Forecast of monthly Real Price S&P 500 stock market data

Working paper for evaluation to 'Análisis Econométrico de los Mercados Financieros'

Name:
Sérgio Nobre
PhD Student

Professor:
Prof. Doctor Alberto Muñoz

Date:
5th August 2010

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
FACULTAD DE CIENCIAS ECONÓMICAS Y EMPRESARIALES
ECONOMIA APLICADA Y ESTADÍSTICA
ESTADÍSTICA APLICADA A LA ECONOMIA Y A LA ADMINISTRATIÓN Y DIRECCION DE EMPRESAS

INDEX

1. Introduction

The S&P 500 is a free-float capitalization-weighted index published since 1957 of the prices of 500 large-cap common stocks actively traded in the United States. The stocks included in the S&P 500 are those of large publicly held companies that trade on either of the two largest American stock market companies; the NYSE Euronext and the NASDAQ OMX.

The index is the best known of the many indices owned and maintained by Standard & Poor's, a division of McGraw-Hill. S&P 500 refers not only to the index, but also to the 500 companies that have their common stock included in the index.

The issue of forecasting stock market data is not new, therefore the S&P 500 forecasting is a very interesting problem for Artificial Neural Networks (NNs) forecasting. The objective for this working paper is to model and forecast the S&P 500 using NNs. On this scope two hypothesis can be made:

H1: It is possible to build a NN forecast model for the S&P 500 stock price data.

H2: It is possible to forecast the S&P 500 using the built model.

In Chapter 2. the data will be presented as well as the pre-processing of it in order to be used on forecasting. On Chapter 3, an overview of the methods used on this working paper is described and the results of its application to the data described are showed on Chapter 4.

Some conclusions where taken based both on the results and on the reviewed bibliography.

2. DATA

The Standard & Poor's 500 Index is calculated using a base-weighted aggregate methodology; that means the level of the Index reflects the total market capitalization of all 500 component stocks relative to a particular base period. The S&P 500's base period is 1941-43

The raw data was subjected to analysis in order to evaluate its performance and confirms the validity of the final data subjected to analysis. The raw data (Price) was analyzed with SATISTICA software.

2.1. RAW DATA

The Price data series has the following time plot

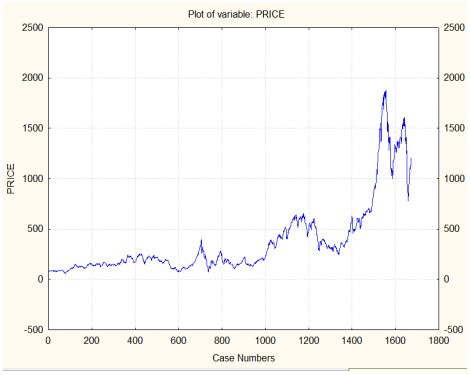


Chart 1 – Time series

Shows a time series with a regular trend for the about three quarters of the series with high volatility on the end.

The autocorrelation function shows a graphic within the confidence intervals showing a dependency on the series, however the partial autocorrelation function shows different results showing the underlying process, Prices, has not a multivariate normal distribution and does have a seasonal pattern.

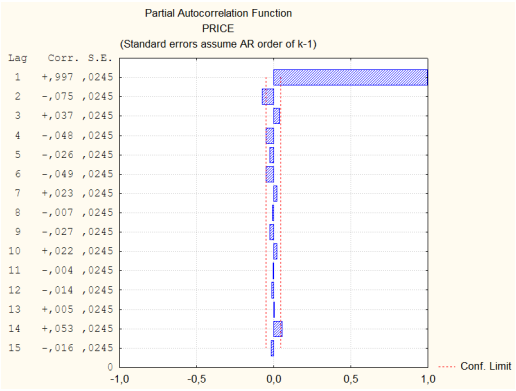


Chart 2 – Autocorrelation function

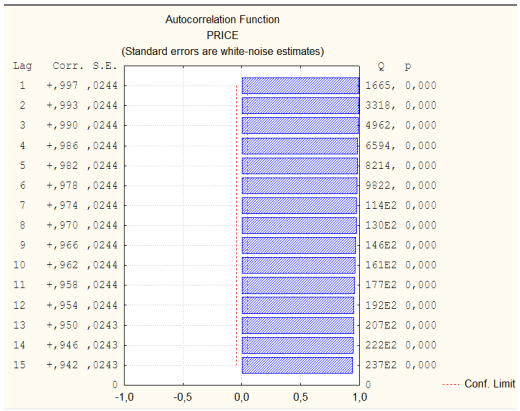


Chart 3 - Partial autocorrelation function

As expected the normal probability plot does not shows a proximity to the normal distribution.

