WSOM'2003 proceedings - Workshop on Self-Organizing Maps
Hibikino (Japan), 11-14 September 2003, pp. 340-345

Time series forecasting with SOM and local non-linear models - Application to the DAX30 index prediction

Simon Dablemont, Geoffroy Simon[1], Amaury Lendasse[2],
Alain Ruttiens[3], François Blayo[4], Michel Verleysen[1]

Universite catholique de Louvain
[1] DICE - Place du Levant, 3
[2] CESAME - Avenue G. Lemaitre, 4
B-1348 Louvain-la-Neuve
BELGIUM

[3]CBC Banque
Grand-Place, 5
B-1000 Bruxelles
BELGIUM

[4] PREFIGURE Espace G2C
Rue de Gerland, 75 B
F-69307 Lyon Cedex 07
FRANCE

[1]Tel : +32 10 47 25 40 [1]Fax : +32 10 47 25 98

simdable@compaqnet.be, simon@dice.ucl.ac.be, lendasse@auto.ucl.ac.be,
alain.ruttiens@cbc.be, francois.blayo@prefigure.com, verleysen@dice.ucl.ac.be

Keywords: time series forecasting, local models, financial prediction, returns

Abstract— A general method for time series forecasting is presented. Based on the splitting of the past dynamics into clusters, local models are built to capture the possible evolution of the series given the last known values. A probabilistic model is used to combine the local predictions. The method can be applied to any time series prediction problem, but is particularly suited to data showing non-linear dependencies and cluster effects, as many financial series do. The method is applied to the prediction of the returns of the DAX30 index.

1 Introduction

The analysis of financial time series is of primary importance in the economical world. This paper deals with a data-driven empirical analysis of financial time series; the goal is to obtain insights into the dynamic of the series and out-of-sample forecasting.

Forecasting future returns on assets is of obvious interest in empirical finance. If one was able to forecast tomorrow's return on an index with some degree of precision, one could use this information in an investment. Unfortunately, we are seldom able to generate a very accurate prediction for assets returns.

The Gaussian random walk paradigm (under the form of the geometric Wiener process) is the core of modeling of financial time series. Its robustness mostly suffice to keep it as the best foundation for any development in financial modeling, in addition to the fact that, on the long run, and with enough spaced out data, it is almost verified by the facts. Failures in its application are however well admitted on the (very) short term (market microstructure). To some extent, such failures are actually caused by the uniqueness of the modeling process.

The first breach in such unique process has appeared with two-regime or switching processes [1], which recognize that a return process could be originated by two different differential equations. But in such case, the switch is governed by an exogenous cause (for example in the case of exchange rates, the occurrence of a central bank decision to modify its leading interest rate or to organize a huge buying or selling of its currency through major banks).

Market practitioners, however, have always observed that financial markets can follow different behaviors over time, such as overreaction, mean reversion, etc, which look like succeeding each other with the passing time. Such observations would justify a rather fundamental divergence from the classic modeling foundations. That is, financial markets should not be modeled by a single process, but rather, by a succession of different processes, even in absence of the exogenous causes retained by existing switching process. Such a multiple switching process should imply, first, the determination of a limited number of competitive sub-processes, and secondly, the identification of the factor(s) causing the switch from one to another sub-processes. The resulting model should not be Markovian, and, without doubt, hard to determine. The aim of this paper is, as a first step, to at least empirically verify, with help of neural networks, that a multiple switching process leads to better short term forecasting.

Neural networks are used here as non-linear prediction tools, able to create clusters in the data; these clusters will correspond to the sub-processes, within which returns and volatility display different dynamic behavior. ARMA-GARCH [2, 3] are aimed to model

WSOM'2003 proceedings - Workshop on Self-Organizing Maps
Hibikino (Japan), 11-14 September 2003, pp. 340-345

the conditional mean and variance of a series. However, the series is still characterized by a single model even if it is adapted to transitions (switching processes).

In this paper we will present a forecasting method based on an empirical analysis of the past of the series. An originality of this method is that it does not make the assumption that a single model is able to capture the dynamics of the whole series. On the contrary, it splits the past of the series into clusters, and generates a specific local model for each of them. The local models are then combined in a probabilistic way, according to the distribution of the series in the past. This forecasting method can be applied to any time-series forecasting problem, but is particularly suited for data showing non-linear dependencies and cluster effects.

