TIME SERIES ANALYSIS WITH AXO-AXONIC CONNECTIONS IN NEURAL NETWORKS*

Mingo L.F.
Dept. OEI - Escuela Informática
Universida Politécnica de Madrid
Crta. Valencia km. 7
28031 Madrid - Spain
lfmingo@eui.upm.es

Castellanos J.
Dept. IA - Facultad Informática
Universida Politécnica de Madrid
Campus de Montegancedo
28660 Madrid - Spain
jcastellanos@fi.upm.es

Arroyo F.
Dept. LPSI - Escuela Informática
Universida Politécnica de Madrid
Crta. Valencia km. 7
28031 Madrid - Spain
farroyo@eui.upm.es

Abstract—**This paper shows properties of neural networks with axo-axonic connections when dealing with time series learning. Such neural networks can perform a global *Taylor* series analysis provided they use linear activation functions. Moreover, when forecasting periodic signals, sinusoidal activation function can be used to obtain a *Fourier* analysis. The *mean squared error* of these nets can be controlled varying the number of hidden layers in such a way that a neural net with n hidden layers outputs a $(n+1)$-degree nonlinear result. *Taylor* or *Fourier* coefficients can be used to setup weight initial values. This architecture performs a global approximation instead of a local approximation obtained by *Taylor* or *Fourier* analysis. A stock market application is also shown. Main purpose is to perform a sensitivity analysis of several stock variables in order to forecast the *IBEX-35* spanish index. Results are compared to those obtained with classics methods. A brief survey of classic time series analysis neural networks is exposed.**

Keywords: Neural Networks, Time Series Analysis, Function Approximation, Signal Forecasting.

1 Introduction

Neural networks are non-linear systems whose structure is based on principles observed in biological neuronal systems. A neural network could be seen as a system that can be able to answer a query or give an output as answer to a specific input. The in/out combination, i.e. the transfer function of the network is not programmed, but obtained through a "training" process on empiric datasets.

In practice the network learns the fuction that links input together with output by processing correct input/output couples. Actually, for each given input, within the learning process, the network gives a certain output which is not exactly the desired output, so the training algorithm modifies some parameters of the network in the desired direction. Hence, every time an example is input, the algorithm adjusts its network parameters to the optimal values for the given solution: in this way the algorithm tries to reach the best solution for all the examples. These parameters we are speaking about are essentially the weights or linking factors between each neuron that forms our network.

Neural Networks' application fields are tipically those where classic algorithms fail because of their unflexibility (they need precise input datasets). Usually problems with unprecise input datasets are those whose number of possible input datasets is so big that they can't be classified. For example in image recognition are used probabilistic algoithms whose efficency is lower than neural networks' and whose caratheristics are low flexibility and high development complexity. Another field where classic algorithms are in troubles is the analisys of those phenomena whose matematical rules are unknown.

There are indeed rather complex algorithms which can analyse these phenomena but, from comparisons on the results, it comes out that neural networks result far more efficient [20, 19]: these algorithms use *Fourier*'s trasform to decompose phenomena in frequential components and for this reason they result highly complex and they can only extract a limited number of harmonics generating a big number of approximations. A neural network trained with complex phenomena's data is able to estimate also frequential components, this means that it realizes in its inside a *Fourier*'s trasform even if it was not trained for that! One of the most important neural networks' applications is undoubtfully the estimation of complex phenomena such as meteorological, financial, socio-economical or urban events. Thanks to a neural network it's possible to predict, analyzing hystorical series of datasets just as with these systems but there is no need to restrict the problem or use *Fourier*'s tranform. A defect common to all those methods it's to restrict the problem setting certain hypothesis that can turn out to be wrong. We just have to train the neural network with hystorical series of data given by the phenomenon we are studying.

Calibrating a neural network means to determinate

*Supported by TIC2003-09319-c03-03 and INTAS INNO 182

the parameters of the connections (synapsis) through the training process. Once calibrated there is need to test the netowrk efficency with known datasets, which has not been used in the learning process. There is a great number of Neural Networks which are substantially distingushed by: type of use, learning model (supervised/non-supervised), learning algorithm, architecture, etc.

This paper focuses on supervised networks with the *backpropagation* learning algorithm and applied to signal analysis with a typical feedforward architecture. Some classic neural architectures (multilayer perceptron, time-lag recurrent network, Jordan and Elman networks and recurrent networks) are briefly explained and a new architecture (*axo-axonic*) is also shown in order to apply such architectures to a given signal forecasting and analysis problem.

1.1 Multilayer Perceptron

Multilayer perceptrons (*MLPs*) are layered feedforward networks [18] typically trained with static backpropagation. These networks have found their way into countless applications requiring static pattern classification. Their main advantage is that they are easy to use, and that they can approximate any input-output map.

