Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2010, Article ID 373648, 20 pages
doi:10.1155/2010/373648

Research Article

Stacked Heterogeneous Neural Networks for Time Series Forecasting

Florin Leon and Mihai Horia Zaharia

Faculty of Automatic Control and Computer Engineering, Technical University "Gheorghe Asachi" of Iaşi, Boulevard Mangeron 53A, 700050 Iaşi, Romania

Correspondence should be addressed to Florin Leon, florinleon@gmail.com

Received 31 January 2010; Accepted 21 February 2010

Academic Editor: Cristian Toma

A hybrid model for time series forecasting is proposed. It is a stacked neural network, containing one normal multilayer perceptron with bipolar sigmoid activation functions, and the other with an exponential activation function in the output layer. As shown by the case studies, the proposed stacked hybrid neural model performs well on a variety of benchmark time series. The combination of weights of the two stack components that leads to optimal performance is also studied.

1. Introduction

Many processes found in the real world are nonlinear. Therefore, there is a need for accurate, effective tools to forecast their behavior. Current solutions include general methods such as multiple linear regression, nonlinear regression, artificial neural networks, but also specialized ones, such as *SETAR*, Self-Exciting Threshold Auto-Regression [1], *MES*, multivariate exponential smoothing [2], and *FCAR*, Functional Coefficient Auto-Regressive models [3]. The *DAN2* model [4] is a dynamic architecture for artificial neural networks (ANNs) for solving nonlinear forecasting and pattern recognition problems, based on the principle of learning and accumulating knowledge at each layer and propagating and adjusting this knowledge forward to the next layer.

Recently, more hybrid forecasting models have been developed, integrating neural network techniques with conventional models to improve their accuracy. A well-known example is *ARIMA*, the Auto-Regression Integrated Moving Average [5]. *ARFIMA*, Auto-Regressive Fractionally Integrated Moving Average, is a time series model that generalizes ARIMA by allowing nonlinear values in modeling events with long memory [6]. The *SARIMABP* model [7] combines *SARIMA*, Seasonal ARIMA, and the back-propagation

training algorithm to predict seasonal time series for machinery production and soft drink time series. Another hybrid model, *KARIMA* [8], combines Kohonen's self-organizing map and ARIMA to predict short-term traffic flow.

Other techniques include a combination of the radial basis functions, *RBF*, neural networks, and the Univariant Box-Jenkins, *UBJ*, model [9]. Another model that combines radial basis functions neural networks with a nonlinear time-varying evolution particle swarm optimization, *PSO*, is developed by other authors [10]. It displays dynamically adaptive optimization for the inertia and acceleration coefficients, accelerating convergence, and shows good performance on the time series prediction of a power system.

The study of nonlinear phenomena is especially important in the field of system dynamics. Research in this area proved that nonlinear differential equations are capable of generating continuous mathematical functions similar to pulse sequences [11]. Also, an extension to the Fourier/Laplace transform needed for the analysis of signals represented by traveling wave equations was proposed [12], along with a mathematical technique for the simulation of the behavior of large systems of optical oscillators.

Artificial neural networks are one of the most accurate and widely used forecasting models, with applications in social, economical, engineering, foreign exchange, stock problems, and so forth. Neural network have some characteristics that make them particularly valuable for forecasting [13].

(i) Unlike traditional model-based methods, NNs are data-driven and self-adaptive, needing little a priori information about the studied problems.

(ii) NNs are known to have good generalization capabilities. After learning the data presented to them, they can correctly predict unseen data, even in the presence of noise.

(iii) NNs are universal approximators [14]; that is, they can approximate any continuous real function to any degree of accuracy.

(iv) NNs are nonlinear models, and therefore better suited to capture the true nature of many natural processes.

2. The Proposed Neural Network Architecture

The proposed model is composed of two neural networks, each with one hidden layer, as shown in Figure 1.

The inputs of the model are the recent values of the time series, depending on the size of the sliding window s. Basically, the stack model predicts the value of the time series at moment t depending on the latest s values:

$$y_t = f\left(y_{t-1}, y_{t-1}, \ldots, y_{t-s}\right). \tag{2.1}$$

The neurons in the hidden layers of both networks that compose the stack are normal multiplayer perceptron (MLP) neurons, with bipolar sigmoid, or hyperbolic tangent, activation functions:

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}. \tag{2.2}$$

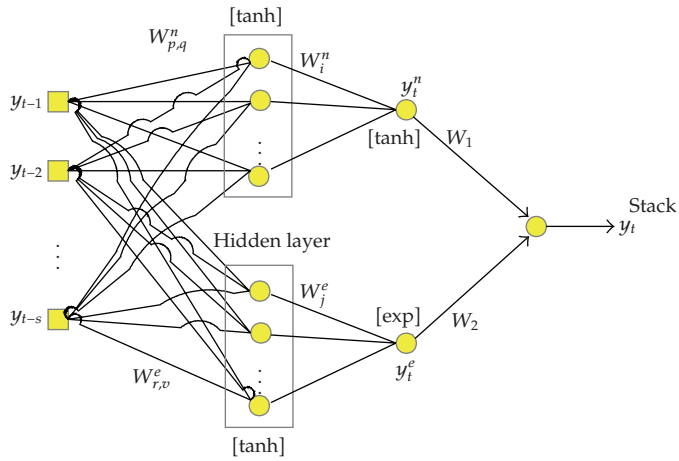

Figure 1: The architecture of the stacked neural network.

The difference between the two neural networks lies in the output layer of the individual neural networks that compose the stack. The "normal" network, represented at the top in Figure 1, has the same activation function, the bipolar sigmoid in the output layer. The "exponential" network has a simple exponential function instead:

$$\exp(x) = e^x. \tag{2.3}$$

Each network respects the basic information flow of an MLP, where the θ values represent the thresholds of the neurons:

$$y^n_t = \tanh\left(\sum_i w^n_i \cdot \left(\tanh\left(\sum_p w^n_{p,q} \cdot y_p - \theta^n_q\right)\right) - \theta^n\right), \tag{2.4}$$

$$y^e_t = \exp\left(\sum_j w^e_j \cdot \left(\tanh\left(\sum_r w^e_{r,v} \cdot y_r - \theta^e_v\right)\right) - \theta^e\right). \tag{2.5}$$

Finally, the stack output y^{stack}_t is computed as a weighted average of the two outputs, y^n_t and y^e_t:

$$y^{\text{stack}}_t = w_1 \cdot y^n_t + w_2 \cdot y^e_t, \tag{2.6}$$

with $w_{1,2} \in [0,1]$ and $w_1 + w_2 = 1$.