In the following of this paper, we will first describe how Self-Organizing Maps (SOM) can be applied to time series data (section 2), and briefly introduce the Radial-Basis Functions Networks that will be used as non-linear models (section 3). Then, we will describe the forecasting method itself (section 4), and illustrate its results on the DAX returns series (section 5).

2 Self-organization of regressors

In the context of time series prediction data have to be manipulated in an adequate way before using any kind of forecasting model. For example, using the last known data from a scalar time series is not informative enough to predict the next value with a sufficient accuracy. The common manipulation is to build a fixed-sized regressor from the data. The regressor contains some of the past data from the series, and is supposed to contain the most useful information to perform an accurate prediction. The problem of determining the most informative or optimal regressor for a given dataset is an interesting and open question [4], but exceeds the scope of this paper. In the following it is supposed that the optimal, or near-optimal regressor for a given dataset is known (for example through physical knowledge of the series or the use of statistical resampling model selection methods).

The order of a regressor is the number of past scalar values of the series used to build this vector. Transforming a scalar time series with n data into regressors of order p thus leads to $n-p+1$ data in a p-dimension space. The prediction method described below will make use of SOMs (Self-Organizing Maps [5]) built on such regressors.

3 RBF Networks

Radial Basis Function Networks (RBFN) are neural networks used in approximation and classification. They share with Multi-Layer Perceptrons the universal approximation property [6].

Classical RBF networks have their inputs fully connected to non-linear units in a single hidden layer. The output of a RBFN is a linear combination of the hidden units outputs. More precisely, the output is a weighted sum of Gaussian functions or kernels (i.e. the nonlinearities) applied to the inputs :

$$y = \sum_{i=1}^{I} \lambda_i \, e^{-\frac{||x-c_i||^2}{\sigma_i^2}}, \qquad (1)$$

where x is the input vector, y is the scalar output of the RBFN, $c_i,\ 1 \leq i \leq I$, are the centers of the I Gaussian kernels, $\sigma_i,\ 1 \leq i \leq I$, are their widths, and $\lambda_i,\ 1 \leq i \leq I$, their weights. Intuitively those last λ_i parameters represent the relative importance of each kernel in the output y.

As shown in equation 1, the RBF network has three sets of parameters $c_i,\ \sigma_i,\ \lambda_i,\ 1 \leq i \leq I$. One advantage of RBFN networks compared to other approximation models is that these three sets can be learned separately with suitable performances. Moreover the learning of the λ_i weights results from a linear system. A description of learning algorithms for RBF networks can be found in [7, 8].

4 The Forecasting Method

This section is devoted to a detailed description of the time series forecasting method. This method will first be sketched to give an intuition of how the forecasting is performed. Then each step of the method will be detailed. The simple case of a scalar time series will be considered. This case can be extended easily to non-scalar time series.

4.1 Method Description

The forecasting method is based on the "looking in the past" principle. To perform a prediction at time $t+1$, a regressor is built with past valued until time t. Then similar regressors are looked for in the past of the series, and the true next values after these regressors are considered. These are combined in a probabilistic way to build the prediction at time $t+1$.

First the original time series is transformed into a set of regressors of order p. This first dataset is quantized using the SOM algorithm.

Another application of the SOM is done on extended regressors. Indeed for $n-p$ of the regressors built on the series, we know the true next value. Therefore we build extended regressors containing the p past values, together with a supplementary one, the respective next value of the series. The SOM algorithm is also applied to these $p+1$-dimensional vectors.

WSOM'2003 proceedings - Workshop on Self-Organizing Maps
Hibikino (Japan), 11-14 September 2003, pp. 340-345

The relation between the first and the second sets of codewords issued from the SOM algorithms is encoded into a frequency table constructed empirically on the datasets.

In each of the classes determined by the second SOM, a RFBN is built to approximate the relation between the output value (the so-called "next value") and the p-dimensional input (the regressor).

Finally, the prediction at time $t+1$ is performed by combining the local models associated to classes in the second SOM, according to their frequencies with respect to the class considered in the first SOM.

4.2 Quantizing the *inputs*

Consider a scalar time series $x(t)$, $1 \leq t \leq n$ where $x(t)$ is the value at time t. As detailed in section 2, we consider here that the order p of an adequate regressor for this series is known. The original scalar series is then transformed into a series of $(p+q)$-dimensional *input* vectors $X(t)$ as follows:

$$\begin{aligned} X(t) &= (x(t-p+1), x(t-p+2), ..., x(t), \\ &\quad u(t-q+1), u(t-q+2), ..., u(t)), \end{aligned} \tag{2}$$

where $u(t)$ is a possible exogenous variable of interest for the prediction of $x(t)$. (Note that other exogenous variables could be added to $X(t)$ without changing the principale of the method.)

