International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:1, No:10, 2007

The Application of an Ensemble of Boosted Elman Networks to Time Series Prediction: A Benchmark Study

Chee Peng Lim and Wei Yee Goh

***Abstract*—**In this paper, the application of multiple Elman neural networks to time series data regression problems is studied. An ensemble of Elman networks is formed by boosting to enhance the performance of the individual networks. A modified version of the AdaBoost algorithm is employed to integrate the predictions from multiple networks. Two benchmark time series data sets, i.e., the Sunspot and Box-Jenkins gas furnace problems, are used to assess the effectiveness of the proposed system. The simulation results reveal that an ensemble of boosted Elman networks can achieve a higher degree of generalization as well as performance than that of the individual networks. The results are compared with those from other learning systems, and implications of the performance are discussed.

***Keywords*—**AdaBoost, Elman network, neural network ensemble, time series regression

I. Introduction

MANY approaches have been examined for tackling time series data regression problems, e.g., moving average, exponential smoothing, decomposition, ARIMA (Autoregressive Integrated Moving Average). Most of the methods involve the construction of a mathematical model that best represents the behavior of the observed system. However, identifying a suitable model requires skilled and experienced forecasters as real-world processes often exhibit non-linear characteristics which are difficult to model. Different forecasters may arrive at different models even if they use the same set of observations. For example, in ocean modeling and weather prediction, individual forecasters can be sensitive to small variations in initial and boundary conditions [1]. In [2], it is explained that Artificial Neural Network (ANN) models are inherently unstable in performance, i.e., small changes in the training set and/or parameter selection can produce large changes, hence resulting in different networks. Thus, the sensitivity to small perturbation in model parameters can be exploited to generate an ensemble of forecasters, and the output from all ensemble members can be analyzed and combined to improve the overall performance.

In general, an ensemble of models can be used in two ways, e.g., (i) selecting the output from the "best" (e.g., lowest error rate, highest posterior probability) of the ensemble members to be the final prediction; (ii) combining the output from the ensemble members using some decision combination algorithm. In this paper, we are concerned with the latter approach, with specific focus on ANN-based methodologies. In particular, the scope of work is on the use of an ensemble of Elman networks [3], generated by perturbation of initial network weights coupled with a modified AdaBoost algorithm [4] for tackling time-series regression problems.

Lately, ANNs have received a lot of attention as an alternative for solving time series data regression problems. ANNs are parameterized approach that can return the best-fit curve that approximates behavior of the training data provided to the network models. ANNs are able to tolerate noisy inputs, including chaotic components having very heavy tails [5]. With a suitable architecture, an ANN can offer good generalization capability and achieve high performance in solving non-linear time series data regression problems. Indeed, ANN-based ensembles have been proposed to form a more accurate and robust learning system [2, 6-9].

In classification tasks, many researchers have investigated the techniques of combining predictions from multiple classifiers to produce a result that is generally more accurate than any of the individual classifiers. Boosting [4] is one of the recent methods that forms an ensemble of individually trained classifiers, e.g. decision trees and ANNs, such that the test set error can be reduced. The theoretical justifications of its efficiency have been discussed [9-12]. In regression problems, Zemel and Pitassi [13] proposed a gradient-based boosting algorithm in which a threshold was introduced to differentiate correct responses from incorrect ones. Boosting was also used to combine multiple regression trees and to reduce the prediction error of a single regressor [14-15], as well as to improve the predictive power of nonparametric regression methods [16].

While different strategies have been proposed, boosting, in general, assigns different voting strengths to classifiers based on their accuracy rates. It maintains a weight count for each sample in the training set to reflect the difficulty needed to

Manuscript received December 25, 2005. This work was sponsored by University of Science Malaysia, Malaysia.

C. P. Lim currently is an Associate Professor at School of Electrical and Electronic Engineering, University of Science Malaysia, Malaysia. (e-mail: cplim@eng.usm.my).

W.Y. Goh currently is an engineer at a multi-national corporation in Penang, Malaysia.

classify the sample in comparison with previously established networks. Adjusting the weights causes the boosted networks to focus on "hard" samples. A good ensemble is produced when the individual classifiers are both accurate and diverse, i.e. the classifiers have low error rates and their errors are on different parts of the input space [17].

In this paper, a specific type of supervised, partial recurrent ANN architecture, namely the Elman network [3], is employed. The Elman network is embedded with feedback connections that offer a convenient way to accumulate previous knowledge as "experiences" and perform future predictions based on these "experiences". While other researchers have employed the Elman network for modeling time series profiles [8, 18], here we investigate the use of an ensemble of Elman networks coupled with a modified AdaBoost (Adaptive Boosting) algorithm [4, 19] for time series data regression problems. The ensemble is created by randomizing the initial weights of a pool of Elman networks. Then, modified AdaBoost is implemented by directly weighting the cost function of Elman networks. This differs from boosting by sampling as proposed in [14, 15]. Two benchmark problems are employed to assess the effectiveness of the modified AdaBoost algorithm coupled with an ensemble of Elman networks.

