Applied Soft Computing 49 (2016) 462–473

Contents lists available at ScienceDirect

Applied Soft Computing

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

Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance

Rohitash Chandra [a,*], Shelvin Chand [b]

[a] Artificial Intelligence and Cybernetics Research Group, Software Foundation, Nausori, Fiji
[b] School of Computing, Information and Mathematical Sciences, University of South Pacific, Suva, Fiji

ARTICLE INFO

Article history:
Received 16 October 2014
Received in revised form 15 August 2016
Accepted 16 August 2016
Available online 31 August 2016

Keywords:
Cooperative neuro-evolution
Feedforward networks
Recurrent neural networks
Time series prediction
Mobile application

ABSTRACT

The fusion of soft computing methods such as neural networks and evolutionary algorithms have given a very promising performance for time series prediction problems. In order to fully harness their strengths for wider impact, effective real-world implementation of prediction systems must incorporate the use of innovative technologies such as mobile computing. Recently, co-evolutionary algorithms have shown to be very promising for training neural networks for time series prediction. Cooperative coevolution decomposes a problem into subcomponents that are evolved in isolation and cooperation typically involves fitness evaluation. The challenge has been in developing effective subcomponent decomposition methods for different neural network architectures. In this paper, we evaluate the performance of two problem decomposition methods for training feedforward and recurrent neural networks for chaotic time series problems. We further apply them for financial prediction problems selected from the NASDAQ stock exchange. We highlight the challenges in real-time implementation and present a mobile application framework for financial time series prediction. The results, in general, show that recurrent neural networks have better generalisation ability when compared to feedforward networks for real-world time series problems.

1. Introduction

Neural networks have been very popular for time series prediction in a wide range of applications that include the field of finance and business [1–3]. In the past, feedforward and recurrent neural network architectures have been trained with a wide range of algorithms that have shown promising performance for time series prediction [4–6]. The algorithms used for training have been mainly gradient based approaches [4,7], neuro-evolution [8,6,9] and hybrid methods [10,11,1,2]. Hybrid neural network architectures [12] and feature reconstruction methods have also been used with exceptional results [5]. Hence, there is motivation to feature neural networks in real-world applications with innovative technologies.

Although gradient based approaches for neural networks have been very promising, alternative methods are have been explored which mainly include the use of evolutionary algorithms in several variations. Cooperative coevolution (CC) is an evolutionary algorithm that divides a problem into subcomponents [13] and has become popular for neuro-evolution [14–16]. CC methods used for training neural networks is known as *cooperative neuro-evolution* [17]. Cooperative neuro-evolution has shown to be very promising for time series prediction using feedforward [18] and recurrent neural networks [6,9]. Cooperative neuro-evolution features more diverse solutions through the sub-components when compared to conventional evolutionary algorithms [14]. In cooperative neuro-evolution, problem decomposition refers to the way the neural network is broken down into subcomponents. The two established problem decomposition methods are those on the synapse level [19] and neuron level [17,20]. Neuron level decomposition has shown good performance in pattern classification problems [21,20]. On the other hand, synapse level decomposition has shown good performance in control and time series prediction problems [19,8,6]; however, they reported poor performance for pattern classification problems [17].

The generalisation capability of neural networks has made them prominent in a wide range of real-world applications [22,23,5]. Although feedforward networks have been popular in time series predictions, recurrent architectures, in principle, are better suited for modelling temporal sequences [24]. It is important to validate if their architectural properties provide better performance for chaotic time series problems.

* Corresponding author.
E-mail address: c.rohitash@gmail.com (R. Chandra).

http://dx.doi.org/10.1016/j.asoc.2016.08.029
1568-4946/

The smartphone revolution has created the potential to feature real-time applications of neural networks for time series prediction problems. There have been several methods developed to guide investors using neural networks with mobile application on expected market trends [25]. Such applications are very valuable to the financial and business community as it would be able to inform them of future risks and benefits [3]. Investors rely on their mobile devices to keep track of their businesses and the stock markets in real-time [26]. However, computational intelligence methods such as neural networks require high levels of computational power for training. This makes it challenging for real-time application support in smartphones that have limitations in battery life and issues related to over-heating. Cloud computing infrastructure provides motivation for mobile implementation of such methods.

In this paper, we evaluate the performance of two problem decomposition methods for training feedforward and recurrent neural networks for chaotic time series problems. We present an extensive comparison of feedforward and recurrent network architectures on a combination of simulated and real-world chaotic time-series problems. We further apply them for financial time series prediction problems selected from the *national association of securities dealers automated quotations* (NASDAQ) stock exchange. We highlight the challenges in real-time implementation and present a mobile application framework. This paper extends preliminary results where cooperative neuroevolution was used for training feedforward networks for time series problems [18].

The rest of the paper is organised as follows. Section 2 gives a background on financial time series and mobile applications. Section 3 gives details of the cooperative co-evolutionary method for training feedforward and recurrent neural networks. Section 4 presents the results with the proposed framework for mobile application. Section 5 presents the discussion followed by the conclusion and directions for future research.

2. Background and related work

2.1. Financial time series prediction

The stock market is very unpredictable which makes investment very risky and uncertain. A number of soft computing methods have been used in the past for financial prediction. Chu et al. presented dual-factor fuzzy models for stock index forecasting [27]. Feng and Chou presented radial basis function networks [28] with promising performance for NASDAQ stock exchange problems. Hidden Markov model-based approaches [29] have been used in forecasting airline stocks while support vector regression [23] has shown good progress with complex problems in security markets. Neural Networks, in particular, have shown good results with financial time series prediction [30,1,2]. Polynomial pipe-lined neural networks [30] are constructed from a number of pi-sigma networks and recurrent pi-sigma networks linked together in sequence. They have shown good results in predicting the exchange rate between the United States (US) dollar and three other currencies. Spiking neural networks, and in particular, polychronous spiking networks [1] have been used in predicting the stock market and exchange rate data. Ridge polynomial neural networks [2] produced similar results when compared to multilayer perceptrons, functional link neural networks, and pi-sigma neural networks for the stock market and exchange rate data sets.

2.2. Mobile application

Mobile devices such as *smartphones* and *tablets* have become part of our lives due to portable access to software applications [25]. Mobile devices are not generally suitable for running applications with computational intelligence methods due to certain limitations that involve processing power, battery life and overheating. Even with quad-core processors, running computationally expensive algorithms on mobile devices consumes high battery power and produces heat. There is requirement of certain level of prioritization in terms of computation and data processing on such devices. Large amounts of data processing can considerably slow down such devices [31].

In the past, soft computing methods such as neural networks have been deployed as Android application for number plate recognition [32]. Although the application showed promising accuracy in prediction, it was not able to function well on devices with low memory and processing speed. There has been an interest on use of mobile phones in medical informatics [33] and several applications have been developed for different platforms for mobile health informatics [33,34]. Mobile applications have become very popular and successful in the education sector since mobile learning have been the focus in universities and schools [35].

Furthermore, neural networks have been used for recognition of handwritten digits via an Android application [36]. Although the application gave promising accuracy, it required extra memory which could be an issue on cheaper smartphones. Moreover, an Android application for automatic classification of environmental sounds employed a neural network for classification [37] and it was reported that the application required high computational power. In further studies, a mobile-based data mining package was successfully implemented using a Java-based framework to extend the *Weka* data-mining tool [38]. Therefore, success in implementations of soft computing based mobile applications in a wide range of areas motivate their implementations for mobile-based financial prediction.

3. Evaluation of co-evolutionary neural network architectures for time series prediction

We provide an overview of the evaluation of coevolutionary methods for training feed-forward and recurrent neural networks for chaotic time series prediction. We first provide details of Elman recurrent networks and then present cooperative neuro-evolution and time series problems.

3.1. Elman recurrent neural networks

Recurrent neural networks (RNNs) are dynamical systems that feature feedback connections that make the main difference when compared to feedforward networks. One popular architecture is the Elman RNN [39], which is composed of an *input layer*, a *context layer* which provides state information, a *hidden layer* and an *output layer* as shown in Fig. 1. Each layer contains one or more neurons which propagate information from one layer to another by computing a non-linear function of their weighted sum of inputs. The equation of the dynamics of the change of hidden state neuron activations in the context layer is given in Eq. (1).

$$y_i(t) = f\left(\sum_{k=1}^{K} v_{ik}\ \ y_k(t-1) + \sum_{j=1}^{J} w_{ij} x_j(t-1)\right) \tag{1}$$

where $y_k(t)$ and $x_j(t)$ represent the output of the context state neuron and input neurons, respectively. v_{ik} and w_{ij} represent their corresponding weights; i represents the number of input neurons while j and k represents the number of hidden and context layer neurons, respectively. $f(.)$ is a sigmoid transfer function. Time (t) refers to each data point of the time series sample used for prediction.

