Processing Techniques for improving Regularization Networks

J.M. Górriz
Department of Electronics
University of Cádiz
Algeciras, Cádiz, Spain
email: juanmanuel.gorriz@uca.es

Carlos G. Puntonet and M. Salmerón
Department of Architecture and Computer Tech.
University of Granada
Granada, Spain
email: carlos@atc.ugr.es

Abstract— **In this paper we summarize various preprocessing techniques including a new method for volatile time series forecasting. These methods improve the performance of Regularization Networks i.e. using Independent Component Analysis (ICA) algorithms and filtering as preprocessing tools. The preprocessed data is introduced into a Artificial Neural Network (ANN) based on radial basis functions (RBFs) and the prediction results are compared with the ones we get without these preprocessing tools, with the high computational effort method based on parallel neural networks (PNN) and with the Principal Component Analysis (PCA) technique.**

I. Introduction

In the history of research of the forecasting problem one can extract various relevant periods such as the following mentioned: a possible solution to this problem was described by Box and Jenkins [1], who developed a time-series forecasting analysis technique based on linear systems. Basically the procedure consisted of suppressing the non-seasonality of the series, performing parameter analysis, which measures time-series correlation, and selecting the model that best fits the data set (a specific order ARIMA model). But in real systems, non-linear and stochastic phenomena crop up, and then time series dynamics cannot be described exactly using classical models. ANNs have improved results in forecasting by detecting the non-linear nature of the data. ANNs based on RBFs allow a better forecasting adjustment; they implement local approximations to non-linear functions, minimizing the mean square error to achieve the adjustment of neural parameters. For example, Platt's algorithm [2], Resource Allocating Network (RAN), consisted of neural network size control, reducing the computational time cost associated with computing the optimum weights in perceptron networks.

Matrix decomposition techniques have been used as an improvement on Platt's model [3]. For example, Singular Value Decomposition (SVD) with pivoting QR decomposition selects the most relevant data in the input space avoiding non-relevant information processing (NAPA-PRED "Neural model with Automatic Parameter Adjustment for PREDiction"). NAPA-PRED also includes neural pruning [4]. An improved version of this algorithm can be found in [5] based on Support Vector Machine philosophy.

The next step was to include exogenous information in these models. There are some choices in order to do that; we can use the forecasting model used in [6] which gives good results but with computational time and complexity cost; Principal Component Analysis (PCA) is a well-established tool in Finance. It was already proved [3] that prediction results can be improved using the PCA technique. This method linear transform the observed signal into principal components which are uncorrelated (features), giving projections of the data in the direction of the maximum variance [7]. PCA algorithms use only second order statistical information; Finally, in [8] we can discover interesting structure in finance using the new signal-processing tool Independent Component Analysis (ICA). ICA finds statistically independent components using higher order statistical information for blind source separation ([9], [10]). This new technique may use Entropy (Bell and Sejnowski 1995, [11]), Contrast functions based on Information Theory (Comon 1994, [12]), Mutual Information (Amari, Cichocki y Yang 1996, [13]) or geometric considerations in data distribution spaces (Carlos G. Puntonet 1994 [14],[15], [16]), etc. Forecasting and analyzing financial time series using ICA can contributes to a better understanding and prediction of financial markets ([6],[8]).

There exist numerous forecasting applications in time series forecasting, as analyzed in [17]: signal statistical preprocessing and communications, industrial control processing, econometrics, meteorology, physics, biology, medicine, oceanography, seismology, astronomy and psychology.

II. Neural Regularization networks based on RBFs

Because of their inherent non-linear processing and learning capabilities, ANNs (Artificial Neural Networks) have been proposed to solve prediction problems. An excellent survey of NN forecasting applications is to be found in [18]. There it is claimed that neural nets often offer better performance, especially for difficult time series than are hard to deal with classical models such as ARIMA models [1]. One of the simplest, but also most powerful, ANN models is the Radial Basis Function (RBF) network model. This consists of locally-receptive activation functions (or neurons) implemented by

means of gaussian functions [19]. In mathematical terms, we have

$$o(\mathbf{x}) = \sum_{i=1}^{N} o_i(\mathbf{x}) = \sum_{i=1}^{N} h_i \exp\left\{-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{\sigma_i^2}\right\} \quad (1)$$

where N denotes the number of nodes (RBFs) used; $o_i(\mathbf{x})$ gives the output computed by the i-th RBF for the input vector $\mathbf{x}$ as an exponential transformation over the norm that measures the distance between $\mathbf{x}$ and the RBF center $\mathbf{c}_i$, whereas σ_i denotes the *radius* that controls the locality degree of the corresponding i-th gaussian response. The global output $o(\mathbf{x})$ of the neural network is, as can be seen, a linear aggregate or combination of the individual outputs, weighted by the real coefficients h_i.