The training of the networks is performed with the classical back-propagation algorithm [15], with a momentum term [16] and adaptive learning rate [17]. Besides training the two components of the stack, the goal of the model is to find the optimal weights w_1 and w_2, such that the error of the whole stack is minimized.

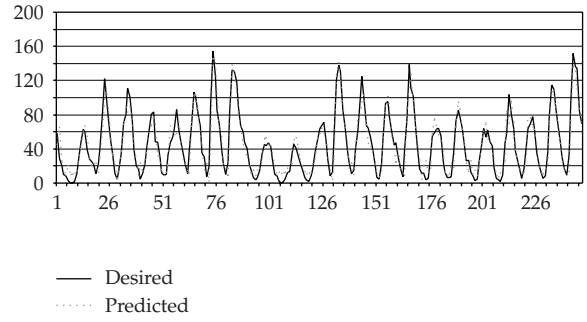

Figure 2: The proposed model performance on the sunspot training data (window size 5).

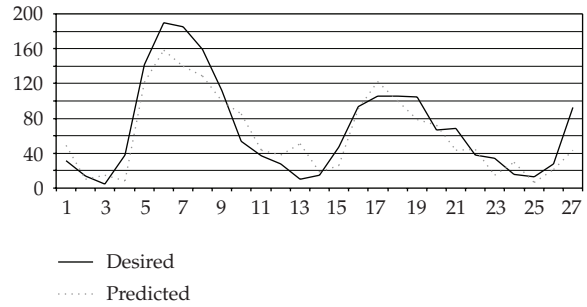

Figure 3: The proposed model predictions for the sunspot data (window size 5).

3. Case Studies

3.1. Test Methodology

In the following sections, we consider four classical benchmark problems and one original, super-exponential growth problem on which we test the performance of our model. In each case, we divide the available data into 90% for training and 10% for testing. We separately consider a sliding window size of 5 and 10, respectively.

3.2. The Sunspot Series

Wolfer's sunspot time series records the yearly number of spots visible on the surface of the sun. It contains the data from 1700 to 1987, for a total of 288 observations. This data series is considered as nonlinear and non-Gaussian and is often used to evaluate the effectiveness of nonlinear models [13, 18].

With a window size of 5 points and considering 28 points ahead, the performance of the model on the training set is displayed in Figure 2.

The forecasting capabilities of the model are displayed in Figure 3.

In Figure 4, the effect of the weights on the mean square error of the stack on the testing data is displayed. One can see that the optimal weights are $w_1 = 100$ and $w_2 = 0$, where w_1 is the weight of the neural network with sigmoid activation functions, and $w_2 = 1 - w_1$ is the weight of the second neural network, whose output neuron has an exponential activation function.

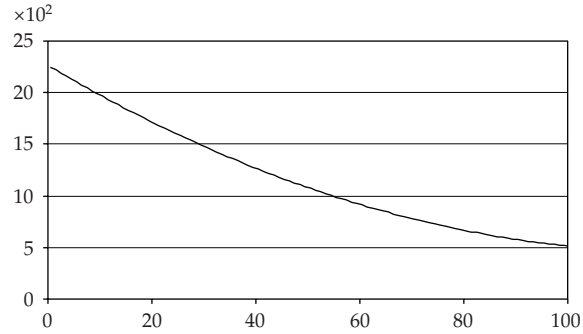

Figure 4: The evolution of MSE when w_1 varies (sunspots, window size 5).

Table 1: The errors of the model for the sunspot data (window size 5).

		MSE on original data	MSE on normalized data
Training	Normal NN	131.865	3.645×10^{-3}
	Exponential NN	827.933	22.886×10^{-3}
	Stack	131.865	3.645×10^{-3}
Testing	Normal NN	511.531	14.140×10^{-3}
	Exponential NN	2248.409	62.151×10^{-3}
	Stack	511.531	14.140×10^{-3}

Table 2: The errors of the model for the sunspot data (window size 10).

		MSE on original data	MSE on normalized data
Training	Normal NN	96.085	2.656×10^{-3}
	Exponential NN	811.737	22.438×10^{-3}
	Stack	96.085	2.656×10^{-3}
Testing	Normal NN	619.387	17.121×10^{-3}
	Exponential NN	2466.148	68.170×10^{-3}
	Stack	619.387	17.121×10^{-3}

Table 1 shows the errors both for the training and for testing. It separately presents the mean square errors for the first, "normal" network, for the second, "exponential" network, and for their stack, respectively. Since the range of the datasets is very different, we also display the MSE on the normalized data between 0 and 1, in order to better compare the performance of our model on data with different shapes.

Next, we increase the size of the sliding window to 10 points. The corresponding performance of the model on the training set is displayed in Figure 5.

The prediction capabilities of the model are displayed in Figure 6.

The evolution of the mean square error of the stack on the testing data is displayed as a function of the normal NN weight, w_1, in Figure 7. In this case, as in the previous one, the optimal weights are $w_1 = 100$ and $w_2 = 0$.

Table 2 shows the mean square errors obtained for the increased window size.

In both cases, we see that the normal neural network can approximate the time series better than the exponential network. When the window size is 10, the model seems to slightly

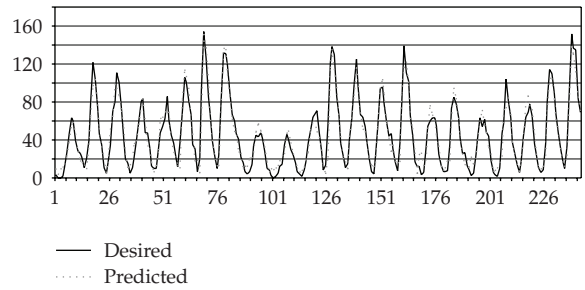

Figure 5: The proposed model performance on the sunspot training data (window size 10).