After those manipulations the SOM algorithm is applied to the *input* vectors $X(t)$. After convergence the algorithm gives an *IN* map of N_{in} codewords IN_i, $1 \leq i \leq N_{in}$ of dimension $p+q$; each codeword is associated to a class of $(p+q)$-dimensional *input* vectors.

4.3 Quantizing the *outputs*

For each input vector $X(t)$, $1 \leq t \leq n-p$, we know the next value of the series $x(t+1)$. New *output* vectors of dimension $p+q+1$ are formed according to

$$\begin{aligned} X'(t) &= (X(t), x(t+1)) \\ &= (x(t-p+1), ..., x(t), x(t+1), \\ &\quad u(t-q+1), ..., u(t)) \end{aligned} \tag{3}$$

Note that, by definition, there is a one-to-one relation between each *input* and each *output* vector.

Another SOM is applied to the *output* vectors $X'(t)$. This results in an OUT map of N_{out} $(p+1)$-dimensional codewords OUT_j, $1 \leq j \leq N_{out}$, and in N_{out} associated classes of *output* vectors.

4.4 Frequency table

The two sets of codewords of the maps *IN* and OUT only contain a static information. This information does not reflect completely the evolution of the time series. The idea here is thus to create a data structure that represents this dynamics, i.e. how each class of output vectors (including the value at time $t+1$) is associated to each class of input ones (without the $t+1$ value).

This structure is the frequency table $T(i,j)$, with $1 \leq i \leq N_{in}$, $1 \leq j \leq N_{out}$. Each element $T(i,j)$ of this table represents the proportion of *output* vectors that belongs to the j^{th} class of the OUT map while their corresponding *input* vectors belong to class i of the *IN* map. Those proportions are computed empirically for the given dataset and sum to one on each line of the table.

Intuitively the frequency table represents all the possible evolutions at a given time t together with the probability that they effectively happen.

4.5 Local RBFN models

When applied to the *output* vectors, the SOM algorithm provides N_{out} classes. In each of these classes a RBFN model is learned. Each RBFN model has $p+q$ inputs (the regressor and the exogenous variables) and one output (the prediction at time $t+1$). These models represent the local evolution of the time series, restricted to a specific class of regressors. The local information provided by these models will be used to predict the future evolution of the time series.

4.6 Forecasting

The relevant information has been extracted from the time series through the two Kohonen maps, the frequency table and the local RBF networks detailed in the previous sections. Having this information, it is now possible to perform the forecasting itself. At each time t, the goal is to estimate the value of $x(t+1)$, denoted $\hat{x}(t+1)$, at the following time step.

First the *input* at time t is built, leading to $X(t)$. This vector is presented to the *IN* map, and the nearest codeword IN_k is identified ($1 \leq k \leq N_{in}$).

In the frequency table, in the k^{th} line, there are some columns corresponding to classes of the OUT map for which the proportions are non zero. That means that those columns represent possible evolutions for the considered data $X(t)$, since $X(t)$ as the same shape than data in the k^{th} class.

For each of those potential evolutions, the respective RBF networks are considered (remember that one RBF has been built for each class in the OUT map). For each of them, a local prediction $\hat{x}_j(t+1)$ is obtained ($1 \leq j \leq N_{out}$). The final prediction is a weighted sum of the different local predictions, the weights being the proportions recorded in the frequency table:

$$\hat{x}(t+1) = \sum_{j=1}^{N_{out}} T(k,j)\hat{x}_j(t+1) \tag{4}$$

WSOM'2003 proceedings - Workshop on Self-Organizing Maps
Hibikino (Japan), 11-14 September 2003, pp. 340-345

5 Experimental Results

5.1 Data

The example presented here deals with the German DAX30 index time series for the period starting on January 02, 1992 and ending on January 08, 2003; it includes the implied volatility VDAX time series.

In a few words, implied volatility reflects the expected average volatility for the market participants, for example over the life of an option. Most market practitioners consider this measure of volatility as the most trustworthy forecast of the short-term volatility of the underlying asset. In an option pricing framework, the implied volatility is defined as the volatility that, when plugged in the option pricing formula, equates the theoretical price of the option with the observed market price.

On top of Fig. 1 and Fig. 2 we can see the respective evolution of the DAX and VDAX time series. The bottom of those two figures are the returns of the DAX and VDAX respectively.