In most RBF network applications, the coefficients h_i are determined after setting up the location and radius for each of the N nodes. The locations can be set, as in [19], by a *clustering* algorithm, such as the *K-means algorithm* [20], and the radius is usually set after taking into account considerations on RBFs close to the one being configured. The adjustment of the linear expansion coefficients can be done using recursive methods for linear least squares problems [21], [22] or the new method based on Regularization-VC Theory presented in [5] characterized by a suitable regularization term which enforces flatness in the input space, so that the actual risk functional over a training data set is minimized, and determined by the previously set parameter values [23]. Recursive specification allows for real-time implementations, but the questions arises of whether or not we are using a simplified-enough neural network, and this is a question we will try to address using matrix techniques over the data processed by the neural net.

In the particular context of the RBF networks, the mapping of a time series prediction problem to the network is performed setting up the input as past values (consecutive ones, in a first approximation) of the time series. The output is viewed as a prediction for the future value that we want to estimate, and the computed error between the desired and network- estimated value is used to adjust parameters in the network.

III. Fundamentals of PCA

PCA is probably the oldest and most popular technique in multivariate data analysis. It transforms the data space into a feature space, in such a way, that the new data space is represented by a reduced number of "effective" features. Its main advantages lie in the low computational effort and the algebraic procedure.

Given a $n \times N$ data set $\mathbf{x}$,where N is the sample size, PCA tries to find a linear transformation $\tilde{\mathbf{x}} = \mathbf{W}^T\mathbf{x}$ into a new orthogonal basis $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$ $m \leq n$ such that:

$$Cov(\tilde{\mathbf{x}}) = E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\} = \mathbf{W}^T Cov(\mathbf{x})\mathbf{W} = \mathbf{\Lambda} \quad (2)$$

where $\lambda = diag(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix. Hence PCA, decorrelates the vector $\mathbf{x}$ as all off-diagonal elements in the covariance matrix of the transformed vector vanish. In addition to the transformation presented in equation 2 (Karhunen-Loeve transformation when $m = n$) the variances of the transformed vectors $\tilde{\mathbf{x}}$ can be normalized to one using:

$$\tilde{\mathbf{x}} = \mathbf{W}_z^T\mathbf{x} \quad (3)$$

with the sphering matrix $\mathbf{W}_z = \left[\frac{\mathbf{w}_1}{\sqrt{\lambda_1}}, \ldots, \frac{\mathbf{w}_m}{\sqrt{\lambda_m}}\right]$ It has been shown that prediction results can be improved using this technique in [3].

IV. ICA in general

ICA has been used as a solution of the blind source separation problem [10], denoting the process of taking a set of measured signal in a vector, $\mathbf{x}$, and extracting from them a new set of statistically independent components (ICs) in a vector $\mathbf{y}$. In the basic ICA each component of the vector $\mathbf{x}$ is a linear instantaneous mixture of independent source signals in a vector $\mathbf{s}$ with some unknown deterministic mixing coefficients:

$$x_i = \sum_{i=1}^{N} a_{ij} s_j \quad (4)$$

Due to the nature of the mixing model we are able to estimate the original sources $\tilde{s}_i$ and the un-mixing weights b_{ij} applying i.e. ICA algorithms based on higher order statistics such as cumulants.

$$\tilde{s}_i = \sum_{i=1}^{N} b_{ij} x_j \quad (5)$$

Using vector-matrix notation and defining a time series vector $\mathbf{x} = (x_1, \ldots, x_n)^T$, $\mathbf{s}$, $\tilde{\mathbf{s}}$ and the matrices $\mathbf{A} = \{a_{ij}\}$ and $\mathbf{B} = \{b_{ij}\}$, we can write the overall process as:

$$\tilde{\mathbf{s}} = \mathbf{B}\mathbf{x} = \mathbf{B}\mathbf{A}\mathbf{s} = \mathbf{G}\mathbf{s} \quad (6)$$

where we define $\mathbf{G}$ as the overall transfer matrix. The estimated original sources will be, under some conditions included in Darmois-Skitovich theorem (chapter 1 in [24]), a permuted and scaled version of the original ones. Thus, in general, it is only possible to find $\mathbf{G}$ such that $\mathbf{G} = \mathbf{PD}$ where $\mathbf{P}$ is a permutation matrix and $\mathbf{D}$ is a diagonal scaling matrix.