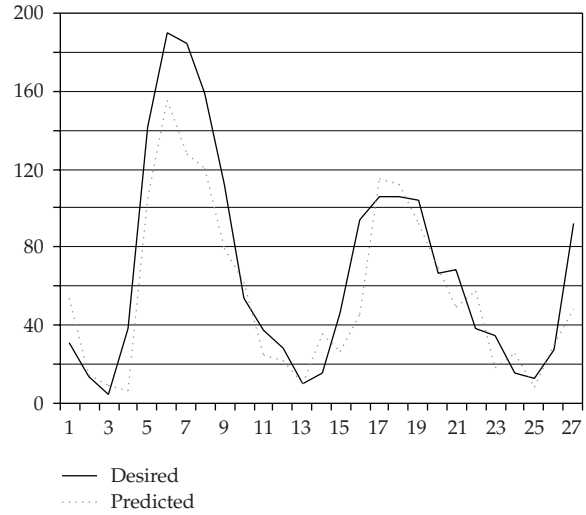

Figure 6: The proposed model predictions for the sunspot data (window size 10).

overfit the data compared to the case when the window size is 5, yielding better errors for the training set, but a little worse errors for the testing set.

3.3. The Canadian Lynx Series

This series contains the yearly number of lynx trapped in the Mackenzie River district of Northern Canada [19]. The data set has 114 observations, corresponding to the period of 1821–1934.

With a window size of 5 points and considering 11 points ahead, the performance of the model on the training set is displayed in Figure 8.

The forecasting capabilities of the model are displayed in Figure 9.

The evolution of the mean square error of the stack on the testing data is displayed as a function of the normal NN weight, w_1, in Figure 10. Here, the optimal weights are $w_1 = 23$ and $w_2 = 77$.

Table 3 shows the mean square errors obtained for a window size of 5.

When size of the window is increased to 10 points, the performance of the model on the training set is that shown in Figure 11.

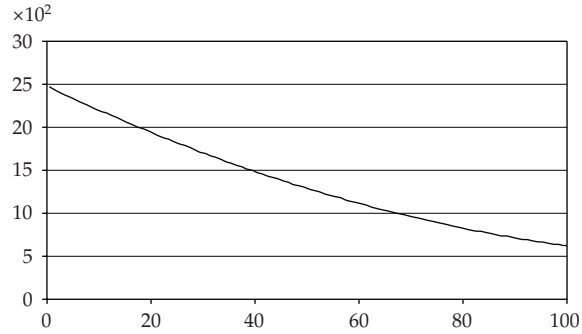

Figure 7: The evolution of MSE when w_1 varies (sunspots, window size 10).

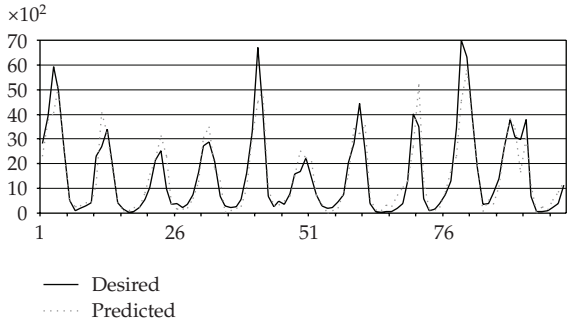

Figure 8: The proposed model performance on the lynx training data (window size 5).

Table 3: The errors of the model for the lynx data (window size 5).

		MSE on original data	MSE on normalized data
Training	Normal NN	530,416.473	10.974×10^{-3}
	Exponential NN	383,963.211	7.944×10^{-3}
	Stack	401,015.900	8.297×10^{-3}
Testing	Normal NN	154,951.311	3.206×10^{-3}
	Exponential NN	113,955.783	2.357×10^{-3}
	Stack	109,902.034	2.273×10^{-3}

The testing performance of the model is displayed in Figure 12.

The optimal weights for this stack are $w_1 = 99$ and $w_2 = 1$, as can be observed from Figure 13.

Table 4 shows the mean square errors obtained for a window size of 10.

For this dataset, the exponential network can contribute to the stack result. When the window size is 5, its weight even dominates the stack, and its contribution decreases for a larger window size. It is possible that this phenomenon appears because for a smaller window size, the model may look exponential when learning the high peaks. For a larger window, the model may have a wider perspective, which includes the peaks, and the problem may seem to become more linear. The errors for a window size of 10 are also much smaller for the training set, and larger for the testing set, compared to the errors found for a window set of 5.

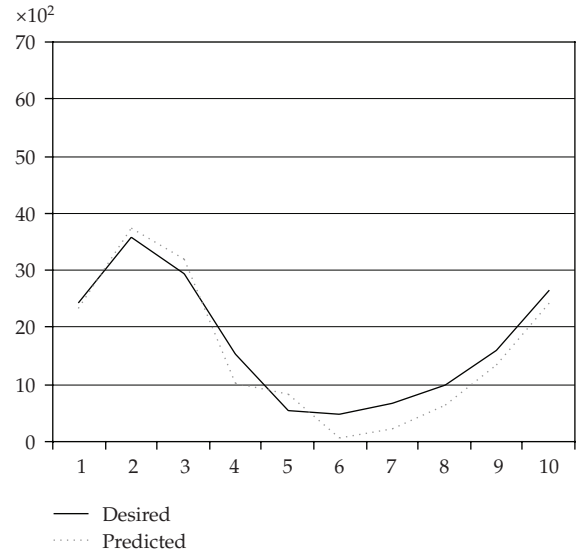

Figure 9: The proposed model predictions for the lynx data (window size 5).

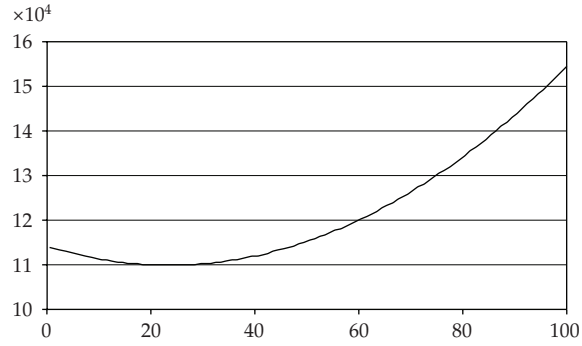

Figure 10: The evolution of MSE when w_1 varies (lynx, window size 5).

