Combining Genetic Algorithms, Neural Networks and Data Filtering for Time Series Forecasting *

José Neves   Paulo Cortez
jneves@di-ia.uminho.pt   paulo@di-ia.uminho.pt
Departamento de Informática
Universidade do Minho
Largo do Paço, 4709 Braga Codex, PORTUGAL
Voice: +351(53)604466/70   Fax: +351(53)604471

Keywords: Genetic Algorithms, Neural Networks, Data Filtering, Time Series.

Abstract

In the last few decades an increasing focus as been put over the field of *Time Series Forecasting (TSF)*, the forecast of a time ordered variable. Contributions from the arenas of *Operational Research*, *Statistics*, and *Computer Science* as lead to solid *TSF* methods (eg. *Exponential Smoothing* or *Regression*) that replaced the old fashion ones, which were primary based on intuition. Although these methods give accurate forecasts on linear *Time Series (TS)*, their handicap is with noise or nonlinear components, which is a commum situation (eg. in financial daily *TS*). An alternative approach for *TSF* as recently emerged from the field of *Artificial Intelligence*, where new optimization algorithms, such as *Genetic Algorithms* and *Artificial Neural Networks* have became popular. Following this trend, the present work reports on a *Genetic Algoritm Neural Network* system, and in its use for *TSF*.

1 Introduction

Although some well established management methods have been in place for the last decades, which were based on the feelings and intuition of the managers, the approach seems no more appropriate. These methods have been replaced by a new brand of decision making techniques, whose principles emerged from disciplines such as *Operational Research*, *Computer Science*, *Statistics* and *Principles of Organization Design*, among others [10].

Indeed, an important point, when making a decision, is the capacity to predict the circumstances that surround it. Thus, it seems natural that enterprises are interested in obtaining well grounded forecasts of the variables that affect strongly the success (or unsuccess) of such proceedings [7]. One way is to use *Time Series Forecasting (TSF)*, forecast of a chronologically ordered variable. The goal in *TSF* is to learn patterns from historical data in order to predict the behavior of the system, and not how it works. Normally, *TSF* models are based on methods of *Exponential Smoothing*, *Decomposition* or *Regression*. Although this kind of approach turns out to be efficient on well behaved *Time Series (TS)*, it presents drawbacks when noise or other nonlinear relations are present, what is a common situation on financial *TS* on a day-to-day basis [18].

New optimization algorithms based on processes that mimic the natural evolution of the living species have gained an increasing importance within the field of *Artificial Intelligence*. In particular, the study of phenomenons of the process of natural selection or the central nervous system, has leaded to successful tools such as *Genetic Algorithms (GAs)* and *Artificial Neural Networks (ANNs)* [17]. One promising approach for *TSF* comes from the use of the last ones. *ANNs* are connectionist models that can learn patterns from past data in order to respond to new situations, even when noise or incomplete data is present. This is the main reason why they have comparative advantages over other methods, with applications ranging from computer vision and data analysis to expert systems, just to name a few.

*This work was supported by JNICT through the PRAXIS' grant XXI/ BD/ 13793/ 97

The application of *ANNs* for *TSF* started on the late eighties. The intention was to overwhelm the limitations of the conventional *TSF* methods, in particular when used on non-linear financial data; indeed, encouraging results appeared over the use of feedforward *ANNs* on those markets [14] [5]. Other comparative studies [3][20][12] have shown that *ANNs* can perform as well or even better that conventional methods, namely the Holt-Winters [7] and the Box-Jenkins ones [2].

A problem that arises with this approximation is the search of the best *ANN* topology, which can demand a huge computational effort. To overcome this problem one can use random search, hill-climbing or *GAs*. The last ones are effective on problems of combinatorial nature, giving good results were other methods seem to fail. Thus is seems natural to combine *GAs* with *ANNs*, in the so called *GANN* systems, in order to maximize the advantages of both tools [9]. In these systems, the *ANN* learning guides the evolutionary search of the best topology. Empirical studies suggest that the *GANNs* systems outperform the *GAs* and the *ANNs* in the search for a solution [8][19]. Other problem with the use of *ANNs* is related with the best way to feed the data into an *ANN*, which is called the preprocessing stage. In this process, the use of normalization techniques or data filtering may be very useful [1].