The organization of this paper is as follows. The concept and architecture of the Elman network are described in section II. After an introductory account of the AdaBoost in section III, the modified AdaBoost algorithm is presented in section IV. In section V, an experimental study on two benchmark data sets with various network configurations using single and multiple boosted Elman networks are discussed. The results are analyzed and compared with those from other learning systems. A summary is included in section VI.

II. The Elman Network

The Elman network is a simple recurrent neural network that only involves partial feedback in the network structure (see Fig. 1.) It comprises four layers, namely the input layer, hidden layer, output layer, and context layer. During operation, outputs of the hidden layer are fed back to the context layer at every time step. The context layer, thus, acts as a "container" that preserves previous information, and this recurrence gives the network dynamical properties, which provide the ability to perform mappings that are functions of time.

Suppose an n-dimensional input vector, $\boldsymbol{x}_t = (x_{1t}, \cdots, x_{nt})$, is given at time t, the learning dynamic of the Elman network with q hidden nodes is as follows

1. Initially, the context nodes are set to zero. The forward weights and biases are generated using the Nguyen-Widrow method [20]. This technique ensures that the active regions of the nodes were distributed roughly evenly over the input space and, thus, helps accelerate the training process. The recurrent connections are non-adjustable and are fixed at unity weights.
2. The output functions of the hidden and output layers are computed, respectively, as follows

$$h_{i,t} = \Psi\left(\gamma_{i0} + \sum_{j=1}^{n} \gamma_{ij} x_{j,t} + \sum_{\ell=1}^{q} \delta_{i\ell} h_{\ell,t-1}\right)$$
$$\equiv \psi_i(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}, \boldsymbol{\theta}) \quad i = 1, \cdots, q \tag{1}$$

$$o_t = \Phi\left(\beta_0 + \sum_{i=1}^{q} \beta_i \psi_i(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}, \boldsymbol{\theta})\right) \tag{2}$$

where $h_{i,t}$ is the output of the i-th hidden node, o_t is the estimator of the target variable, $\boldsymbol{\theta}$ is the vector of parameters containing all γ's and δ's, where γ's, δ's, β's are the weights from the input layer to the hidden layer, the context layer to the hidden layer, and the hidden layer to the output layer, respectively, and $\Psi(\cdot)$ and $\Phi(\cdot)$ are the activation functions.

3. The prediction error, e.g. mean-squared-error (MSE), is calculated by comparing the predicted output with the target sequence.
4. The adjustable weights and biases are updated using truncated gradient back-propagation [21] with an additional momentum term and an adaptive learning rate.

Steps 2 to 4 are repeated until a predefined error rate or number of training epochs is reached.

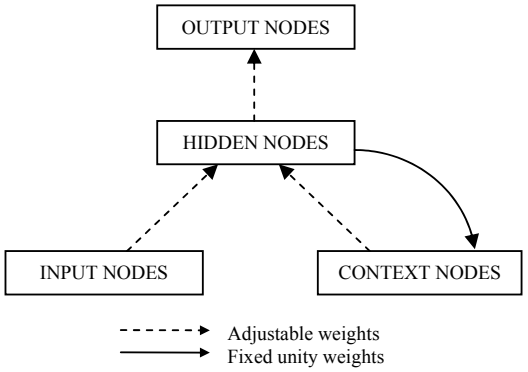

Fig. 1 Architecture of the Elman network

III. AdaBoost

Boosting is a general method for improving the accuracy of any given learning algorithm. It produces a final solution by combining rough and moderately inaccurate decisions offered by different classifiers, which is at least slightly better than random guessing. In boosting, the training set used for each classifier is produced (weighted) based on the performance of the earlier classifier(s) in the series. Therefore, samples that are incorrectly classified by previous classifiers in the series are emphasized more than samples that are correctly classified.

AdaBoost solved many of the practical difficulties of the earlier boosting algorithms. The algorithm first receives a

training set, $(x_1,y_1),\ldots,(x_n,y_n)$, $i=1,\ldots,n$ as inputs where x_i is a sample in space *X*, and y_i is a class label in set *Y* associated with x_i. Without loss of generality, assume $Y=\{-1,+1\}$ for binary class case. AdaBoost repeatedly calls a *weak (base) learning algorithm* in a series of rounds $k=1,\ldots,K$. In each round *k*, the algorithm assigns a distribution or a set of weights over the training set. The weight of training sample *i* during round *k* is denoted as $D_k(i)$. The weak learner can use D_k on the training samples. Alternatively, the training inputs can be sampled according to D_k, and these resampled inputs can be used to train the weak learner. Initially, all the weights are set equally, and the weights of incorrectly classified samples are increased on each repetition so that the learner is forced to focus on the "hard" samples in the training set.