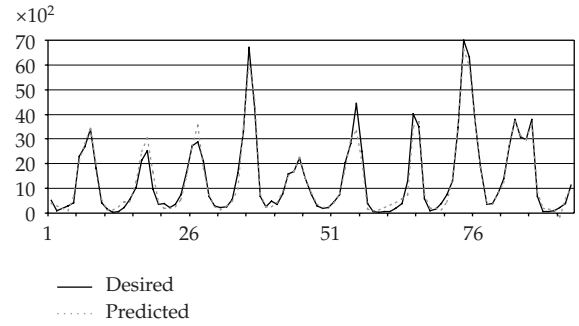

Figure 11: The proposed model performance on the lynx training data (window size 10).

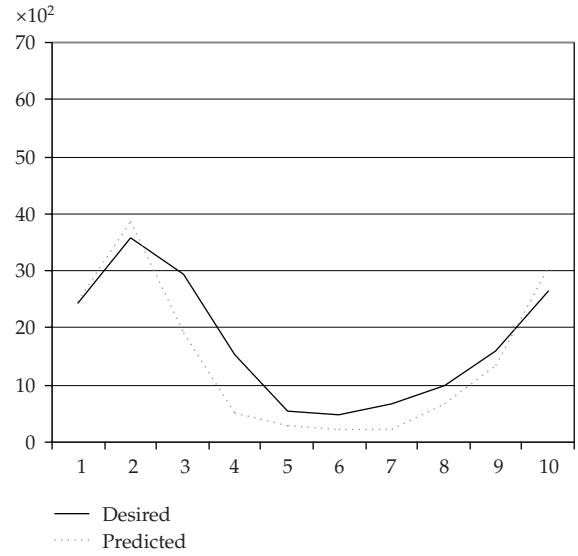

Figure 12: The proposed model predictions for the lynx data (window size 10).

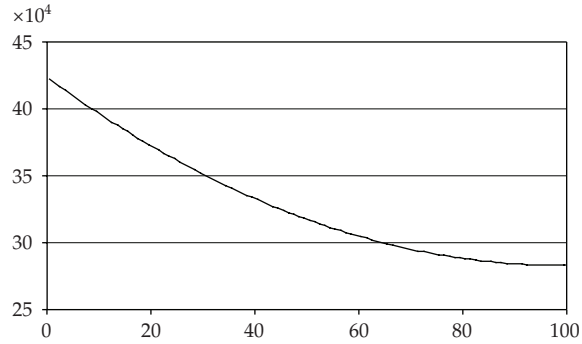

Figure 13: The evolution of MSE when w_1 varies (lynx, window size 10).

Table 4: The errors of the model for the lynx data (window size 10).

		MSE on original data	MSE on normalized data
Training	Normal NN	66,484.341	1.375×10^{-3}
	Exponential NN	88,404.178	1.829×10^{-3}
	Stack	66,426.519	1.374×10^{-3}
Testing	Normal NN	283,132.166	5.858×10^{-3}
	Exponential NN	421,951.130	8.730×10^{-3}
	Stack	283,105.757	5.857×10^{-3}

3.4. Ozone

This data represents monthly ozone concentrations in parts per million from January 1955 to December 1972 made in downtown Los Angeles [20].

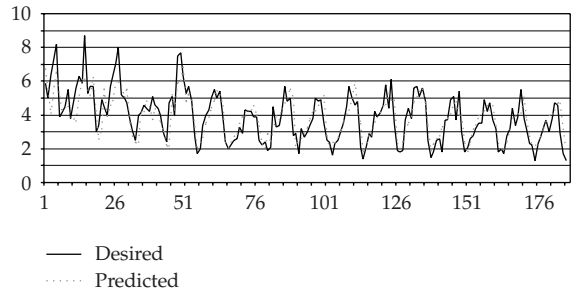

Figure 14: The proposed model performance on the ozone training data (window size 5).

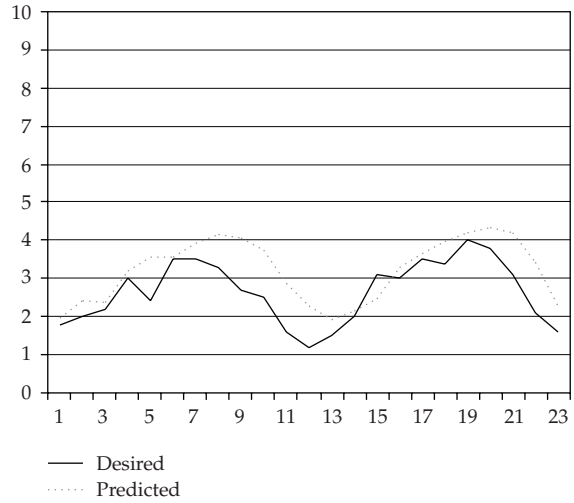

Figure 15: The proposed model predictions for the ozone data (window size 5).

Table 5: The errors of the model for the ozone data (window size 5).

		MSE on original data	MSE on normalized data
Training	Normal NN	0.704	12.519×10^{-3}
	Exponential NN	0.680	12.089×10^{-3}
	Stack	0.675	12.016×10^{-3}
Testing	Normal NN	0.702	12.483×10^{-3}
	Exponential NN	0.592	10.539×10^{-3}
	Stack	0.589	10.485×10^{-3}

We consider a window size of 5 points and 24 points ahead for prediction. The performance of the model on the training set is displayed in Figure 14.

The prediction performance of the model results from Figure 15.

The evolution of the mean square error of the stack on the testing data is displayed as a function of the normal NN weight, w_1, in Figure 16. Here, the optimal weights are $w_1 = 14$ and $w_2 = 86$.

Table 5 shows the mean square errors obtained for this time series.