2 The Artificial Neural Network Architecture

Some decisions have to be made when using *ANNs*, which depend on the specificity of the problem under consideration or on the kind of data used. Therefore, it was decided to use feedforward *ANNs*, a popular architecture that has been used by most of the studies on this field [14][20][3]. Then, in order to reduce the searching space and guided by some empirical evidence, it were chosen fully connected *ANNs* with bias, with just one hidden layer and without shortcut connections (Figure 1). The topology will be represented by productions in the form $L_i - L_h - Lo$, for an *ANN* with L_i input nodes, L_h hidden nodes, and L_o output nodes [15]. The ***R****esilent Back****PROP****agation (RPROP)* algorithm was adopted for the *ANN* training [16]. The initial weights were randomly generated within the range $[\frac{-2}{z};\frac{2}{z}]$ for a node with z inputs [6]. Finally, eight activation functions were tested (Table 1) [1].

There are several ways to feed a *TS* into an *ANN*, in order to perform *TSF*. In this work it was decided

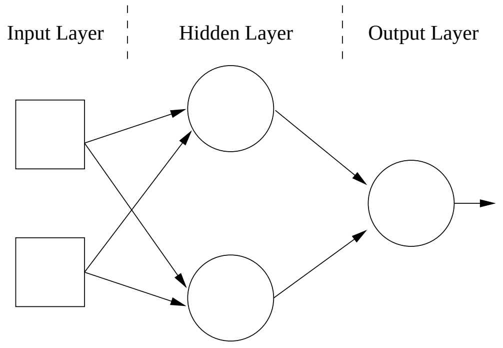

Figure 1: Structure of the *ANN* used

Table 1: Activation Functions

Name	**Function** $f(x)$	**Codomain**
linear	x	$]-\infty,+\infty[$
sigmoid	$\frac{1}{1+\exp(-x)}$	$[0,1]$
sigmoid1	$\frac{2}{1+\exp(-x)}-1$	$[-1,1]$
sigmoid2	$\frac{x}{1+\lvert x\rvert}$	$[-1,1]$
tanh	$\tanh(x)$	$[-1,1]$
cos	$\sin(x \bmod 2\pi)$	$[-1,1]$
sin	$\cos(x \bmod 2\pi)$	$[-1,1]$
gaussian	$\exp(\frac{-x^2}{2})$	$[-1,1]$

to use a sliding time window of size k, for an *ANN* with k input nodes. The goal is to have the output as a function of k previous *TS* values, thus reducing the weights to be estimated [4]. The *ANN* will take the form $k - L_h - 1$. The training cases, created by the time window, are given bellow:

$$\begin{array}{lcl} x_1, x_2, ..., x_k & \rightarrow & x_{k+1} \\ x_2, x_3, ..., x_{k+1} & \rightarrow & x_{k+2} \\ \ldots & \rightarrow & \ldots \\ x_{p-k}, ..., x_{p-1} & \rightarrow & x_p \end{array}$$

for a *TS* of length p $(x_1, ..., x_p)$.

Once some activation functions require that the data must be bounded by a certain range (Table 1), the training cases have to be normalized. This range was set to $[0,1]$ after some empirical tests. One way to perform the normalization is to use a simple linear transformation. The problem with this approach is that a *TS* with a *trend* component can present future values above that range. In principle, a higher range permits a better assimilation by the *ANN*, so it was decided to use the range $[0.2, 0.8]$ (value suggest in [20]) for *TS* with *trend* and the range $[0,1]$ otherwise.