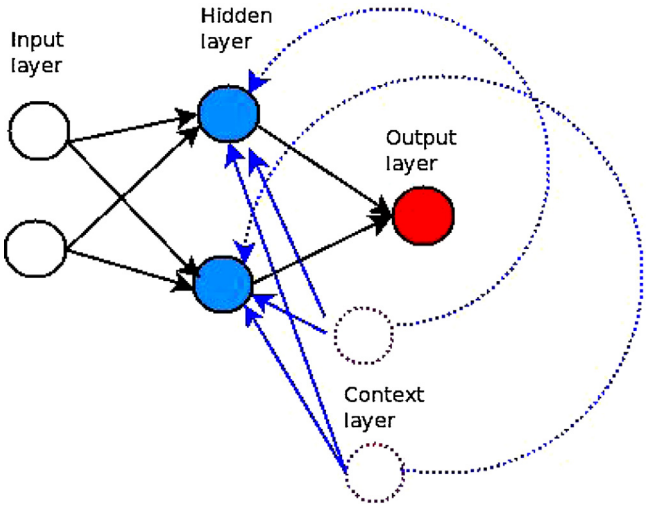


Fig. 1. Elman RNN showing 2 input and hidden neurons with recurrent connections through the context layer. Note that the Elman RNN used for time series problems is composed of 1 neuron in the input layer and 1 output neuron. It unfolds by n time steps which is given by the embedding dimension of the time series.

Elman RNNs have shown to be very promising for time series prediction problems and have been trained using back-propagation through-time [4,5] and cooperative neuro-evolution [6,9].

3.2. Cooperative neuro-evolution and problem decomposition

Cooperative coevolution employs problem decomposition strategies that decompose the network topology into subcomponents. It determines how the weights are encoded into the subcomponents that are implemented as sub-populations. The two main problem decomposition methods are *neuron level* (NL) and *synapse Level* (SL) problem decomposition.

In SL problem decomposition, the neural network is decomposed to its lowest level where each weight connection (synapse) forms a subcomponent. Examples include cooperatively co-evolved synapses neuro-evolution [19] and neural fuzzy network with cultural cooperative particle swarm optimisation [8]. In NL problem decomposition, the neurons in the network act as the reference point for the decomposition. Examples include enforced sub-populations [16,21] and neuron-based sub-population [17,20].

Algorithm 1. Cooperative coevolution.

Step 1: Decompose the problem (NL or SL)
Step 2: Initialise and cooperatively evaluate each sub-population
for each *cycle* until termination **do**
  for each Sub-population **do**
    for n Generations **do**
      i) Select and create new offspring
      ii) Cooperatively evaluate the new offspring
      iii) Update sub-population
    end for
  end for
end for

In Algorithm 1, the neural network is decomposed using the NL or SL method into k subcomponents. In SL, k is the number of weights and biases within the neural network which apply for both feedforward and recurrent neural networks. In NL for feedforward networks, k is the total number of hidden and output neurons [6], whereas in RNNs, k is total number of context, hidden and output neurons [20]. These are explicitly shown in Figs. 2 and 3, respectively.

In the *initialisation* phase, all the sub-populations are populated with random values and evaluated. An individual whose fitness is to be evaluated is concatenated with best individuals from the rest of the sub-populations. Note that arbitrary fitness values are assigned during initialisation since the respective individuals in all the sub-populations do not have a fitness. The second phase is the *evolution* phase in which an individual whose fitness has to be evaluated is concatenated with the best individuals from the rest of the sub-populations. In the later case, all the individuals obtain their true fitness in the first cycle of evolution. A cycle is completed when all the sub-populations have been evaluated in a round-robin fashion for a given number of generations which is known as the *depth of search*.

The sub-populations are evolved in isolation and cooperation takes place for fitness evaluation [13]. The goal of the evolutionary process is to increase the fitness which tends to decrease the neural network error which is defined as the root-mean-squared-error for time series problems.

Evaluation of the fitness of each individual for a particular sub-population is done cooperatively with the fittest individuals from the other sub-populations [14]. Cooperative evaluation for an individual j, in a particular sub-population i, is done by concatenating the fittest individuals from the rest of the sub-populations and encoding them into a neural network where its fitness is evaluated [13,40,20,6]. The fitness of the overall neural network is then assigned to that particular individual even though it was just a small component of the overall network. This is further illustrated in Figs. 3 and 2.

All the sub-populations are evolved for a fixed number of generations. The generalized generation gap model with parent-centric crossover operator (G3-PCX) evolutionary algorithm [41] is the designated evolutionary algorithm in the sub-populations.

3.3. Time series data prepossessing for neural networks

Time series prediction involves a large series of data points that are taken at regular intervals. We need to transform them in order to build prediction models using soft computing methods such as neural networks. The basic transformation or reconstruction requires the time series data to be broken down into smaller pieces or windows that are taken at regular intervals. The process is described in detail hereafter.

Given an observed time series $x(t)$, an embedded phase space $Y(t) = [(x(t), x(t-T), \ldots, x(t(D-1)T)]$ can be generated, where, T is the time delay, D is the embedding dimension, $t = 0, 1, 2, \ldots, N - DT - 1$ and N is the length of the original time series [42]. Taken's theorem expresses that the vector series reproduces many important characteristics of the original time series. The right values for D and T must be chosen in order to efficiently apply Taken's theorem [43]. Taken proved that if the original attractor is of dimension d, then $D = 2d + 1$ will be sufficient to reconstruct the attractor [42].

The reconstructed vector is used to train the feedforward and recurrent network for one-step-ahead prediction where 1 neuron is used in the output layer. In the case of feedforward network, D is the number of input neurons. In the case of recurrent networks, 1 is the number of input neuron. The recurrent network unfolds k steps in time which correspond to the embedding dimension D.

The root mean squared error (RMSE) is used to measure the prediction performance of the recurrent and feed-forward network. The normalised mean squared error (NMSE) is used for further comparison with results from the literature. These are given in Eqs. (2) and (3).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2}$$

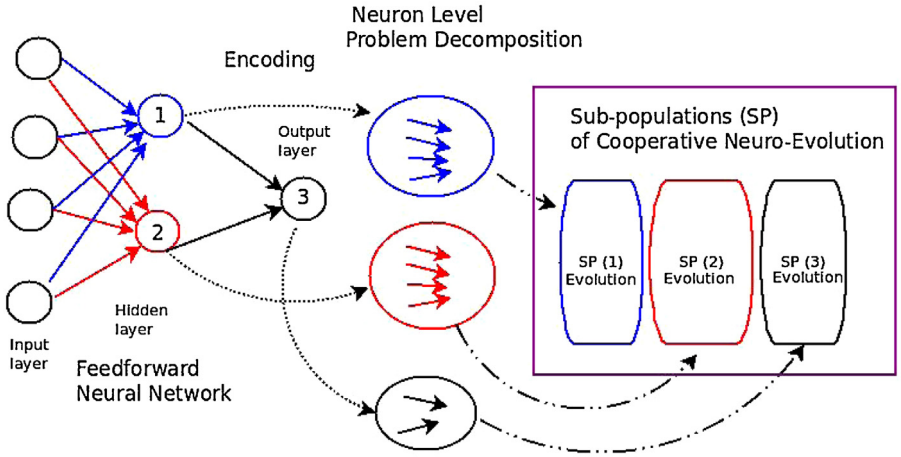


Fig. 2. FNN with NL problem decomposition. Note that 4 input neurons represent time series reconstruction with embedding dimension of 4.

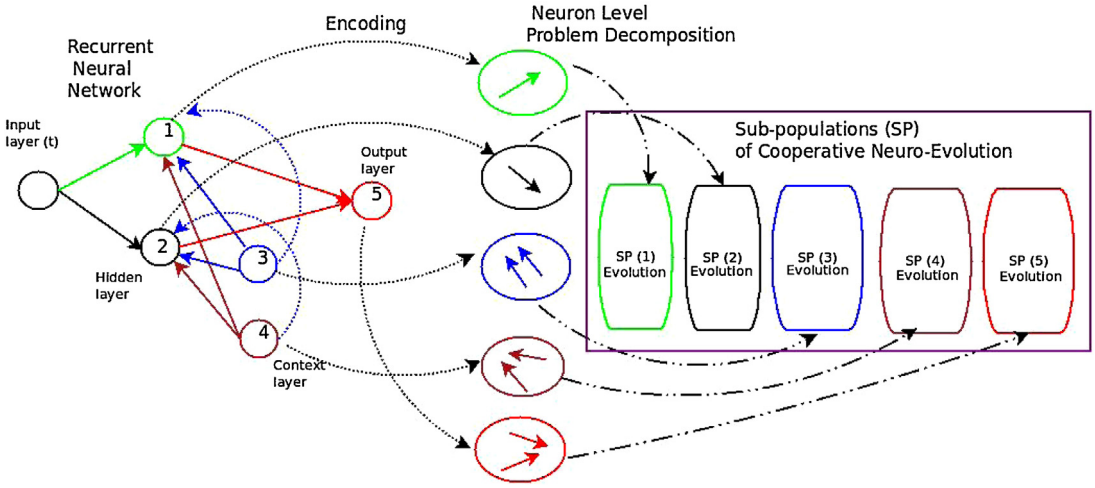


Fig. 3. Elman RNN with NL problem decomposition. Note that 1 input neuron can represent time series reconstruction with any embedding dimension. The network unfolds to the size of the embedding dimension.

$$NMSE = \left(\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y}_i)^2} \right) \tag{3}$$

where y_i, $\hat{y}_i$ and $\bar{y}_i$ are the observed data, predicted data and average of observed data, respectively. N is the length of the observed data. These two performance measures are used in order to compare the results with the literature.

