have been used as benchmark data in forecasting research to evaluate the performance of predictors. Many simulation tests have been conducted based on data from (16) corresponding to different initial conditions. Each test is repeated over 10 times to mitigate random effects related to initial values of network weights. Fig. 3 summarizes the compassion results for predictors based on feedforward NNs and recurrent NNs over ten steps, in terms of the root mean squares errors (RMSE). Both predictors are trained by the use of the Levenberge-Marquet algorithm [14]. Comparing the curves 1 vs. 2, and curves 3 vs. 4, it is clear that both the NN and the recurrent NN predictors based on step inputs perform better than those based on the sequential inputs. The main reason is that, if the same number of inputs is employed, the step input pattern can cover a wider information horizon. On the other hand, it is seen that the performance of the feedward NN predictor based on step inputs (curve 2) is a little better than that from the recurrent NNs (curve 4). However, the recurrent NN predictor takes much longer time in training operations, or the average time of 174 seconds compared with 18 seconds for the feedforwar NNs over 100 epochs; that is mainly due to its recurrent network characteristics. Furthermore, the forecasting performance of the feedforward NN

predictor does not vary dramatically over a wide range of horizon.

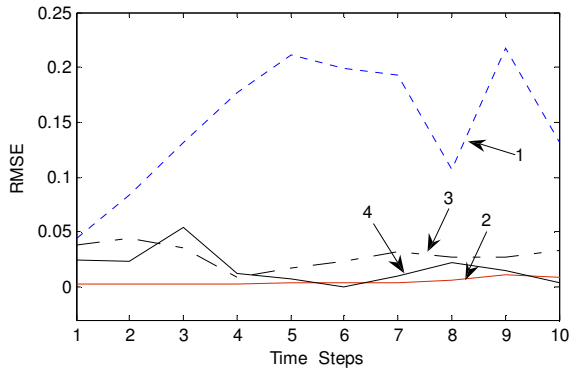

Fig. 3: Performance comparison: 1-feedforward NNs with sequential inputs, 2-feedforward NNs with step inputs, 3-recurrent NNs with sequential inputs, 4-recurrent NNs with step inputs

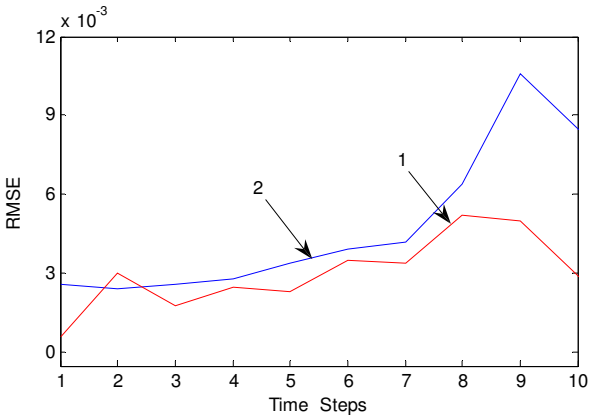

Fig. 4: Performance comparison between the recurrent NF predictor (curve 1) and the feedforward NN predictor (curve 2)

Fig. 4 shows the comparison between the feedforward NN predictor and the recurrent NF predictor, both with step inputs. The recurrent NF is trained by using a general hybrid algorithm [9], that is, premise and consequent fuzzy parameters in (1) are trained by the gradient decent algorithm and LSE [14], respectively. As shown in Fig. 4, it is clear that the recurrent NF predictor performs better than the feedforward NNs, due to its effective reasoning and appropriate training. Furthermore, recurrent context nodes can store cues from previous time steps. This function of recurrent networks is especially valuable for predictors with limited and step inputs (i.e., $s>1$) to provide more information to the network so as to improve forecasting accuracy. On the other hand, the training speed in both predictors is comparable, that is, the average time is 17 and 18 seconds over 100 epochs, respectively, for the recurrent NF and feedforward NN predictors.

Performance Comparison in terms of Training Algorithms: The suggested hybrid adaptive training method adopts two algorithms: the RTRL and the LES. In each training epoch, the fuzzy system parameters in the recurrent context are adaptively optimized by applying the RTRL (7) in the back pass, whereas the linear consequent parameters are updated by using the weighted LSE (14) in the forward pass. The RTRL trains the recurrent network while the network continues to perform its signal processing function, rather than at the end of the presented sequences. In initialization, the synaptic weights of the algorithm are set to small values (0.025 in this case) from a uniform distribution, while both the state vector and its partial derivative (with respect to the weight vector) are set to zero.

As an example, consider a Mickey-Glass data set from (16) with initial conditions of $\tau = 30$, $x(0) = 1.2$, $dt = 1$, and $x(t) = 0$ for $t < 0$. If 1600 data pairs are selected, the first 800 samples are used for training and the remaining 800 samples for testing the identified model. Fig. 5 shows the forecasting performance of the NF predictor based on the hybrid training of the gradient-LSE over 100 epochs. Fig. 6 demonstrates the corresponding results by using the hybrid learning with the RTRL-LSE. Apparently, the latter is superior to the former (with the root mean squares error of 0.0258 vs. 0.0472).