In principle, backpropagation provides a way to train networks with any number of hidden units arranged in any number of layers [14, 15]. In fact, the network does not have to be organized in layers - any pattern of connectivity that permits a partial ordering of the nodes from input to output is allowed. In other words, there must be a way to order the units such that all connections go from "earlier" (closer to the input) to "later" ones (closer to the output). This is equivalent to stating that their connection pattern must not contain any cycles. Networks that respect this constraint are called feedforward networks; their connection pattern forms a directed acyclic graph or dag.

Backpropagation algorithm can be expressed as equation (1). Note that in order to calculate the error for unit j, we must first know the error of all its posterior nodes (forming the set P_j). Again, as long as there are no cycles in the network, there is an ordering of nodes from the output back to the input that respects this condition. For example, we can simply use the reverse of the order in which activity was propagated forward.

$$\delta_j = f_j^{'}(net_j) \sum_{i \in P_j} \delta_j w_{ij} \tag{1}$$

1.2 Jordan/Elman Networks

Jordan and Elman networks extend the multilayer perceptron with context units, which are processing elements that remember past activity. Context units provide the network with the ability to extract temporal information from the data. In Elman networks, the activity of the first hidden layer are copied to the context units, while the Jordan network copies the output of the network.

Jordan and Elman networks [17, 16] combine the past values of the context unit with the present input x to obtain the present net output. The Jordan context unit acts as a so called lowpass filter, which creates an output that is the weighted (average) value of some of its most recent past outputs. The output y of the network is obtained by summing the past values multiplied by the scalar parameter t^n. The input to the context unit is copied from the network layer, but the outputs of the context unit are incorporated in the net through their adaptive weights.

$$y(n) = \sum_{i=0}^{n} x(n) y^{n-i} \tag{2}$$

In these networks, the weighting over time is inflexible since we can only control the time constant (i.e. the exponential decay). Moreover, a small change in time is reflected as a large change in the weighting (due to the exponential relationship between the time constant and the amplitude). In general, we do not know how large the memory depth should be, so this makes the choice of t problematic, without having a mechanism to adopt it.

In linear systems, the use of past input signals creates the moving average (MA) models. They can represent signals that have a spectrum with sharp valleys and broad peaks. The use of the past outputs creates what is known as the autoregressive (AR) models. These models can represent signals that have broad valleys and sharp spectral peaks. The Jordan net is a restricted case of a non-linear AR model, while the configuration with context units fed by the input layer is a restricted case of non-linear MA model. Elman?s net does not have a counterpart in linear system theory. These two topologies have different processing power.

1.3 Time-Lag Recurrent Networks

Time lagged recurrent networks are *MLPs* extended with short term memory structures. Most real-world data contains information in its time structure. Yet, most neural networks are purely static classifiers. *TLRNs* are the state of the art in nonlinear time series prediction, system identification and temporal pattern classification.

We may incorporate time into the design of a neural network implicitly or explicitly. A straightforward method of implicit representation of time is to add a short-term memory structure in the input layer of a static neural network (e.g., multilayer perceptron).The resulting configuration is sometimes called a focused time-lagged feedforward network (TLFN).

The short-term memory structure may be implemented in one of two forms [20], as described here:

- Tapped-Delay-Line (TDL) Memory. This is the

most commonly used form of short-term memory. It consists of p unit delays with $(p+1)$ terminals which may be viewed as a single input-multiple output network. The memory depth of a TDL memory is fixed at p, and its memory resolution is fixed at unity, giving a depth resolution constant of p.

- Gamma Memory. We may exercise control over the memory depth by building a feedback loop around each unit delay. In effect, the unit delay z^{-1} of the standard TDL memory is replaced by the transfer function.

$$G(z) = \frac{\mu z^{-1}}{1 - 1(1-\mu)z^{-1}} = \frac{\mu}{z - (1-\mu)} \tag{3}$$

where μ is an adjustable parameter. For stability, the only pole of $G(z)$ at $z = (1-\mu)$ must lie inside the unit circle in the z plane. This, in turn, requires that we restrict the choice of μ to the following range of values: $0 < m < 2$.

The overall impulse response of the gamma memory, consisting of p sections, is the inverse z transform of the overall transfer function

$$G_p(z) = \left(\frac{\mu}{z - (1-\mu)}\right)^p \tag{4}$$

Denoting the impulse response by $g_p(n)$, we have

$$g_p(z) = \begin{pmatrix} n-1 \\ p-1 \end{pmatrix} \mu^p (1-\mu)^{n-p} \tag{5}$$

where (:) is a binomial coefficient.The overall impulse response $g_p(n)$ for varying p represents a discrete version of the integrand of the gamma function, hence the name gamma memory.