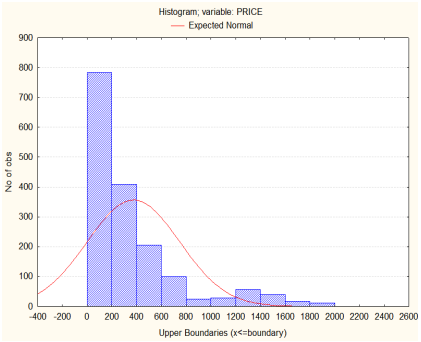


Chart 4 – Histogram of expected normal

Consequently the QQ plot has not a linear graph, it is highly curvilinear.

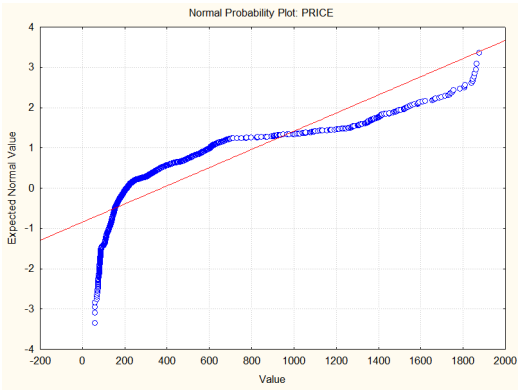


Chart 5 – QQ plot

The spectral analysis does not show a clear seasonal pattern, it shows irregularity.

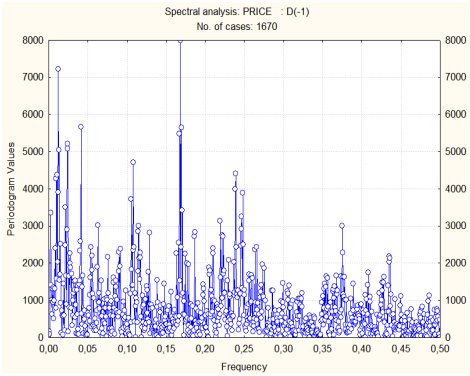


Chart 6 – Spectral analysis

The several ANN models tested result in an invariably a good adjustment on almost series, but extremely poor on the last quarter of the series, as the chart 7 shows.

The raw Price data shows results that are not reliable to take in consideration for NN forecasting, shows a bad adjustment on the model and a extremely poor forecasting capacity.

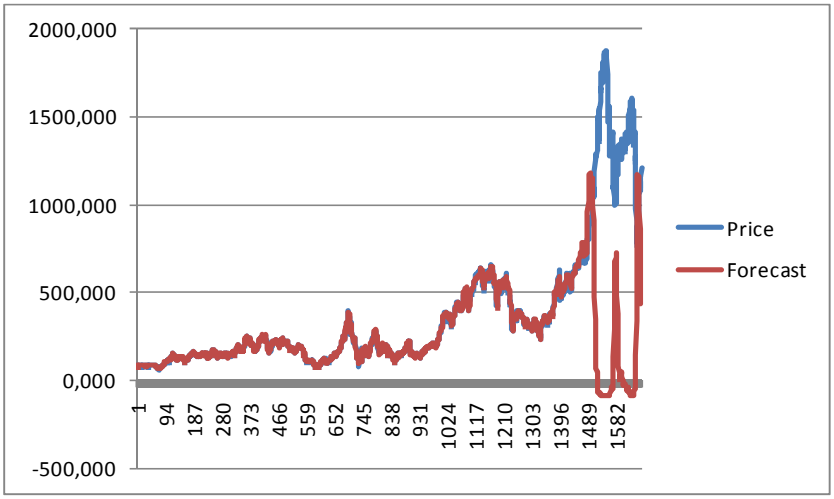


Chart 7 – Price time series and Forecast time series for raw data

2.2. DATA PRE-PROCESSING

Three issues where arisen with the raw data analysis,

1) Bad data pre-diagnosis
2) Bad model adjustment

3) Bad model forecasting capability

In order to improve the model adjustment and data diagnosis, the series was reduced to the most recent 475 observations due to the over fitting on the total series. Resulting on the following series,

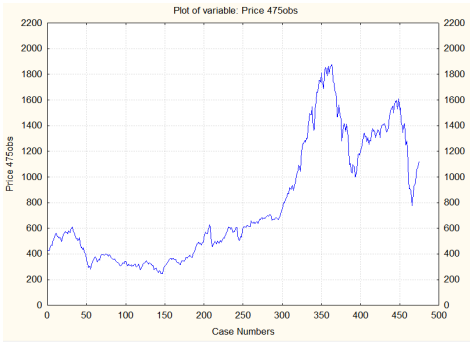


Chart 8 – 475 observation Price time series

Then the series was turned into logarithmical for deasonalise it, resulting on the following

