Hybrid Swarm System for Time Series Forecasting

Paulo S. G. de M. Neto[1], Ricardo de A. Araújo[1], Gustavo G. Petry[1], Tiago A. E. Ferreira[1], Germano C. Vasconcelos[1]

[1]Center for Informatics – Federal University of Pernambuco
Av. Prof. Luiz Freire, s/n, CDU, 50732-970, Recife - PE - Brazil

{psgmn,raa,ggp2,taef,gcv}@cin.ufpe.br

Abstract. *In this paper, a hybrid swarm system is presented for time series forecasting. It consists of an intelligent hybrid model composed of an Artificial Neural Network (ANN) and a Particle Swarm Optimizer (PSO), which search the relevant time lags for a correct characterization of the time series, as well as the number of processing units in the hidden layer, the training algorithm and the modeling of ANN. The proposed method shows to be an efficient procedure to training and adjusting the ANN parameters through the use of a particle swarm optimization mechanism. An experimental analysis is conducted with the proposed method using four real world time series and the results are compared to standard MLP networks according to five performance measures.*

1. Introduction

Approaches based on Artificial Neural Networks (ANNs) have been proposed for the nonlinear modeling of time series [Zhang et al. 1998]. However, in order to define a solution to a given problem, an ANN requires the setting up of system configuration parameters, which are not always easy to determine. In addition to those, in the particular case of time series forecasting, another element that demands definition is the relevant time lags to represent the time series. In this context, interesting works have been proposed in literature [Mattos et al. 2005, Ferreira et al. 2006, Araújo et al. 2007].

The Particle Swarm Optimizer (PSO) [vandenBergh and Engelbrecht 2004] is a stochastic optimization technique based on a flock of birds or the sociological behavior of a group of people. The PSO was introduced by Kennedy and Eberhart [Eberhart and Kennedy 1995] and is widely applied to optimization problems due to its high search power in state spaces. It has been used to solve many optimization problems, like ANN training [Eberhart and Hu 1999], [Engelbrecht and Ismail 1999], [van den Bergh and Engelbrecht 2000] and function minimization [Shi and Eberhart 1998], [Shi and Eberhart 1999].

This paper presents a hybrid swarm system for the time series forecasting problem. The proposed method is an intelligent hybrid model composed of a Particle Swarm Optimizer (PSO) [vandenBergh and Engelbrecht 2004] and an ANN. The hybrid swarm system starts by choosing one of three distinct models used to describe the ANN architecture, where each model is trained and adjusted by the PSO, which determines the relevant time lags for a correct characterization of the time series, as well as the number of processing units in the hidden layer, the training algorithm and the model structure ANN. The proposed method is described in section 4. It is shown how this procedure can enhance forecasting performance making use of four real world time series: Brightness of a Variable Star Series, Sunspot, Dow Jones Industrial Average (DJIA) Index and Dollar Real Exchange Rate. The experimental results are compared to Multilayer Perceptron Networks(MLPs) and Random Walk Model according to five performance measures: mean square error (MSE), mean absolute percentage error (MAPE), U of Theil Statistics, prediction of change in direction (POCID), average relative variance (ARV).

2. The Time Series Forecasting Problem

A time series is a set of points, generally time equidistant, defined by,

$$X_t = \{x_t \in \mathbb{R} \mid t = 1, 2, 3 \ldots N\}, \tag{1}$$

where t is the temporal index and N is the number of observations. Therefore X_t is a sequence of temporal observations orderly sequenced and equally spaced.

The main objective when applying forecasting techniques to a time series is to identify certain regular patterns present in the data in order to create a model capable of generating the future patterns. In this context, a crucial factor for a good forecasting performance is the correct choice of the time lags considered for representation of the time series. Such relationship structures among historical data constitute a d-dimensional phase space, where d is the dimension capable of representing the relationship. Takens [Takens 1980] proved that if d is sufficiently large, such built phase space is homeomorphic to the phase space which generated the time series.