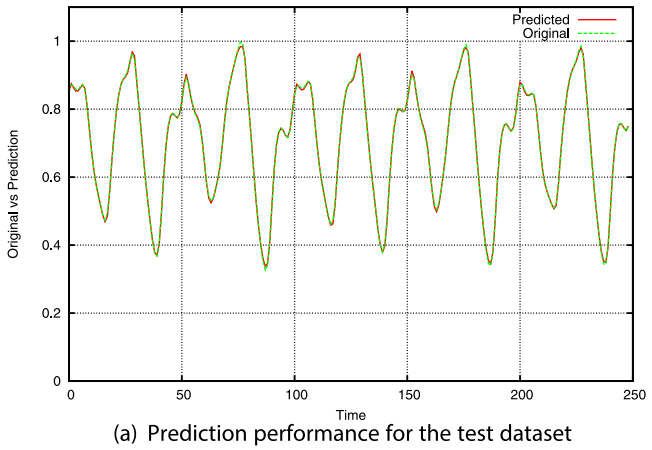
3.4. Mobile application framework for financial time series prediction

We present a mobile application framework which can notify potential investors on possible market closing prices through which the investors can make appropriate investment decisions. The framework is composed of a mobile application and the intelligent prediction system that consists the selected neural network trained on a cloud computing infrastructure [44]. Each day the system is trained with new data that is fetched using web services [45]. The prediction from the respective neural network architectures can be updated in a relational database which maintains the history of predictions to generate interactive graphs and visualisations for the mobile application [46]. Each time the application is loaded, it checks with the cloud computing platform if any data has been added. The main computation in terms of training the intelligent prediction system is assigned to the cloud infrastructure. Hence, the application can be installed on any type of mobile device regardless of their processing power.

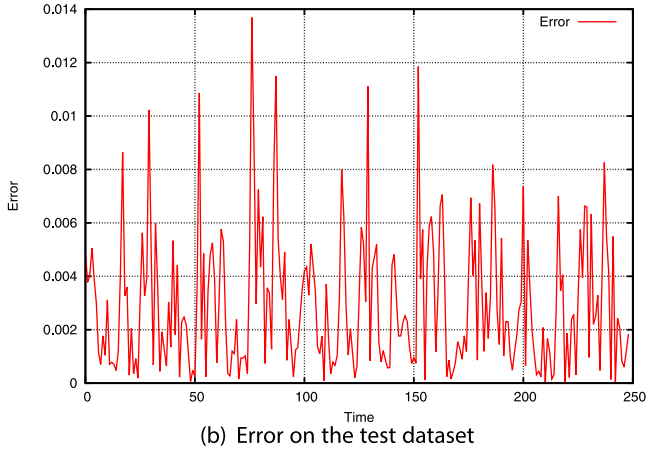
The mobile application can display the previous prediction results along with their errors in the prediction of the respective neural network architectures so that investors are aware of the risks involved. Strategies can be implemented for further access to the prediction performance from the previous timespan that can be given by weeks or months.

It is important to prioritize in terms of what information to send to the mobile device so that it is not overloaded with large amounts of data. An alternative approach would be to train the neural network on the cloud computing infrastructure and then transfer the weights or knowledge to the mobile device. The mobile application will receive the knowledge and apply it to the local neural networks built within the application. As a result, each day the application will fetch the latest market prices from the cloud server and input the prices into its local neural network. Both options will require the same amount of prediction data and can be refined as more data is gathered automatically using web services.

The mobile application needs to offer a high level of security to its users that can be implemented as three-factor remote authentication scheme that includes biometric information for authentication [47]. Security can be further enhanced by the implementation of robust biometric features that can include signature



(a) Prediction performance for the test dataset



(b) Error on the test dataset

Fig. 4. Overview of mobile application framework for real-time financial prediction.

and iris recognition [48,49]. Fig. 4 shows an overview of the proposed mobile application framework that highlights the need for secure data flow, biometrics for the mobile devices and use of cloud computing infrastructure for the intelligent prediction system.

4. Simulation and analysis

This section presents an experimental study that evaluates the performance of feedforward networks and Elman RNNs for time series prediction. In the beginning, three benchmark time series problems are used and the results are compared with the literature. Afterwards, the proposed approach is applied to three NASDAQ financial time series problems. A mobile application framework for real-time implementation also given.

The NL and SL decomposition methods are used for feedforward networks (FNNs) and Elman RNNs. Their performance is evaluated in terms of training and generalisation. In the benchmark chaotic time series problems, the Mackey–Glass times series [50] and Lorenz time series [51] are the two simulated time series. The real-world problem is the Sunspot time series [52]. The experimental setup for Elman RNNs is taken from our previous work [6].

4.1. Benchmark chaotic time series problems

The *Mackey–Glass time series* has been used in literature as a benchmark problem due to its chaotic nature [50]. The differential equation used to generate the Mackey-Glass time series is given in Eq. (4).

$$\frac{\delta x}{\delta t} = \frac{ax(t-\tau)}{[1+x^c(t-\tau)]} - bx(t) \tag{4}$$

In Eq. (4), the delay parameter τ determines the characteristic of the time series, where $\tau > 16.8$ produces chaos. The selected parameters for generating the time series is taken from the literature [11,5]; where the constants $a = 0.2$, $b = 0.1$ and $c = 10$. The chaotic time series is generated by using time delay $\tau = 17$ and initial value $x(0) = 1.2$.

The experiments use the chaotic time series with a length of 1000 generated by Eq. (4). The first 500 samples are used for training the respective neural network architecture while the rest are used for testing. The time series is scaled in the range [0,1]. The phase space of the original time series is reconstructed with the embedding dimensions $D = 4$ and $T = 2$.

The *Lorenz time series* was introduced by Edward Lorenz who has extensively contributed to the establishment of Chaos theory [51]. The Lorenz equation are given in Eq. (5).

$$\begin{aligned} \frac{dx(t)}{dt} &= \sigma[y(t) - x(t)] \\ \frac{dy(t)}{dt} &= x(t)[r - z(t)] - y(t) \\ \frac{dz(t)}{dt} &= x(t)y(t) - bz(t) \end{aligned} \tag{5}$$

where η, r, and b are dimensionless parameters. The typical values of these parameters are $\eta = 10$, $r = 28$, and $b = 8/3$ [53,5]. The x-coordinate of the Lorenz time series is chosen for prediction and 1000 samples are generated. The time series is scaled in the range $[-1,1]$. The first 500 samples are used for training and the remaining 500 is used for testing. The phase space of the original time series is reconstructed with the embedding dimensions $D = 4$ and $T = 2$.

The *Sunspot time series* is a good indication of the solar activities for solar cycles which impact Earth's climate, weather patterns, satellite and space missions [54]. The prediction of solar cycles is difficult due to its complexity. The monthly smoothed Sunspot time series has been obtained from the World Data Centre for the Sunspot Index [52]. The Sunspot time series from November 1834 to June 2001 is selected which consists of 2000 points. This interval has been selected in order to compare the performance the proposed methods with those from literature [11,5]. The time series is scaled in the range $[-1,1]$. The first 1000 samples are used for training while the remaining 1000 samples are used for testing. The phase space of the original time series is reconstructed with the embedding dimensions $D = 5$ and $T = 2$. Note that the selected problems are scaled in the range of [0,1] and $[-1,1]$ in order to provide a fair comparison with the literature.

4.2. Experimental design

The respective neural network architectures use sigmoid units in the hidden layer for the different problems. In the output layer, a sigmoid unit is used for the Mackey–Glass time series while hyperbolic tangent unit is used for Lorenz and Sunspot time series. The RMSE and NMSE given in Eq. (2 and 3) are used as the main performance measures.