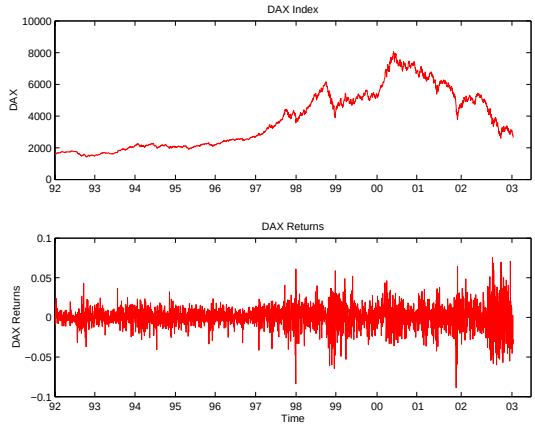

Figure 1: Above : the DAX time series from 2^{nd} January 1992 to 8^{th} January 2003; below : the corresponding DAX returns time series (same period).

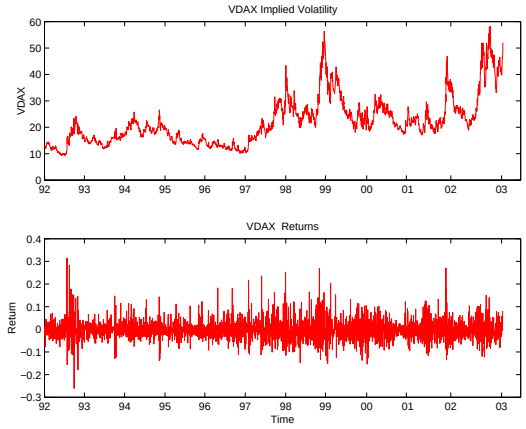

Figure 2: Above : the VDAX time series evolution (same period as DAX); below : the corresponding VDAX returns time series (also same period).

5.2 Methodology

This section begins with a few precisions over the data considered in this experimental example.

To test the performances of the method data are divided in three subsets : the learning, validation and test subsets. Usually a proportion of respectively 80%, 10% and 10% is used. The learning set is used to learn the SOM resulting in the *IN* and *OUT* maps. Those learning data divided into clusters are used to learn the local RBF models. The validation set is used to optimize some meta-parameters (size of the maps, numbers of neurons in the RBFs, etc.) within a selected range or set. The test set is used to evaluate the performances of the forecasting method.

Vectors $X(t)$ and $X'(t)$ considered here are composed according to equations 2 and 3, with $x(t)$ the DAX returns and $u(t)$ the VDAX returns series.

The size of the *IN* and *OUT* maps have been chosen according to the Mean Absolute Error criterion computed over the validation set V_{set} as :

$$\text{MAE} = \frac{1}{\#V_{set}} \sum_{t=0}^{\#V_{set}-1} || \, x(t+1) - \hat{x}(t+1) \, || \quad (5)$$

Another criteria that will be used is the Normalized Mean Absolute Error defined as :

$$\text{NMAE} = \frac{\sum_{t=0}^{\#V_{set}-1} || \, x(t+1) - \hat{x}(t+1) \, ||}{\sum_{t=0}^{\#V_{set}-1} || \, x(t+1) - \text{E}[\text{x(t+1)}] \, ||} \quad (6)$$

with E[.] the statistical expectation. The intuitive interpretation of this second criterion is straightforward : we compare the obtained MAE to the MAE that we would obtain if the orignal time series $x(t)$ was modelised by a random walk. Since the values $x(t)$ are here returns of the price of a market value, under the assumption of random walk, the expectation is zero. Therefore equation 6 simplifies in :

$$\text{NMAE} = \frac{\sum_{\text{t=0}}^{\#\text{V}_{\text{set}}-1} || \, \text{x(t+1)} - \hat{\text{x}}\text{(t+1)} \, ||}{\sum_{\text{t=0}}^{\#\text{V}_{\text{set}}-1} || \, \text{x(t+1)} \, ||}. \quad (7)$$

Intuitively, a NMAE value of 1 would mean that the model does not perform better than a random walk modeling while a value near 0 value indicates a quite perfect modeling.

Both those error criteria can be computed over each dataset. In pratice, there are used on the validation and test sets, to choose meta-parameters and assess the performances respectively.