A crucial problem in reconstructing the original state space is the correct choice of the variable d, or more specifically, the correct choice of the time lags.

3. Particle Swarm Optimizer Fundamentals

The Particle Swarm Optimizer (PSO) [vandenBergh and Engelbrecht 2004] is an optimization technique based on the social behavior (swarm) that a population of individuals adapts to its environment. In each epoch, each individual of the population of solutions adjusts its position based on its own experience and the experience of neighbors, including the current velocity and position and the best previous position experienced by itself and its neighbors. In this way, if any particle discovers a promising solution, the swarm is guided to the new solution in order to explore more thoroughly the region found.

The swarm size is given by s. Each individual ($1 \leq i \leq s$) has a current position in the search space (x_i), a current velocity (v_i) and a personal best position in the search space (y_i). Assuming that the function f is to be minimized, the swarm consists of n particles, and at each iteration, each swarm particle velocity is updated by

$$\begin{aligned} v_{i,j}(t+1) = w v_{i,j}(t) + c_1 r_1 [y_{i,j}(t) - x_{i,j}(t)] \\ + c_2 r_2 [\hat{y}_j(t) - x_{i,j}(t)] \end{aligned} , \tag{2}$$

where $j \in 1, 2, \ldots, n$, $\hat{y}_j(t)$ denotes the current position in the search space (found by any swarm particle), $y_{i,j}(t)$ represents the personal best position in the search space (found by each swarm particle), $v_{i,j}$ is the velocity of the j-th dimension of the i-th particle, c_1 and c_2 represent the acceleration coefficients, which control how far a particle will move in a single iteration, and $r_1 \sim U(0,1)$ and $r_2 \sim U(0,1)$ are elements from two uniform random sequences in the interval [0,1]. The term w is referred to as inertia weight, this value is typically set to vary linearly from 1 to near 0 during the course of the procedure. It is worth to mention that this is reminiscent from the temperature adjustment schedule found in Simulated Annealing algorithms [vandenBergh and Engelbrecht 2004].

Thus, the new particle position is updated by

$$x_i(t+1) = x_i(t) + v_i(t+1). \tag{3}$$

The personal best particle position and the global best particle found by any particle during all previous iterations are updated by equations (4) and (5), respectively.

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)), \\ x_i(t+1) & \text{otherwise.} \end{cases} ; \tag{4}$$

$$\hat{y}(t+1) = \text{argmin} f(y_i(t+1)). \tag{5}$$

The term v_i is normalized in the range $[-v_{max}, v_{max}]$ in order to reduce the likelihood of particles leaving the search space. It is worth to mention that this mechanism doesn't restrict the values of x_i in the range of v_i, it only limits the maximum distance that a particle will move during each iteration [vandenBergh and Engelbrecht 2004].

4. The Proposed Method

The method proposed in this paper uses a Particle Swarm Optimizer (PSO) [vandenBergh and Engelbrecht 2004] search mechanism for time series forecasting. It is based on the definition of the three main elements necessary for building an accurate forecasting system: 1. The minimum number of time lags adequate for representing the series, 2. The structure of the model capable of representing such underlying information for the purpose of prediction, and 3. The ANN training algorithm.

The proposed hybrid swarm system consists of an intelligent hybrid model composed of an ANN and a PSO [vandenBergh and Engelbrecht 2004]. The procedure consists of: three distinct models used to describe the ANN (multilayer perceptron – MLP) architecture, where each model is trained and adjusted by the PSO, which determines: 1. The minimum number of time lags for a correct characterization of the time series (initially, a maximum number ($MaxLags$) is previously defined and then the PSO can choose any value in the interval $[1, MaxLags]$ for each individual of the population), 2. The maximum number of processing units in the ANN hidden layer ($NHiddenMax$) (set up initially and then the PSO can choose any value in the interval $[1, NHiddenMax]$ for each individual of the population), 3. The architecture, network parameters configuration and the initial weights of the ANN, and 4. The training algorithm ($ANNTrain$) for the ANN (Levenberg-Marquardt [Hagan and Menhaj 1994], RPROP [Reidmiller and Braun 1993], scaled conjugate gradient [Moller 1993] or one step secant [Battiti 1992]).