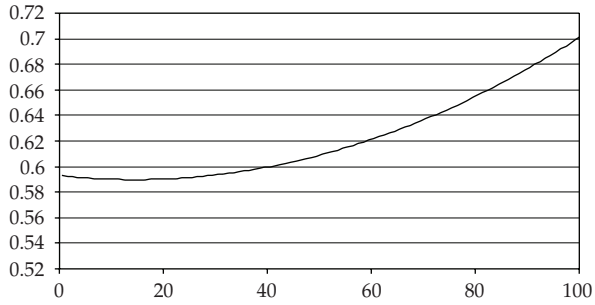

Figure 16: The evolution of MSE when w_1 varies (ozone, window size 5).

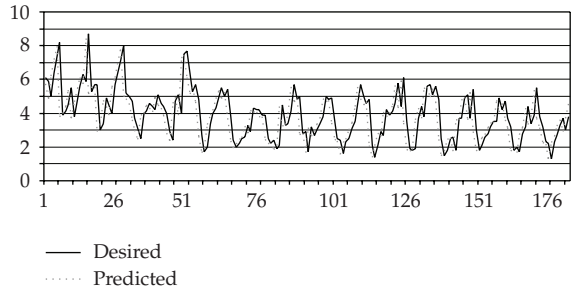

Figure 17: The proposed model performance on the ozone training data (window size 10).

Table 6: The errors of the model for the ozone data (window size 10).

		MSE on original data	MSE on normalized data
Training	Normal NN	0.203	3.609×10^{-3}
	Exponential NN	0.222	3.949×10^{-3}
	Stack	0.201	3.569×10^{-3}
Testing	Normal NN	1.328	23.615×10^{-3}
	Exponential NN	1.238	22.014×10^{-3}
	Stack	1.312	23.336×10^{-3}

In the case when the size of the window is increased to 10 points, the performance of the model on the training set is that shown in Figure 17.

The forecasting capabilities of the model are displayed in Figure 18.

The optimal weights are $w_1 = 96$ and $w_2 = 4$, as it can be seen in Figure 19.

Table 6 shows the mean square errors obtained for a window size of 10.

The behavior of the model on this time series is very similar to that of the lynx time series, regarding both the change in the weights and the comparison of training and testing errors.

3.5. UK Industrial Production

This data series contains the index of industrial production in the United Kingdom, from 1700 to 1912 [20].

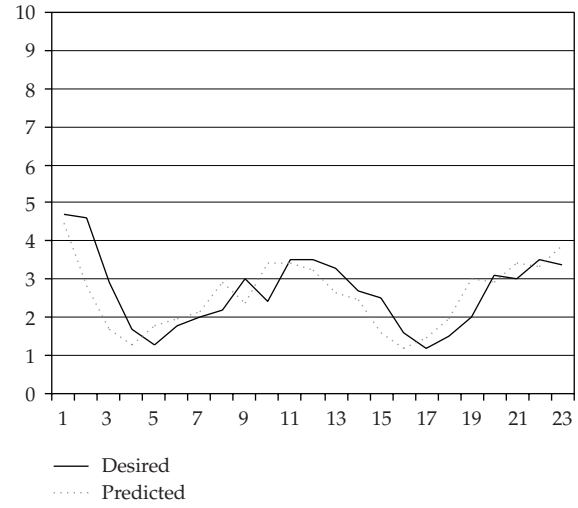

Figure 18: The proposed model predictions for the ozone data (window size 10).

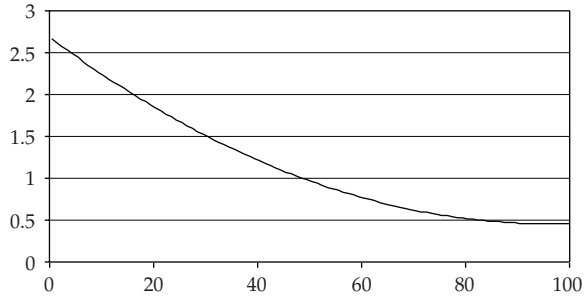

Figure 19: The evolution of MSE when w_1 varies (ozone, window size 10).

Table 7: The errors of the model for the UK industrial production data (window size 5).

		MSE on original data	MSE on normalized data
Training	Normal NN	1.185	0.132×10^{-3}
	Exponential NN	0.988	0.111×10^{-3}
	Stack	0.988	0.111×10^{-3}
Testing	Normal NN	296.895	33.281×10^{-3}
	Exponential NN	9.810	1.099×10^{-3}
	Stack	9.810	1.099×10^{-3}

We first consider the performance of the model on the training set with a window size of 5 points and 21 points ahead, as displayed in Figure 20.

The forecasting capabilities of the model are shown in Figure 21.

The evolution of the mean square error of the stack on the testing data is displayed as a function of the normal NN weight, w_1, in Figure 22. The optimal weights are $w_1 = 0$ and $w_2 = 100$.

Table 7 shows the mean square errors obtained for a window size of 5.

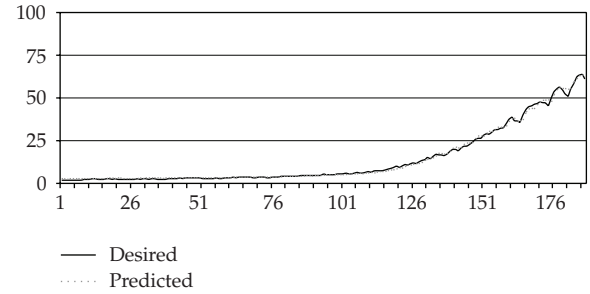

Figure 20: The proposed model performance on the UK industrial production training data (window size 5).