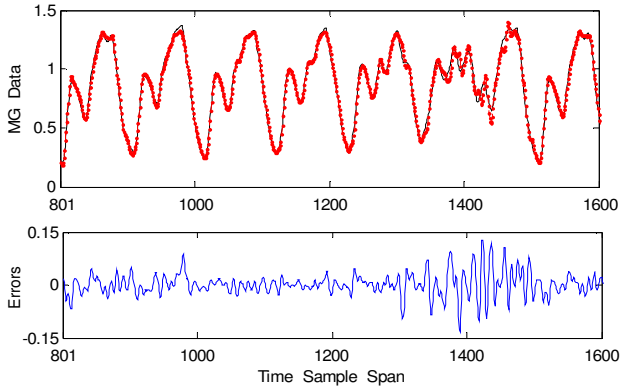

Fig. 5: Upper graph: The forecasting result (ten-steps-ahead) of a Mackey-Glass data set (solid curve) by using the gradient-LSE algorithm (dotted curve). Lower graph: the forecasting errors

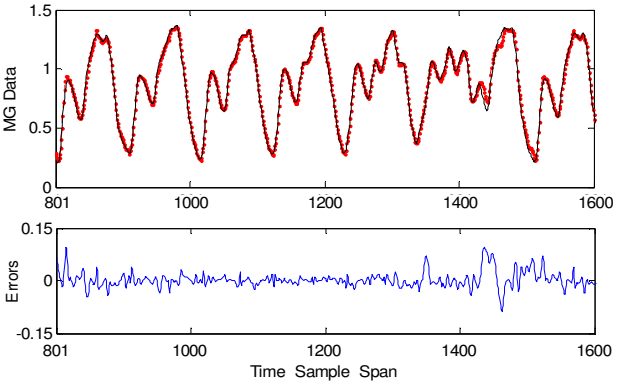

Fig. 6: Upper graph: The forecasting result (ten-steps-ahead) of a Mackey-Glass data set (solid curve) by using the RTRL-LSE algorithm (dotted curve). Lower graph: the forecasting errors

From comparison it is seen that the suggested adaptive hybrid method is an effective training method. It integrates the benefits of both the RTRL and weighted LSE, and possesses the randomness that is helpful to escape from local minima. The RTRL-LSE algorithm is fast in training, compared with other commonly used training algorithms, such as the Levenberge-Marquet method [14], due to its simple gradient-related operations. In fact, it is a little slower than the gradient-LSE training (17 seconds vs. 21 seconds per 100 epochs), because of its reliance on instantaneous estimates of gradients and relatively smaller learning rate. But this hybrid training algorithm can provide high training performance in terms of rate of convergence and quality of solution.

Material Fatigue Property Forecasting: Another objective of this work is to implement the proposed recurrent NF predictor for real-world applications. Material fatigue property testing is used as an example. Material fatigue testing is usually a very time-consuming process. A reliable forecasting tool is critically needed quickly estimate the material's dynamic characteristics so as to effectively shorten the experimental time. This test is conducted on the experimental setup similar to Fig. 7. An aluminum specimen has a thickness of 3 mm, and its both ends are clamped to the test rig. The monitoring time-step is set at s = 3,000 cycles. Fig. 8 shows the 6-steps-ahead forecasting results of the crack propagation trend, based on the feedforward NN predictor (Fig. 8a) and the recurrent NF scheme (Fig. 8b), respectively. It is seen that feedforward NN predictor (Fig. 8a) generates larger forecasting errors due to its slow convergence, and has a relatively poor robustness especially to the high frequency variations. The recurrent NF predictor (Fig. 8a), however, can overcome this problem by an adaptive training process. It can catch system's dynamic behavior quickly and track the propagation trend accurately.