These processes offer the system the effective capacity to seek the most compact ANN, and thus reduce the computational cost and the probability of the overfitting problem. Each swarm element represents an ANN (three-layer), where the first layer is defined by the number of time lags, the second layer is composed by the number of hidden processing units (sigmoidal units) and the third layer is composed by one processing unit (prediction horizon of one step ahead).

The PSO individuals are evaluated by the fitness function defined by,

$$fitness = \frac{POCID}{1 + MSE + MAPE + NMSE + ARV} \tag{6}$$

where MSE, $MAPE$, $NMSE$, $POCID$ and ARV are the mean square error, the mean absolute percentage error, the normalized mean square error (or U of Theil Statistics), the prediction of change in direction and the average relative variance used for ANN performance evaluation, respectively, and will be formally defined in section 5.

The termination conditions for the PSO are,

1. The number of PSO iterations ($MaxGer$);
2. The increase in the validation error or generalization loss (Gl) [Prechelt 1994]: $Gl > 5\%$;
3. The decrease in the training error or process training (Pt) [Prechelt 1994]: $Pt \leq 10^{-6}$.

4.1. PSO Individuals Modeling

Each individual of the PSO population is an ANN (three-layer MLP). These individuals are represented by particles that have the following parameters (ANN parameters):

- W_{ij}: weights of connections between the input layer and the hidden layer
- W_{jk}: weights of connections between the hidden layer and the output layer;
- b_j^1: bias of the hidden layer;
- b_k^2: bias of the output layer;
- $NetMod$: ANN model.
- $NHidden$: the number of processing units in the ANN hidden layer;
- $NLags$: the number of relevant time lags;
- $ANNTrain$: the ANN training algorithm.

Three distinct forms of modeling the ANN are proposed ($NetMod = 1, 2, 3$), where each is described in the following subsections.

4.1.1. First ANN model

The first architecture for modeling ANNs ($NetMod = 1$) uses the sigmoidal activation function for all hidden processing units. The output processing unit uses a linear activation function where a sigmoidal function is applied to its bias. This architecture was used by Leung et al [Leung et al. 2003], where the output of ANN is given by

$$y_k(t) = \sum_{j=1}^{n_h} W_{jk} Sig \left[\sum_{i=1}^{n_{in}} W_{ij} Z_i(t) + b_j^1 \right] + Sig(b_k^2), \tag{7}$$

where $Z_i(t)$ $(i = 1, 2, \ldots, n_{in})$ are the ANN input values, n_{in} denotes the number of ANN input and n_h is the number of hidden units. Since the prediction horizon is one step ahead, only one output unit is necessary ($k = 1$). The term Sig is a sigmoidal function defined by:

$$Sig(x) = \frac{1}{1 + exp(-x)}. \tag{8}$$

4.1.2. Second ANN model

The second model ($NetMod = 2$) consists of hidden units activated by a sigmoidal function with its output layer using a linear function. The output of ANN is given by:

$$y_k(t) = \sum_{j=1}^{n_h} W_{jk} Sig \left[\sum_{i=1}^{n_{in}} W_{ij} Z_i(t) + b_j^1 \right] + b_k^2. \tag{9}$$

4.1.3. Third ANN model

The third architecture ($NetMod = 3$) applies the sigmoidal activation function to all processing units. The output of ANN is given by:

$$y_k(t) = Sig \left\{ \sum_{j=1}^{n_h} W_{jk} Sig \left[\sum_{i=1}^{n_{in}} W_{ij} Z_i(t) + b_j^1 \right] + b_k^2 \right\}. \tag{10}$$

5. Evaluation Measures

Most of the works found in the literature of time series prediction frequently employ only one performance criterion for model evaluation. The measure used is usually the MSE (Mean Squared Error),

$$MSE = \frac{1}{N} \sum_{j=1}^{N} (target_j - output_j)^2, \tag{11}$$

where N is the number of patterns, $target_j$ is the desired output for pattern j and $output_j$ is the predicted value for pattern j.