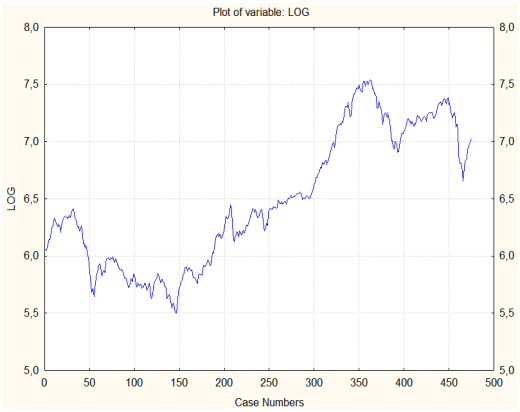


Chart 9 – Logarithmical time series

Then, in order to make the series stationary, the first differences where made on it, resulting on the following graph

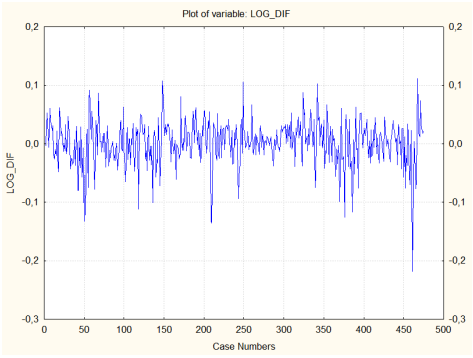


Chart 10 – First difference time series

As the series continued showing a not constant variance the second difference was made on the series,

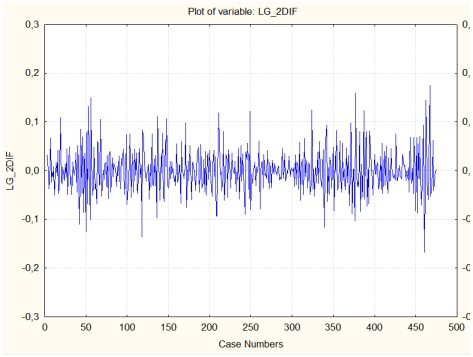


Chart 11 – Second difference time series

Resulting on a stationary time series ready to be used for forecasting purposes.

3. METHODS

3.1. BASIC CONCEPTS

3.1.1. TIME SERIES

A time series can be regarded simply as any set of observations ordered in time. A time series can be expressed by:

$$Z_t = \{Z_t \in \Re \mid t = 1, 2, 3, \ldots, N\}$$

Where t is a time index and N is the number of observations. Considering the existence of a time series of observations until time t, the forecast at time t + h is denoted by Z_t (h), whose origin is t and the forecast horizon is h.

Besides the forecast horizon, another parameter used by the forecasting process is the number of historical elements prior to the forecast horizon. It is called the forecast window and is present in most of the methods of time series forecasting. The preview window is used to train the examples (patterns) in which some forecasting methods perform the extraction of knowledge (learning) for application in the prediction of future values. The element that immediately follows the preview window is the target, i.e. the element you want to predict. The time series is usually divided into two sets of elements: the first is for the forecasting method to obtain the learning (training set) and the second is used to verify its performance in predicting future values (test set). Figure 01 shows an example of definition of these components to predict a time series: the division series in sets for training and testing, a preview window and the forecast horizon (target).

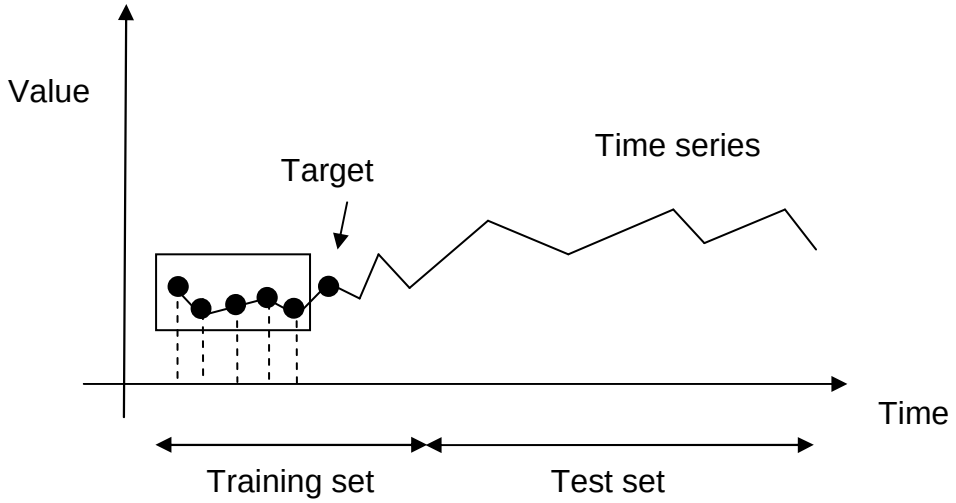


Figure 1

3.1.2. ANN COMPONENTS

The components of a neural network can be described by eight main elements:

- Processing units set
- Activation state
- Output function
- Interconnection pattern
- Propagation rule
- Learning rule
- Environment

3.1.2.1. PROCESSING UNIT

The processing unit is the basic component of the ANN and corresponds to the human neuron. The input represent the dendrite, each one has a signal (x) which is added (Σ). After each addition, this signal is processed through a function that represents the limit or activation function f(x), which produces a output signal. The artificial neuron can be interpreted as a simplified representation of biological neurons.

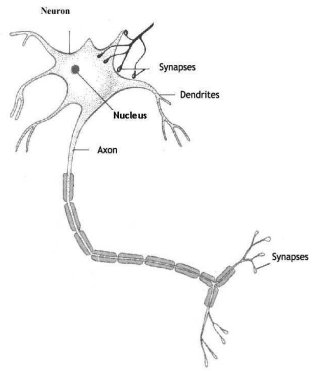


Figure 2 – Biological Neuron

3.1.2.2. ACTIVATION STATE

The activation state of all neurons of a network, i.e. the system activation state, specifies what is being represented on the network on ant instant *t*. the activation values exiting on the networks can be discrete.

3.1.2.3. OUTPUT FUNCTION

Neuron interact between themselves through a value that is transmitted through connections. This value is determined by the stimulating neuron activation. The

output function is the activation state mapping on a output signal. The output function can be the identity function, that is, the output signal is equal to the activation state. In many cases the output function is a limit function which only provide the output when its state goes over a certain limit.

3.1.2.4. INTERCONNECTION LIMIT

It is possible to represent the network interconnection pattern by a weight matrix *w*, where the element *wij*, corresponds to the influence of neuron *ui* over the neuron *uj*. Connections with positive weights, indicate the neuron *uj* activation disabling. The set of disabling and enabling connections existing on the network determines the behaviour of it.

Topologically the ANN can be organised in layers. The network entry layer does not recomputes its exits, it is used only for data input. The resulting values of the cells pertaining to the output layer are considered the network final results as a whole. The neuron that does not belong neither to the input layer neither to the output layer are called intermediary or hidden.

3.1.2.5. PROPAGATION RULE

Each neuron u_i computes its new activation through a propagation rule. In general, it is defined as being a sum function of the net weight input of the directly connected to u_i neurons. This is made through the multiplication of the j-th neuron by the connection weight of the i-th to the j-th neuron, for each one of the *j* neurons that are connected to the *i* neuron input. The propagation rule completes through the result of this computation when goes over the limit. This limit can be null, should be overcame for the cell activation occur.

3.1.2.6. ACTIVATION RULE

It is necessary a rule that computes the activation value of a neuron on the instant *t*. it is needed a function *f* that computes the new activation a(t) using the net input. Usually, this function has the form of $a_i(t+1)=f[a_i(t), net_i(t)]$ where *f* is the activation function also called limit function. This function maps the input for a pre-specified output. The most common activation function are linear, ramp, step and sigmoid.

3.1.2.7. LEARNING RULE

The modification of the processing or knowledge structure of a NN involves the interconnection pattern changing. In principle it can be made through three ways:

- New connections development
- Lost of existing connections on the network
- Modification of the existing weights

When the interconnection pattern is a weight matrix W, the first two items can be simulated by the last one. Taking a connection with weight zero, changing it to a positive or negative value, is equivalent to develop this connection. On the same way, changing its weight to zero means disconnecting it. Therefore, the learning rules change the networks connection weights through experience.

Specifically if a neuron *ui* receives an input of other *ui* and both are strongly active, the weight wij *(*from ui to ui*)* must be strengthened.

3.1.2.8. ENVIRONMENT

This last component of ANNs is the environment where the network should work. It is necessary to specify the environment nature, establishing the possible output and input patterns. Usually the environment is represented a probability function stable over a set of input patterns. This distribution can be independent or not, of input and output on the same environment.

3.2. ARTIFICIAL NEURAL NETWORKS

In intuitive terms, artificial neural networks (ANN) are mathematical models inspired by the principles of operation of biological neurons and in brain structure. ANNs are units of numerical processing, which produces a layer architecture in information flow with or without feedback, having a structure of signal processing with great power of adaptation and capacity for representation of non-linear processes. Among the usual applications of ANNs have: recognition and pattern classification, clustering or grouping, time series prediction, function approximation, optimization, signal processing, image analysis and process control. A neural network resembles the brain in two respects:

- Knowledge is acquired by the network from its environment through a learning process, and
- Forces of connection between neurons, known as synaptic weights are used to store the acquired knowledge.