The weak learner has to compute a *hypothesis*, $h_k : X \to \{-1,+1\}$ with respect to D_k. The hypothesis is measured by its *error* [4], $\varepsilon_k = \Pr_{i\sim D_k}[h_k(x_i)\neq y_i] = \sum_{i:h_k(x_i)\neq y_i} D_k(i)$. Once hypothesis h_k is calculated, AdaBoost uses a parameter $\alpha_k = \frac{1}{2}\ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$ to measure the reliability of h_k. Note that $\alpha_k \geq 0$ if $\varepsilon_k \leq 1/2$. Then, update $D_{k+1}(i) = \frac{D_k(i)\exp(-\alpha_k y_i h_k(x_i))}{Z_k}$, where Z_k is a normalization factor (chosen so that D_{k+1} is a distribution). In response, the weights of samples misclassified by h_k are increased, and the weights of correctly classified samples are decreased, hence the algorithm is forced to focus on "hard" samples. Finally, the weak hypotheses are combined into a single *final hypothesis*, $H(x)=\text{sign}\left(\sum_{k=1}^{K}\alpha_k h_k(x)\right)$, which is a weighted majority vote of the *K* hypotheses.

Schwenk and Bengio [9] compared three versions of AdaBoost using ANNs, i.e., (R) training each hypothesis with a fixed training set obtained by resampling with replacement from the original training set; (E) training by resampling, after each epoch, a new training set from the original training set; and (W) training by directly weighting the cost function of the ANN. Their experimental results agreed with the findings in [4] that improvement in generalization error brought by AdaBoost is mainly due to emphasizing the margin, rather than variance reduction due to randomization of the resampling process. The (W) approach, which is preferred in ANN framework, is implemented in this paper.

IV. Modified AdaBoost

In a previous study [19], modifications to the AdaBoost algorithm were proposed to suit time series prediction problems. Generally, use of boosting in regression/prediction is harder than classification. Classifiers map the inputs with several classes, while regressors need to predict the expected values that are associated with the given inputs. In prediction problems, the outputs are neither correct nor incorrect (in contrast to classification problems). Rather, performance of a predictor is measured by the prediction error. This leads to the proposed modifications of the AdaBoost algorithm so that it is applicable to real-valued prediction tasks.

The modified AdaBoost algorithm is shown in Fig. 2. It first takes a sequence of *n* samples as the training set, $(x_1,y_1),\ldots,(x_n,y_n)$, $1\leq i\leq n$, where x_i is an instance in space *X*, and y_i is an instance in space *Y* associated with x_i. In each round *k*, the Elman network is trained with respect to a probability distribution $D_k(i)$ over an original training set. The initial distribution D_1 is uniform over the training set, so $D_1(i)=1/n$ for all *i*. After each round, the network computes the hypothesis or the predicted output. A normalized unit error function, ε_k, is used to assess the difference between the predicted and actual outputs, which is then weighted by D_k. The normalized error function used can be the difference or the relative error between the predicted and actual outputs. Other error functions such as those proposed in [14, 15] are also applicable. Then, D_{k+1} is computed by changing the probabilities of samples in accordance with the error function. This makes the network put more emphasis on the "hard" samples. A normalization factor is used to constrain D_{k+1} as a distribution. The final output is computed by taking weighted outputs from the sequentially trained networks. A weighted median approach was proposed to combine the results from multiple regressors [14, 15]. The final output is obtained by simply taking the weighted outputs of the sequentially trained networks in accordance with parameter α_k that reflects the importance/reliability of the corresponding network in the ensemble.

Given: $(x_1,y_1),\ldots,(x_n,y_n)$ where $x_i\in X, y_i\in Y$

Initialize: $D_1(i)=1/n$

Repeat: For $k=1,\ldots,K$:

1. Train the Elman network with respect to distribution D_k.
2. Get weak hypothesis, h_k, with the error function, ε_k, with respect to D_k.
3. Choose $\alpha_k = \frac{1}{2}\ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$.
4. Update $D_{k+1}(i) = \frac{D_k(i)\exp(\alpha_k \bullet \varepsilon_k(i)/\varepsilon_k)}{Z_k}$, where Z_k is a normalization factor (chosen so that D_{k+1} is a distribution).