Each neural network architecture was tested with different numbers of hidden neurons. The FNN uses one hidden layer and the RNN unfolds in time according to the embedding dimension (D) of the particular time series. For the Mackey-Glass and Lorenz time series, we used embedding dimension $D = 4$ while for Sunspot time series we used $D = 5$. A time-lag of 2 ($T = 2$) was used for all

Table 1
The prediction training and generalisation performance (RMSE) of NL and SL Mackey–Glass time series.

Prob.	*H*	Training	Generalization	Best
FNN-NL	3	0.0107 ± 0.0013	0.0107 ± 0.0013	0.0050
	5	0.0089 ± 0.0010	0.0088 ± 0.0010	**0.0037**
	7	**0.0078 ± 0.0008**	**0.0078 ± 0.0008**	0.0040
	9	0.0080 ± 0.0006	0.0079 ± 0.0006	0.0038
RNN-NL	3	0.0114 ± 0.0010	0.0114 ± 0.0011	0.0055
	5	0.0108 ± 0.0006	0.0108 ± 0.0007	0.0061
	7	0.0111 ± 0.0008	0.0111 ± 0.0009	**0.0053**
	9	**0.0107 ± 0.0007**	**0.0107 ± 0.0007**	0.0063
FNN-SL	3	0.0228 ± 0.0022	0.0229 ± 0.0022	0.0125
	5	0.0195 ± 0.0012	0.0195 ± 0.0012	0.0124
	7	0.0177 ± 0.0009	0.0178 ± 0.0009	0.0121
	9	**0.0166 ± 0.0009**	**0.0166 ± 0.0009**	**0.0112**
RNN-SL	3	0.0181 ± 0.0021	0.0182 ± 0.0021	0.0098
	5	**0.0164 ± 0.0019**	**0.0164 ± 0.0019**	**0.0082**
	7	0.0191 ± 0.0041	0.0191 ± 0.0041	0.0090
	9	0.0296 ± 0.0069	0.0297 ± 0.0069	0.0105

three problems. These values have shown to give good results in literature [6].

We employ 1 generation as the *depth of search* as it has given optimal performance for both NL and SL decomposition methods in the previous study [20]. Note that all the sub-populations evolve for the same depth of search. The termination condition of the three problems is when a total of 50,000 function evaluations have been reached.

The G3-PCX algorithm uses the *generation gap model* [41] for selection. We use G3-PCX parameters as follows. The pool size of 2 parents and 2 offspring with a population size of 300 is used [6].

4.3. Results

This section provides the results of the evaluation of feedforward (FNN) and recurrent neural networks (RNN) with NL and SL decomposition methods for the benchmark problems. The results report the mean, 95% confidence interval and the best performance (RMSE) from the 50 independent experimental runs. Note that the best cases for each problem have been highlighted in bold by the lowest values in the RMSE. Moreover, we also note that SL and NL methods for recurrent neural networks have been proposed previously [6]. Here, we further evaluate their performance on the same datasets in order to compare with feedforward networks as a preliminary study for financial prediction.

In the Mackey–Glass problem, the FNN was able to give better generalisation in comparison to the RNN for both the problem decomposition methods as shown in Table 1. We observe that the RNN gave better generalisation performance when compared to FNN for the Lorenz problem as shown in Table 2. Moreover, NL was not able to outperform SL. In the Sunspot problem, FNN produced a poorer performance when compared to the RNN as shown in Table 3. NL gave better performance when compared to SL for both the network architectures. The results, in general, show that the neural network with the lowest training error does not always give the best generalisation performance. This is due to over-fitting and noise in the dataset. In the Lorenz and Sunspot problems, the performance was generally better with fewer hidden neurons. This was different in the case of the Mackey–Glass problem.

Overall, the RNN outperformed the FNN for two of the three benchmark problems. Furthermore, NL has given better performance when compared to SL for most cases for both the neural network architectures.

Figs. 5–7 shows the prediction performance from the best experimental runs of the respective problems. We observe that the error bars in the Sunspot problem show higher variations at specific regions when compared to Mackey-Glass and Lorenz problems. This is due to their difference in category of application which is distinguished by real-world and simulated time series.

Table 4 further compares the performance of the evaluated methods with the literature which reports the mean and the 95%

Table 2
The prediction training and generalisation performance (RMSE) of NL and SL for the Lorenz time series.

Prob.	*H*	Training	Generalization	Best
FNN-NL	3	**0.0173 ± 0.0028**	**0.0180 ± 0.0032**	0.0042
	5	0.0252 ± 0.0071	0.0272 ± 0.0072	**0.0022**
	7	0.0319 ± 0.0083	0.0351 ± 0.0089	0.0039
	9	0.0347 ± 0.0074	0.0365 ± 0.0075	0.0062
RNN-NL	3	0.0178 ± 0.0015	0.0184 ± 0.0016	0.0072
	5	**0.0132 ± 0.0014**	**0.0136 ± 0.0015**	**0.0031**
	7	0.0143 ± 0.0015	0.0147 ± 0.0016	0.0051
	9	0.0149 ± 0.0016	0.0155 ± 0.0017	0.0051
FNN-SL	3	0.0608 ± 0.0294	**0.0451 ± 0.0220**	0.0153
	5	**0.0526 ± 0.0081**	0.0546 ± 0.0084	0.0082
	7	0.0574 ± 0.0072	0.0605 ± 0.0074	**0.0078**
	9	0.0679 ± 0.0156	0.0717 ± 0.0156	0.0128
RNN-SL	3	0.0209 ± 0.0024	0.0214 ± 0.0025	0.0078
	5	0.0168 ± 0.0020	0.0175 ± 0.0021	**0.0043**
	7	0.0164 ± 0.0028	0.0172 ± 0.0029	0.0059
	9	**0.0144 ± 0.0019**	**0.0151 ± 0.0020**	0.0064

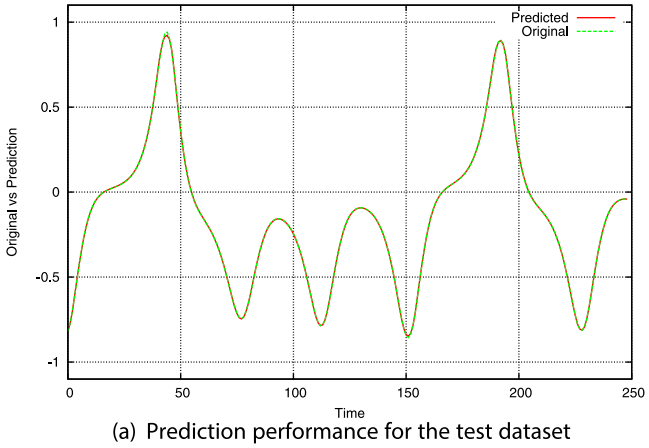
Table 3
The prediction training and generalisation performance (RMSE) of NL and SL Sunspot time series.

Prob.	*H*	Training	Generalization	Best
FNN-NL	3	0.0217 ± 0.0028	**0.0558 ± 0.0074**	**0.0165**
	5	**0.0213 ± 0.0025**	0.0631 ± 0.0077	0.0181
	7	0.0273 ± 0.0032	0.0724 ± 0.0075	0.0169
	9	0.0292 ± 0.0043	0.0737 ± 0.0087	0.0186
RNN-NL	3	0.0190 ± 0.0015	**0.0391 ± 0.0089**	0.0144
	5	**0.0160 ± 0.0011**	0.0437 ± 0.0111	**0.0124**
	7	0.0168 ± 0.0012	0.0434 ± 0.0113	0.0152
	9	0.0178 ± 0.0011	0.0689 ± 0.0179	0.0147
FNN-SL	3	**0.0542 ± 0.0028**	**0.0593 ±0.0035**	**0.0336**
	5	0.0728 ± 0.0166	0.1024 ±0.0264	0.0341
	7	0.0787 ± 0.0143	0.1260 ± 0.0211	0.0478
	9	0.0835 ± 0.0304	0.1575 ± 0.0341	0.0582
RNN-SL	3	0.0207 ± 0.0022	**0.0512 ± 0.0123**	0.0169
	5	0.0179 ± 0.0019	0.0537 ± 0.0128	0.0166
	7	**0.0165 ± 0.0010**	0.0566 ± 0.0155	0.0151
	9	0.0171 ± 0.0016	0.0651 ± 0.0189	**0.0150**

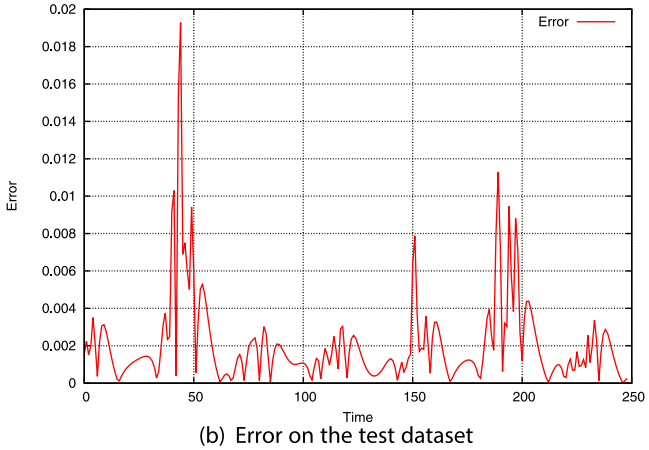
Table 4
Comparison with the literature.