1.4 Recurrent Networks

Fully recurrent networks feed back the hidden layer to itself. Partially recurrent networks start with a fully recurrent net [19, 18] and add a feedforward connection that bypasses the recurrency effectively treating the recurrent part as a state memory. These recurrent networks can have an infinite memory depth and thus find relationships through time as well as through the instantaneous input space.

We need an algorithm that computes equation (6) at each time step t. Since we know $e_k(t)$ at all times (the difference between our targets and outputs), we only need to find a way to compute the second factor $\partial y_k(t)/\partial w_{ij}$.

$$-\frac{\partial E(t)}{\partial w_{ij}} = \sum_{k \in U} \frac{-\partial E(t)}{\partial y_k(t)} \frac{\partial y_k(t)}{\partial w_{ij}} = \sum_{k \in U} e_k(t) \frac{\partial y_k(t)}{\partial w_{ij}} \tag{6}$$

The key to understanding *RNs* is to appreciate what this factor expresses. It is essentially a measure of the sensitivity of the value of the output of unit k at time t to a small change in the value of w_{ij}, taking into account the effect of such a change in the weight over the entire network trajectory from t_0 to t. Note that w_{ij} does not have to be connected to unit k. Thus this algorithm is non-local, in that we need to consider the effect of a change at one place in the network on the values computed at an entirely different place.

$$p_{ij}^k(t+1) = f_k^{'}(net_k(t)) \sum_{I \in U} w_{kI} p_{ij}^I(t) + \delta_{ik} z_j(t) \tag{7}$$

The algorithm then consists of computing, at each time step t, the quantities $p_{ij}^k(t)$ using the differences between targets and actual outputs to compute weight changes, and the overall correction to be applied to w_{ij} is given by:

$$\begin{aligned} \Delta w_{ij}(t) &= \mu \textstyle\sum_{k \in U} e_k(t) p_{ij}^k(t) \\ \Delta w_{ij} &= \textstyle\sum_{t=t_0+1}^{t_1} \Delta w_{ij}(t) \end{aligned} \tag{8}$$

2 Axo-axonic Networks

The most usual connection type in neural networks is the axo-dendritic connection. This connection is based on the fact that the axon of an afferent neuron is connected to another neuron via a synapse on a dendrite, and modelized in *ANN* model by a weighted activation transfer function. But, there exists many other connection types as: axo-somatic, axo-axonic and axo-synaptic [1]. This paper is focused on the second kind of connection type *axo-axonic*. Merely, the structure of the axo-axonic connection can be sketched by three neurons with a classical axo-dendritic connection and the synaptic axonal termination of N_3 connected to the synapse S_{12}. The principle consists on propagating the action of neuron N_3 as synapse S_{12}. In order to model previous connection type, two neural networks are required [5]. The first (assistant) one will compute the weight matrix of the second (principal) one. And, the second network will output a response, using the previously computed weight matrix, this architecture is named Enhanced Neural Networks *ENN* [2, 3, 4].

2.1 Taylor Approximation

Taylor approximation degree 2 of a function n-differentiable at a point $x = a$ can be obtained using the following expression as a power series:

$$\hat{f}(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + e(\xi) \tag{9}$$

, where ξ belongs to interval $[x, a)$.

If $f'''(x)$ is a continuos function in the closed interval $[a, x]$ then this derivate has a maximun M in such

interval, and therefore, the error in the aproximation (equation 9) is measure by [8, 9]:

$$\max |f'''(x)| \leq M \tag{10}$$

$$|e(x)| \leq \frac{1}{6} M |x-a|^3 \tag{11}$$

In case an approximation degree n of function $f(x)$ must be obtained, previous equations can be generalized in order to get:

$$\hat{f}(x) = \sum_{i=0}^{n} \frac{f^{i)}(a)(x-a)^i}{i!} + \frac{f^{n+1)}(\xi)(x-a)^{(n+1)}}{(n+1)!} \tag{12}$$

provided following constraints are verified:

1. $f^{i)}(x)$ corresponds to the i-derivate of $f(x)$. Besides $f^{0)}(x) = f(x)$.
2. If $i = 0$ then $i! = 1$.
3. ξ is a point at interval $[x, a)$.

The approximation error, that is $f(x) - \hat{f}(x)$, can be measured if the $(n+1)$-derivate is a continuos function in interval $[a, x)$. Approximation error has a maximum defined by:

$$|e(x)| \leq \frac{1}{(n+1)!} M |x-a|^{(n+1)} \tag{13}$$

2.2 ENN as Taylor series approximators.

Above section has shown that a function can be approximated with a given error using a polynomial $P(x) = \hat{f}(x)$ with a degree n. The error $f(x) - P(x)$ is measure by equation (13) in such a way that in order to find a suitable approximation (error lower than a known threshold) it is only needed to compute sucessive derivates of function $f(x)$ until a certain degree n.