Although the MSE measure may be used to guide the prediction model in the training process, it cannot be considered alone as a conclusive measure for comparison of different prediction models [Clements and Hendry 1993]. For this reason, other performance criteria should be considered for allowing a more robust performance assertiveness.

A second relevant measure is the MAPE (Mean Absolute Percentage Error), given by

$$MAPE = \frac{100}{N} \sum_{j=1}^{N} \left| \frac{target_j - output_j}{X_j} \right|, \tag{12}$$

where X_j is the time series at point j.

A third performance measure is the U of Theil Statistics, which is given by

$$UofTheil = \frac{\sum_{j=1}^{N} (target_j - output_j)^2}{\sum_{j=1}^{N} (output_j - output_{j+1})^2}, \tag{13}$$

which associates the model performance with a random walk model [Mills 2003]. If the U of Theil is equal to 1, the predictor has the same performance of a random walk model [Mills 2003]. If the U of Theil is greater than 1, then the predictor has a worse performance than a random walk model [Mills 2003], and if the U of Theil is less than 1, the predictor is better than a random walk model [Mills 2003]. In the perfect model, the U of Theil tends to zero.

Another relevant evaluation measure considers the calculation of the correctness of Prediction of Change in Direction, or POCID for short,

$$POCID = 100 \frac{\sum_{j=1}^{N} D_j}{N}, \tag{14}$$

where

$$D_j \begin{cases} 1 & \text{if } (target_j - target_{j-1})(output_j - output_{j-1}) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

There is also another relevant evaluation measure, the ARV (Average Relative Variance), which is given by

$$ARV = \frac{\sum_{j=1}^{N} (output_j - target_j)^2}{\sum_{j=1}^{N} (output_j - \overline{target})^2}, \tag{16}$$

which associates the model performance with the mean of the time series. Term $\overline{target}$ is the mean of the time series. If the ARV is equal to 1, the predictor has the same performance of the mean of the time series. If the ARV is greater than 1, then the predictor has a worse performance than the mean of the time series, and if the ARV is less than 1, the predictor is better than a mean of the time series. In the ideal model, ARV tends to zero.

6. Simulations and Results

Six real world time series were used for evaluation of the proposed method, two natural phenomena time series (Star and Sunspot) and four financial time series (Dow Jones Industrial Average (DJIA) Index, National Association of Securities Dealers Automated Quotation (NASDAQ) Index Series, Petrobras Stock Values and Dollar Real Exchange Rate). All series investigated were normalized to lie within the interval $[0, 1]$ and were divided in to three sets [Prechelt 1994]: training set with 50% of the series data, validation set with 25% of the series data and test set with 25% of the series data. The PSO parameters are the same for all experiments. The number of PSO iterations is 1000. The acceleration coefficients ($c1$ and $c2$) are set to 2.05, the inertia weight (w) is set to 0.9, the v_{max} is set to 2.048 and the terms r_1 and r_2 are random numbers in the range $[0, 1]$. The PSO population is composed of 10 particles, where each individual is an ANN with the maximum architecture: a 10-10-1 multilayer perceptron (MLP) network, which denotes 10 units in the input layer, 10 units in the hidden layer and 1 unit in the output layer (prediction horizon of one step ahead).

It is worth to mention that each PSO individual is trained by one of the four algorithms below: levenberg-marquardt (LM) [Hagan and Menhaj 1994] or RPROP [Reidmiller and Braun 1993], scaled conjugate gradient [Moller 1993], or one step secant [Battiti 1992], for a period of 10^3 epochs. The termination conditions for the ANN training are the maximum number of epochs, the increase in the validation error or generalization loss (Gl) beyond 5% and the decrease in the training error process training (Pt) under 10^{-6}.