Problem	Method	RMSE (Mean ± CI)
Sunspot	Wavelet packet multilayer perceptron [10]	1.25E−01
	Hybrid NARX-Elman recurrent neural network [5]	1.19E−02
	Evolutionary multi-objective recurrent neural network ensembles [55]	1.56E−02 ± 1.11E−03
	Co-evolutionary recurrent neural networks (Synapse level) [6]	6.88E−02 ± 2.66E−02
	Co-evolutionary recurrent neural networks (Neuron level) [6]	5.58E−02 ± 8.01E−03
	Competitive co-evolutionary recurrent neural networks (two islands) [56]	4.06E−02 ± 5.90E−03
	Competitive co-evolutionary recurrent neural networks (three islands) [56]	5.95E−02 ± 1.62E−02
	Evaluated CCRNN-SL	5.12E−02 ± 1.23E−02
	Evaluated CCRNN-NL	3.91E−02 ± 8.90E−03
	Evaluated CCFNN-SL	5.93E−02 ± 3.50E−03
	Evaluated CCFNN-NL	5.58E−02 ± 7.40E−03
Lorenz	Pseudo Gaussian - radial basis neural network	9.40E−02
	Auto regressive moving average with neural networks [53]	8.76E−02
	Hybrid NARX-Elman recurrent neural network [5]	1.08E−04
	Back-propagation and genetic algorithm with residual analysis [57]	2.96E−02
	Co-evolutionary recurrent neural networks (Synapse level) [6]	1.95E−02 ± 2.59E−03
	Co-evolutionary recurrent neural networks (Neuron level) [6]	1.82E−02 ± 2.82E−03
	Competitive co-evolutionary recurrent neural networks (two islands) [56]	9.15E−03 ± 1.01E−03
	Competitive co-evolutionary recurrent neural networks (three islands) [56]	1.78E−02 ± 1.90E−03
	Evaluated CCRNN-SL	1.51E−02 ± 2.00E−03
	Evaluated CCRNN-NL	1.36E−02 ± 1.50E−03
	Evaluated CCFNN-SL	4.51E−02 ± 2.20E−02
	Evaluated CCFNN-NL	1.80E−02 ± 3.20E−03
Mackey	Adaptive neuro fuzzy inference system [12]	1.50E−03
	Genetic fuzzy predictor ensemble [58]	3.80E−02
	Pseudo Gaussian - radial basis network [59]	2.80E−03
	Neural fuzzy network with particle swarm optimisation (PSO) [8]	2.10E−02
	Neural fuzzy network with cultural cooperative PSO [8]	8.42E−03
	Neural fuzzy network with cooperative PSO [8]	1.76E−02
	Neural fuzzy network with differential evolution [8]	1.62E−02
	Neural fuzzy network with genetic algorithm [8]	1.63E−02
	Back-propagation and genetic algorithm with residual analysis [57]	1.30E−03
	Hybrid NARX-Elman recurrent neural network [5]	3.72E−05
	Evolutionary multi-objective recurrent neural networkensembles [55]	1.23E−02 ± 4.49E−03
	Co-evolutionary recurrent neural networks (Synapse level) [6]	9.39E−03± 5.57E−04
	Co-evolutionary recurrent neural networks (Neuron level) [6]	1.23E−02 ± 9.16E−04
	Competitive co-evolutionary recurrent neural networks (two islands) [56]	8.47E−03 ± 6.30E−04
	Competitive co-evolutionary recurrent neural networks (three islands) [56]	7.89E−03 ± 5.36E−04
	Evaluated CCRNN-SL	1.64E−02 ± 1.90E−03
	Evaluated CCRNN-NL	1.07E−02 ± 7.00E−04
	Evaluated CCFNN-SL	1.66E−02± 9.00E−04
	Evaluated CCFNN-NL	7.80E−03 ± 8.00E−04

confidence interval (CI). We note that Co-evolutionary RNNs with SL and NL proposed in the literature [6] used a smaller population size and hence the results slightly vary with those evaluated in this work. We observe that in the MAckey-Glass problem, the evaluated methods which are highlighted in bold outperformed some of the methods such as neural fuzzy network with genetic algorithm (GA) [8] and neural fuzzy network with differential evolution (DE) [8]. In the case of the Lorenz problem, we observe that the evaluated methods outperform some of the methods such as the autoregressive moving average (ARMA) with feedforward networks [53]. Furthermore, in the Sunspot problem, the evaluated methods outperformed some of the methods in the literature such

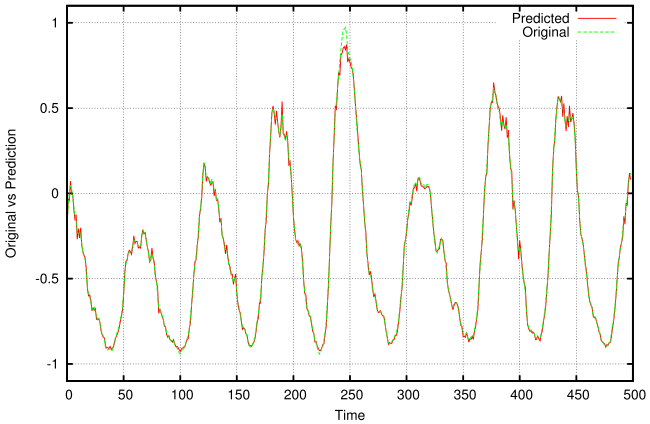


(a) Prediction performance for the test dataset

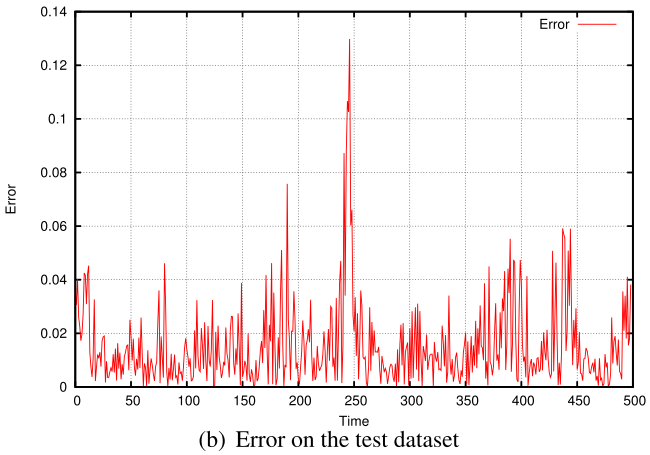


(b) Error on the test dataset

Fig. 5. Performance given by NL (feedforward network) for a typical experimental run on the testing set for Mackey Glass time series.



(a) Prediction performance for the test dataset



(b) Error on the test dataset

Fig. 6. Performance given by NL (feedforward network) for a typical experimental run on the testing set for Lorenz time series.

as the wavelet packet multilayer perceptron [10]. We note that several methods from the literature have better performance as they have used hybrid methods in the prediction model architectures and training [5].

4.4. Financial time series prediction

This section presents the results of the experiments for financial time series problem using selected problem decomposition methods cooperative neuro-evolution for feedforward and recurrent neural networks.

4.4.1. Experimental design

The financial time series data set is taken from the NASDAQ stock exchange [60]. It contains daily closing prices for ACI Worldwide, Staples Inc., and Seagate Technology Holdings. The data set for each company contained closing stock prices from December 2006 to October 2010 which is equivalent to around 800 data points. The stock prices fluctuated over the 3 year period, therefore, the dataset also features the recession that hit the United States market in 2008. We used embedding dimension $D = 5$ and time lag $T = 2$, to reconstruct the time series data using Taken's theorem in order to generate the training and testing datasets.

The closing stock prices were scaled between 0 and 1. The maximum number of function evaluations was set at 50,000 which was used as a termination condition for the algorithm. Similar to the previous data sets, we used a population size of 300. In the respective neural network architectures, sigmoid units were employed in the hidden and output layers.