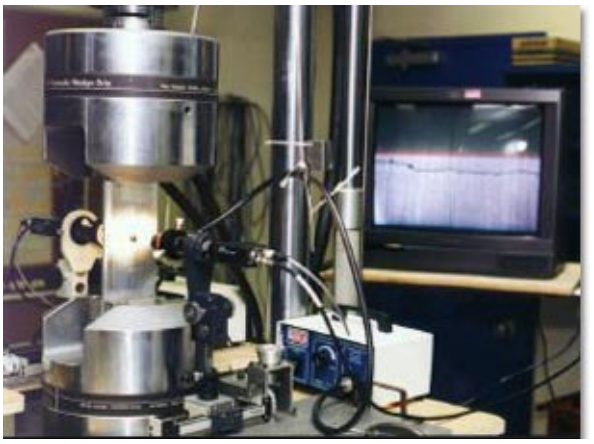

Fig. 7: The experimental setup for material property testing

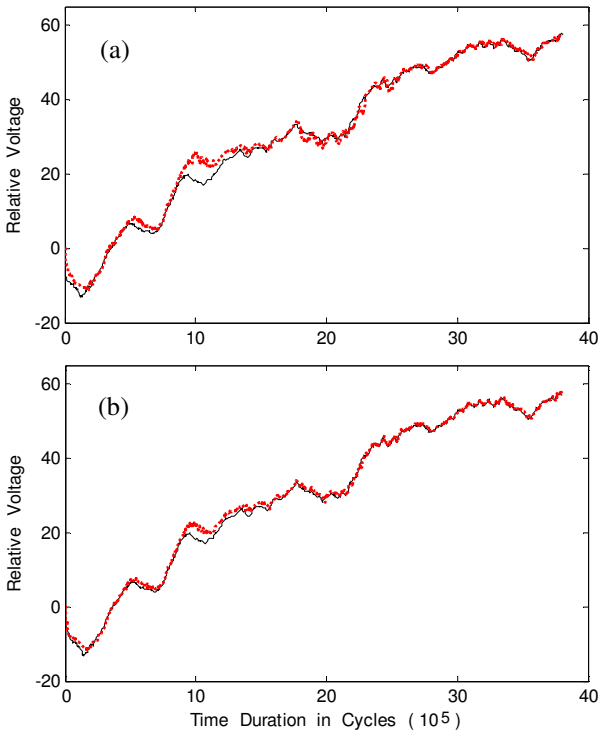

Fig.8: Forecasting results (three-steps-ahead) for a crack propagation (solid curves): (a) by using the feedforward NN scheme (dotted curve); (b) by using the recurrent NF predictor (dotted curve)

CONCLUSIONS

In this paper, the knowledge-based data-driven multi-step-ahead forecasting schemes are evaluated in terms of performance and efficiency. Of the feedforward NNs, recurrent NNs, and recurrent NF systems, analysis results have shown that predictors based on step inputs perform better than those based on sequential inputs, as long as the same number inputs are employed. On the other hand, the recurrent NF predictors perform better than those based on feedforward and recurrent NNs, in multi-step-ahead forecasting operations. A hybrid training algorithm is adopted to improve the robustness and reliability of the recurrent NF predictor, by which the fuzzy parameters in the recurrent context layer are trained by using the real-time recurrent learning, whereas the fuzzy consequent parameters are updated by applying a weighted least squares estimate. Simulation results have shown that this suggested algorithm has a better convergence and higher forecasting accuracy than a generally used gradient-LSE algorithm. In addition, the proposed recurrent NF predictor is implemented for material property testing. Online test results have shown that the developed NF predictor is a reliable forecasting tool. It can capture the system's dynamic behavior quickly and track the system's characteristics accurately.

ACKNOWLEDGEMENT

This project was partly supported by MC Technologies Inc.

REFERENCES

1. M. Pourahmadi, 2001. Foundations of Time Series Analysis and Prediction Theory, John Wiley & Sons Inc.
2. D. Chelidze and J. Cusumano, 2004. A dynamical systems approach to failure prognosis. J. Vib. Acous. 126: 1-7.
3. C. Li and H. Lee, 2005. Gear fatigue crack prognosis using embedded model, gear dynamic model and fracture mechanics. Mech. Syst. Signal Process. 9: 836-846.
4. J. Connor, R. Martin, and L. Atlas, 1994. Recurrent neural networks and robust time series prediction. IEEE Trans. Neural Net. 5: 240-254.
5. Y. Liang and X. Liang, 2006. Improving signal prediction performance of neural networks through multiresoltion learning approach. IEEE Trans. Syst. Man Cyber. - Part B, 36: 341-352.
6. D. Husmeier, 1999. Neural Networks for Conditional Probability Estimation: Forecasting beyond Point Prediction, Springer-Verlag London Ltd.
7. J. Jang, 1993. ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans. Syst. Man Cyber. 23: 665-685.
8. W. Wang, 2007. An adaptive predictor for dynamic system forecasting. Mech. Syst. Signal Process. 21: 809-823.
9. W. Wang, F. Ismail, and F. Golnaraghi, 2004. A neuro-fuzzy approach for gear system monitoring. IEEE Trans. Fuzzy Syst. 12: 710-723.
10. M. Figueiredo, R. Ballini, S. Soares, M. Andrade, and F. Gomide, 2004. Learning algorithms for a class of neurofuzzy network and applications. IEEE Trans. Syst. Man Cyber. - Part C, 34: 293-301.
11. H. Ishibuchi and T. Yamamoto, 2005. Rule weight specification in fuzzy rule-based classification systems. IEEE Trans. Fuzzy Syst. 13: 428-435.
12. D. Nauck, 2000. Adaptive rule weights in neuro-fuzzy systems. J Neural Comput. Appli. 9: 60-70.
13. R. Williams and D. Zisper, 1989. A learning algorithm for continually running fully recurrent neural networks. Neural Comput. 1: 270-280.
14. S. Haykin, 1998. Neural Networks: A Comprehensive Foundation, 2nd edition, Prentice Hall.
15. M. Mackey and L. Glass, 1977. Oscillation and chaos in physiological control systems. Science, 197: 287-289.