1st International Conference "From Scientific Computing to Computational Engineering"
1st IC–SCCE
Athens, 8–10 September, 2004
©IC–SCCE

TIME SERIES FORECASTING USING COMPUTATIONAL INTELLIGENCE METHODS

Nicos G. Pavlidis and Michael N. Vrahatis
Department of Mathematics,
University of Patras Artificial Intelligence
Research Center (UPAIRC),
University of Patras, GR–26110 Patras, Greece
email: {npav, vrahatis}@math.upatras.gr

Keywords: Time Series Forecasting, Recurrent Neural Networks, Differential Evolution

Abstract: *Forecasting the future evolution of a system based only on past information comprises a central scientific problem. In this work we investigate the comparative performance of recurrent multi–layer perceptrons, trained through backpropagation through time and the differential evolution algorithm, to perform one–step–ahead predictions for the laser time series (Data set A) from the Santa–Fe Time Series Prediction and Analysis Competition* [1].

1 INTRODUCTION

One of the central problems of science is forecasting. This problem can be summarized in the following question: "Given the past how can we predict the future?" [2]. The classic approach is to build an explanatory model from first principles and measure initial data. This approach is not feasible in fields where we still lack the first principles necessary to construct reliable models of the underlying system dynamics. Here we assume knowledge of just the time series.

We investigate the comparative performance of Recurrent Multi–Layer Perceptrons (RMLPs) [3] trained using Backpropagation Through Time (BPTT) [4] and the Differential Evolution algorithm (DE) [5], on the task of generating one–step–ahead predictions for the laser time series (Data set A) from the *Santa–Fe Time Series Prediction and Analysis Competition* [1]. The obtained experimental results indicate that training RMLPs using the DE algorithm produces networks with higher generalization capability compared to BPTT.

The rest of the paper is organized as follows: Section 2 presents theoretical aspects of artificial neural networks relevant to the problem of function approximation and prediction of time series; Section 3 provides a brief overview of the training algorithms employed, namely DE and BPTT; in Section 4 the experimental results obtained are presented; the paper ends with conclusions in Section 5.

2 ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) are parallel computational models comprised of densely interconnected, simple, adaptive processing units, characterized by an inherent propensity for storing experiential knowledge and rendering it available for use. ANNs resemble the human brain in two fundamental respects; firstly, knowledge is acquired by the network from its environment through a learning process, and secondly, interneuron connection strengths, known as synaptic weights are employed to store the acquired knowledge [6].

Two critical parameters for the successful application of ANNs are the appropriate selection of network architecture and training algorithm. The problem of identifying the optimal network architecture for a specific task remains up to date an open and challenging problem. For the general problem of function approximation, the *universal approximation theorem* proved in [7, 8, 9] states that:

Theorem 2.1 *Standard Feedforward Networks with only a single hidden layer can approximate any continuous function uniformly on any compact set and any measurable function to any desired degree of accuracy.*

An immediate implication of this theorem is that any lack of success in applications must arise from inadequate learning, insufficient number of hidden units, or the lack of a deterministic relationship between the input and the target. A second theorem proved in [10] provides upper bounds for the architecture of a feedforward neural network destined to approximate a continuous function defined on the unit cube in $\mathbb{R}^n$.

Theorem 2.2 *On the unit cube in $\mathbb{R}^n$ any continuous function can be uniformly approximated, to within any error by using a two hidden layer network having 2n+1 units in the first layer and 4n+3 units in the second layer.*

Assuming, that the input vector for the network consists of a number of delayed observations of the time series, and the target is the next value, then the *universal myopic mapping theorem* [11, 12] states that any shift-invariant map can be approximated arbitrarily well by a structure consisting of a bank of linear filters feeding a static Feedforward Neural Network (FNN). This type of ANN is known as *Focused Time–Lagged Feedforward Neural Network* (FTLFN). The universal myopic theorem is limited to shift invariant maps. An immediate implication of this limitation is that FTLFNs are suitable for modeling stationary processes [6]. To overcome this limitation one alternative is to distribute the impact of time throughout the structure of a feedforward network rather than incorporating time only at the input end. This approach has given rise to the *Distributed Time Lagged Feedforward Networks* (DTLFNs) [13]. DTLFNs employ connections with time delays. In other words, synaptic weights act as finite impulse response filters.