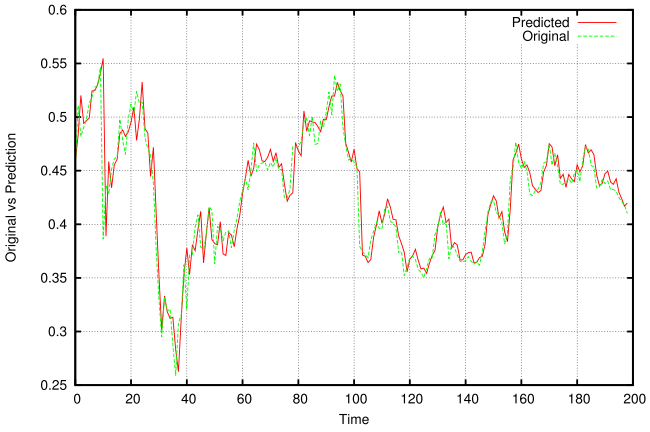


Fig. 7. Performance given by NL (feedforward network) for a typical experimental run on the testing set for Sunspot time series.

4.4.2. Results

The results for 50 experimental runs are given in Tables 5–7. The mean, 95% confidence interval, and the best performance (RMSE) has been given for the respective training and test datasets. The best results are highlighted in bold.

Table 5
The prediction training and generalisation performance (RMSE) of NL and SL for the ACI Worldwide financial data.

Prob.	*H*	Training	Generalization	Best
FNN-NL	3	0.0208 ± 0.0004	0.0213 ± 0.0004	**0.0191**
	5	0.0202 ± 0.0003	0.0209 ± 0.0003	0.0194
	7	0.0198 ± 0.0003	0.0208 ± 0.0003	0.0195
	9	**0.0197 ± 0.0003**	**0.0207 ± 0.0003**	0.0196
RNN-NL	3	0.0207 ± 0.0004	0.0211 ± 0.0013	**0.0193**
	5	0.0202 ± 0.0003	**0.0204 ± 0.0002**	**0.0193**
	7	**0.0201 ± 0.0002**	**0.0204 ± 0.0004**	**0.0193**
	9	0.0203 ± 0.0002	0.0205 ± 0.0007	**0.0193**
FNN-SL	3	0.0474 ± 0.0023	0.0483 ± 0.0042	0.0299
	5	0.0297 ± 0.0016	0.0258 ± 0.0017	**0.0192**
	7	0.0269 ± 0.0008	**0.0246 ± 0.0011**	0.0200
	9	**0.0266 ± 0.0007**	0.0247 ± 0.0009	0.0195
RNN-SL	3	0.0226 ± 0.0007	0.0219 ± 0.0008	**0.0191**
	5	0.0219 ± 0.0007	**0.0210 ± 0.0005**	0.0193
	7	0.0211 ± 0.0005	0.0211 ± 0.0006	0.0193
	9	**0.0210 ± 0.0004**	0.0217 ± 0.0006	0.0195

Table 6
The prediction training and generalisation performance (RMSE) of NL and SL for the Staples Inc. financial data.

Prob.	*H*	Training	Generalization	Best
FNN-NL	3	0.0172 ± 0.0006	**0.0802 ± 0.0102**	0.0301
	5	0.0171 ± 0.0005	0.0883 ± 0.0110	0.0353
	7	**0.0167 ± 0.0003**	0.0977 ± 0.0124	0.0276
	9	0.0168 ± 0.0004	0.1082 ± 0.0129	**0.0271**
RNN-NL	3	0.0167 ± 0.0002	**0.0767 ± 0.0104**	**0.0299**
	5	**0.0165 ± 0.0001**	0.0851 ± 0.0109	0.0330
	7	0.0166 ± 0.0001	0.0777 ± 0.0072	0.0379
	9	0.0167 ± 0.0001	0.0841 ± 0.0083	0.0357
FNN-SL	3	0.0341 ± 0.0021	0.1264 ± 0.0099	0.0444
	5	0.0275 ± 0.0010	0.0983 ± 0.0113	0.0260
	7	0.0254 ± 0.0009	**0.0757 ± 0.0107**	**0.0249**
	9	**0.0237 ± 0.0006**	0.0892 ± 0.0111	0.0259
RNN-SL	3	0.0193 ± 0.0007	**0.0532 ± 0.0056**	**0.0218**
	5	**0.0192 ± 0.0008**	0.0882 ± 0.0147	0.0244
	7	0.0205 ± 0.0012	0.0884 ± 0.0126	0.0303
	9	0.0250 ± 0.0032	0.0992 ± 0.0130	0.0329

Table 7
The prediction training and generalisation performance (RMSE) of NL and SL for the Seagate Technology Holdings financial data.

Prob.	*H*	Training	Generalization	Best
FNN-NL	3	0.0188 ± 0.0004	0.2015 ± 0.0359	0.0373
	5	0.0186 ± 0.0004	**0.1609 ± 0.0278**	0.0458
	7	0.0185 ± 0.0002	0.1948 ± 0.0365	0.0274
	9	**0.0184 ± 0.0001**	0.1944 ± 0.0359	**0.0224**
RNN-NL	3	0.0186 ± 0.0002	0.2141 ± 0.0361	0.0318
	5	**0.0184 ± 0.0001**	0.1839 ± 0.0300	**0.0169**
	7	0.0186 ± 0.0002	**0.1443 ± 0.0244**	0.0212
	9	0.0185 ± 0.0002	0.1759 ± 0.0391	0.0229
FNN-SL	3	0.0365 ± 0.0025	0.2179 ± 0.0097	0.1197
	5	0.0284 ± 0.0015	**0.1956 ± 0.0283**	**0.0374**
	7	0.0257 ± 0.0009	0.2316 ± 0.0476	0.0439
	9	**0.0237 ± 0.0007**	0.3229 ± 0.0509	0.0593
RNN-SL	3	**0.0206 ± 0.0005**	0.1902 ± 0.0302	0.0388
	5	0.0218 ± 0.0011	**0.1709 ± 0.0329**	**0.0240**
	7	0.0238 ± 0.0018	0.2134 ± 0.0334	0.0282
	9	0.0293 ± 0.0040	0.2275 ± 0.0434	0.0410

In the ACI Worldwide Inc. problem (Table 5), the generalisation performance was very competitive for both network architectures. The RNN performed slightly better than the FNN in terms of training and generalisation. We further observe that the NL decomposition gave better performance in comparison to SL decomposition. The performance on NL improved as the number of hidden neurons increased. In Staples Inc. problem (Table 6), the RNN outperformed FNN in both the training and generalisation performance. We further observe that SL decomposition gave better performance than NL for both the neural network architectures. The Seagate Technology Holdings problem (Table 7) has shown to be the most difficult problem. We observe that the RNN outperformed FNN and NL outperformed SL on both the network architectures. The proposed methods showed signs of over-fitting with this problem as the

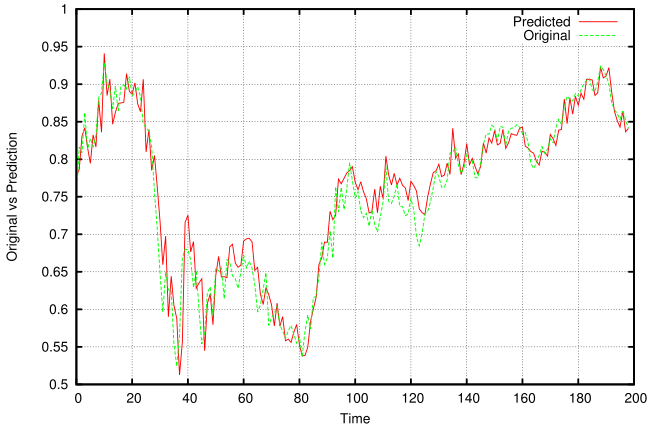


Fig. 8. Performance given by NL (feed-forward neural network) for a typical experimental run on the testing set for ACI Worldwide.

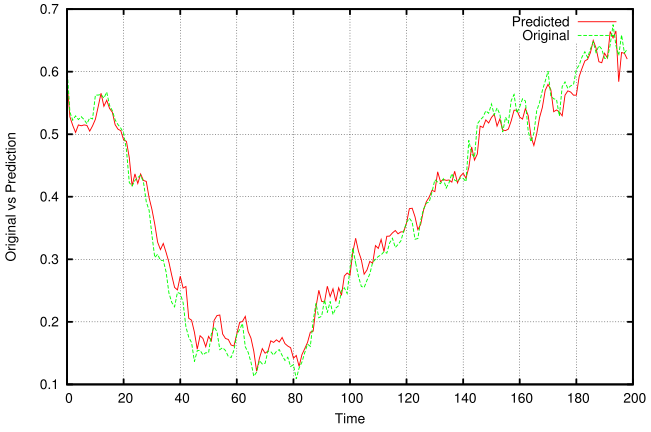


Fig. 9. Performance given by NL (feed-forward neural network) for a typical experimental run on the testing set for Staples Inc.

training and generalisation performance had a relatively large variance. We further highlight that the training performance is much better than the generalisation performance. This could be due to the difference in the trend taken by the stock market that highly differentiates the training dataset from the testing.