For each time series, five experiments were repeated and the experiment with the largest validation fitness function is chosen to represent the model. The next subsections show the experimental results achieved with the proposed model for the four series. For the sake of comparison results are also shown with a random walk (RW) model [Mills 2003] and with standard MLPs (multilayer perceptron) in exactly the same conditions. The MLP was trained with the Levenberg-Marquardt (LM) [Hagan and Menhaj 1994] algorithm with architecture $X - Y - 1$, which denotes X units in the input layer, Y units in the hidden layer and 1 unit in the output layer (prediction horizon of one step ahead). The term X represents the relevant time lags, and for each time series the same time lags chosen by the proposed model are used. The term Y represents the number of units in the hidden layer, and for each time series the same number chosen by the proposed model is also used. The termination conditions for the MLP training with LM [Hagan and Menhaj 1994] are the maximum number of epochs (10^4), the increase in the validation error or generalization loss ($Gl > 5\%$) and the decrease in the error of the process training ($Pt < 10^{-6}$).

6.1. Star Series

The Star series corresponds to daily observations in the same place and hour of an oscillating shine star, constituting a database of 600 points.

For the prediction of the Star series (with 1 step ahead of prediction horizon), the proposed method automatically chose the lags 1, 2, 3, 4, 5, 6 and 8 as the relevant lags for the time series representation, defined the second ANN model (Equation 9) with architecture 7-7-1, in which $NetMod = 2$, $NLags = 7$ e $NHidden = 7$, and chose the Levenberg-Marquard algorithm for ANN training. Table 1 shows the results for all performance measures.

Table 1. Results for the Star series.

	RW Model	MLP Model	Proposed Model
MSE	$3.7191 \cdot 10^{-3}$	$6.4973 \cdot 10^{-4}$	$2.1960 \cdot 10^{-4}$
MAPE	16.14%	6.83%	3.85%
NMSE	1.0000	0.1746	0.0573
POCID	65.98%	73.46%	76.19%
ARV	$5.4643 \cdot 10^{-2}$	$9.5463 \cdot 10^{-3}$	$3.2266 \cdot 10^{-3}$
fitness function	3.6257	9.1655	15.5149

According to Table 1, the proposed model prediction obtained superior performance (in terms of fitness function) than the RW model and MLP model. The obtained NMSE measure (smaller than 1) indicates that the proposed model has a better behavior than a random walk model [Mills 2003] and the POCID measure (greater than 50%) shows that the proposed model performs better than a coin tossing experiment. Figure 1(a) shows the actual Star values (solid line) and the predicted values generated by the proposed model (dashed line) for the last 100 points of the test set.

6.2. Sunspot Series

The selected Sunspot series consisted of the total annual measures of the sun spots from the years of 1700 to 1988, constituting a database of 289 points.

For the prediction of the Sunspot series (with 1 step ahead of prediction horizon), the proposed method automatically chose the lags 1, 2, 3 and 4 as the relevant lags for the

time series representation, defined the second ANN model (Equation 9) with architecture 4-10-1, in which $NetMod = 2$, $NLags = 4$ e $NHidden = 10$, and chose the Levenberg-Marquard algorithm for ANN training. Table 2 shows the results for all performance measures.

Table 2. Results for the Sunspot series.

	RW Model	MLP Model	Proposed Model
MSE	$2.7003 \cdot 10^{-2}$	$1.1634 \cdot 10^{-2}$	$9.5536 \cdot 10^{-3}$
MAPE	55.27%	34.28%	31.95%
NMSE	1.0000	0.3980	0.3237
POCID	76.81%	81.15%	91.30%
ARV	0.4050	0.1744	0.1432
fitness function	1.3325	2.2625	2.7310

According to Table 2, the proposed model prediction obtained much superior performance (in terms of fitness function) than RW model and MLP model. The obtained NMSE measure (smaller than 1) indicates that the proposed model has a better behavior than a random walk model [Mills 2003] and the POCID measure (greater than 50%) shows that the proposed model performs better than a coin tossing experiment. Figure 1(b) shows the actual Sunspot values (solid line) and the predicted values generated by the proposed model (dashed line) for the last 70 points of the test set.