Enhanced Neural Networks behave as n-degree polynomial approximators depending on the number of hidden layer in the architecture. In order to obtain such behavior all activation functions of the net must be lineal function $f(x) = ax + b$.

As shown in figure 1 and output equations, the number of hidden layers can be increased in order to increase the degree of the output polynomial, that is, the number n of hidden layers control, in some sense, the degree $n + 2$ of output polynomial of the net.

Table 1 shows how the degree of the output polynomial increases according to the number of hidden layers in the net.

The only condition that the learning algorithm must verified is that weights must be adjusted to values related with the sucesive derivates of function $f(x)$ that pattern set represents. Usually such function is unkown therefore, if the network converges with a low mean squared error then all weights of the net have converged to the derivates of function $f(x)$ (the pattern set unkown function), and such weights will gather some information about the function and its derivates that the pattern set represents.

Table 1: Number hidden layers vs. degree of output polynomial

Hidden Layers	Degree $P(x)$	Output Polynomial
0	2	$o = a_2x^2 + a_1x + a_0$
1	3	$o = a_3x^3 + a_2x^2 + a_1x + a_0$
...	...	...
n	$n+2$	$o = \sum_{i=0}^{n+2} a_i x^i$

As an example, function $f(x) = sen(x)cos(x)$ can be approximated using equation (12), with a given point $a = 0$. Such equation can be reduced to $\tilde{f}(x) = x - \frac{4}{6}x^3$, using a polynomial $P(x)$ degree 3. This is a mathematical approach, but what happens if such function is the pattern set to an enhanced neural network mentioned before?.

A one hidden layer neural network must be used in order to obtain a 3-degree polynomial as the output expression. Figure 2 shows such architecture, after the training stage, the final configuration is shown. Output equation of the net is $o = x - \frac{4}{6}x^3$, equivalent equation with $\tilde{f}(x)$.

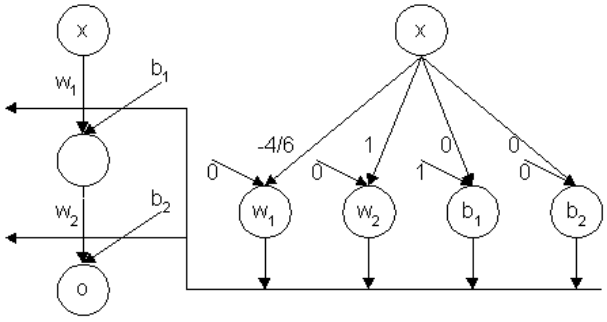
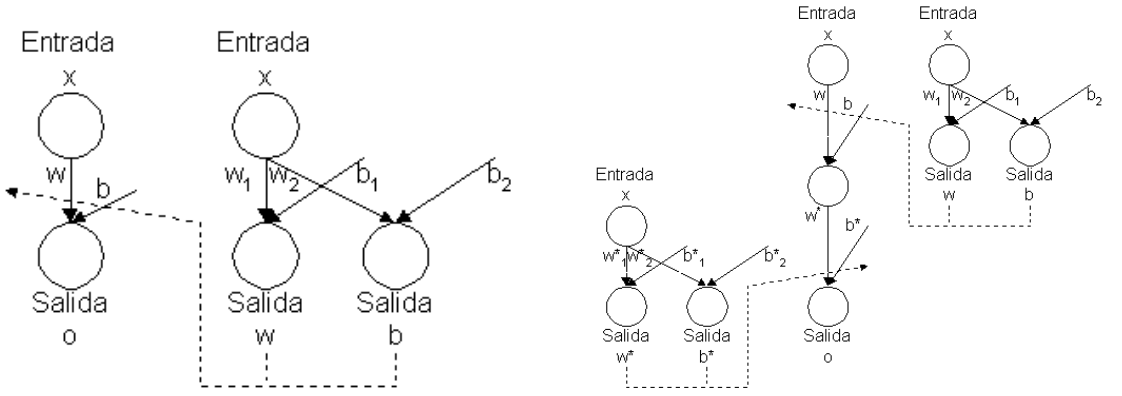


Fig.2: Approximation of $f(x) = sen(x)cos(x)$ with a one hidden layer

The approximation error using net in figure 2 can be computed using equation (13), and therefore $MSE \leq |e(x)|$. Such approximation is not the only one nor the best one, but it can be computed theoretically in order to provide the net some initial weights in order to speed up the learning process and to obtain a better approximation that the initial one with a lower error ratio. In sumary, Enhanced Neural Networks can be initialized to some weights computed using the Taylor Series of the function that the pattern set defines and after this initial stage the learning algorithm must be applied in order to achieved the best solution (the one that improves the Taylor Series error).