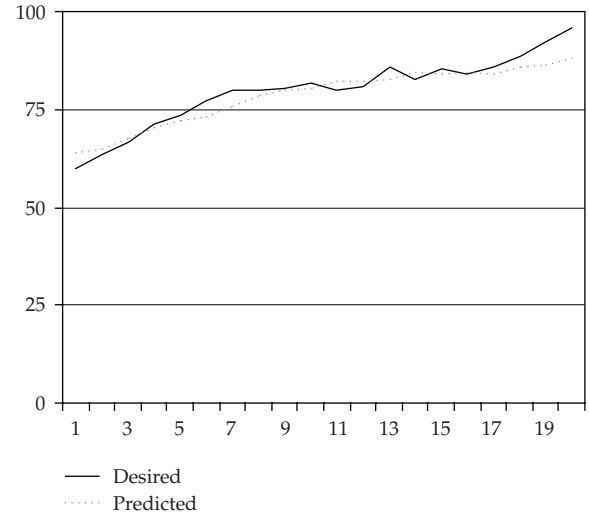

Figure 21: The proposed model predictions for the UK industrial production data (window size 5).

Table 8: The errors of the model for the UK industrial production data (window size 10).

		MSE on original data	MSE on normalized data
Training	Normal NN	0.861	9.682×10^{-5}
	Exponential NN	0.862	9.702×10^{-5}
	Stack	0.862	9.702×10^{-5}
Testing	Normal NN	319.766	3597.439×10^{-5}
	Exponential NN	7.264	81.724×10^{-5}
	Stack	7.264	81.724×10^{-5}

When the size of the window is increased to 10 points, the performance of the model on the training set is the one shown in Figure 23.

The prediction capabilities of the model are displayed in Figure 24.

Just like in the previous case, the optimal weights are $w_1 = 0$ and $w_2 = 100$, as one can see in Figure 25.

Table 8 shows the mean square errors obtained for a window size of 10.

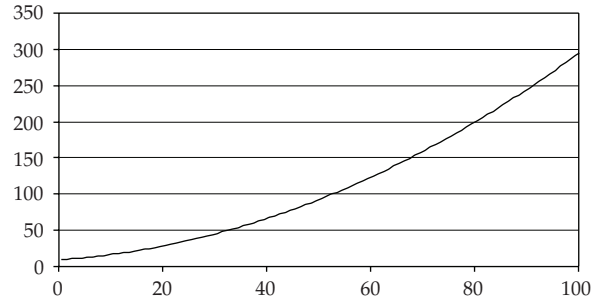

Figure 22: The evolution of MSE when w_1 varies (UK industrial production, window size 5).

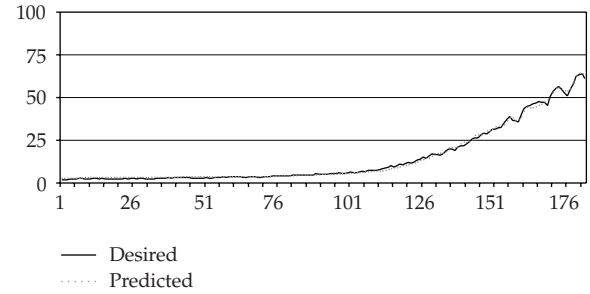

Figure 23: The proposed model performance on the UK industrial production training data (window size 10).

Unlike the previous problems, the exponential nature of this time series makes it difficult for a normal neural network. Therefore, the exponential network dominates the stack, independent of the size of the window. One can notice that although the normal network can approximate the training set fairly well, with errors comparable to those of the exponential network, there is a clear difference in performance for the prediction phase, where only the exponential network can find a good trend for the time series.

3.6. Super-Exponential Growth

In order to test the limits of our model, we devised a function given by the following equation:

$$f(x) = \left(\frac{x}{100} + 1\right)^{x/15+1} + 5 \cdot \sin\left(\frac{x}{10}\right) \cdot \sqrt{x}. \tag{3.1}$$

The first term of this function is chosen in such a way that both the base and the exponent increase with x, thus producing a super-exponential growth. The second term is a kind of sinusoid that would account for fluctuations to the regular growth model. The values of the coefficient are chosen in such a way that at first, for low values of x, the oscillatory behavior is more important, and then, as x increases, the super-exponential term begins to dominate the value of the function.

Since the function is super-exponential, and the activation function of the second neural network is only exponential, it is expected that our stack model will learn the training

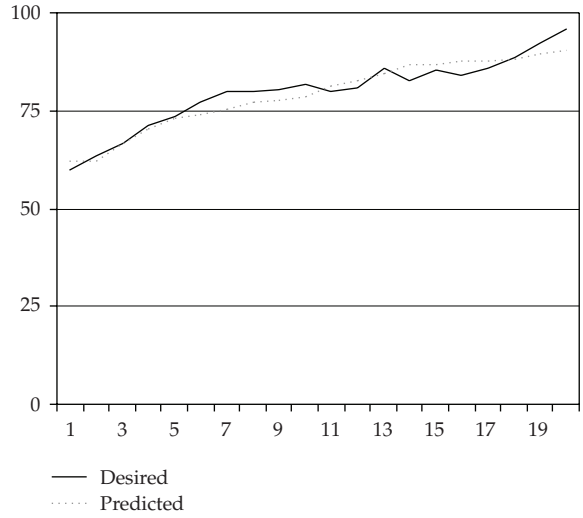

Figure 24: The proposed model predictions for the UK industrial production data (window size 10).

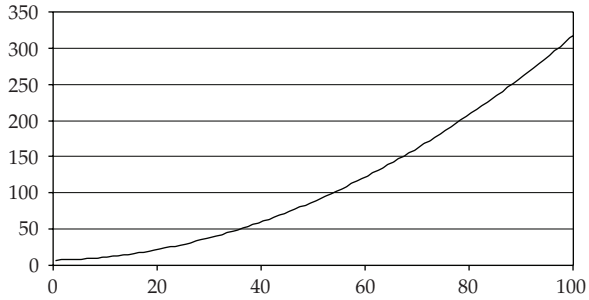

Figure 25: The evolution of MSE when w_1 varies (UK industrial production, window size 10).

data well, but will fail to extrapolate to the prediction set. This drawback can be compensated by allowing different activation functions, such as a double-exponential function a^{b^x}, to the output layer of the neural network.

With a window size of 5 points and considering 21 points ahead, the performance of the model on the training set is displayed in Figure 26.

The forecasting capabilities of the model are displayed in Figure 27.

The evolution of the mean square error of the stack on the testing data is displayed as a function of the normal NN weight, w_1, in Figure 28. Here, the optimal weights are $w_1 = 0$ and $w_2 = 100$.