6.3. Dow Jones Industrial Average Index Series

The Dow Jones Industrial Average (DJIA) Index series corresponds to daily records from January 1st 1998 to August 26th 2003, constituting a database of 1,420 points.

For the prediction of the DJIA Index series (with 1 step ahead of prediction horizon), the proposed method automatically chose the lags 1, 2, 4, 5, 7 and 8 as the relevant lags for the time series representation, defined the second ANN model (Equation 9) with architecture 6-10-1, in which $NetMod = 2$, $NLags = 6$ e $NHidden = 10$, and chose the Levenberg-Marquard algorithm for ANN training. Table 3 shows the results for all performance measures.

Table 3. Results for the DJIA Index series.

	RW Model	MLP Model	Proposed Model
MSE	$8.3911 \cdot 10^{-4}$	$8.4089 \cdot 10^{-4}$	$8.3853 \cdot 10^{-4}$
MAPE	9.68%	9.85%	9.59%
NMSE	1.0000	1.0023	1.0072
POCID	46.30%	50.72%	51.44%
ARV	$3.5051 \cdot 10^{-2}$	$3.5126 \cdot 10^{-2}$	$3.3447 \cdot 10^{-2}$
fitness function	3.9509	4.2663	4.4224

According to Table 3, the proposed model prediction obtained superior performance (in terms of fitness function) than RW model and MLP model. The obtained NMSE measure (around 1) indicates that the proposed model has a similar behavior of a random walk model [Mills 2003] and the POCID measure (around 50%) shows that the proposed model performs similar than a coin tossing experiment. Prediction for the DJIA Index series is dislocated one step ahead the original values, characterizing the results obtained by the NMSE. Figure 1(c) shows the actual DJIA Index values (solid line) and the predicted values generated by the proposed model (dashed line) for the last 100 points of the test set.

6.4. Dollar Real Exchange Rate Series

The Dollar Real Exchange Rate series corresponds to daily observations from January 3rd 2000 to December 29th 2005, constituting a database of 1,508 points.

For the prediction of the Dollar Real Exchange Rate series (with 1 step ahead of prediction horizon), the proposed method automatically chose the lags 1, 2, 9 and 10 as the

relevant lags for the time series representation, defined the second ANN model (Equation 9) with architecture 4-1-1, in which $NetMod = 2$, $NLags = 4$ e $NHidden = 1$, and chose the Levenberg-Marquard algorithm for ANN training. Table 4 shows the results for all performance measures.

Table 4. Results for the Dollar Real Exchange Rate series.

	RW Model	MLP Model	Proposed Model
MSE	$6.1495 \cdot 10^{-5}$	$6.8091 \cdot 10^{-5}$	$6.0887 \cdot 10^{-5}$
MAPE	1.81%	1.82%	1.75%
NMSE	1.0000	1.1070	0.9918
POCID	56.14%	59.89%	59.97%
ARV	$4.6040 \cdot 10^{-3}$	$5.0979 \cdot 10^{-3}$	$4.4853 \cdot 10^{-3}$
fitness function	14.7112	15.2253	16.0075

According to Table 4, the proposed model prediction obtained superior performance (in terms of fitness function) than RW model and MLP model. The obtained NMSE measure (around 1) indicates that the proposed model has a similar behavior of a random walk model [Mills 2003] and the POCID measure (around 50%) shows that the proposed model performs similar than a coin tossing experiment. Prediction for the Dollar Real Exchange Rate series is dislocated one step ahead the original values, characterizing the results obtained by the NMSE. Figure 1(d) shows the actual Dollar Real Exchange Rate values (solid line) and the predicted values generated by the proposed model (dashed line) for the last 100 points of the test set.