An early stopping procedure has been adopted to prevent overfitting; i.e. the training cases will be divided into two sets: a training set (to assimilate the data) and a validation set (to test if the *ANN* has a generalization capacity) [15]. In *TSF*, recent past data affects strongly the forecast, so it was decided to use only 10% of the available data in the validation set.

After training, one-step ahead forecast is produced by feeding the *ANN* with the last known values of the *TS*:

$$F_{1,p} = ANN(x_{p-k+1}, ..., x_p)$$

where $ANN()$ stands for the function modeled by the *ANN* and $F_{i,j}$ for the forecast in the j period to i periods ahead of j. Thus, a short term one-ahead forecast, for l periods, will be given by the set:

$$\{F_{1,p}; F_{1,p+1}; ...; F_{1,p+l-1}\}$$

As an example, one will consider the seasonal monthly *TS* given by $\{..., 5, 16, 20, 16, 20, 24\}$, being the last values relative to the months between January and June. Assuming that the forecasts are performed by an *ANN* with 3 input nodes, and that the present month is March, then the forecast for April will be given by $F_{1,\text{March}} = ANN(5, 16, 20)$. In a similar way, in April one will have $F_{1,\text{April}} = ANN(16, 20, 16)$. Thus, a short term forecast, during 3 months ($l = 3$), will be given by the set $\{F_{1,\text{March}}; F_{1,\text{April}}; F_{1,\text{May}}\}$.

Multi-step ahead forecasts are done by feedback of the previous forecasts:

$$\begin{array}{l} F_{1,p} = ANN(x_{p-k+1}; ...; x_p) \\ F_{2,p} = ANN(x_{p-k+2}; ...; x_p; F_{1,p}) \\ F_{3,p} = ANN(x_{p-k+3}; ...; x_p; F_{1,p}; F_{2,p}) \\ \ldots \\ F_{l,p} = ANN(F_{l-k,p}; ...; F_{l-1,p}) \\ \ldots \end{array}$$

A long term forecast, until l periods ahead of p, will then be given by the set:

$$\{F_{1,p}; F_{2,p}; ...; F_{l,p}\}$$

3 The *GANN* System

After some experiences with *ANNs* it was possible to conclude that the forecasting is a function of four factors: L_i - the number of input nodes; Δ_0 and Δ_{max} - the *RPROP* parameters; f - the activation function; and L_h - the number of hidden nodes. This can be easily explained: the first factor sets the size of the time window, while the others affect the way how the *ANN* learns.

This is where the *GA* enters. The *GA* will be used to select the best *ANN* to each *TS*, by evolution. The first step in one's approach is to choose the *GA* chromosome. The four parameters were coded using the base 2 (two) gray code (Figure 2). The number of nodes (L_i and L_h) were encoded with 4 (four) bits, within the range $[3, 14]$. There is some evidence that this is the correct range [3], and a higher one would increase exponentially the searching time. In the case of the *RPROP* parameters, Δ_{max} was settled to 50 (fifty) (the advised value [16]) while Δ_0 was encoded to the discrete range of $\{0.1, 0.2, ..., 0.8\}$, using only 3 (three) bits. The eight activation functions were also codified with 3 (three) bits.

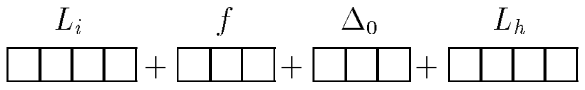

Figure 2: The *ANN* chromosome

One crucial aspect of the *GANN* system is the selection of the the fitness function. Normaly, when one works with *GAs* the fitness function is well determined and easy to compute, which is not the present case. The fitness will be indirectly given by the *ANN* training, using the error from the validation cases (MSE_{va}); i.e. a fitness value will be given by

$$fitness = \frac{1}{MSE_{va}}$$

The *GA* works as an optimization procedure of second order, so the tuning of its parameters is not so crucial. Based on some empirical tests, intuition, and having in mind to cut some of the searching time, it was decided to use an *GA* with a population of 30 (thirty) individuals; rank-based selection and one point crossover operators; and a mutation rate of 0.02. Figure 3 shows how the *GA* works. An initial set of individuals are randomly created. Then the *ANNs* are conceived according to the chromosome information. The evaluation occurs after the *ANN* training. Each individual is ranked according to its fitness. The crossover and mutation operators will create a new population, which will also be evaluated. Finally, the rank-based selection operator will select the best *ANNs* from both populations, breading a new one. The process continues until a minimum of energy for the chromosomes social setting is reached [13].