Table 9 shows the mean square errors obtained for a window size of 5.

When size of the window is increased to 10 points, the performance of the model on the training set is that shown in Figure 29.

The forecasting capabilities of the model are displayed in Figure 30.

In a similar way to the case with a window size of 5, the optimal weights are $w_1 = 0$ and $w_2 = 100$, as it can be seen in Figure 31.

Table 10 shows the mean square errors obtained for a window size of 10.

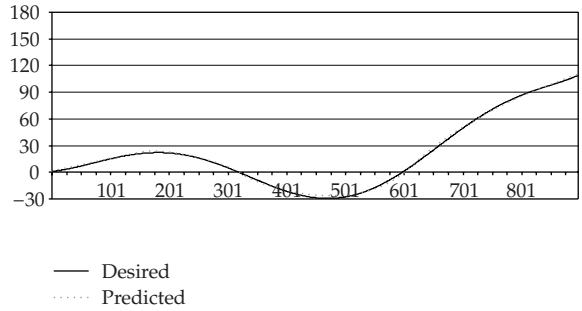

Figure 26: The proposed model performance on the super-exponential growth training data (window size 5).

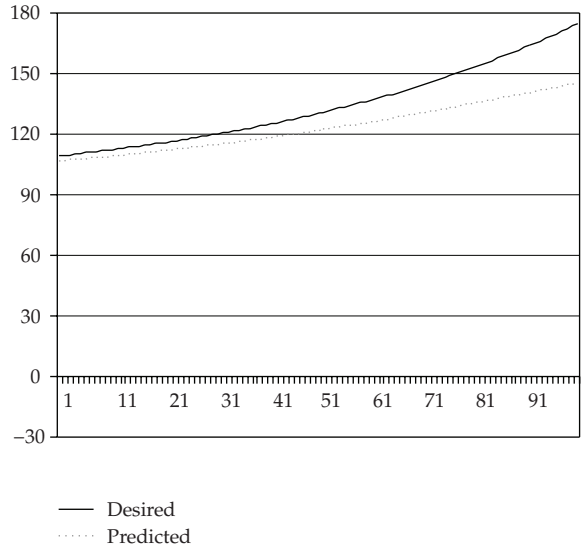

Figure 27: The proposed model predictions for the super-exponential growth data (window size 5).

Table 9: The errors of the model for the super-exponential growth data (window size 5).

		MSE on original data	MSE on normalized data
Training	Normal NN	2.825	6.783×10^{-5}
	Exponential NN	1.593	3.824×10^{-5}
	Stack	1.593	3.824×10^{-5}
Testing	Normal NN	826.936	19.851×10^{-3}
	Exponential NN	172.645	4.144×10^{-3}
	Stack	172.645	4.144×10^{-3}

This time series poses similar problems as the previous one. The only difference is that the super-exponential nature of the proposed function exceeds the prediction possibilities of the exponential network. For this kind of problems, other types of activation functions can be used. The stacked model proposed here is flexible enough to accommodate different types of neural networks, with different activation functions.

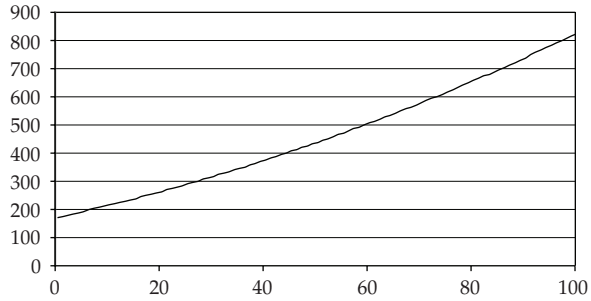

Figure 28: The evolution of MSE when w_1 varies (super-exponential growth, window size 5).

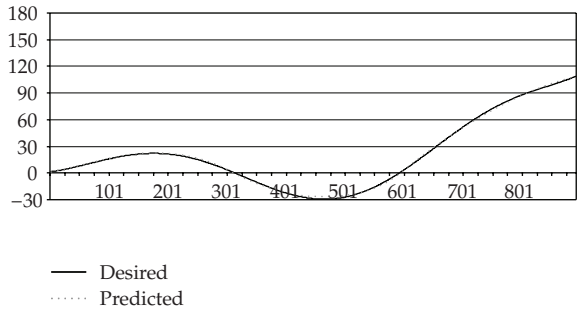

Figure 29: The proposed model performance on the super-exponential growth training data (window size 10).

Table 10: The errors of the model for the super-exponential growth data (window size 10).

		MSE on original data	MSE on normalized data
Training	Normal NN	5.270	12.651×10^{-5}
	Exponential NN	1.047	2.513×10^{-5}
	Stack	1.047	2.513×10^{-5}
Testing	Normal NN	939.119	22.544×10^{-3}
	Exponential NN	76.707	1.841×10^{-3}
	Stack	76.707	1.841×10^{-3}

3.7. Comparison with Other Forecasting Models

We compare the model fitting performance of our stack neural network with several other models, using the implementations in the Statistical Analysis System (SAS) 9.0 software package [21]. We compare the mean square error of our model for the two different window sizes with the error of the best model in SAS. Table 11 presents this comparison.

The best error for a problem is shown in bold letters. It can be seen that our model outperforms the models implemented in SAS for all but the last benchmark problems. The reason for the greater error for the super-exponential growth problem is the inherent limitation of choosing an exponential instead of a super-exponential activation function.

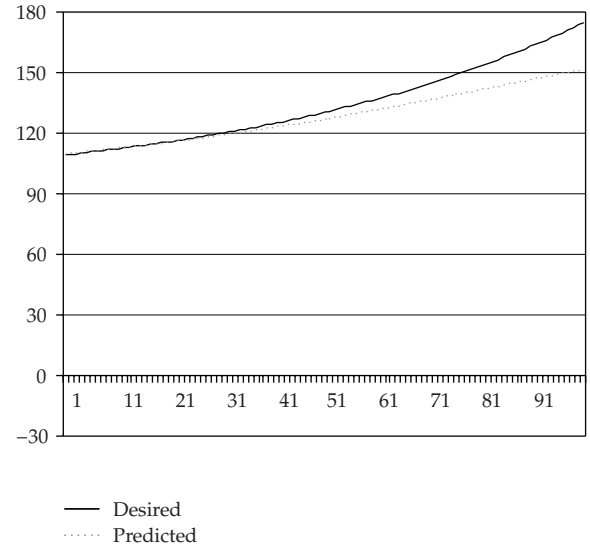

Figure 30: The proposed model predictions for the super-exponential growth data (window size 10).