Indeed, various square SOM sizes were tried with an increasing number of units. The model selected is the one having the lowest MAE value on the validation set. For both the *IN* and *OUT* datasets, 4x4 maps have been chosen. Fig. 3 shows the $X(t)$ vectors in their respective classes after training of the *IN* map. Fig. 4 shows the corresponding code vectors for each class.

Fig. 5 and Fig. 6 show respectively the $X'(t)$ vectors in the classes obtained for the OUT map and the corresponding code vectors for the OUT map classes.

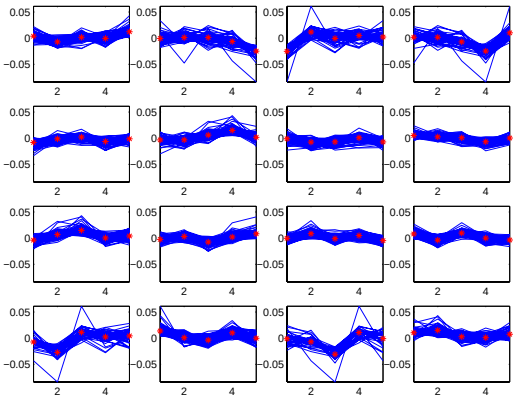

Figure 3: The data of the training data sample in their respective classes for the IN map; those classes were obtained using the SOM algorithm.

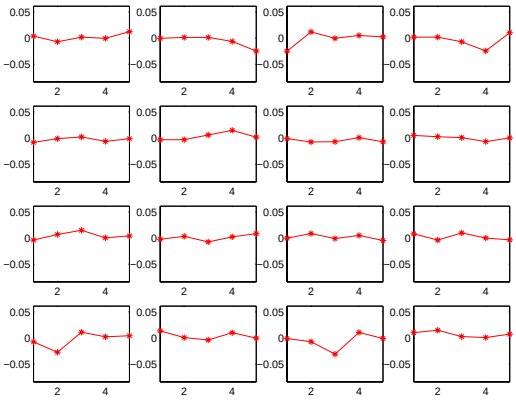

Figure 4: The code vectors of the IN map obtained by the SOM algorithm (training sample).

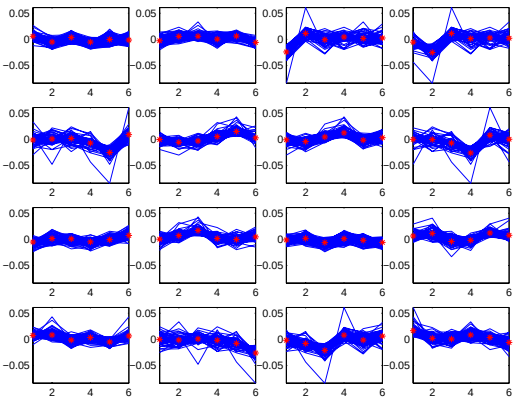

Figure 5: The data of the training data sample in their respective classes for the OUT map after application of the SOM algorithm.

Each local RBF model may have a different number of Gaussian kernels. Here again, the validation set has been used to determine the optimal number of Gaussian kernels in each RBF model. Those optimal numbers are comprised between 5 and 8.

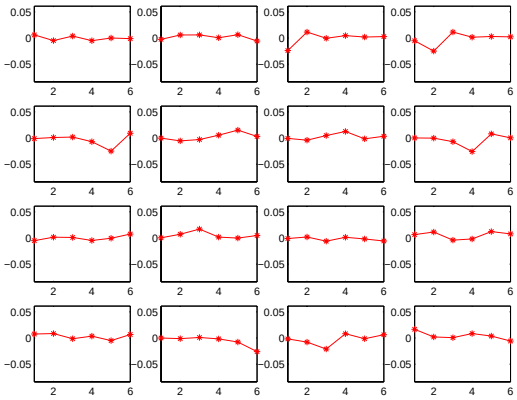

Figure 6: The code vectors of the OUT map obtained by the SOM algorithm (training sample).

Table 1 presents a part of the whole 16x16 frequency table; lines correspond to IN_i codewords, and columns to OUT_j ones. This table reads as follows: none of the possible evolutions for a data looking like the ones in the first class of the IN map are present in classes 1 to 8 of the OUT map (in fact 95% of those are in class number 13); 24% of the possible evolutions for a data looking like the ones in class two of the IN map are in class two of the OUT map, 13% belongs to class four of the OUT map, 3% to class five, etc.