3.1 Results for the *GANN* system

In this work there was some care in selecting *TS* that were related with real problems, from different domains such as financial markets or natural processes,

```
BEGIN
    Initialize generation ($g = 0$)
    Create initial population ($P_0$)
    WHILE NOT (Minimum-of-Energy) DO
        Evaluate population ($P_g$)
        Select individuals for crossover
        Cross parents to breed new individuals ($P'_g$)
        Mutate new population ($P'_g$)
        Evaluate new population ($P'_g$)
        Select individuals from both populations for
        the next one ($P_{g+1}$)
        Increase current generation ($g = g + 1$)
    END WHILE
END
```

Figure 3: Structure of the *GA*

which have already been studied by other conventional methods (Table 2) [2][7][10], where s stands for the seasonal factor, when applied. The number of forecasts, l, was set to 12 for the monthly *TS*, and to 10 for the rest.

Table 2: Characteristics of the *TS*

Series (S)	**Period**	**Trend**	s
1	Yearly	No	-
2	Monthly	Yes	12
3	Monthly	Yes	12
4	Daily	Yes	-
5	Monthly	Yes	12
6	Daily	Yes	-
7	Daily	No	-
8	-	No	-
9	-	No	-
10	Monthly	Yes	-
11	Monthly	Yes	12

Table 3 shows the best *ANN* for each series of Table 2. As expected, the *ANNs* are distinctive, thus justifying the need of the *GA*. As expected, seasonal series seem to require larger time windows, with an L_i between 12 (twelve) and 13 (thirteen). These series present a seasonal factor of 12 (twelve) (small time windows will not capture the seasonal patterns). For the other series, the time window falls within the range $[3, 7]$, which indicates that the data from recent series affects strongly the forecast. The remaining factors do not seem to interfere on the *TS* make up; i.e. these factors only affect the way the *ANN* learns, and not how the cases are created.

Table 3: The best *ANN* for each series

S	**Topology**	f	Δ_0	MSE_{va}
1	$4-4-1$	*sin*	0.7	22
2	$13-11-1$	*linear*	0.4	227
3	$13-3-1$	*linear*	0.8	243966
4	$7-9-1$	*linear*	0.2	49.7
5	$12-8-1$	*gaussian*	0.5	16612
6	$4-3-1$	*cos*	0.8	38.5
7	$6-14-1$	*sigmoid2*	0.3	173.6
8	$7-6-1$	*sin*	0.8	33.3
9	$6-6-1$	*sigmoid2*	0.2	2.72
10	$3-11-1$	*sigmoid2*	0.6	5593
11	$13-7-1$	*gaussian*	0.5	2287

One shall discuss the short term results (Table 4), using as a referential the ARIMA [2] and the Holt-Winters [7] methods. The results from Holt-Winters outperform those obtained with the other methods for the seasonal series 2, 3, 5 and 11, what is not surprising, since this method was conceived to this kind of series. With series presenting a strong trend (series 4, 6 and 10), the *GANN*'s system gives better results than ARIMA for the first two series. Series 10 has monthly periods, which may make it seasonal, with Holt-Winters, which is seasonal, presenting the best results. For the series without trend and seasonal components (series 1, 7, 8 and 9), the system's results are similar to those obtained by ARIMA, being even better for series 1 and 8.