We observed that NL outperformed SL in two of the three financial time series problems. In the comparison of the two different network architectures, we observe that the RNN gave a better generalisation performance than FNN. The RNN gave superior performance in all three financial time series problems. Therefore, in general, we can conclude that the RNN is more suited to financial prediction. Figs. 8–10 provide the prediction performance from the best experimental runs using feedforward networks. They give an overview of the prediction performance when compared to the actual values.

5. Discussion

In determining the best neural network architecture, we found that the RNN outperformed the FNN on two out of the three benchmark problems that consist of the Lorenz and Sunspot time series. Furthermore, in the Sunspot and Lorenz time series, lower number of hidden neurons produced better generalisation when compared to Mackey-Glass time series. A case-study was done in which both network architectures and the co-evolutionary problem decomposition methods were applied to real-world financial time series problems from the NASDAQ stock exchange. The datasets from ACI Worldwide, Staples Inc., and Seagate Technology Holdings were used for comparison. The RNN outperformed the FNN on all three financial time series problems with better generalisation performance.

We included experiments on four real-world problems (Sunspot, ACI Worldwide, Staples Inc. and Seagate Technology Holdings). These datasets seem to include a higher level of noise and uncertainty which made them more difficult for prediction. We gathered that the RNN outperformed the FNN on all four real-world problems. Moreover, NL outperformed SL decomposition on five of the six problems, hence, NL is suited for both simulated and real-world problems.

In our past research [6], we employed training time of 100,000 function evaluations. In this study, we have used 50,000 function evaluations which becomes a significant reduction. We have seen better results in some of the cases in this study as we used improved parameter settings such as bigger population size in the

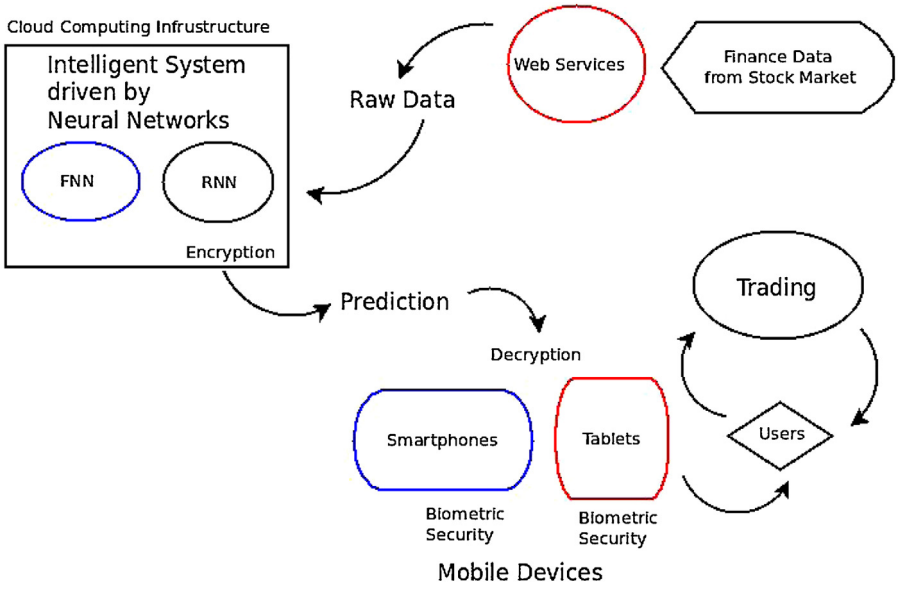


Fig. 10. Performance given by NL (feed-forward neural network) for a typical experimental run on the testing set for Seagate Technology Holdings.

sub-populations. During the training phase, we have observed that training accuracy continues to improve for the entire 50,000 function evaluations. We acknowledge that the extent of improvement decreases towards the end which has been common for similar evolutionary methods [56,6].

We further note that the experiments were designed without the validation set in order to compare the performance of the algorithms and the neural network architectures only on the training data with fixed training time. In future, a validation set can also be used in order to avoid over-fitting.

6. Conclusions and future work

We presented a comparison of feedforward and recurrent neural networks performance for time series prediction where cooperative neuro-evolution was used as the training algorithm. We used two different problem decomposition methods in cooperative neuro-evolution. The results show that the recurrent neural network architecture with neuron level decomposition gave the best results across the different problems. A mobile application framework was also presented that has the potential to efficiently synchronise the prediction system with smartphones taking into account the challenges of over-heating and power consumption. It was highlighted that security issues need to be death with in the implementation of biometric features in the mobile application. This can give potential investors real-time information on market behaviour which is useful in making investments.

In future research, it would be interesting to apply the prediction method to different financial problems that deal with foreign exchange rates, interest rates, and dividend rates. Multi-variate time series can also be used to further improve the prediction. Furthermore, machine learning approaches such as transfer learning could be used to exploit fundamental knowledge in different markets that generate the time series.

References

[1] D. Reid, A. Hussain, H. Tawfik, Spiking neural networks for financial data prediction, in: The 2013 International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1–10, http://dx.doi.org/10.1109/IJCNN.2013.6707140.

[2] R. Ghazali, A.J. Hussain, P. Liatsis, H. Tawfik, The application of ridge polynomial neural network to multi-step ahead financial time series prediction, Neural Comput. Appl. 17 (3) (2008) 311–323, http://dx.doi.org/10.1007/s00521-007-0132-8.

[3] M. Tkác, R. Verner, Artificial neural networks in business: two decades of research, Appl. Soft Comput. 38 (2016) 788–804, http://dx.doi.org/10.1016/j.asoc.2015.09.040 http://www.sciencedirect.com/science/article/pii/S1568494615006122.

[4] T. Koskela, M. Lehtokangas, J. Saarinen, K. Kaski, Time series prediction with multilayer perceptron, FIR and Elman neural networks, in: Proceedings of the World Congress on Neural Networks, San Diego, CA, USA, 1996, pp. 491–496.

[5] M. Ardalani-Farsa, S. Zolfaghari, Chaotic time series prediction with residual analysis method using hybrid Elman-NARX neural networks, Neurocomputing 73 (13–15) (2010) 2540–2553.

[6] R. Chandra, M. Zhang, Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction, Neurocomputing 186 (2012) 116–123, http://dx.doi.org/10.1016/j.neucom.2012.01.014.

[7] D. Mirikitani, N. Nikolaev, Recursive Bayesian recurrent neural networks for time-series modeling, IEEE Trans. Neural Netw. 21 (2) (2010) 262–274.

[8] C.-J. Lin, C.-H. Chen, C.-T. Lin, A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications, IEEE Trans. Syst. Man Cybern. C: Appl. Rev. 39 (1) (2009) 55–68.

[9] R. Chandra, Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction, IEEE Trans. Neural Netw. Learn. Syst. 26 (2015) 3123–3136.

[10] K.K. Teo, L. Wang, Z. Lin, Wavelet packet multi-layer perceptron for chaotic time series prediction: effects of weight initialization, in: Proceedings of the International Conference on Computational Science-Part II, ICCS '01, 2001, pp. 310–317.

[11] A. Gholipour, B.N. Araabi, C. Lucas, Predicting chaotic time series using neural and neurofuzzy models: a comparative study, Neural Process. Lett. 24 (2006) 217–239.

[12] J.-S. Jang, ANFIS adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (3) (1993) 665–685.

[13] M. Potter, K. De Jong, A cooperative coevolutionary approach to function optimization, in: Y. Davidor, H.-P. Schwefel, R. Mönner (Eds.), Parallel Problem Solving from Nature – PPSN III, Vol. 866 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1994, pp. 249–257.

[14] M.A. Potter, K.A. De Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, Evol. Comput. 8 (2000) 1–29.

[15] N. García-Pedrajas, D. Ortiz-Boyer, A cooperative constructive method for neural networks for pattern recognition, Pattern Recogn. 40 (1) (2007) 80–98.

[16] F. Gomez, R. Mikkulainen, Incremental evolution of complex general behavior, Adapt. Behav. 5 (3-4) (1997) 317–342, http://dx.doi.org/10.1177/105971239700500305.

[17] R. Chandra, M. Frean, M. Zhang, On the issue of separability for problem decomposition in cooperative neuro-evolution, Neurocomputing 87 (2012) 33–40, http://dx.doi.org/10.1016/j.neucom.2012.02.005.