Alternatively, one can employ Recurrent Neural Networks (RNNs) to model time–varying processes. The general property that renders such networks interesting is their ability to manifest highly nonlinear dynamical behavior [14]. Unlike feedforward neural networks, units in RNNs are fed by activities from previous time steps through recurrent (feedback) connections. Thus, contextual information is retained enabling RNNs to model successfully time series. RNNs were successfully applied in many real–life applications where processing time–dependent information was necessary.

3 SUPERVISED TRAINING ALGORITHMS

The efficient supervised training of ANNs is a subject of considerable ongoing research and numerous algorithms have been proposed to this end. Training RNNs can be considered as the problem of identifying a particular dynamical system among a parameterized family of such systems that best fits the desired specification. Supervised training amounts to the global minimization of an appropriately chosen network error function E. The rapid computation of a set of weights that minimizes this error is a rather difficult task since, in general, the number of network weights is large and the resulting error function generates a complex surface in the weight space, characterized by multiple local minima and broad flat regions adjoined to narrow steep ones. Next, a brief exposition of the training algorithms considered, is provided.

3.1 Backpropagation through time

BPTT is a powerful tool with applications to pattern recognition, dynamic modeling, sensitivity analysis, and the control of systems among others [4]. Similarly to the backpropagation algorithm for FNNs, BPTT is based on the steepest descent unconstrained optimization method. The description of the underlying principles of this algorithm that follows is based on [14].

Let $\mathcal{N}$ denote a single layer RNN to be trained. Also assume that $\mathcal{N}$ has n units and that it is to run from time t_0 to time t_{end}. As described in [6] we can "unroll" $\mathcal{N}$ in time to obtain a feedforward network $\mathcal{N}^*$ which has a layer for each time step in the interval $[t_0, t_{end}]$, and n units in each layer. Each unit in $\mathcal{N}$ has a copy in each layer of $\mathcal{N}^*$, and each connection from unit i to unit j in $\mathcal{N}$ has a copy connecting unit i in layer τ, to unit j in layer $\tau + 1$, for $\tau \in [t_0, t_{end})$. A simple example of this procedure is illustrated in Fig. 1. The key insight driving BPTT is that to compute $\partial E(t_0, t_{end})/\partial w_{ij}$ in $\mathcal{N}$, one simply computes the partial derivatives $\partial E(t_0, t_{end})$ with respect to each of the $(t_{end} - t_0)$ copies of w_{ij} in $\mathcal{N}^*$ and adds them up. Thus, the problem of computing the error gradient in $\mathcal{N}$

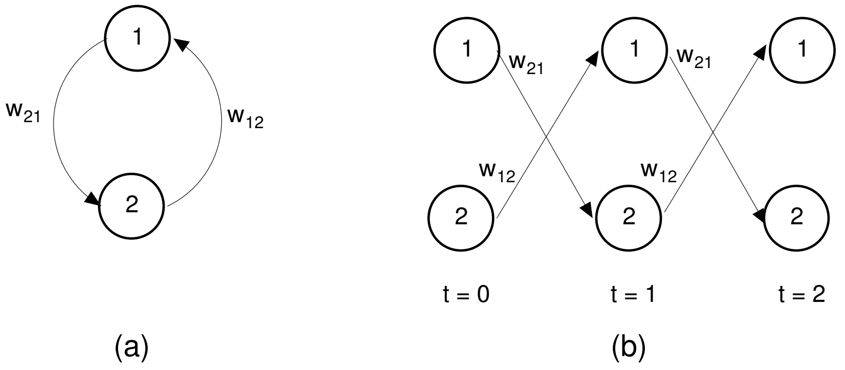

Figure 1: (a): Recurrent network $\mathcal{N}$ (b): "Unrolled" network $\mathcal{N}^*$

reduces to the problem of computing the error gradient in $\mathcal{N}^*$, for which the standard backpropagation algorithm can be used. For a thorough mathematical derivation refer to [4, 14].