In the process of learning, generalization means that the neural network will produce a desired output for input values not used during this process, this being formed, usually by means of examples. The basic element of building an

ANN is the artificial neuron, whose description is the initial stage for understanding the concepts related to the structures of existing networks.

3.2.1. ARTIFICIAL NEURON

A neuron is a processing unit of information that is critical to the operation of a neural network. The model of one neuron is illustrated in Figure 03 and its structure is described below.

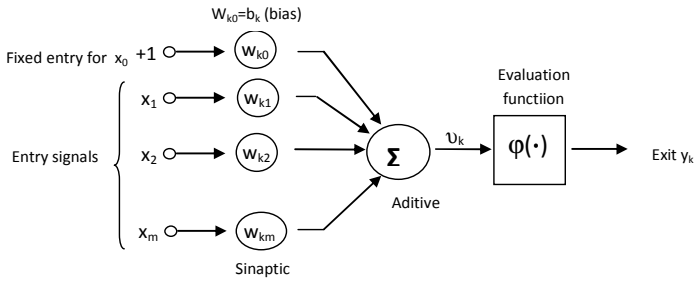


Figure 3 – Artificial Neuron

The structure of a neuron allows one set of values as inputs (x_1, x_2, ..., x_m) to produce a single output (y_k). Such entries are weighted by their respective synaptic weights (w_{k1}, w_{k2}, w_{k3} ...) and added to the value of an externally applied bias b_k. The bias b_k has the effect of increasing or decreasing the net inflow of activation function, depending on whether it is positive or negative. Then an activation function φ(·) Is used to restrict the amplitude of the output signal.

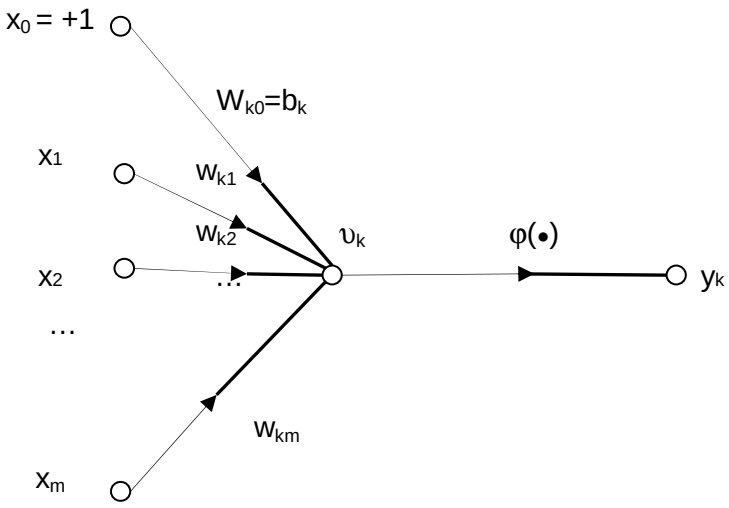


Figure 4

Typically the normalized range output of a neuron is described as the closed unit interval [0,1] or alternatively [-1,1]. Considering V_k as the potential

activation or induced local field, the neuron can be described by the following equations:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j$$

$$y_k = \varphi(v_k)$$

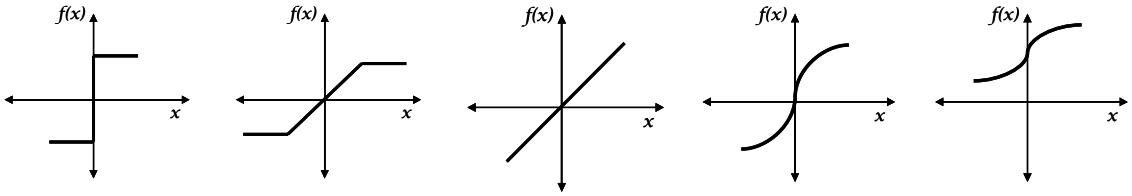
The activation function can display linear or nonlinear characteristics, determining the output of one neuron from its potential for activation. Among those that have linear behavior stand out linear function and the function of threshold (commonly referred to in the literature of engineering as a Heaviside function, function step or unipolar step). In general, a nonlinear function simulates more accurately the biological neurons, the sigmoid function is one of the most used. The sigmoid function can be defined by:

$$\varphi(v) = \frac{1}{1 + e^{-v}}$$

The activation functions described above extend from zero to +1. For the use of negative values in the corresponding form of a sigmoid function, we can start to employ the hyperbolic tangent function defined by:

$$\varphi(v) = \tanh(v)$$

The following graphics show the activation functions



(a) **(b)** **(c)** **(d)** **(e)**

3.2.2. ARTIFICIAL NEURAL NETWORK ARCHITECTURE

Neural networks are characterized by the following proprieties:

- The pattern of connections between the various network layer (network type)
- Number of neurons in each layer (complexity)
- Learning algorithm
- Neuron activation functions

The structure of the neural network, composed of interconnections of neurons, can vary on the number of layers, neurons in each layer, activation function of neurons in a layer and how the layers are connected (fully or partially).