In financial time series this model (equation 4) can be applied to the stock series where there are some underlying factors like seasonal variations or economic events that affect the stock time series simultaneously and can be assumed to be quite independent [25].

V. ICA operation and general preprocessing stage

The main goal, in the preprocessing step, is to find non-volatile time series including exogenous information i.e. financial time series, easier to predict using ANNs based on RBFs. This is due to smoothed nature of the kernel functions used in regression over multidimensional domains [26]. We propose the following Preprocessing Steps

- After Whitening the set of time series $\{x_i\}_{i=1}^n$(subtract the mean of each time series and removing the second order statistic effect or covariance matrix diagonalization process)

- we apply some ICA algorithm to estimate the original sources s_i and the mixing matrix $\mathbf{A}$ in equation 4. Each IC has information of the stock set weighted by the components of the mixing matrix. In particular, we use a well known Fast ICA algorithm based in High Order Statistics proposed in by A. Hyvarinen in [27] with the code available in:
  http : //www.maths.lth.se/help/R/.R/library/fastICA/html/fastICA.html
  The un-mixing matrix is calculated according the measure of non-gaussianity via the NegEntropy function:

$$\mathbf{J(y)} = E[\mathbf{G(y)}] - E[\mathbf{G(v)}] \qquad \mathbf{v} \in \mathbf{N}(0,1) \tag{7}$$

  where $\mathbf{G}$ is a function with different shape choices.
  Once convergence, which is related with the measure of non-gaussianity, is reached , we estimate the mixing matrix $\mathbf{A}$.
- Filtering.
  1) We neglect non-relevant components in the mixing matrix $\mathbf{A}$ according to their absolute value. We consider the rows $\mathbf{A}_i$ in matrix $\mathbf{A}$ as vectors and calculate the mean Frobenius Norm [1] of each one. Only the components bigger than mean Frobenius Norm will be considered. This is the principal preprocessing step using PCA tool, in this case is not enough.

$$\tilde{\mathbf{A}} = \mathbf{Z} \cdot \mathbf{A} \tag{8}$$

  where $\{\mathbf{Z}\}_{ij} = [\{\mathbf{A}\}_{ij} > \frac{||\mathbf{A}_i||_{Fr}}{n}]$
  2) We apply a low band pass filter to the ICs. We can choose between two choices:
     a) The well know FFT domain with slightly bigger computational cost. This election implies $\mathcal{H}(f)$ low pass filter design (for both positive and negative frequencies), multiply the FFT output by this filter function, and then do an inverse FFT to get back a filtered data set in time domain. The particular situation we consider is that there is some underlying, uncorrupted signal we want to measure $\mathbf{x}$. The measurement process is imperfect and what comes out of observed signals is a corrupted signal $\hat{\mathbf{x}}$. The relation between these signals is straight forward in frequency domain:

$$\mathcal{X}(f) = \mathcal{R}(f)\hat{\mathcal{X}} \tag{9}$$

        Modelling $\mathcal{R}(f)$ we can obtain filtered ICA signals to feed artificial Neural Networks.
     b) The well-adapted for data smoothing Savitsky-Golay smoothing filter [28] for two reasons: a)ours is a real-time application for which we must process a continuous data stream and wish to output filtered values at the same rate we receive raw data and b) the quantity of data to be processed is so large that we just can afford only a very small number of floating operations on each data point. This filter is also call Least-Squares [29] or DISPO [30]. These filters derive from a particular formulation of the data smoothing problem in the time domain and their goal is to find filter coefficients c_n in the expression:

$$\bar{s}_i = \sum_{n=-n_L}^{n_R} c_n s_{i+n} \tag{10}$$

        where $\{s_{i+n}\}$ represent the values for the ICs in a window of length $n_L + n_R + 1$ centered on i and $\tilde{s}_i$ is the filter output (the smoothed ICs), preserving higher moments [31].
        For each point s_i we least-squares fit a m order polynomial for all $n_L + n_R + 1$ points in the moving window and then set $\tilde{s}_i$ to the value of that polynomial at position i. As shown in [31] there are a set of coefficients for which equation 10 accomplishes the process of polynomial least-squares fitting inside a moving window:

$$c_n = \{(\mathbf{M}^T \cdot \mathbf{M})^{-1}(\mathbf{M}^T \cdot \mathbf{e}_n)\}_0 \sum_{j=0}^{m} \{(\mathbf{M}^T \cdot \mathbf{M})^{-1}\}_{0j} \cdot n^j \tag{11}$$

        where $\{\mathbf{M}\}_{ij} = i^j$, $i = -n_L, \ldots, n_R$, $j = 0, \ldots, m$, and $\mathbf{e}_n$ is the unit vector with $-n_L < n < n_R$.

- Reconstructing the original signals using the smoothed ICs and filtered $\tilde{\mathbf{A}}$ matrix we get a less high frequency variance version including exogenous influence of the old ones. We can write using equations 8 and 7.

$$\mathbf{x} = \tilde{\mathbf{A}} \cdot \bar{\mathbf{s}} \tag{12}$$

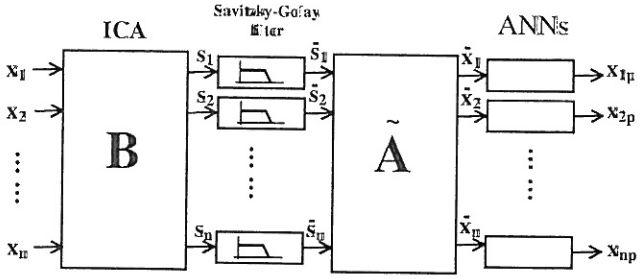

Fig. 1. Schematic representation of prediction and filtering process.

VI. Multidimensional Regularization Networks

We use an Multidimensional RN based on RBFs to forecast a series x_i i.e. from the Stock Exchange, building a forecasting function $\mathbf{P}$ with the help of Improved NAPA-PRED algorithm [5], for one of the set of signals $\{x_1, \ldots, x_n\}$. As shown in

[1] Let $x \in \mathbf{R}^n$ Frobenius Norm $||\mathbf{x}||_{Fr} \equiv \sqrt{\sum_{i=1}^{n} x_i^2}$

section II the individual forecasting function can be expressed in term of RBFs as:

$$\mathbf{F}(\mathbf{x}) = \sum_{i=1}^{N} f_i(\mathbf{x}) = \sum_{i=1}^{N} h_i exp\left\{-\frac{||\mathbf{x}-\mathbf{c}_i||^2}{\mathbf{r}_i^2}\right\} \qquad (13)$$

where $\mathbf{x}$ is a p-dimensional vector input at time $\mathbf{t}$, $\mathbf{N}$ is the number of neurons (RBFs) , $\mathbf{f}_i$ is the output for each neuron i-th , $\mathbf{c}_i$ is the centers of i-th neuron which controls the situation of local space of this cell and $\mathbf{r}_i$ is the radius of the i-th neuron. The global output is a linear combination of the individual output for each neuron with the weight of $\mathbf{h}_i$. Thus we are using a method for moving beyond the linearity where the core idea is to augment/replace the vector input $\mathbf{x}$ with additional variables, which are transformations of $\mathbf{x}$, and then use linear models in this new space of derived input features. RBFs are one of the most popular kernel methods for regression over the domain $\mathbf{R}^n$ and consist on fitting a different but simple model at each query point $\mathbf{c}_i$ using those observations close to this target point in order to get a smoothed function. This localization is achieved via a weighting function or kernel $\mathbf{f}_i$.

The preprocessing step suggested in section V is necessary due to the dynamic of the series and it will be shown that results improve sensitively. Thus we will use as input series the ones we got in equation 12.

The MRN model in [5] is archived applying/extending regularization concept to extra series, including a row of neurons (equation 13) for each series, and weight these values by a factor $\mathbf{b}_{ij}$. Finally, the global smoothed function for the stock $\mathbf{j}$ is defined as:

$$\mathbf{P}_j(\mathbf{x}) = \sum_{i=1}^{S} \mathbf{b}_{ij} F_i(x_i, j) \qquad (14)$$

where $\mathbf{F}_i$ is the smoothed function of each series, $\mathbf{S}$ is the number of input series and $\mathbf{b}_{ij}$ are the weights for j-stock forecasting. Obviously one of these weight factors must be relevant in this linear fit ($\mathbf{b}_{jj} \sim 1$, or auto weight factor).