Table 4: Comparative results for short term forecasting (using *MSE* as the error measure)

S	**GANN**	**ARIMA**	**Holt-Winters (HW)**
1	**139.6**	154	-
2	340.3	452	**271**
3	834578	-	**530654**
4	63.9	**36.3**	-
5	20672	15290	**11435**
6	**62.9**	72.1	-
7	3670	**2939**	-
8	**66.8**	141.5	-
9	265	**240.4**	-
10	20711	24217	**18446**
11	6238.7	2581.3	**1885**

Similar results were achieved for long term forecasts (Table 5), being the exception given by Holt-Winters. In fact, these results fall when compared with other methods, for series 2 and 11. In a surprising way, the *GANN* system forecasts outperform all the others for series 2. It was not possible to perform comparisons with ARIMA, for the series with trend; indeed, ARIMA was not conceived for this kind of forecast. On the other hand the *GANN*'s systems results were better that the Holt-Winter's ones. For the rest of the series, the system's results only outperformed the ARIMA ones for series 7.

Table 5: Comparative results for long term forecasting (using *MSE* as the error measure)

S	GANN	ARIMA	HW
1	562.1	**267**	-
2	**370.4**	521.8	621.5
3	3888567	-	**2927255**
4	214.25	-	-
5	21542	20289.3	**16954**
6	170.4	-	-
7	**2867**	2897	-
8	270.3	**161.3**	-
9	316	**205.2**	-
10	**33548**	-	52012
11	6258.5	**2707.7**	3046.4

4 Data Filtering

In the previous section it was shown that seasonal series require a large time window (with a size between 12 and 13). However, not all time lags have the same influence on the forecast. Indeed, a large time window can increase the system entropy, affecting the *ANN* learning capacity and the accuracy of the forecast. On the other hand, the results obtained with the *GANN* system for this kind of series where somehow disappointing. It seems therefore natural that selecting the correct time lags to feed the *ANN* may improve the forecast, in particular for seasonal *TS*.

The structure of the previous used floating time window was given in the form $< 1, 2, .., n-1, n >$, where n stands for the size of the window (Figure 4). With the data filtering approach one pretends to use small time windows, with only the relevant data. For example, ARIMA suggests the use of the windows $< 1, 12, 13 >$ or $< 1, 2, 12, 13 >$ for seasonal monthly *TS*, which will be the ones used to test this system (series 2, 3, 5, 10 and 11 from Table 2).

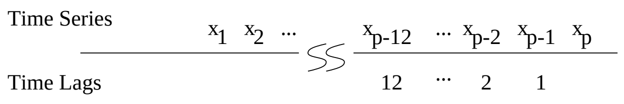

Figure 4: Time lags of a *TS*

A new functionality will be added to the *GANN* system. This new system will be called ***GANNF*** (***GANN*** system with **F**iltering). Twelve sets of lags will be used (Table 4). These sets were based on the work of Faraday and Chatfield [4]. The type of window was encoded with 4 bits, replacing the L_i factor of the *GANN* system (Figure 5). Excepting this fact, the *AG* behaves in a similar way to the one used in the previous system.

Table 6: Sets of lags

Window (W)	Lags	L_i
1	$< 1, 12 >$	2
2	$< 1, 2, 12 >$	3
3	$< 1, 12, 13 >$	3
4	$< 1, 2, 3, 12 >$	4
5	$< 1, 2, 12, 13 >$	4
6	$< 1, 2, 3, 4, 12 >$	5
7	$< 1, 2, 12, 13, 14 >$	5
8	$< 1, 2, 3, 4, 12, 13 >$	6
9	$< 1, 2, 3, 4, 12, 13, 14 >$	7
10	$< 1, 2, ..., 12 >$	12
11	$< 1, 2, ..., 13 >$	13
12	$< 1, 2, ..., 14 >$	14