There are basically three types of architecture: single-layer feedforward networks, multilayer feedforward networks and recurrent networks.

The single-layer feedforward networks are composed of an input layer of source nodes projected on an output layer of neurons, but not vice versa. An example of Multilayer Feedforward Networks are networks based on multilayer perceptron (MLP). Perceptron is a network model of a level only able to classify patterns that are linearly separable. MLP networks have one or more hidden layers (hidden) as well as layers of input and output, the structure of these networks meets the shortcomings of perceptron networks. Typically, neurons in each layer of an MLP only have as its inputs the output signals of previous layer. The MLP networks are typically used to determine a mapping between two data sets. Figure 05 illustrates an example of an MLP.

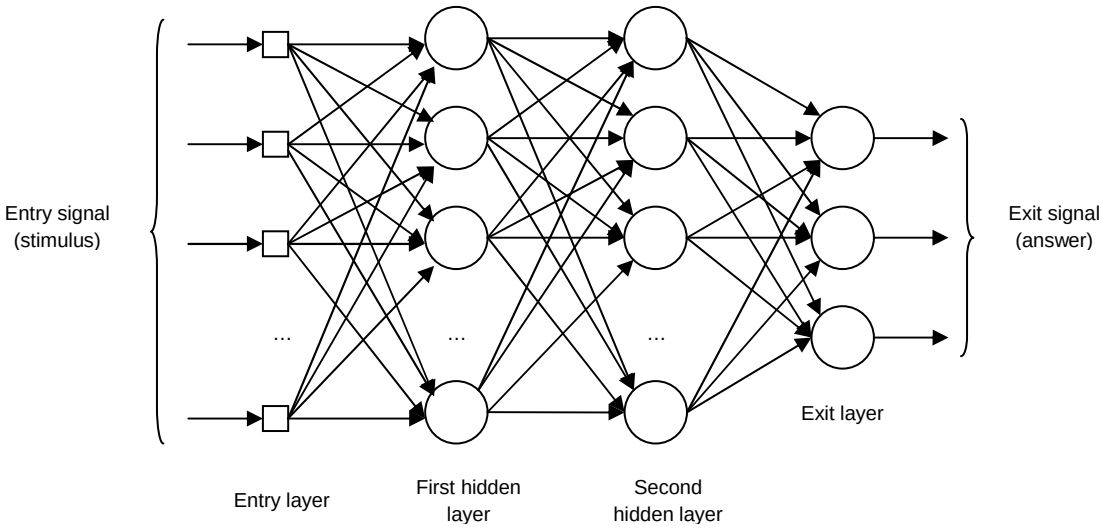


Figure 5 – Multilayer perceptron

A recurrent neural network differs from a feedforward network to have at least one feedback loop. Due to this characteristic, are also called ANNs with memory, dynamically responding to stimuli, i.e., after applying a new input, the output is calculated and fed back to modify the entry. The MLP architectures are models of artificial neural networks most commonly used and known. The inherent capacity of its network infrastructure into three layers promotes any mapping of input-output, describing the MLP networks for efficient prediction of a time series.

The choice of the structure of the neural network directly determines the quality of the obtained model. Even for a defined structure, there can be high or even

infinite number of possible solutions. Several activities are necessary for defining the architecture of a neural network for which a specific problem should be solved optimally.

After determining the type of network architecture that will be used as options exemplified above, various activities are required for assembly of the network structure, some of which are described below. In most applications of prediction, only one hidden layer is used, although some exceptional circumstances justify a higher number. The number of neurons in layers of input and output depends on the dimensionality of the data, while the number of neurons in the hidden layers depends on the complexity of the problem. The greater the number of neurons in intermediate layers, more complex functions is mapped to the ANN.

For the case of prediction of future values is employed where the concept of prediction window, the amount of input nodes of the network is directly determined by the size of this window.

Figure 06 illustrates the amount of input neurons to predict the next value (forecast also known as a step forward) in an MLP network, considering the current value x (t) and a preview window of n elements. Illustrated in this configuration, the number of neurons needed for the output layer is only one. Another step configuration of network layers is the selection of activation functions of neurons, which are identical for all neurons of one layer.