Output: The final output

$$H(x)=\sum_{k=1}^{K}\left(h_k(x)\bullet\alpha_k \Big/ \sum_{k=1}^{K}\alpha_k\right)$$

Fig. 2 The modified AdaBoost algorithm

The differences between the original AdaBoost algorithm and the modified version shown in Fig. 2 occur in steps 2 and 4 and the "output" section. Parameter ε_k in modified AdaBoost (step 2) is a user-defined unit error function weighted by D_k. In contrast, ε_k of original AdaBoost is the summation of the probabilities of misclassified samples. In step 4, the exponential part of D_{k+1} is updated in accordance with the error distribution, while weights of correctly/incorrectly classified samples in original AdaBoost are decreased/increased by a factor. In addition, the final output is modified from the weighted majority vote (AdaBoost) to the weighted predictions of the sequentially trained networks in accordance with their assigned reliabilities.

V. Simulation Studies

Two sets of simulations using benchmark data were conducted to examine the effectiveness of an ensemble of boosted Elman networks coupled with the modified AdaBoost algorithm in time series prediction problems. They were the Sunspot and Box-Jenkins gas furnace data sets. An ensemble of diverse Elman networks was created by randomly initializing the weights of the constituting networks. Then, modified AdaBoost was used to directly weight the cost function of ANN predictors, which differs from the use of boosting for resampling a training set [14, 15]. Each network was trained by weighting the cost function, i.e. the MSE, according to the (W) approach proposed in [9]. The cost function was weighted by the probability of each sample in D_k. The weighted MSE is computed as follows

$$\text{weighted MSE} = \frac{1}{n}\sum_{i=1}^{n}\left[D_k(i)(y_i - o_i)\right]^2 \quad (3)$$

where y_i is the expected output presented to the network, o_i is the output predicted by the network, D_k is the probability distribution of the original training set, and *n* is the number of samples. Thus, the network can focus on the "hard" samples that contribute more error to the cost function.

A. The Sunspot Series

The Sunspot series, a well-known benchmark time series prediction problem, comprises yearly average sunspot activity recorded for the period 1700-1979. The data set can be downloaded from a public repository with explanation [22]. The procedure in [23] was followed in this study in order to compare the network performance with those of other models. As stated in [23], the Sunspot data from year 1700 to 1920 and year 1921 to 1979 were used for training and test, respectively. The inputs were the past samples of sunspot activities, $x_i(t-1), x_i(t-2), \ldots, x_i(t-12)$, while the output was the next in-coming sample, $y_i = x_i(t)$. The network structure contained 12 input nodes, 8 hidden sigmoid nodes, and a single output node (12-8-1).

In the experiment, the modified AdaBoost algorithm shown in Fig. 2 was employed. By following the procedure in [23], 4000 iterations were employed in the Elman network and the sum of squared errors (SSEs) as in equation (4) (with O_i the final predicted output) was used as the performance measurement in both the training and test phases.

$$\text{SSEs} = \sum_{i=1}^{n}\frac{1}{2}(y_i - O_i)^2 \quad (4)$$

The weighted error of each sample was evaluated by multiplying the weight of the corresponding sample, $D_k(i)$. The difference (error) between the predicted and the desired outputs of each sample was calculated and normalized between 0 and 1. Each error was weighted by $D_k(i)$, and sum of all errors of the training samples computed. As shown in [10, 11], the training error would drop exponentially fast if each weak hypothesis is slightly between than random guess. Thus, the network training process was terminated either when the total error was greater than 0.5 or when convergence occurred, i.e. improvement in the error rate was insignificant.

After several repeated experiments, the results converged after combining 22 individual networks. The best three results of single and boosted Elman networks are tabulated in Table I. The boosting algorithm can improve the network performance. The errors in the training and test phases were, respectively, reduced by at least 26% and 28%. The mean error reduction was 27% in the training phase, and 43% in the test phase. The second experiment yielded the lowest error rates in the training and test phases. Fig. 3 shows the original Sunspot series and the predicted outputs of the second experiment.

From Fig. 3, we can observe that the prediction errors were in the range of ± 0.15 during training, while the prediction errors were in the range of ± 0.21 during testing, except in year 1959 where the prediction error was 0.4. Note that year 1959 generated the highest sunspot activity among the monitored period. This highest peak was hard to forecast because it constituted an extrapolation condition. Extrapolation arises when the prediction process occurs beyond the range of the training samples, while interpolation refers to the conversed situation [24]. Basically, the network performs better in interpolated (as compared to extrapolated) regions of the input data. Notice that all the models in [23] yielded poor performances around the particular year too.