0	0	0	0	0	0	0	0	$\cdots$
0	.24	0	.13	.03	.13	0	.01	$\cdots$
0	0	.95	0	0	.03	0	0	$\cdots$
.82	0	0	0	0	0	0	0	$\cdots$
0	0	0	.09	0	0	0	.91	$\cdots$
0	.03	0	0	.79	0	.06	0	$\cdots$
0	.01	0	.6	.01	0	.09	0	$\cdots$
.15	0	.03	.06	.01	0	.21	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 1: Extract of the frequency table: upper left corner of the whole 16x16 table.

5.3 Prediction results

This section presents a comparison of the NMAE results obtained on the DAX30 index series by the method presented in this paper, and by two reference methods. The first reference method is a global RBF network, trained on the same series. Indeed it is known that global neural networks (RBFN models) perform equivalently as the best econometrics models [9, 10, 11]. Various global RBF networks have been learned on the whole learning dataset, and the best one has been chosen according to the MAE criterion over the validation set. The performances presented here are computed over the test set. The second reference model is the random walk that has, of course, a

NMAE equal to 1. Table 2 presents a comparison between the results obtained with a randow walk model, a global RBF model and our SOMs plus local RBF models.

	Random Walk	Global RBF	SOMs + RBFs
NMAE	1	0.674	0.454

Table 2: Forecasting results (288 observations in test set).

It can be noticed that better predictions are obtained by the SOMs together with the local RBF models, compared to the two reference methods. As argued in the introduction, this is probably a consequence of the data splitting into clusters, according to the multiple switching process argument.

6 Conclusion

In this paper, a general forecasting method valid for any series of data has been presented. The method is particularly suited to series showing different dynamics over time, therefore benefiting from the clustering of their past. The SOM algorithm is used to create cluster containing local information. This local information is in turn modeled using local RBF networks. A global prediction is obtained by combining the local predictions obtained with the various RBF models.

The method has also been tested on a specific financial time series, namely the daily evolution of the German DAX30 index over a period of ten years. The dynamics of this time series and of its related implied volatility are modeled by the SOMs plus RBFs method. The results obtained are promising, and largely exceed those obtained under the random walk hypothesis, and by global neural network models.

Acknowledgements

G. Simon is funded by the Belgian F.R.I.A.

M. Verleysen is Senior Research Associate of the Belgian F.N.R.S.

The work of A. Lendasse is supported by the Interuniversity Attraction Poles (IAP), initiated by the Belgian Federal State, Ministry of Sciences, Technologies and Culture. The scientific responsibility rests with the authors.

References

[1] F. Diebold, J.-H. Lee, G. Weinbach, "Regime switching with time-varying transition probabilities, in non stationary time series analysis and cointegration", *C. Hargreaves editors*, Oxford University Press, pp 283-302, 1994.

[2] J. Hamilton, "Time Series Analysis",*Princeton University Press*, 1994.

[3] R.F. Engel, "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation",*Econometrica*, Vol. 50, pp. 987-1007, 1982.

[4] M. Verleysen, E. de Bodt, A. Lendasse, "Forecasting financial time series through intrinsic dimension estimation and non-linear data projection", in Proc. of International Work-conference on Artificial and Natural Neural networks (IWANN'99), *Springer-Verlag Lecture Notes in Computer Science*, n 1607, pp. II596-II605, June 1999.

[5] T. Kohonen, "Self-Organizing Maps", *Springer Series in Information Sciences*, Vol. 30, 2nd edition, 1997.

[6] S. Haykin, "Neural Networks - A comprehensive foundation", *Prentice Hall*, 2nd edition, 1999.

[7] M. J. Orr, "Optimising the Widths of Radial Basis Functions", in Proc. of Vth Brazilian Symposium on Neural Networks, Belo Horizonte, Brazil, December 1998.

[8] N. Benoudjit, C. Archambeau, A. Lendasse, J. Lee, M. Verleysen, "Width optimization of the Gaussian kernels in Radial Basis Function Networks", in Proc. of 10th European Symposium on Artificial Neural Networks (ESANN'2002), *d-side*, Brussels, Belgium, April 2002.

[9] A.P. Blake, G. Kapetanios, "A radial basis function artificial neural network test for ARCH", *Economics Letters*, n 69, pp 15-23, 2000.

[10] P.H. Franses, D. van Dijk, "Nonlinear Time series models in Empirical Finance",*Cambridge University Press*, 2000.

[11] A.P. Refenes, A.N. Burgess, Y. Bentz, "Neural Networks in Financial Engineering: A Study in Methodology",*IEEE transactions on Neural networks*, Vol. 8, n. 6, pp.1222-1267, November 1997.