3.2 Differential evolution algorithm

DE is a novel minimization method [5], capable of handling nondifferentiable, nonlinear and multimodal objective functions. DE has been designed as a stochastic parallel direct search method, that utilizes concepts borrowed from the broad class of evolutionary algorithms. The method typically requires few, easily chosen, control parameters. A comparative investigation of the DE algorithm on the problem of training FNNs is provided in [15]. DE has been applied to train FNNs with integer weights and threshold activation functions [16], DTLFNs [17], as well as spiking neural networks [18].

DE is a population–based stochastic algorithm that exploits a population of potential solutions, *individuals*, to probe the search space. New individuals are generated by the combination of randomly chosen individuals from the current population. This operation is referred to as *mutation*. Specifically, for each individual w_g^k, $k = 1, \ldots, NP$, where g denotes the current generation index, a new individual v_{g+1}^i (mutant individual) is generated according to one of the following *mutation strategies*:

$$v_{g+1}^i = w_g^{\text{best}} + \mu(w_g^{r_1} - w_g^{r_2}), \tag{1}$$

$$v_{g+1}^i = w_g^{r_1} + \mu(w_g^{r_2} - w_g^{r_3}), \tag{2}$$

$$v_{g+1}^i = w_g^i + \mu(w_g^{\text{best}} - w_g^i) + \mu(w_g^{r_1} - w_g^{r_2}), \tag{3}$$

$$v_{g+1}^i = w_g^{\text{best}} + \mu(w_g^{r_1} - w_g^{r_2}) + \mu(w_g^{r_3} - w_g^{r_4}), \tag{4}$$

$$v_{g+1}^i = w_g^{r_1} + \mu(w_g^{r_2} - w_g^{r_3}) + \mu(w_g^{r_4} - w_g^{r_5}), \tag{5}$$

$$\begin{aligned} v_{g+1}^i = {} & (w_g^{r_1} + w_g^{r_2} + w_g^{r_3})/3 + (p_2 - p_1)(w_g^{r_1} - w_g^{r_2}) \\ & +(p_3 - p_2)(w_g^{r_2} - w_g^{r_3}) + (p_1 - p_3)(w_g^{r_3} - w_g^{r_1}) \end{aligned} \tag{6}$$

where w_g^{best} is the best member of the previous generation; $\mu > 0$ is a real parameter, called *mutation constant*, which controls the amplification of the difference between two individuals so as to avoid the stagnation of the search process; and $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, \ldots, k-1, k+1, \ldots, NP\}$, are random integers mutually different and different from the running index k. The mutation strategy of Eq. (6) is known as the trigonometric mutation operator, and has been recently proposed in [19]. This mutation strategy performs a mutation according to Eq. (6) with probability τ_μ and a mutation according to Eq. (2) with probability $(1 - \tau_\mu)$. The formulae to compute p_i, $i = \{1, 2, 3\}$ and p' are given in [19]. Note that in the experimental results reported in Section 4 we also investigate the performance differences among the various possible mutation strategies.

To increase further the diversity of the mutant individuals, the resulting individuals are combined with other predetermined individuals – the *target* individuals – and this operation is called *crossover*. Specifically, for each component l ($l = 1, 2, \ldots, D$) of the mutant individual v_{g+1}^k, we randomly choose a real number r in the interval $[0, 1]$. Then, we compare this number with the *recombination constant*, ρ. If $r \leqslant \rho$, then we select, as the l–th component of the trial individual u_{g+1}^k, the l–th component of the mutant individual v_{g+1}^k. Otherwise, the l-th component of the target vector, w_g^k, becomes the l–th component of the trial vector. This operation yields the *trial* individual. Finally, the trial individual is accepted for the next generation if and only if, it yields a reduction in the value of the error function. This is the *selection* operation.