Matrix notation can be used to include the set of forecasts in an S-dimensional vector $\mathbf{P}$ ($\mathbf{B}$ in figure2):

$$\mathbf{P}(\mathbf{x}) = diag(\mathbf{B} \cdot \mathbf{F}(\mathbf{x})) \qquad (15)$$

where $\mathbf{F} = (\mathbf{F}_1, \ldots, \mathbf{F}_S)$ is an $S \times S$ matrix with $F_i \in R^S$ and B is an $S \times S$ weight matrix. The operator $diag$ extracts the main diagonal. Because the number of neurons and the input space dimension increases in prediction function (equation15), we must control them (*parsimony*) to reduce curse of dimensionality problem and the well known overfitting problem.

VII. Simulations

We work with indexes of different Spanish banks and companies during the same period to investigate the effectiveness of ICA techniques for financial time series (Figure 2). We have specifically focussed on the IBEX35 from Spanish stock, which we consider the most representative sample of the Spanish stock movements, using closing prices in 2000.

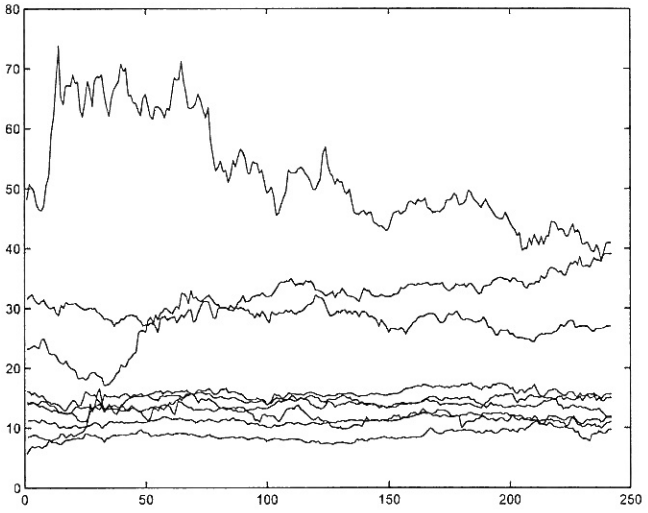

Fig. 2. Set of Stock Series.

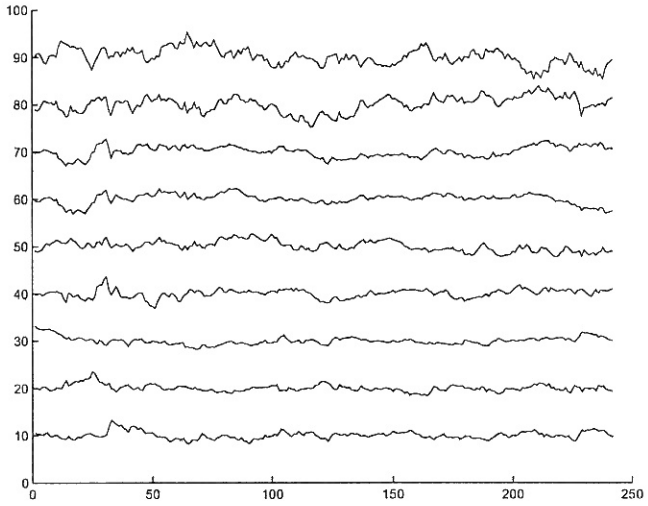

Fig. 3. ICs for the stock series.

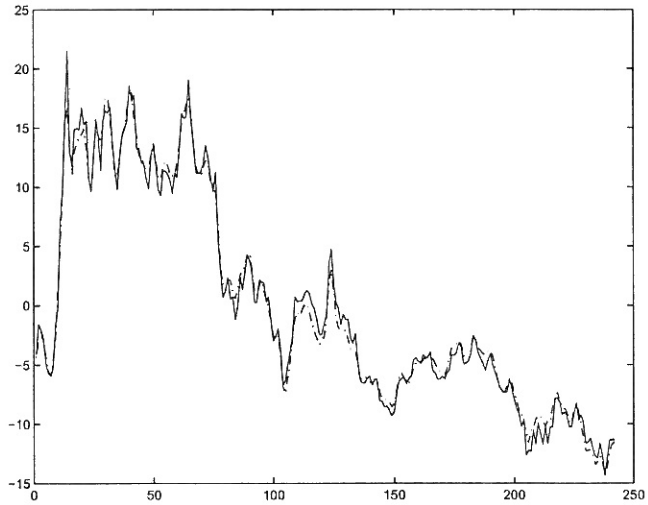

Fig. 4. Real Series from ICA reconstruction (scaled old version)(line) and Preprocessed Real Series (-.).