The mean of the three best results shown in Table I was compared with those from a number of different models reported in [23]. The results of different models with specified sizes (shown in bracket) are listed in Table II. In [23], a cascade-form predictor that consisted of a non-linear sub-predictor (NSP) and a linear sub-predictor (LSP) was proposed. The ML-WDC model is a multi-layer network with direct linear connections from the input layer to the output layer. From the results shown in Table II, the boosted Elman network performed better than other models. The boosted network achieved the lowest errors in both the training and test phases. By comparing with Khalaf and Nakayama's

model, the boosted network reduced the SSEs by 40% in the training phase and by 5% in the test phase.

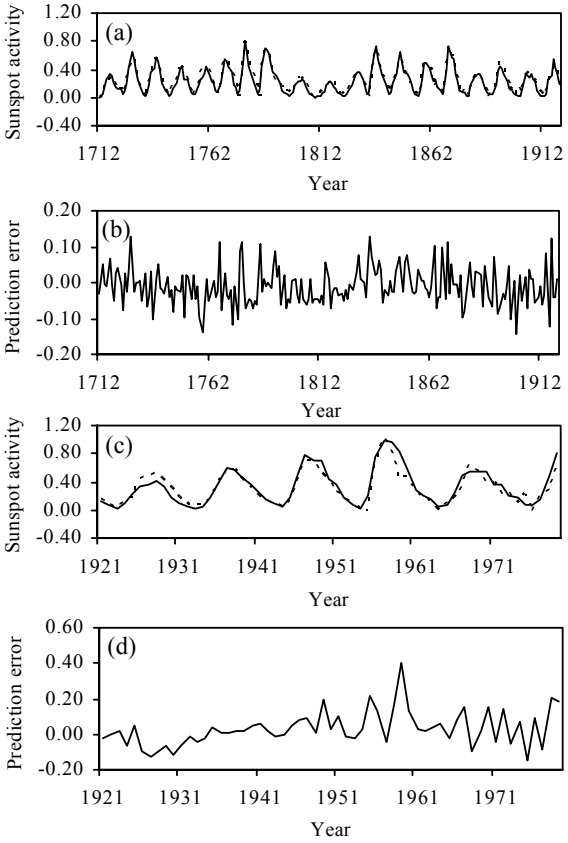

Fig. 3 Predictions of the boosted network in the training and test phases, [(a), (b)] and [(c), (d)], respectively, of the Sunspot series. (a) original series (solid line) and predicted series (dashed line); (b) prediction errors from year 1712 to 1920; (c) original series (solid line) and predicted series (dashed line); (d) prediction errors from year 1921 to 1979

TABLE I
THE SUM OF SQUARED ERRORS OF SINGLE AND BOOSTED NETWORKS AND THE PERCENTAGES OF ERROR REDUCTION (SUNSPOT SERIES)

No.	Training		Error reduction (%)	Test		Error reduction (%)
	Single	Boosted		Single	Boosted	
1	0.361	0.263	27	0.667	0.295	56
2	0.331	0.244	26	0.368	0.264	28
3	0.347	0.255	27	0.529	0.289	45
Mean	0.346	0.254	27	0.521	0.283	43

TABLE II
THE SUM OF SQUARED ERRORS OF VARIOUS MODELS FOR THE SUNSPOT SERIES

No	Model Name	Training	Test
1	Multi-layer perceptron model (12-8-1)	0.4087	0.4476
2	ML-WDC model (12-8-1)	0.5786	0.7152
3	Sandwich model LSP(5)+NSP(12-8-1)+LSP(5)	0.6514	0.4093
4	The reverse order model LSP(6)+NSP(12-8-1)	0.4787	0.4235
5	Khalaf and Nakayama NSP(12-8-1)+LSP(10)	0.4227	0.2963
6	Boosted Elman network model	0.2537	0.2828

B. The Box-Jenkins Series

The Box-Jenkins series is another frequently used benchmark for time series prediction. The data set used in this study is the gas furnace data (series *J*) [25]. It was recorded from a combustion process of a mixture of methane-air. The data set contained 296 samples. The input $u(t)$ was the gas flow into the furnace and the output $y(t)$ was the CO_2 (Carbon Dioxide) concentration in the outlet gas.

This study followed closely the experimental procedure in [26] in order to compare the performance of the boosted Elman network with a list of performances from other approaches. The Elman network was provided with two inputs and an output: the values of methane at time (*t*-4) and CO_2 produced at time (*t*-1) as inputs, and CO_2 produced at time (*t*) as output. Hence, the original 296 data pairs were reduced to 292 data pairs of $[u(t-4), y(t-1); y(t)]$. A total of 200 data samples were used for training and the remaining for test.