4 EXPERIMENTAL RESULTS

The first step in the analysis and prediction of time series originating from real–world systems is the choice of an appropriate time delay, T, and the determination of the embedding dimension, D. To select T a standard approach is to use the value that yields the first minimum of the mutual information function [20]. For the considered time series the first minimum occurs at $T = 2$, as illustrated in Fig. 2. To determine the minimum embedding dimension for state space reconstruction we applied the method of "False Nearest Neighbors" [21]. As illustrated in Fig. 2 the proportion of false nearest neighbors as a function of D drops to zero for $D = 5$. For the computation of the mutual information and the false nearest neighbors we employed the corresponding routines from the TISEAN software package [22].

Numerical experiments were performed using a Neural Network C++ Interface developed under the Fedora Linux 1.0 operating system using the GNU compiler collection (gcc) version 3.3.2. After experimentation with alternative RMLP architectures we selected the one hidden layer topology 5–5–1. The learning rate for the BPTT algorithm was set to 0.1 and the algorithm was allowed to perform 1500 iterations per experiment. For all the mutation strategies of the DE algorithm, the parameter setup used was; a population size of 30 individuals; a

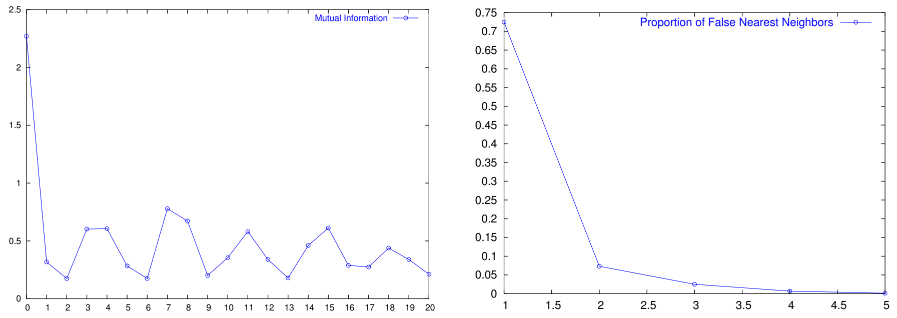

Figure 2: Mutual information as a function of T (left) and proportion of "false nearest neighbors" as a function of D (right).

maximum number of iterations equal to 300; μ was set to 0.5; ρ was equal to 0.7; and finally for the trigonometric mutation operator of Eq. (6) the parameter τ_μ was set to 0.1.

As training set for the RMLPs we used the first 1000 observations of the data set, as was the case in the competition. The next 100 observations were used to formulate the test set on which the performance of the RMLPs was evaluated. To evaluate the generalization capability of the trained RMLPs, three performance measures were employed, the Mean Relative Error (MRE), the Normalized Mean Squared Error (NMSE), and the Root Mean Squared Error (RMSE). The definitions of these measures are given in Eqs. (7)–(9):

$$\mathrm{MRE} = \frac{1}{N}\sum_{t=t_0}^{t_{end}}\left|\frac{y_t-\hat{y}_t}{y_t}\right|, \tag{7}$$

$$\mathrm{NMSE} = \frac{1}{N}\frac{\sum_{t=t_0}^{t_{end}}(y_t-\hat{y}_t)}{\sum_{t=t_0}^{t_{end}}(y_t-\overline{y})}, \tag{8}$$

$$\mathrm{RMSE} = \frac{1}{N}\sum_{t=t_0}^{t_{end}}(y_t-\hat{y}_t)^2, \tag{9}$$

where N stands for the length of the series ($N = t_{end} - t_0$), y_t is the true value of the series at time t, $\hat{y}_t$ is the predicted value for y_t, and $\overline{y}$ is the mean value of the series over the period $[t_0, t_{end}]$.