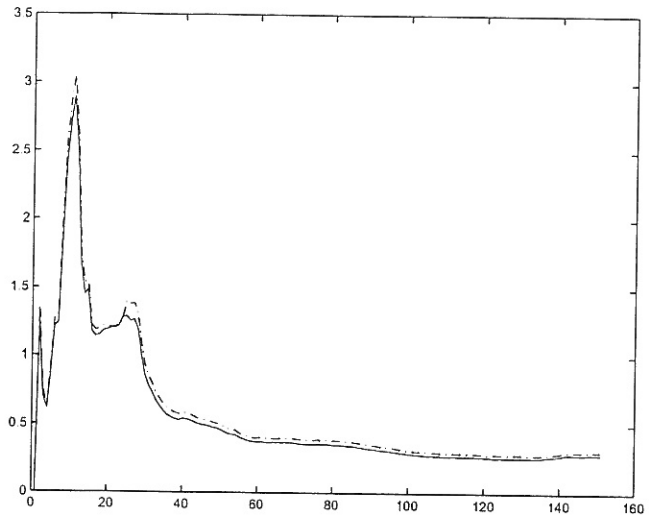

Fig. 5. NRMSE evolution for ANN with ICA+SG (line) and without (-.).

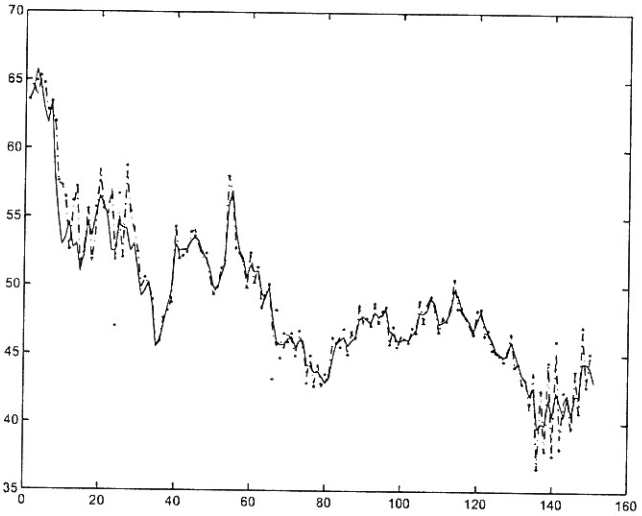

Fig. 6. Real Series(line) Predicted Series with ICA+SG(-.) Predicted Series without preprocessing (.).

We considered the closing prices of Bankinter for prediction and 8 more stock of different Spanish companies (ACS, Aguas de Barcelona, Banco Popular, Banco Santander, BBVA, Dragados, Carrefour and Amadeus). Each time series includes 200 points corresponding to selling days (quoting days).

We performed ICA on the Stock returns using the ICA algorithm presented in section V assuming that the number of stocks equals the number of sources supplied to the mixing model. This algorithm whiten the raw data as the first step. The ICs are shown in Figure 3. These ICs represents independent and different underlying factors like seasonal variations or economic events that affect the stock time series simultaneously. Via the rows of $\mathbf{A}$ we can reconstruct the original signals with the help of these ICs i.e. Bankinter stock after we preprocess the raw data:

- Frobenius Filtering: the original mixing matrix[2]:

$$\bar{\mathbf{A}} = \begin{pmatrix} 0.3262 & -0.2294 & 0.1650 & 0.0360 \\ \mathbf{-0.2766} & \mathbf{1.9506} & \mathbf{-0.3282} & \mathbf{4.7013} \\ 0.3305 & -0.1884 & 0.0454 & -0.2316 \end{pmatrix} \tag{16}$$

is transformed to:

$$\tilde{\mathbf{A}} = \begin{pmatrix} 0.3262 & -0.2294 & 0.1650 & 0.0360 \\ \mathbf{0} & \mathbf{1.9506} & \mathbf{0} & \mathbf{4.7013} \\ 0.3305 & -0.1884 & 0.0454 & -0.2316 \end{pmatrix} \tag{17}$$

thus we neglect the influence of two ICs on the original 5^{th} stock. Thus only a few ICs contribute to most of the movements in the stock returns and each IC contributes to a level change depending its amplitude transient [8].