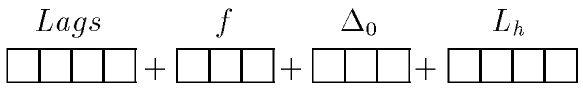

Figure 5: Chromosome of the *GANNF* system

The training cases were created according to each time window. As an example, for the window 2, one has:

$$\begin{array}{lcl} x_1, x_{11}, x_{12} & \rightarrow & x_{13} \\ x_2, x_{12}, x_{13} & \rightarrow & x_{14} \\ \ldots & \rightarrow & \ldots \\ x_{p-12}, x_{p-2}, x_{p-1} & \rightarrow & x_p \end{array}$$

As with the previous system, after training, one can perform forecasts by feeding the *ANN* with the lastest time window or with feedbacks of the previous forecasts. For the example depicted above, one has:

$$\begin{array}{l} F_{1,p} = RNA(x_{p-11}, x_{p-1}, x_p) \\ F_{2,p} = RNA(x_{p-10}, x_p, F_{1,p}) \\ \ldots \\ F_{l,p} = RNA(F_{l-12,p}, F_{l-2,p}, F_{l-1,p}) \\ \ldots \end{array}$$

5 Data Filtering Results

Table 7 shows the best *ANN* for each series. Although the *ANNs* are also distinctive, they are smaller than the ones obtained by the *GANN* system. This fact adds up to the utility of data filtering, which led to simple *ANNs*. A confrontation with the previous system (Table 3) shows that better fitness values (MSE_{va}) are attained for series 2, 5 and 11. For series 10, a worse result is not surprising, since the better *ANN* obtained with the *GANN* system follows a topology of $3-11-1$, with a window $< 1, 2, 3 >$, a solution out of range of the *GANNF*'s searching space. However, the behavior of series 3 is strange, since the solution is in the searching space of the filtering system. An explication may have to be found on the criterion Minimum-of-Energy, referred to above.

Table 7: The best *ANN* for each series

S	W	Topology	f	Δ_0	MSE_{va}
2	3	$3-8-1$	*linear*	0.7	120.4
3	8	$6-6-1$	*linear*	0.2	277101
5	4	$4-7-1$	*gaussian*	0.5	9911
10	2	$3-13-1$	*linear*	0.2	6353
11	4	$4-3-1$	*gaussian*	0.5	1213.7

For short term forecasting (Table 8), Holt-Winters still gives better results, being the exception the series 10, where the *GANNF* results outperformed all the others. A parallel with other methods shows the filtering system with a performance that outperforms that of the *GANN*'s system for series 3, 10 and 11, and ARIMA for series 2 and 10.

Table 8: Comparative results for short term forecasting (using *MSE* as the error measure)

S	GANN	GANNF	ARIMA	HW
2	340.3	368.3	452	**271**
3	834578	759581	-	**530654**
5	20672	25103.8	15290	**11435**
10	20711	**12486.3**	24217	18446
11	6238.7	4500	2581.3	**1885**

The advantages of the filtering system are stressed on long term forecasting (Table 9). In fact, the *GANNF*'s system outperforms the *GANN*'s one for almost all the series (the exception is series 5). Another aspect that has to be stressed is the loss of superiority from Holt-Winters, that outperforms the other methods only for series 5 and 11.

Table 9: Comparative results for long term forecasting (using *MSE* as the error measure)

S	GANN	GANNF	ARIMA	HW
2	370.4	**308**	521.8	621.5
3	3888567	**657445**	-	2927255
5	21542	31279	20289.3	**16954**
10	33548	**14534**	-	52012
11	6258.5	4792	**2707.7**	3046.4

6 Conclusions

The surge of new optimization algorithms, like *ANNs* and *GAs*, has created new exciting possibilities for new developments in vast fields. More recently, appeared the possibility to combine *GAs* with *ANNs*, in the so called *GANN*'s systems, in order to potentiate the advantages from both tools. Although there is a lot of work done on this promising approach, the applications world does not follows the pattern [9]. This work contributes to preempt it, with the application of the *GANN*'s systems to *TSF*.