Figure 3 shows the surface computed by a net as the number of hidden layers is increased. The mean squared



$$
\begin{aligned}
o &= wx + b = \\
&= (w_1 x + b_1)x + w_2 x + b_2 = \\
&= w_1 x^2 + (b_1 + w_2)x + b_2
\end{aligned}
$$

$$
\begin{aligned}
o &= w^*(wx + b) + b^* = \\
&= (w_1^* x + b_1^*)[(w_1 x + b_1)x + w_2 x + b_2] + = \\
&+ w_2^* x + b_2^* = \\
&= Ax^3 + Bx^2 + Cx + D
\end{aligned}
$$

Fig.1: ENN architectures and output expressions

error is decreasing as the number of hidden layers goes up. This figure shows that this kind of neural net is very suitable when approximating functions, a given function or a function defined by the pattern set.

2.3 Non-Linear Activation

According to previous ideas, *linear ENNs* are better than linear *MLPs*, or at least, they are able to generate complex regions in order to divide the output space. When working with a *MLP*, only hyperplanes can be obtained. And moreover, the degree of the output equation increases according to the number of hidden layers.

In order to obtain a functional basis, one constraint must be made. It consists on implementing the network architecture with lineal *PE*s except the output neurons of assistant network. These neurons must have an activation function $g(x)$ which is used to computed the functional basis as the application of $g(x)$ to a non lineal combination of inputs. Figure 4 shows an example of a functional basis and the main network ourput.

Depending on the activation function of output neurons belonging to assistant network, the main network will output an approximation function based on non lineal combination of elements belonging to the basis. That is if a sinusoidal activation function is implemented, then a cuasi-Fourier approximation is computed by the network; is a Ridge activation function is implemented, then a cuasi-Ridge approximation is computed and so on.

Main advantage of this new approximation method is that is absolutely easy to implement. And moreover, a global approximation to all the pattern set is perform. This way, if there are enough input patterns, then the generalization error will be minimized if there are enough learning iterations.

2.3.1 Enhanced Neural Networks as Universal Approximators

Along the paper [3], this new architecture has shown that it is very suitable when dealing with any problem. Decision surfaces generated by the net are complex enough to represent any data set. The powerfull of these nets is in the number of hidden layers, that is, in the degree of the output polinomial associated to one output unit.

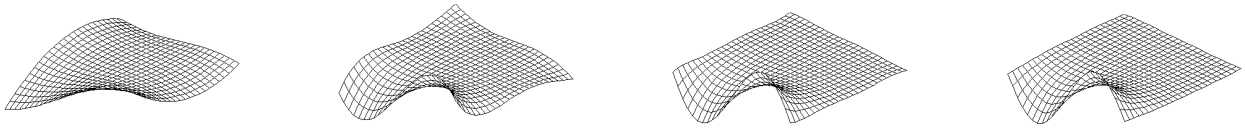
Funahashi Theorem can be directly apply to *Enhanced Neural Networks* in order to proof the universal approximation property of proposed networks, provided that activation function in hidden and output neurons belongs to a given class of functions stated by *Funahashi*. This way, *ENN* behave as universal approximators, that is, they are able to learn any pattern set.

3 Stock Market Forecasting Results

Some studies [10, 13] have shown that the mathematical analysis of stock movements does not work. Technical and Fundamental Analysts are proven wrong time and again - and with good reason. The numbers, formulas and charts that Analysts generate and use are not trading stocks - real people are trading stocks. No matter how experienced a Trader may be, there will always be a degree of subjectivity or even emotion involved in every trade. Still, as with any form of prediction there is always a degree of ambiguity, which can be magnified further when applied to something as volatile as the stock market. There will always be errors or 'upsets' - but you may find they are few and far in between.