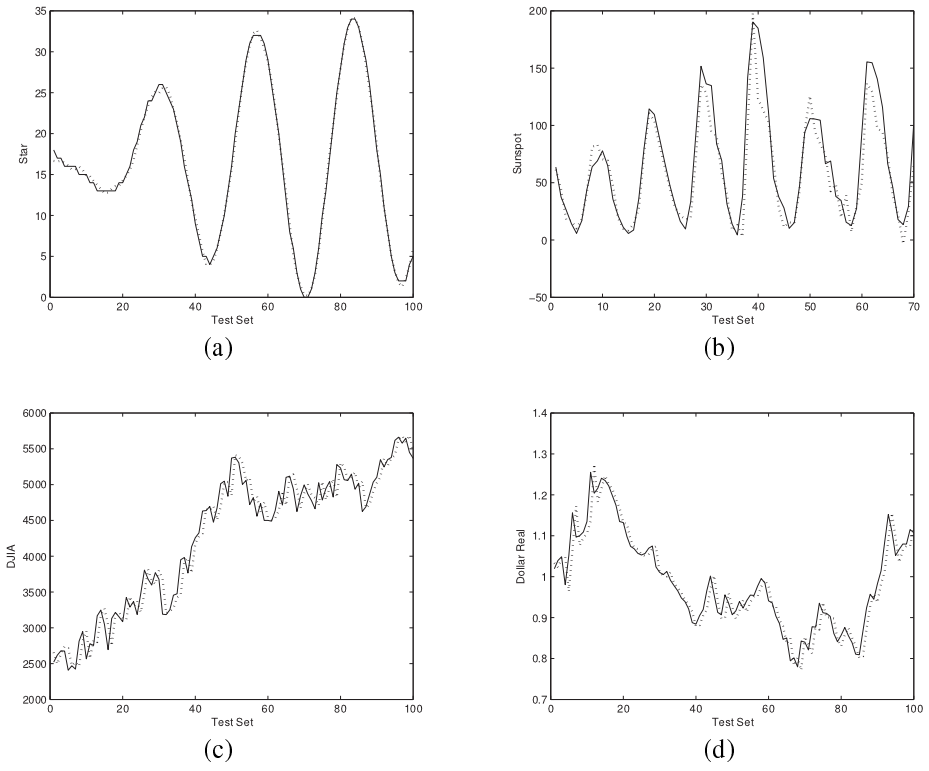

Figure 1. Prediction results for the analyzed time series (test set): actual values (solid line) and predicted values (dashed line).

7. Conclusion

In this paper has presented a hybrid swarm system for time series forecasting, which consists of an intelligent hybrid model composed of an Artificial Neural Network (ANN)

and a Particle Swarm Optimizer (PSO). This method searches for the minimum number of time lags for a correct characterization of the time series and the best neural network structure in terms of the number of hidden processing units, the model, the parameters configuration, the initial weights and the training algorithm of the ANN.

The experimental results using a set of consistent performance measures with five different metrics, the mean square error (MSE), the mean absolute percentage error (MAPE), U of Theil, the prediction of change in direction (POCID) and the average relative variance (ARV). The method was applied to tow naturals phenomena time series, Sunspot and Star, two real world time series from the financial market with all their dependence on exogenous and uncontrollable variables (Dow Jones Industrial Average Index and Dollar Real Exchange Rate).

It was observed that the proposed model obtained a result better than random walk model [Mills 2003] and ANN model for all the analyzed natural phenomena series. For the analyzed financial time series, the proposed model obtained a slightly superior behavior than random walk and MLP model.

Prediction for all analyzed financial series is dislocated one step ahead the original values with all used models.This observation is also in consonance with the work of Sitte and Sitte [Sitte and Sitte 2002], which have shown that the predictions of financial time series exhibit a characteristic of one step shift with respect to the original data.

The experimental results have shown that the proposed method is a valid option for the time series forecasting, therefore automatic adjusting the parameters of the ANN, finding the optimum or sub-optimum values for this parameters obtaining good prediction results with an acceptable computational cost. Other time series are being reaped for the efficiency confirmation of the proposed method. Future works will consider the phase prediction adjust procedure, like the method proposed by Ferreira et al. [Ferreira et al. 2006].