		max	mean	min	std
BPTT	MRE	2.66	1.25	0.67	0.58
	NMSE	2.16	1.05	0.48	0.60
	RMSE	0.56	0.37	0.26	0.10
DE1	MRE	1.09	0.70	0.47	0.19
	NMSE	0.92	**0.62**	0.42	0.14
	RMSE	0.36	**0.30**	0.24	0.03
DE2	MRE	1.27	0.80	0.54	0.22
	NMSE	1.43	0.68	0.48	0.28
	RMSE	0.45	0.31	0.26	0.05
DE3	MRE	1.31	0.84	0.45	0.27
	NMSE	0.98	0.70	0.58	**0.12**
	RMSE	0.37	0.32	0.29	**0.02**
DE4	MRE	1.48	0.73	0.44	0.35
	NMSE	**0.86**	0.66	**0.28**	0.16
	RMSE	**0.35**	0.30	**0.20**	0.04
DE5	MRE	1.21	0.82	**0.42**	0.21
	NMSE	1.21	0.78	0.59	0.17
	RMSE	0.42	0.33	0.29	0.03
DE6	MRE	**0.94**	**0.69**	0.49	**0.15**
	NMSE	0.90	0.70	0.40	0.19
	RMSE	0.36	0.31	0.24	0.04

Table 1: One–step–ahead prediction performance on the test set

Table 1 reports the performance of the considered algorithms on the problem of one–step–ahead prediction on the test set over ten experiments. The notation DEx is used to refer to the DE algorithm incorporating the mutation strategy of Eq.(x). For each of the three performance measures, the maximum (max), mean, minimum (min), and the standard deviation (std), is reported. Furthermore, the lowest value attained for each measure is bold–faced.

The results reported in Table 1 indicate that training through the DE algorithm yields a significant improvement in the performance of the considered RMLPs. With respect to all the measures considered, all the mutation strategies of the DE algorithm managed to outperform BPTT. In particular, the lowest maximum (best worst case performance) and mean MRE, as well as, the lowest standard deviation with respect to this measure, were attained using the trigonometric mutation operator DE6, while DE5 attained the lowest minimum MRE value (best case performance). DE4 produced RMLPs with the lowest maximum and minimum performance with respect to the NMSE and RMSE measures. The lowest mean values for these measures were achieved by DE1, while training through DE3 yielded the lowest standard deviation.

5 CONCLUSIONS

In this paper we investigated the comparative performance of Recurrent Multi–Layer Perceptrons trained using Backpropagation Through Time and the Differential Evolution algorithm, on the task of generating one–step–ahead predictions for the laser time series (Data set A) from the *Santa–Fe Time Series Prediction and Analysis Competition* [1]. To evaluate the performance of the trained networks, three measures were used, namely the Mean Relative Error, the Normalized Mean Squared Error, and the Root Mean Squared Error. The Differential Evolution algorithm managed to outperform Backpropagation Through Time with respect to the maximum, mean, minimum and standard deviation of all three performance measures, irrespective of the choice of mutation strategy. From the six alternative mutation strategies considered, the most promising results were obtained by the first, fourth and sixth (refer to Section 3.2). In the future we intend to investigate the application of Differential Evolution and alternative Evolutionary Algorithms to the problem of iterated prediction using recurrent neural networks.

Acknowledgment

We acknowledge partial support by the "Pythagoras" research grant awarded by the Greek Ministry of Education and Religious Affairs and the European Union.

References

[1] Weigend, A. S. and Gershenfeld, N. A., editors. (1994). "Time series prediction: Forecasting the future and understanding the past, proceedings of the nato advanced research workshop on comparative time series analysis". Addison–Wesley Publishing Company.

[2] Farmer, J. D. and Sidorowich, J. J. (1987). "Predicting chaotic time series". *Physical Review Letters*, Vol., 59(8), pp. 845–848.