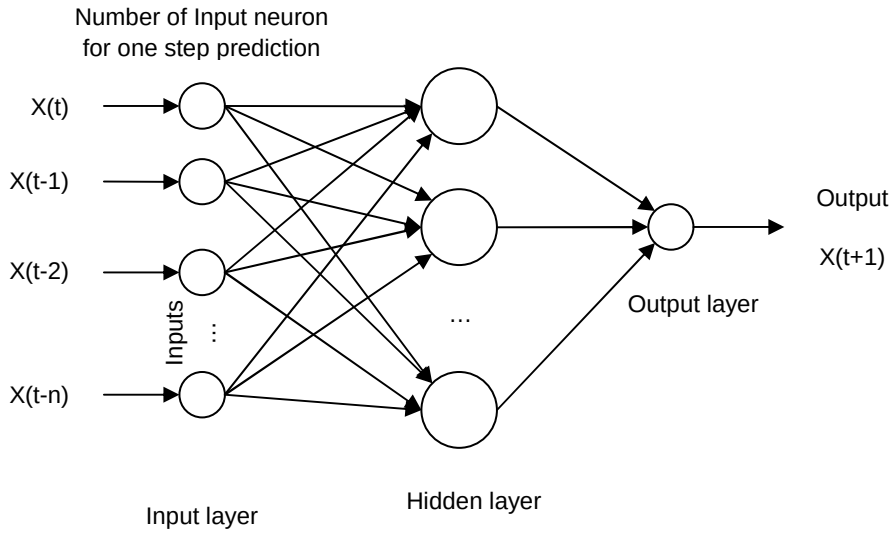


Figure 6

The pattern of interconnection of neurons in the neural network refers to how the output of each neuron of one layer is connected to the input of each subsequent layer neuron. An ANN is said to be fully connected (full interconnection) when each neuron in a layer except the input layer, has its

inputs connected to the outputs of all neurons of the layer that precedes it. If this condition is not met, the ANN is said to be partially connected.

3.2.2.1. LEARNING PROCESS

The property that is of paramount importance for a neural network is its ability to learn from their environment and improve their performance through the eight learning, whose learning algorithms differ by how the adjustment is made to a weight a synaptic neuron.

These algorithms iteratively adjust the weights of the connections between neurons until the desired pairs of input information (s) and output (s) is obtained and the relations of cause and effect can be established. Learning algorithms can be classified into three distinct paradigms: supervised learning, unsupervised learning and reinforcement learning. Move forward some ideas taken from.

The supervised learning, also known as learning with a professor, is characterized by the existence of a teacher or supervisor, the external network which has the function of monitoring the response of the same for each input vector. In unsupervised learning the network is autonomous, it does not require a "Teacher", and the process is driven by correlations in the input data.

The reinforcement learning paradigm can be considered an intermediary between supervised learning and unsupervised. The training set is formed only by the input vectors, but there is a critical external replacement the supervisor of the supervised learning.

3.3. NNS TIME SERIES FORECAST METHODS

The forecasting methodology is generally understood as a set of approaches, methods and tools for collecting time-series data for use in forecasting future values based on past values.

The operational steps are: data preparation; determination of network architecture, design of training strategies, and evaluating the results of the forecast. The preparation step involves data, among other things, the form of acquisition, preprocessing, normalization, the data structure and define the training sets and test.

The activation functions require normalized values and thus there is need for the use of procedures dedicated to preparing the input data of neural networks, adapting them to non-linearity of neurons, in order not to exceed the limits of saturation. According to these authors, the linear normalization is to consider

the minimum and maximum values of each attribute in the setting of the scale, this attribute mapping on the closed interval from zero to one (1). The equation that defines the linear normalization is described as:

$$x_n = \frac{x_t - x_{\min}}{x_{\max} - x_{\min}}$$

Where:

x_n = normalized value
x_i = attribute value to be normalized
x_{min} = minimum value of the attribute to be standardized
$x_{máx}$ = minimum value of the attribute to be standardized

It should be noted the need for a post-processing of data to end network throughput (expected value), reversing the normalization done previously. In structuring the data, pairs of data entries relating to the desired outputs should be built to the stage of supervised learning. As illustrated in Figure 6, the expected value of x (t +1), denoted as target values interrelate passed to x (t), inclusive. Similarly, a target value x (T +2) to use historical data x (t +1), following so on for other pairs of values.

Using a training set consisting of pairs of data (examples) defined above, will allow the neural network to extract knowledge (learning) necessary for its use as a predictor of future values, whose capacity can be tested on a test set of data pairs remaining. The learning will be the result of several presentations of a given set of training examples. A complete presentation of all this training set is called an epoch. Recommendations on the division of these groups in training and testing range from a factor of 90% to 10% by a factor of 50% to 50%.