- We do a polynomial fit in the ICs using the library supported by *MatLab* and the reconstruction of the selected stock (Figure 6) to supply the ANN.

In Figures 4,5 and 6 we show the results we got using our ANN with the above mentioned algorithm. We can say that prediction is better with the preprocessing step avoiding the disturbing peaks or convergence problems in prediction. As is shown in Figure 5 NRMSE is always lower using techniques we discussed in section VI.

Finally we compare the ICA method with the ones presented in [6], [3]. We show in Figure 7 how NRMSE are similar when data set is large enough. At the beginning statistical estimators are not valid in ICA algorithms. The advantage of ICA method is lower computational time. ICA methods also improve the previous technique for exogenous data inclusion using PCA. As we mentioned in previous sections PCA is based in second order statistics thus this method can't extract relevant information for the ANN operation.

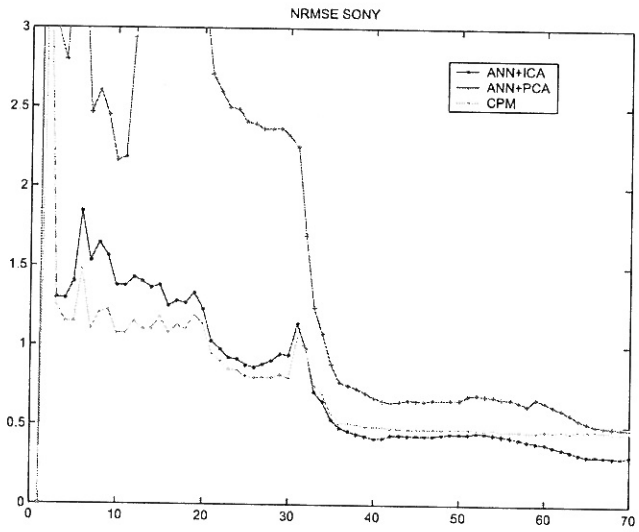

Fig. 7. NRMSE evolution ICA method vs PNN and vs ANN with PCA (Principal Component Analysis) for some indexes.

[2]we show and select the relevant part $\bar{\mathbf{A}} \subset \mathbf{A}$of the row corresponding to Bakinter Stock.

VIII. Conclusions

In this paper we showed that prediction results can be improved with the help of techniques such as ICA, PCA or MRN. ICA algorithms decompose a set of 9 returns from the stock into independent components which fall in two categories: a) large components responsible of the major changes in level prices and b) small fluctuations. Smoothing this components and neglecting the non-relevant ones we can reconstruct a new version of the Stock easier to predict. Moreover we describe a new filtering method to volatile time series that are supplied to ANNs (following INAPA-PRED algorithm to update parameters) in real-time applications. On the other hand, PCA technique, well established in numerous fields, is other possible way. PCA is probably the oldest and most common technique in multivariate data analysis. It intends to transform the data space into a feature space in such a way that the data set may be represented by a reduce number of "effective" features. Matrix spectral properties and Eigenvalues decomposition have special relevance in the later transformation. The basic idea is consider a large set of input variables and transforms it to a new set of variables containing without loss of much information. This is possible due to that very often, multivariate data contains redundant information of 2^{nd} order thus in this method of data compression using the maximum variance principle, PCA is basically regarded as a standard statistical technique. Moreover, PCA can be used to include exogenous information [3], in other words, we can increase *input space dimension* using variables related to the original series. The MRN forecasting model for time-series is characterized by:

- The enclosing of external information. We avoid pre-processing and data contamination applying by ICA and PCA for limited data sets or sudden new shocks. These techniques can be included in CPM under better conditions. Series are introduced into the net directly.
- The forecasting results are improved using hybrid techniques like GA.
- The possibility of implementing in parallel programming languages (i.e. PVM); and the improved performance and lower computational time achieved using a parallel neural network.

References

[1] G. Box, G. Jenkins, and G. Reinsel, *Time Series Analysis. Forecasting and Control.* Prentice Hall, 1994.