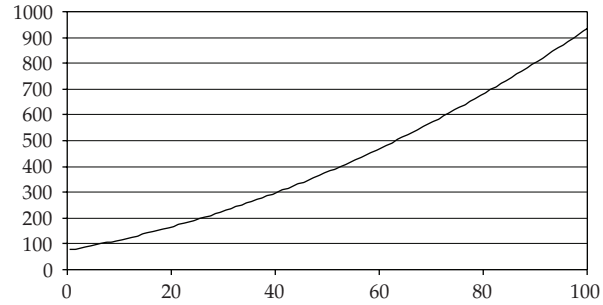

Figure 31: The evolution of MSE when w_1 varies (super-exponential growth, window size 10).

Table 11: Comparative performance of the proposed model with other forecasting models.

Time series	MSE of the stacked neural network with a window size of 5	MSE of the stacked neural network with a window size of 10	MSE of the best SAS model	Name of the best SAS model
Sunspots	131.865	**96.085**	549.21	Simple exponential smoothing
Lynx	401,015.900	**66,426.519**	1,410,768	Simple exponential smoothing
Ozone	0.704	**0.201**	1.079	Simple exponential smoothing
UK Industrial Production	0.988	**0.862**	1.339	Log random walk with drift
Super-Exponential Growth	1.593	1.047	$\mathbf{1.59 \times 10^{-5}}$	Double exponential smoothing

4. Conclusions

Despite its simplicity, it seems that the stacked hybrid neural model performs well on a variety of benchmark problems for time series. It is expected that it can have good results for other important problems that show dynamical and predictive aspects. The model can be easily extended to incorporate other activation functions that can be suitable for a particular problem, such as a double-exponential function a^{b^x}. It is also possible to include nondifferentiable functions in the model, if one adopts an evolutionary algorithm for training the neural networks, instead of the classical back-propagation algorithm.

Acknowledgment

This work was supported in part by CNCSIS grant code 316/2008, *Behavioral Patterns Library for Intelligent Agents Used in Engineering and Management*.

References

[1] H. Tong, *Threshold Models in Nonlinear Time Series Analysis*, vol. 21 of *Lecture Notes in Statistics*, Springer, New York, NY, USA, 1983.
[2] D. Pfeffermann and J. Allon, "Multivariate exponential smoothing: method and practice," *International Journal of Forecasting*, vol. 5, no. 1, pp. 83–98, 1989.
[3] J. L. Harvill and B. K. Ray, "A note on multi-step forecasting with functional coefficient autoregressive models," *International Journal of Forecasting*, vol. 21, no. 4, pp. 717–727, 2005.
[4] M. Ghiassi and S. Nangoy, "A dynamic artificial neural network model for forecasting nonlinear processes," *Computers and Industrial Engineering*, vol. 57, no. 1, pp. 287–297, 2009.
[5] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*, Springer Series in Statistics, Springer, New York, NY, USA, 2nd edition, 1991.
[6] N. Ravishanker and B. K. Ray, "Bayesian prediction for vector ARFIMA processes," *International Journal of Forecasting*, vol. 18, no. 2, pp. 207–214, 2002.
[7] F. M. Tseng, H. C. Yu, and G. H. Tzeng, "Combining neural network model with seasonal time series ARIMA model," *Technological Forecasting and Social Change*, vol. 69, no. 1, pp. 71–87, 2002.
[8] M. V. D. Voort, M. Dougherty, and S. Watson, "Combining Kohonen maps with ARIMA time series models to forecast traffic flow," *Transportation Research Part C*, vol. 4, no. 5, pp. 307–318, 1996.
[9] D. K. Wedding II and K. J. Cios, "Time series forecasting by combining RBF networks, certainty factors, and the Box-Jenkins model," *Neurocomputing*, vol. 10, no. 2, pp. 149–168, 1996.
[10] C.-M. Lee and C.-N. Ko, "Time series prediction using RBF neural networks with a nonlinear time-varying evolution PSO algorithm," *Neurocomputing*, vol. 73, no. 1–3, pp. 449–460, 2009.
[11] G. Toma, "Specific differential equations for generating pulse sequences," *Mathematical Problems in Engineering*, vol. 2010, Article ID 324818, 11 pages, 2010.
[12] E. G. Bakhoum and C. Toma, "Mathematical transform of traveling-wave equations and phase aspects of quantum interaction," *Mathematical Problems in Engineering*, vol. 2010, Article ID 695208, 15 pages, 2010.
[13] M. Khasei and M. Bijari, "An artificial neural network (p, d, q) model for timeseries forecasting," *Expert Systems with Applications*, vol. 37, no. 1, pp. 479–489, 2010.
[14] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.
[15] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*, Blaisdell, New York, NY, USA, 1969.
[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds., vol. 1, pp. 318–362, MIT Press, Cambridge, Mass, USA, 1986.
[17] F. M. Silva and L. B. Almeida, "Acceleration techniques for the backpropagation algorithm," in *Neural Networks*, L. B. Almeida and C. J. Wellekens, Eds., pp. 110–119, Springer, Berlin, Germany, 1990.
[18] M. Ghiassi and H. Saidane, "A dynamic architecture for artificial neural networks," *Neurocomputing*, vol. 63, pp. 397–413, 2005.

[19] L. Stone and D. He, "Chaotic oscillations and cycles in multi-trophic ecological systems," *Journal of Theoretical Biology*, vol. 248, no. 2, pp. 382–390, 2007.
[20] G. Janacek, *Practical Time Series*, Oxford University Press, Oxford, UK, 2001.
[21] SAS Institute, "SAS (Statistical Analysis System)," http://www.sas.com/.