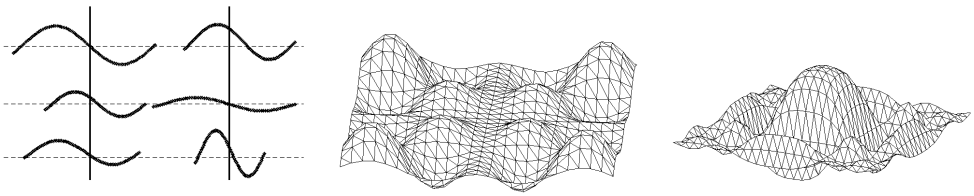
In the past, when trading was not dominated by computers, most financial analysts used macro and micro economic theory as well as classical "linear" modeling techniques [12, 11]. However, today the market moves faster and more chaotically, exhibiting disjointed and nonlinear relationships between market

Pattern set defined by $f(x, y) = sin(x)e^y$



2 hidden layers 4 hidden layers 6 hidden layers 8 hidden layers

Fig.3: Surface approximation depending on the number of hidden layers



a) Sinusoidal basis b) 3D Surfaces Approximation

Fig.4: Approximation with sinusoidal activation functions using basis of figure a).

forces. Neural networks can work well [2, 5] because they are inherently nonlinear and can behave better than linear models in this environment and they can be automatically retrained over and over to accommodate new behavior in the markets.

The data set has been obtain from the *Buse MERVAL, Eurostoxx, nikkey, DOW JONES* indexes. Each pattern represents a day in the stock market and it consists on the close value. The pattern set is made up of 4 years. The desired output is the close value of the *DJ Eurostoxx* index in time $t + 1$, that is a forecasting of the future behavior of the signal, without iterative prediction.

Table 2: Statistical Analysis with all variables.

Variable	Coeff.	Std. Err.	t-St.	Prob
Eurostoxx	0.9813	0.0082	119.4	0
Nikkey	0.0002	0.0001	1.665	0.0961
Dow	0.0002	0.0003	0.7982	0.4249
Dolar	-2.291	4.226	-0.5422	0.5878
Buse	0.0012	0.0024	0.5219	0.6018
Euribor	0.4500	0.4783	0.9409	0.3469
C	-2.249	5.9324	-0.0422	0.9665
R-Sq.	0.9948	Mean var		336.81

3.1 Statistical Analysis

The statistical analysis performs an approximation of the whole pattern set, but it does not divide it into train, cross validation and test pattern set. We have studied the correlation among the all six variables in order to forecast one of them at time $t + 1$. First of all this model performs a 99% approximation of all patterns. Figure 5 (first chart) shows that the desired and output signals are quite similar, but we can see that the only variable that is really significant is the proper one that is forecasted. If we use this model we can not explain the relationship among the all six variables and output one (*EuroStoxx*), see table 2.

Next method uses all variables except the forcasting one. This one achieves a 92% of approximation and all variables are correlated with the output (see table 3 and figure 5, second chart).

The last model only uses one variable (*Dow Jones*) to forecast the desired one (*Eurostoxx*), figure (5). In this case, we obtain a 63% of approximation.

Table 3: Statistical Analysis with all variables except the forecasting one.

Variable	Coeff.	Std. Err.	t-St.	Prob
Nikkey	0.0114	0.0004	29.592	0
Dow	0.0121	0.0013	0.292	0
Dolar	-138.177	15.916	-8.6812	0
Buse	-0.0024	0.0097	-0.2553	0.7985
Euribor	38.24	1.403	27.25	0
C	36.081	23.1689	1.5573	0.1197
R-Sq.	0.9206	Mean var		336.86

Table 4: Statistical Analysis with only one variable.

Variable	Coeff.	Std. Error	t-St.	Prob
Dow	0.0616	0.0014	42.19	0
C	-289.6	14.90	-19.43	0
R-Sq.	0.6396	Mean var		336.86

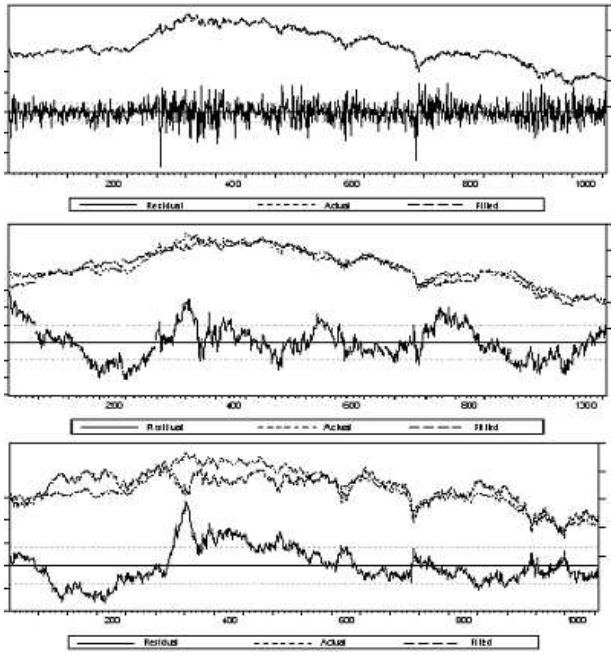


Fig.5: Results of statistical analysis: a) using all variables, b) using all variables except the forecasting one, c)using only the DJ variable.