[2] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, pp. 213–225, 1991.

[3] M. Salmerón-Campos, "Predicción de Series Temporales con Redes Neuronales de Funciones Radiales y Técnicas de Descomposición Matricial," Ph.D. dissertation, University of Granada , Departamento de Arquitectura y Tecnología de Computadores, 2001.

[4] M. Salmerón, J. Ortega, C. G. Puntonet, and A. Prieto, "Improved ran sequential prediction using orthogonal techniques," *Neurocomputing*, vol. 41, pp. 153–172, 2001.

[5] J. Górriz, C. G. Puntonet, M. S. Campos, and J. dela Rosa, "New model for time-series forecasting using rbfs and exogenous data," *Neural Computation and Applications, Vol 13 Issue 2. Jun. 2004.*, 2003.

[6] J. Górriz-Sáez, "Algoritmos híbridos para la modelización de series temporales con técnicas ar-ica," Ph.D. dissertation, University of Cádiz , Departamento de Ing. de Sistemas y Aut. Tec. Electrónica y Electrónica, 2003.

[7] T. Masters, *Neural, Novel and Hybrid Algorithms for Time Series Analysis Prediction.* John Miley & Sons, 1995.

[8] A. D. Back and A. S. Weigend, "Discovering structure in finance using independent component analysis," *Computational Finance*, 1997.

[9] A. D. Back and T. P. Trappenberg, "Selecting inputs for modelling using normalized higher order statistics and independent component analysis," *IEEE Transactions on Neural Networks*, vol. 12, 2001.

[10] A. Hyvarinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 1, pp. 411–430, 2000.

[11] A. Bell and T. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, pp. 1129–1159, 1995.

[12] P. Comon, "Independent component analysis: A new concept?" *Signal Processing*, vol. 36, pp. 287–314, 1994.

[13] S. Amari, A. Cichocki, and H. Yang, "A new learning algorithm for blind source separation," *Advances in Neural Information Processing Systems. MIT Press*, vol. 8, pp. 757–763, 1996.

[14] C. Puntonet, "Nuevos Algoritmos de Separación de Fuentes en Medios Lineales," Ph.D. dissertation, University of Granada , Departamento de Arquitectura y Tecnología de Computadores, 1994.

[15] F. J. Theis, A. Jung, E. Lang, and C. Puntonet, "Multiple recovery subspace projection algorithm employed in geometric ica," *In press on Neural Computation*, 2001.

[16] A. Mansour, N. Ohnishi, and C. Puntonet, "Blind multiuser separation of instantaneous mixture algorithm based on geometrical concepts," *Signal Processing*, vol. 82, pp. 1155–1175, 2002.

[17] D. Pollock, *A handbook of time series analysis, signal processing and dynamics.* Academic Press, 1999.

[18] G. Zhang, B. Patuwo, and M. Hu, "Forecasting with artificial neural networks: the state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[19] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 284–294, 1989.

[20] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1. U. of CA Press, 1967, pp. 281–297.

[21] C. Lawson and R. Hanson, *Solving Least Squares Problems.* SIAM Publications. Philadelphia, U.S.A., 1995.

[22] G. Zelniker and F. Taylor, *Advanced Digital Signal Processing: Theory and Applications.* Marcel Dekker. New York, U.S.A., 1994.

[23] S. Haykin, *Neural Networks: A Comprehensive Foundation.* Macmillan College Publishing Company. New York, U.S.A., 1994.

[24] S. Cruces, *An unified view of BSS algorithms(in Spanish).* University of Vigo, Spain, 1999.

[25] K. Kiviluoto and E. Oja, "Independent component analysis for paralell financial time series," *Proc. in ICONIP98*, vol. 1, pp. 895–898, 1998.

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of Statistical Learning.* Springer, 2000.

[27] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis.* Wiley Interscience, 2001.

[28] A. Savitzky and M. Golay, *Analytical Chemestry*, vol. 36, pp. 1627–1639, 1964.

[29] R. Hamming, *Digital Filters*, 2^a ed. Prentice Hall, 1983.

[30] H. Ziegler, *Applied Spectroscopy*, vol. 35, pp. 88–92, 1981.

[31] W. Press, S. Teukolsky, W. Vertterling, and B. Flannery, *Numerical Recipes in C++*, 2^a ed. Cambridge University Press, 2002.