[3] Puskorius, G. V., Feldkamp, L. A., and Davis, L. I. Jr. (1996). "Dynamic neural network methods applies to on–vehicle idle speed control". *Proceedings of the IEEE*, Vol., 84, pp. 1407–1420.

[4] Werbos, P. J. (1990). "Backpropagation through time: What it does and how to do it". *Proceedings of the IEEE*, Vol., 78(10), pp. 1550–1560.

[5] Storn, R. and Price, K. (1997). "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces". *Journal of Global Optimization*, Vol., 11, pp. 341–359.

[6] Haykin, S. (1999). "Neural networks: A comprehensive foundation". New York: Macmillan College Publishing Company.

[7] Cybenko, G. (1989). "Approximations by superpositions of sigmoidal functions". *Mathematics of Control, Signals, and Systems*, Vol., 2, pp. 303–314.

[8] Hornik, K. (1989). "Multilayer feedforward networks are universal approximators". *Neural Networks*, Vol., 2, pp. 359–366.

[9] White, H. (1990). "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings". *Neural Networks*, Vol., 3, pp. 535–549.

[10] Pinkus, A. (1999). "Approximation theory of the MLP model in neural networks". *Acta Numerica*, Vol.8, pp. 143–195.

[11] Sandberg, I. W. and Xu, L. (1997). "Uniform approximation and gamma networks". *Neural Networks*, Vol., 10(5), pp. 781–784.

[12] Sandberg, I. W. and Xu, L. (1997). "Uniform approximation of multidimensional myopic maps". *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol., 44(6), pp. 477–485.

[13] Wan, E. A. (1994). "Time series prediction by using a connectionist network with internal delays". In Weigend, A. S. and Gershenfeld, N. A., editors, *Time Series Prediction: Forecasting the Future and Understanding the Past, Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis*. Addison–Wesley Publishing Company.

[14] Williams, R. J. and Zipser, D. (1995). "Gradient–based learning algorithms for recurrent networks and their computational complexity". In Chauvin, Y. and Rumelhart, D. E., editors, *Back–propagation: Theory, Architectures and Applications*, pp. 433–486. Hillsdale, N.J. : Lawrence Erlbaum Associates.

[15] Ilonen, J., Kamarainen, J. K., and Lampinen, J. (2003). "Differential evolution training algorithm for feed–forward neural networks". *Neural Processing Letters*, Vol., 17, pp. 93–105.

[16] Plagianakos, V.P. and Vrahatis, M.N., (1999), "Neural network training with constrained integer weights", In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Congress of Evolutionary Computation (CEC'99)*, pp. 2007–2013, Washington D.C., U.S.A., IEEE Press.

[17] Pavlidis, N G., Tasoulis, D. K., Androulakis, G. S., and Vrahatis, M. N. (2004). "Exchange–rate forecasting through distributed time–lagged feedforward neural networks". In Pardalos, P. M., Migdalas, A., and Baourakis, G., editors, *Supply Chain & Finance*, volume 2 of *Series on Computers & Operations Research*, pp. 284–298. World Scientific.

[18] Pavlidis, N. G., Tasoulis, D. K., Plagianakos, V. P., and Vrahatis, M. N., 2004), "Spiking neural network training using evolutionary algorithms", In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2004), Budapest, Hungary*.

[19] Fan, H. Y. and Lampinen, J. (2003). "A trigonometric mutation operation to differential evolution". *Journal of Global Optimization*, Vol., 27, pp. 105–129.

[20] Fraser, A. M. (1989). "Information and entropy in strange attractors". *IEEE Transactions on Information Theory*, Vol., 35, pp. 245–262.

[21] Kennel, M. B., Brown, R., and Abarbanel, H. D. (1992). "Determining embedding dimension for phase–space reconstruction using a geometrical construction". *Physical Review A*, Vol., 45(6), pp. 3403–3411.

[22] Hegger, R., Kantz, H., and Schreiber, T. (1999). "Practical implementation of nonlinear time series methods: The tisean package". *CHAOS*, Vol., 9(2), pp. 413–435.