In accordance with the procedure in [26], the Elman network was trained using only 200 iterations, and the performance was measured by MSE of all the outputs as in (5), including the outputs in the training and test phases.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - O_i)^2 \tag{5}$$

As in the previous study, the error between the predicted output and actual output for each sample was computed, normalized, and weighted by $D_k(i)$. The sum of all errors was then calculated. After several trials, the best network architecture was formed with 26 hidden nodes (2-26-1). The results converged after combining 18 networks. Three best results of the single and boosted networks are tabulated in Table III. All the MSEs produced by the single network were reduced significantly (54%–76%) with the use of boosting. The mean of the MSEs was reduced by 65%. These results, again, demonstrate that boosting is able to greatly improve the performance of the individual networks. Among the results, the third experiment yielded the best performance. Thus, the original and predicted outputs, together with the prediction errors of the third experiment are shown in Fig. 4.

Table IV tabulates the MSEs of other approaches listed in [26] that employed the same number of inputs. Referring to Table IV, the boosted Elman networks (mean of the three best obtained results) ranked six out of twelve. The HyFIS (Hybrid Neural Fuzzy Inference System), FuNN (Fuzzy Neural Network) and ANFIS (Adaptive-Network-Based Fuzzy Inference System) models yielded very good results. These three models are fuzzy inference systems implemented in the framework of adaptive networks. The models constructed the input-output mapping based on both human knowledge (in the form of fuzzy if-then rules) and stipulated input-output data pairs. The HyFIS model has a two-phase learning scheme. Phase one is the rule finding phase, which the fuzzy rules are generated from the desired input-output pairs and the structure of the neural fuzzy system is established from these fuzzy rules. Phase two is the parameter learning phase in which the gradient descent learning is used

to optimally adjust the membership functions for the desired outputs. Because the initial structure is properly set in the first phase, the network only needs 200 epochs to converge. On the other hand, the Elman network needs more training iterations to converge since it does not perform a coarse tuning phase (before network training). Fig. 5 illustrates the MSE curve of a single Elman network where convergence (with MSE less than 0.001) occurred slowly after a large number of training epochs. However, for comparison purpose, the number of network iterations was fixed to 200 as what had been used in [26].

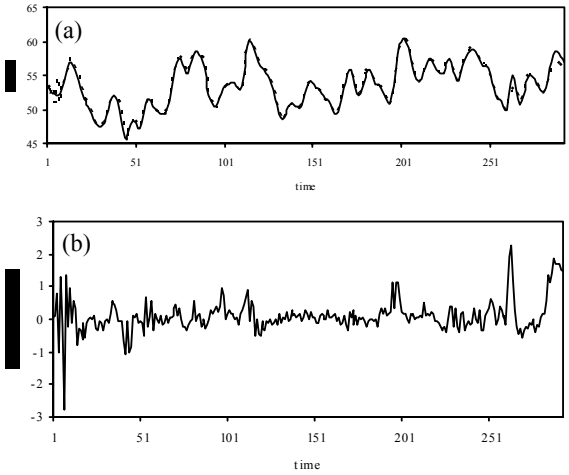

Fig. 4 Predictions of the boosted network of the Box-Jenkin series (a) original series (solid line) and predicted series (dashed line); (b) prediction errors

TABLE III
THE MEAN SQUARE D ERRORS OF SINGLE AND BOOSTED NETWORKS AND THE PERCENTAGES OF ERROR REDUCTION (BOX-JENKINS SERIES)

No.	Single	Boosted	Error reduction (%)
1	0.612	0.223	64
2	0.526	0.242	54
3	0.868	0.206	76
Mean	0.669	0.224	65

TABLE IV
THE RESULTS (MSE) OF VARIOUS MODELS FOR THE BOX-JENKINS SERIES

Model & Reference	MSE	Model & Reference	MSE
HyFIS model [26]	0.00042	Pedrycz's model [31]	0.320
FuNN model [27]	0.00051	Xu's model [32]	0.328
ANFIS model [28]	0.00073	Sugeno's model [33]	0.355
Hauptmann's model [29]	0.134	Pedrycz's model [34]	0.395
Surmann's model [30]	0.160	Lee's model [35]	0.407
Boosted Elman networks	0.224	Tong's model [36]	0.469

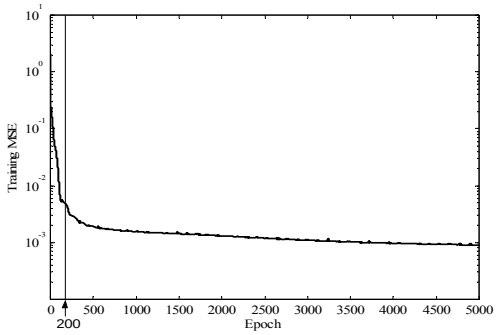

Fig. 5 The mean-squared-error curve of a single network of the Box-Jenkins series