Pattern set in the training process set consists of 6 variables and the aim is to forecast the second variable at time $t+1$. An Enhanced Neural Network with Time-Delays has been implemented to take advantage of its forecasting properties against classical Multilayer Perceptron. But, in order to define the best architecture some genetic controllers has been added. The pattern set has been splitted into the training, cross validation and test sets to measure the performance of the net. After the genetic learning stage we have obtained a two-hidden layer neural network with 13 and 17 hidden units and 5 tap delays with a 10 depth.

Figures 6, 7 and 8 show graphical results of *DJ Eurostoxx* forecasting at time $t+1$. You can see that the training and cross validation set are quite close to the desired response, the have a mean squared error $(0.000127, 0.001843)$ repectively.

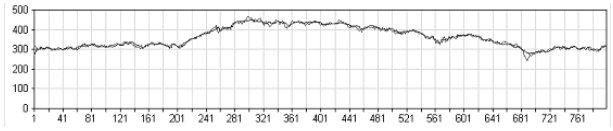


Fig.6: Training output and desired output.

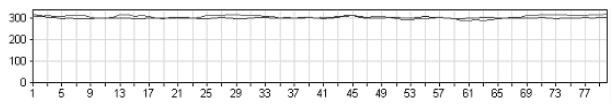


Fig.7: Cross Validation output and desired output.

The first 10 outputs of neural network in the test stage are not valid since this architecture needs a depth of 10 data in order to output a real desired response, that is the way there exists a great error in the prediction of these 10 outputs (see figure 8), but the rest of the forecasting results are good enough to consider this architecture very suitable for this task.

Table 5: MSE of ENN with genetic controllers with 6 and 4 inputs variables.

Training	**Cross Validation**	**Test**
Mean Squared Error with 6 inputs		
0.000127	0.001843	0.023849
Mean Squared Error with 4 inputs		
0.000425	0.002143	0.031442

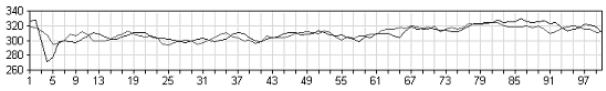


Fig.8: Test output and desired output.

Figure 9 shows the sensitivity analysis of neural networks inputs (*Euribor, DJ Eurostoxx, Dow Jones, Buse Merval, Nikkei, USD/Euro*) that affect the desired output. Inputs *DJ Eurostoxx* and *Buse Merval* are the most relevant inputs in order to output the *DJ Eurostoxx* at time *t+1*. Both of them have a sensitivity $20.8 + 21.4 = 42.2$ of all inputs. But if the previous neural network is only trained with these two inputs the performance is poor. So, we have to take, at least a 80% sensitivity, four inputs *Euribor, DJ Eurostoxx, Dow Jones, Buse Merval* in order to get 76% of sensitivity.

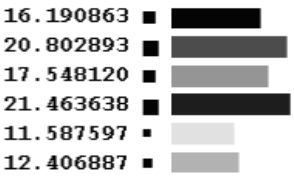


Fig.9: Sensitivity Analysis.

Taking into account previous considerations, the neural architecture was modified to get only four inputs variables and results are similar to those with six variables (see table 3.1).

According to figure 5 of statistical analysis, only the *Dow Jones* variable is needed to obtain a relatively good forecasting, but not so closed to desired response as the neural network output with 4 variables (table 3.1 and figure 9).

Next figure 10 shows the results of the neural network if we only consider one input (*Dow Jones*) as in the statistical analysis. Obtained *MSE* is 0.0539485 in the training stage and 0.0584853 in the test stage.

We have to consider that the statistical analysis per-

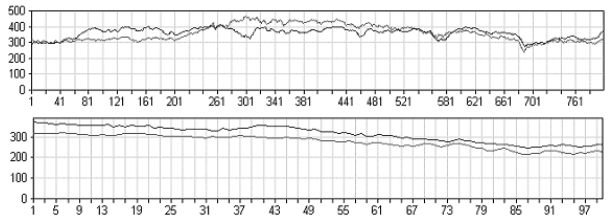


Fig.10: Training and testing outputs with only one input.

forms the computation over the whole pattern set, it does not divide the pattern set in traning/test sets. That is the reason why the test output of the neural network seems worse than the statistical one, but the training output of the net is similar to the statistical one.

4 Conclusions

This paper shows the classical statistical analysis applied to stock market forcasting and a neural network approach with axo-axonic controllers. First of all, the statistical approach does not consider train, cross validation and test pattern sets; that is the reason why results may appear better than neural results. But, the neural approach takes into account these three sets in order to improve the net performance. And if only the train set is compared with the statistical model results are better in the *NN* model.