Prediction for all analyzed financial series is dislocated one step ahead the original values. This observation is also in consonance with the work of Sitte and Sitte [Sitte and Sitte 2002], which have shown that the predictions of financial time series exhibit a characteristic of one step shift with respect to the original data. They argued that the financial time series were a random walk model. Thus, in this approximation of the financial time series model is given by

$$x_t = x_{t-1} + r_t \tag{17}$$

where r_t is a gaussian noise term. In this way, the prediction of the time series x_t, given by $\widehat{x}_t$, requires a minimum forecast error, or,

$$E[\widehat{x}_t - x_t] \rightarrow 0 \tag{18}$$

Solving the Equation 18, it obtains,

$$E[\widehat{x}_t] \rightarrow E[x_{t-1}] \tag{19}$$

justifying the prediction of one step shift with respect to the original data for financial times series. Thus, the result expected is given by Equation 19. However, Ferreira et al. [Ferreira et al. 2006] showed that this behavior, which is like a random walk model for financial time series, can be corrected by a phase prediction adjustment.

References

Araújo, R. A., Sousa, R. P., and Ferreira, T. A. E. (2007). An intelligent hybrid approach for designing increasing translation invariant morphological operators for time series forecasting. In *ISNN (2)*, volume 4492 PART II of *Lecture Notes in Computer Science*, pages 602–611. Springer-Verlag.

Battiti, R. (1992). One step secant conjugate gradient. *Neural Computation*, 4:141–166.

Clements, M. P. and Hendry, D. F. (1993). On the limitations of comparing mean square forecast errors. *Journal of Forecasting*, 12(8):617–637.

Eberhart, R. C. and Hu, X. (1999). Human tremor analysis using particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Washington, USA.

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Int. Symp. Micro Machine and Human Science*, Nagoya, Japan.

Engelbrecht, A. P. and Ismail, A. (1999). Training product unit neural networks. In *Stability Control: Theory Appl.*, volume 2, pages 59–74.

Ferreira, T. A. E., Vasconcelos, G. C., and Adeodato, P. J. L. (2006). A new intelligent methodology for times series ferecasting. *New Mathematics and Natural Computation*, 2(3).

Hagan, M. and Menhaj, M. (1994). Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993.

Leung, F. H. F., Lam, H. K., Ling, S. H., and Tam, P. K. S. (2003). Tuning of the structure and parameters of the neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, 14(1):79–88.

Mattos, P. S. G., Petry, G. G., and Ferreira, T. A. E. (2005). Combinação de redes neurais artificiais com algoritmo genético modificado para a previsão de séries temporais. In *V Encontro Nacional de Inteligência Artificial (ENIA 2005)*. SBC.

Mills, T. C. (2003). *The Econometric Modelling of Financial Time Series*. Cambridge University Press, Cambridge.

Moller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533.

Prechelt, L. (1994). Proben1: A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94.

Reidmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proceedings of the IEEE Int. Conf. on Neural Networks (ICNN)*, pages 586–591, San Francisco.

Shi, Y. and Eberhart, R. C. (1998). A modified particle swarm optimizer. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Anchorage, AK.

Shi, Y. and Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Washington, USA.

Sitte, R. and Sitte, J. (2002). Neural networks approach to the random walk dilemma of financial time series. *Applied Intelligence*, 16(3):163–171.

Takens, F. (1980). Detecting strange attractor in turbulence. In Dold, A. and Eckmann, B., editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381, New York. Springer-Verlag.

van den Bergh, F. and Engelbrecht, A. P. (2000). Cooperative learning in neural networks using particle swarm optimizers. In *South African Comput. J.*, volume 26, pages 84–90.

vandenBergh, F. and Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE Trans. Evolutionary Computation*, 8(3):225–239.

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14:35–62.