VI. SUMMARY

In this paper, an ensemble of Elman recurrent networks coupled with a modified AdaBoost algorithm has been employed to handle time series data regression problems. AdaBoost is simple and easy-to-use, and can be flexibly combined with other learning systems that establish only weak hypotheses because it requires minimum prior knowledge about the weak learners. Furthermore, AdaBoost comes with theoretical justifications [37], i.e. the bias-variance decomposition and the margin distribution of generalization error that can iteratively drive the final prediction error to zero given sufficient data and weak learners that can reliably provide only moderately accurate weak hypotheses, with their errors kept below 50%.

It has been recognized that good and effective ensembles must have accurate but diverse members. Here, an ensemble of Elman networks that differ only in their random initial weights was able to perform very well by using the modified AdaBoost algorithm. Modified AdaBoost is used for directly weighting the cost function of ANN predictors, rather than for sampling the training samples. Empirical evaluations of the boosted networks using two benchmark data sets have been presented. The simulation studies indicated the boosted Elman networks produced reasonably good ensembles that demonstrate good generalization capabilities and high performances in time series prediction tasks. The results from the boosted networks always outperformed those from the individual networks. The results also showed that the boosted networks were comparable with those from other models.

For further work, the findings in [37] serve as a good source to study the theoretical properties of the modified AdaBoost algorithm. In addition, more real-world time-series prediction problems have to be performed to further ascertain the applicability of the ensemble of Elman networks. The results have to be compared not only with those from ANN-based systems but also other established statistical techniques that have been widely used in undertaking time-series regression problems.

REFERENCES

[1] R.N. Miller and L.L. Ehret, "Ensemble generation for models of multimodal systems", *Monthly Weather Review*, vol. 130, pp. 2313-2333, 2002.

[2] J. G. Carney and P. Cunningham, "The NeuralBAG algorithm: Optimizing generalization performance in bagged neural networks," *Proc. 7th European Symp. Artificial Neural Networks*, M. Verleysen, Ed. D-Facto, Brussels, 1999 pp. 35-40.

[3] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179-211, 1990.

[4] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, 1997.

[5] T. Masters, *Practical Neural Network Recipes in C++*. San Diego, CA: Academic Press, Inc., 1993.

[6] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 10, pp. 993-1001, 1990.

[7] Y. Liu and X. Yao, "Negatively correlated neural networks can produce best ensembles," *Australian Journal of Intelligent Information Processing Systems*, vol. 4, no. 3/4, pp. 176-185, 1997.

[8] F. Fessant, S. Bengio, and D. Collobert, "On the prediction of solar activity using different neural network models," *Annales Geophysicae*, vol. 14, pp. 20-26, 1995.

[9] H. Schwenk and Y. Bengio, "Boosting Neural Networks," *Neural Computation*, vol. 12, no. 8, pp. 1869-1887, 2000.

[10] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297-336, 1999.

[11] J.R. Quinlan, "Bagging, Boosting, and C4.5" *Proc of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pp. 725-730, 1996.

[12] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139-158, 2000.

[13] R. S. Zemel and T. Pitassi, "A gradient-based boosting algorithm for regression problems," *Advances in Neural Information Processing Systems 13*, T. Leen, T. Dietterich, and V. Tresp eds., the MIT Press, 2001, pp. 696-702.

[14] H. Drucker, "Fast committee machines for regression and classification," *Third Int. Conf. Knowledge Discovery and Data Mining (KDD'97)*, D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy eds., Menlo Park, CA: AAADietterichI Press, 1997, pp. 159-162.

[15] H. Drucker, "Improving regressors using boosting techniques," *Proc. Fourteenth Int. Conf. Machine Learning (ICML'97)*, D. H. Fisher, Ed. Morgan Kaufmann, 1997, pp. 107-115.

[16] S. Borra and A. Di Ciaccio, "Improving nonparametric regression methods by bagging and boosting," *Computational Statistics & Data Analysis*, vol. 38, pp. 407-420, 2002.

[17] G. Giacinto and F. Roli, "An approach to the automatic design of multiple classifier systems," *Pattern Recognition Letters*, vol. 22, pp. 25-33, 2001.

[18] P. Stagge and B. Sendhoff, "An extended Elman net for modeling time series," *Int. Conf. Artificial Neural Networks (ICANN'97)*, W. Gerstner, A. Germond, M. Hasler, and J. Nicoud, eds., Springer Verlag, 1997, vol. 1327 of Lecture Notes in Computer Science, pp. 427-432.