[18] S. Chand, R. Chandra, Cooperative coevolution of feed forward neural networks for financial time series problem, in: International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6–11, 2014, 2014, pp. 202–209, http://dx.doi.org/10.1109/IJCNN.2014.6889568.

[19] F. Gomez, J. Schmidhuber, R. Miikkulainen, Accelerated neural evolution through cooperatively coevolved synapses, J. Mach. Learn. Res. 9 (2008) 937–965.

[20] R. Chandra, M. Frean, M. Zhang, C.W. Omlin, Encoding subcomponents in cooperative co-evolutionary recurrent neural networks, Neurocomputing 74 (17) (2011) 223–3234, http://dx.doi.org/10.1016/j.neucom.2011.05.003.

[21] F.J. Gomez, Robust non-linear control through neuroevolution, Technical Report AI-TR-03-303 (PhD Thesis), Department of Computer Science, The University of Texas at Austin, 2003.

[22] E. Lorenz, The Essence of Chaos, University of Washington Press, 1993.

[23] X. Liang, R. Chen, Y. He, Y. Chen, Associating stock prices with web financial information time series based on support vector regression, Neurocomputing 115 (2013) 142–149.

[24] C.L. Giles, C.W. Omlin, K.K. Thornber, Equivalence in knowledge representation: automata, recurrent neural networks, and dynamical fuzzy systems, Proc. IEEE 87 (9) (1999) 1623–1640.

[25] C. Sørensen, A. Al-Taitoon, Organisational usability of mobile computing-volatility and control in mobile foreign exchange trading, Int. J. Hum. Comput. Stud. 66 (2008) 916–929.

[26] Z. Chen, R. Li, X. Chen, H. Xu, A survey study on consumer perception of mobile-commerce applications, Procedia Environ. Sci. 11 (2011) 118–124.

[27] H.-H. Chu, T.-L. Chen, C.-H. Cheng, C.-C. Huang, Fuzzy dual-factor time-series for stock index forecasting, Expert Syst. Appl. 36 (1) (2009) 165–171, http://dx.doi.org/10.1016/j.eswa.2007.09.037.

[28] H. Feng, H. Chou, Evolutional RBFNs prediction systems generation in the applications of financial time series data, Expert Syst. Appl. 38 (2011) 8285–8292.

[29] M.R. Hassan, B. Nath, Stockmarket forecasting using hidden Markov model: a new approach, in: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, ISDA '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 192–196, http://dx.doi.org/10.1109/ISDA.2005.85.

[30] A.J. Hussain, A. Knowles, P. Lisboa, W. El-Deredy, D. Al-Jumeily, Polynomial pipelined neural network and its application to financial time series prediction, in: Proceedings of the 19th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence, AI'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 597–606, http://dx.doi.org/10.1007/11941439_64.

[31] C. Wang, W. Duan, J. Ma, C. Wang, The research of android system architecture and application programming, in: International Conference on Computer Science and Network Technology (ICCSNT) 2, 2011, pp. 785–790.

[32] A. Mutholib, T.S. Gunawan, M. Kartiwi, Design and implementation of automatic number plate recognition on android platform, in: International Conference on Computer and Communication Engineering (ICCCE 2012), 2012, pp. 540–543.

[33] P. Klasnja, W. Pratt, Healthcare in the pocket: Mapping the space of mobile-phone health interventions, J. Biomed. Inf. 45 (2012) 184–198.

[34] C. Liu, Q. Zhu, K. Holroyd, E. Seng, Status and trends of mobile-health applications for ios devices: a developer's perspective, J. Syst. Softw. 84 (2011) 2022–2033.

[35] A. Fernández-López, M. Rodríguez-Fórtiz, M. Rodríguez-Almendros, M. Martínez-Segura, Mobile learning technology based on IOS devices to support students with special education needs, Comput. Educ. 61 (2013) 77–90.

[36] Z. Dan, C. Xu, The recognition of handwritten digits based on BP neural network and the implementation on android, in: Third International Conference on Intelligent System Design and Engineering Applications, 2013, pp. 1498–1501.

[37] M. Mielke, R. Brück, Smartphone application for automatic classification of environmental sound, MIXDES 2013, in: 20th International Conference on Mixed Design of Integrated Circuits and Systems, 2013, pp. 512–515.

[38] P. Liu, Y. Chen, W. Tang, Q. Yue, Mobile weka as data mining tool on android, Adv. Electr. Eng. Autom. 139 (2012) 75–80.

[39] J.L. Elman, Finding structure in time, Cogn. Sci. 14 (1990) 179–211.

[40] R. Chandra, M. Frean, M. Zhang, An encoding scheme for cooperative coevolutionary neural networks, in: 23rd Australian Joint Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, Springer-Verlag, Adelaide, Australia, 2010, pp. 253–262.

[41] K. Deb, A. Anand, D. Joshi, A computationally efficient evolutionary algorithm for real-parameter optimization, Evol. Comput. 10 (4) (2002) 371–395.
[42] F. Takens, Detecting strange attractors in turbulence, in: Dynamical Systems and Turbulence, Warwick 1980, Lecture Notes in Mathematics, 1981, pp. 366–381.
[43] C. Frazier, K. Kockelman, Chaos theory and transportation systems: instructive example, Transp. Res. Rec.: J. Transp. Res. Board 20 (2004) 9–17.
[44] R. Moreno-Vozmediano, R.S. Montero, I.M. Llorente, Key challenges in cloud computing: enabling the future internet of services, IEEE Internet Comput. 17 (4) (2013) 18–25, http://dx.doi.org/10.1109/MIC.2012.69.
[45] D. Lee, J. Kwon, S. Lee, S. Park, B. Hong, Scalable and efficient web services composition based on a relational database, J. Syst. Softw. 84 (12) (2011) 2139–2155, http://dx.doi.org/10.1016/j.jss.2011.05.068 http://www.sciencedirect.com/science/article/pii/S0164121211001440.
[46] S. Mahmud, R. Iqbal, F. Doctor, Cloud enabled data analytics and visualization framework for health-shocks prediction, Future Gener. Comput. Syst. (2015), http://dx.doi.org/10.1016/j.future.2015.10.014 http://www.sciencedirect.com/science/article/pii/S0167739X15003271.
[47] F. Wu, L. Xu, S. Kumari, X. Li, A novel and provably secure biometrics-based three-factor remote authentication scheme for mobile client-server networks, Comput. Electr. Eng. 45 (2015) 274–285.
[48] R. Blanco-Gonzalo, R. Sanchez-Reillo, O. Miguel-Hurtado, E. Bella-Pulgarin, Automatic usability and stress analysis in mobile biometrics, Image Vis. Comput. 32 (12) (2014) 1173–1180.
[49] R.R. Jillela, A. Ross, Segmenting IRIS images in the visible spectrum with applications in mobile biometrics, Pattern Recogn. Lett. 57 (2015) 4–16.
[50] M. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science 197 (4300) (1977) 287–289.
[51] E. Lorenz, Deterministic non-periodic flows, J. Atmos. Sci. 20 (1963) 267–285.
[52] SILSO World Data Center, The International Sunspot Number (1834–2001), International Sunspot Number Monthly Bulletin and Online Catalogue. http://www.sidc.be/silso/, (accessed 02.02.15).
[53] I. Rojas, O. Valenzuela, F. Rojas, A. Guillen, L. Herrera, H. Pomares, L. Marquez, M. Pasadas, Soft-computing techniques and ARMA model for time series prediction, Neurocomputing 71 (4-6) (2008) 519–537.
[54] S.S., Solar cycle forecasting: a nonlinear dynamics approach, Astron. Astrophys. 377 (2001) 312–320.
[55] C. Smith, Y. Jin, Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction, Neurocomputing 143 (2014) 302–311.
[56] R. Chandra, Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction, IEEE Trans. Neural Netw. Learn. Syst. 26 (12) (2015) 3123–3136, http://dx.doi.org/10.1109/TNNLS.2015.2404823.
[57] M. Ardalani-Farsa, S. Zolfaghari, Residual analysis and combination of embedding theorem and artificial intelligence in chaotic time series forecasting, Appl. Artif. Intell. 25 (2011) 45–73.
[58] D. Kim, C. Kim, Forecasting time series with genetic fuzzy predictor ensemble, IEEE Trans. Fuzzy Syst. 5 (4) (1997) 523–535.
[59] I. Rojas, H. Pomares, J.L. Bernier, J. Ortega, B. Pino, F.J. Pelayo, A. Prieto, Time series analysis using normalized PG-RBF network with regression weights, Neurocomputing 42 (1-4) (2002) 267–285.
[60] NASDAQ Exchange Daily: 1970–2010 Open, Close, High, Low and Volume. http://www.nasdaq.com/symbol/aciw/stock-chart, (accessed 02.02.15).