Artificial Neural Networks can be applied to many complex problems such as market forecasting. This paper has shown the *EuroStoxx* index forecasting using a neural network architecture with axo-axonic controllers and Time-Delays. Enhanced nets with time-delays is the best architecture in order to forecast the index, this is due to the special architecture and the universal approximation property that this kind of networks has.

Performed sensitivity analisys in the neural network gives us significant variables in order to obtain a valid forecasting method. The best model was obtained with four variables (including the forecasting one) with a really low error in the test pattern set. If we try to implement a neural architecture with only one variable (statistical analysis third figure 5) the net behaves better than the classical model.

References

[1] Delacour, J.; *Apprentissage et Memoire: Une Approache Neurobiologique.* Masson(Ed.) September. (1987).

[2] Mingo L.F., Arroyo F., Luengo C., Castellanos J.; *Learning HyperSurfaces with Neural Networks.* 11th Scandinavian Conference on Image Analysis. SCIA'99. June 7-11. Kangerlussuaq, Greenland. Pp: 731-737. 1999.

[3] Mingo L.F., Arroyo F., Luengo C., Castellanos J.; *Enhanced Neural Networks and Medical Imaging.* 8th International Conference on Computer Analysis of Images and Patterns. CAIP'99. September 1-3. Ljubljana, Slovenia. 1999.

[4] Mingo L.F., Giménez V., Castellanos J.; *Interpolation of Boolean Functions with Enhanced Neural Networks.* Second Conference on Computer Science and Information Technologies. CSIT'99. August 17-22. Yerevan, Armenia. 1999.

[5] Mingo L.F., Castellanos J., Giménez V.; *A New Kind of Neural Networks and Its Learning Algorithm.* Information Processing and Management of Uncertainty in Knowledge Based Systems. IPMU'98. Paris, France. July 6-10. Pp: 1913-1914. 1998.

[6] Mingo L.F., Aslanyan L., Riazanov V., Castellanos J., Diaz M.A.; *Context Neural Network for Temporal Correlation and Prediction.* International Conference on Neural Networks and Genetic Algorithms. Praha, Czech Republic. Springer: Computer Science. Pp.: 110 - 113. 2001.

[7] Peter Andras: *Orthogonal RBF Neural Network Approximation.* Neural Processing Letters, 9(2). Pp.: 141-151. 1999.

[8] Blum, E. K. & Leong, L.: *Approximation Theory and Feedforward Networks.* Neural Networks, 4. Pp. 511-515. 1991.

[9] Cheney E.W., Chui C.K. & Shumaker L.L.: *Ridge Functions, Sigmoidal Functions and Neural Networks.* Approximation Therory VII. Academic Press, Boston. Pp. 158-201. 1992.

[10] Rosenblatt, Taylor S. J.; *Modelling Financial Time Series* Chichester: John Wiley and Sons. 1986.

[11] Palus M.; *Testing for nonlinearity using redundancies: Quantitative and qualitative aspects.* Physica D. Vol: 80. Pp: 186-205. 1995.

[12] Theiler J., Linsay P. S., Rubin D. M. ; *Detecting nonlinearity in data with long coherence times.* Time series prediction: Forecasting the future and understanding the past. Addison Wesley. Ed.: A. S. Weigend and N. A. Gershenfeld. Pp: 429-455. 1993.

[13] Hsieh D.A.; *Chaos and nonlinear dynamics: Applications to Financial Markets.* Journal of Finance. Vol: 46, Pp: 1839-1877. 1991.

[14] Hu YH; *Pattern Classification with Multiple Classifiers.* Dept. of Elect. Comput. Engineering. University of Wisconsin-Madison. 1996.

[15] Hu YH, Tompkins WJ, Urrusti JL, Afonso VX; *Applications of Artificial Neural Networks for ECG*. Signal Detection and Classification. Journal of Electrocardiology, Vol. 26, pp. 66-73. 1994.

[16] A. D. Back and A. S. Weigend; *A first application of independent component analysis to extracting structure from stock returns*. Int. J. on Neural Systems, 8(4):473-484, 1998.

[17] P. Comon; *Independent component analysis - a new concept?* Signal Processing, 36:287-314, 1994.

[18] T. M. Cover and J. A. Thomas.; *Elements of Information Theory*. John Wiley Sons, 1991.

[19] Shigeru Katagiri; *Handbook of Neural Networks for Speech Processing*. Artech House. ISBN 0890069549. 2000

[20] YuHenHu, Jeng-NengHwang; *Handbook of Neural Network Signal Processing VE Profiling*. ISBN: 0849323592. CRC Press. September2001