[19] W.Y. Goh, C.P. Lim, and K.K. Peh, "Predicting Drug Dissolution Profiles with an Ensemble of Boosted Neural Networks: A Time-series Approach", *IEEE Trans. on Neural Networks*, vol. 14, pp. 459-463, 2003. Y. Freund and R. E. Schapire, "A short introduction to boosting," Journal of Japanese Society for Artificial Intelligence, vol. 14, no. 5, pp. 771-780, (Appearing in Japanese, translation by Naoki Abe.), 1999.

[20] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," *Proc. Int. Joint Conf. Neural Networks*, vol. 3, pp. 21-26, 1990.

[21] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation". *Parallel Distributed Processing: Explorations in the microstructure of cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986 vol. 1, pp. 318-362.

[22] M. Nørgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen. (2000). *Neural Networks for Modelling and Control of Dynamic Systems*. London: Springer-Verlag. [Online]. Available: http://www.iau.dtu.dk/nnspringer.html

[23] A. A. M. Khalaf and K. Nakayama, "A cascade form predictor of neural and FIR filters and its minimum size estimation based on nonlinearity analysis of time series," *IEICE Trans. Fundamentals*, vol. E81-A, no. 3, pp. 364-373, 1998.

[24] J. A. Leonard, M. A. Kramer, and L. H. Ungar, "A neural network architecture that computes its own reliability," *Computers & Chemical Engineering*, vol. 16, no. 9, pp. 819-835, 1992.

[25] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco: Holden-Day, 1970.

[26] J. Kim and N. Kasabov, "HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems," *Neural Networks*, vol. 12, pp. 1301-1319, 1999.

[27] N. Kasabov, J. Kim, M. Watts, and A. Gray, "FuNN/2 — A fuzzy neural network architecture for adaptive learning and knowledge acquisition," *Information Sciences*, vol. 101, no.3-4, pp. 155-175, 1997.

[28] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ: Prentice-Hall, 1997.

[29] W. Hauptmann and K. Heesche, "A neural net topology for bidirectional fuzzy-neuro transformation," *Proc. IEEE Int. Conf. Fuzzy Systems (FUZZ-IEEE/IFES)*, Yokohama, Japan, 1995, pp. 1511-1518.

[30] H. Surmann, A. Kanstein, and K. Goser, "Self-organizing and genetic algorithms for an automatic design of fuzzy control and decision systems," *Proc. First European Congress on Fuzzy and Intelligent Technologies (EUFIT'93)*, Aachen, 1993, vol. 1, pp. 1097-1104.

[31] W. Pedrycz, "An identification algorithm in fuzzy relational systems," *Fuzzy Sets and Systems*, vol. 13, pp. 153-167, 1984.

[32] C.-W. Xu and Y.-Z. Lu, "Fuzzy model identification and self-learning for dynamic systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 17 no. 4, pp. 683-689, 1987.

[33] M. Sugeno and T. Yasukawa, "Linguistic modelling based on numerical data," *Proc. Fourth Int. Fuzzy Systems Association World Congress (IFSA'91)*, R. Lowen and M. Roubens, Eds. Brüssels, Belgium: Computer, Management & Systems Science, 1991, pp. 264-267.

[34] W. Pedrycz, P. C. F. Lam, and A. F. Rocha, "Distributed fuzzy system modelling," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 5, pp. 769-780, 1995.

[35] Y.-C. Lee, C. Hwang, and Y.-P. Shih, "A combined approach to fuzzy model identification," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 5, pp. 736-744, 1994.

[36] R. M. Tong, "The evaluation of fuzzy models derived from experimental data," *Fuzzy Sets and Systems*, vol. 4, pp. 1-12, 1980.

[37] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Annals of Statistics*, vol. 26, no. 5, pp. 1651-1686, 1998.

CP Lim received the BEng(Elect.) degree with first class honors from the University of Technology Malaysia in 1992, the MSc(Eng) and PhD degrees from the University of Sheffield, UK in 1993 and 1997. He is currently Associate Professor at School of Electrical & Electronic Engineering, University of Science Malaysia. His research interests include soft computing, pattern classification, medical diagnosis, and fault detection and diagnosis.

Dr. Lim has published more than 100 technical papers, and received six best paper awards at national and international conferences. He is recipient of the Japan Society for the Promotion of Science Research Fellowship (2002), Fulbright Scholarship (2002), and Commonwealth Fellowship (2003), as well as The Outstanding Young Malaysians Award (2001) and National Young Scientist Award (2002) of Malaysia.

WY Goh received her BTech and MSc(Eng.) degrees from University of Science Malaysia in 1999 and 2002, respectively. Her research interests include theory and application of artificial neural network models to time series data regression and drug dissolution profile prediction.