The results suggest that the *ANNs* can be an alternative for *TSF*, specially for *TS* with a high non-linear degree (such as series 1). Comparative results show that the *GANN*'s systems have a performance near the one given by ARIMA, outperforming it to some series. The data filtering facility enhances the forecast of seasonal series, specially for long term forecasts. However, for this kind of series, and for short term forecasts, Holt-Winters outperforms all the others. Although very simple, this method continues to be presented as the ideal one for seasonal *TSF*. On the other hand, for long term forecasts, and when an accurate forecast is mandatory, the use of the *GANN*'s systems may be a good alternative. One positive aspect of the presented system is that it works in a automatically way, with a minimum of human intervention, contrary to ARIMA, which requires the presence of experts.

In future work one intends to test the system robustness; explore the use of other topologies, like recurrent *ANNs*; expand the *GA* parameters; and using other filters, such as the Kalman's ones [11].

References

[1] E. Azoff. *Neural Network Time Series Forecasting of Financial Markets.* John Wiley & Sons, 1994.

[2] G. Box and G. Jenkins. *Time Series Analysis: Forecasting and Control.* Holden Day, San Francisco, USA, 1976.

[3] P. Cortez, M. Rocha, J. Machado, and J. Neves. A Neural Network Based Forecasting System. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, Western Australia, November 1995.

[4] J. Faraday and C. Chatfield. Times Series Forecasting with Neural Networks: A Case Study. Research Report, Statistics Group, University of Bath, UK, 1995.

[5] B. Freisleben and K. K. Ripper. Economic Forecasting Using Neural Networks. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, Western Australia, November 1995.

[6] S. Gallant. *Neural Network Learning and Expert Systems.* MIT Press, Cambridge, USA, 1993.

[7] J. Hanke and A. Reitsch. *A Business Forecasting.* Allyn and Bancon Publishing Company Inc., Massachussetts, USA, 1989.

[8] H. Kitano. Empirical Studies on the Speed of Convergence of Neural Network Training using Genetic Algorithms. In *Proceedings of the International Joint Conference on Neural Networks*, pages 397–404, IEEE, 1990.

[9] P. Kohen. Combining Genetic Algorithms and Neural Networks. Thesis for Master Science Degree, University of Tennessee, Knoxville, USA, 1994.

[10] S. Makridakis and S. Wheelwright. *Forecasting Methods for Management.* John Wiley & Sons, New York, fifth edition, 1989.

[11] I. Nabney, A. McLachlan, and D. Lowe. Practical methods of tracking of nonstationary time series applied to real-world data. In Applications and Science of Artificial Neural Networks II, *SPIE - The International Society for Optical Engineering*, USA, March 1996.

[12] J. Neves and P. Cortez. An Artificial Neural-Network Genetic Based Approach for Time Series Forecasting. In *Proceedings of IV Brazilian Symposium on Neural Networks*, Goiania, Brazil, 1997.

[13] J. Neves, P. Cortez, and M. Rocha. A Stopping Criterion for Genetic Algorithms based on the Principle of Minimum of Energy of a Chromosomes Population. Research Report, The Department of Informatics, University of Minho, Braga, Portugal, 1998.

[14] G. Papadourakis, G. Spanoudakis, and A. Gotsias. Application of Neural Networks in Short-Term Stock Price Forecasting. In *First International Workshop on Neural Networks in the Capital Markets*, London, UK, November 1993.

[15] L. Prechelt. PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Research Report, Fakultät für Informatik, Universität Karlsruhe Germany, 1994.

[16] M. Riedmiller and H. Braun. A Direct Adaptative Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proceeedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, USA, 1993.

[17] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, New Jersey, USA, 1995.

[18] E. Shoneburg. Price Prediction using Neural Networks: A Project Report. *Neurocomputing*, 2:17–27, 1990.

[19] D. White. GANNet: A Genetic Algoritm for Searching Topology and Weight Spaces in Neural Network Design. Dissertation, University of Maryland, 1993.

[20] Tang Z. and Fishwick F. Feed-forward Neural Nets as Models for Time Series Forecasting. Technical Report, University of Florida, 1991.