3.3.1. METHODS

A diverse set of methods for forecasting time series are available. The existing modeling range from statistical models until those derived from computational intelligence technologies, which latter theory can be verified in more detail in. Moreover, such modeling can be combined, either in the form of hybrid and in committees.

The choice of algorithms and modeling described here had as its main scope the use of some artificial intelligence techniques such as fuzzy logic and artificial neural networks. These techniques allow the creation of learning machines, which are systems capable of acquiring knowledge automatically.

When managing committees, seeks to merge the knowledge gained by forecasting methods built basically by learning machines in order to reach a

global decision that aims to be superior to that achievable by any of them acting alone. For this approach to literature uses a range of terms, such as committee, ensembles, combiner, Classifier fusion, aggregation, and others to describe sets of learning machines that work to solve a coupled problem of machine learning. Aiming to incorporate the changes in classifications and concepts relating to the above combination methods, this study committee will adopt the term learning to describe some associated strategies.

3.3.1.1. BACKPROPAGATION ALGORITHM

The backpropagation training algorithm has emerged as the standard algorithm for training multilayer perceptrons (MLP neural networks). The term backpropagation arises from the fact that the algorithm is based on the backpropagation of errors to make adjustments to the weights of the intermediate layers.

Briefly, extracted from, the backpropagation algorithm derives its name from the fact that the partial derivatives of the cost function (performance measure) in relation to the free parameters (synaptic weights and bias levels) of the network are determined by backpropagation of error signals (computed by the output neurons) through the network, layer by layer.

Learning by back-propagation algorithm can be described basically by two computational steps: processing to front (spread) and backward processing (backpropagation). In the process forward (spread), the input vector is applied to the neurons of the network and its effect propagates through the network, layer by layer.

The synaptic weights w remain fixed and the signal function that appears in the output of neuron j is calculated as:

$$y_i(n) = \varphi(v_j(n))$$

where φ (v_j (n)) is the activation function of neuron j, and the n-th training pattern and v_j (n) the activation potential of neuron j. The activation function defines the output of the neuron. The basic types of activation functions are: Linear, Sigmoid and hyperbolic tangent. Regarding the potential for activation, which is represented by:

$$v_j(n) = \sum_{i=o}^{p} w_{ij}(n) y_i(n)$$

where p is the total number of inputs applied to neuron j, w_{ij} (n) is the synaptic weight which connects neuron i to neuron j, and y_i (n) is the input from neuron j

(or equivalently, functional signal at the output of neuron i). The error signal for the j-th neuron of output layer is defined by:

$$e_i = d_i(n) - y_i(n)$$

where d_j (n) is the desired response. The calculation of these error signals terminates the propagation phase of the algorithm. In the processing performed by backpropagation, the adjustment of weights is performed by the gradient method and can be described by the following equation:

$$\Delta w_{ji}(n) = -\eta\left(\frac{\partial E}{\partial w_{ji}}\right)$$

Where:

η is the learning rate parameter (which defines the magnitude of updating the weights) and $\partial E / \partial w_{ji}$ is the partial derivative of error and in the weight w_{ij}. The above equation is known as Rule Delta. Another way to represent this set of weights is expressed by:

$$\Delta w_{ji}(n) = -\eta\delta_j(n)y_i(n)$$

where the local gradient δ_j (n) defined by

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n))$$

The last equation shows that the local gradient δ_j (n) to output neuron j is equal to the corresponding error signal derived by φ'_j (v_j (n)) the activation function-related. Figure 15 illustrates a diagram of the implementation of this training to one output neuron.

3.3.1.2. LEARNING COMMITTEES

A committee of learning represents the aggregation of more than one machine learning to produce a single solution for a given computational problem. These machines will refer basically to the forecasting methods from artificial intelligence techniques, but may include other derived statistics, for example.

Following a description of the overall process:

- Each forecaster receives the input data of the problem, which may not be the same for different forecasters;
- Each forecaster (method base) is a mapping f: $R^m \rightarrow R$ output from input f;
- Each exit of each predictor is multiplied by a weight, where the sum of all weights placed on the outputs should be equal to 1;

- The outputs are selected to form the committee, so that not all the outputs of the predictors are combined, but only those that improve the performance of the committee, and
- Selected and weighted outputs are summed to compose the solution of the committee.

After the step of selecting committee members, combining the individual results of each of these can be done several ways. The most common are multiple voting or voting majority for the task of pattern classification and arithmetic mean or weighted average for the task of regression.

In a broader context and not necessarily related to the prediction of time series. A classification task is to predict a categorical value, for example, predict if the client is good or bad credit. In regression, the attribute to be provided consists of a continuous value, for example, predict the profit or loss on a loan.

3.3.1.3. LINEAR COMBINATION

The simplest strategy is to apply a combination of weighted sum of the outputs of individual committee members. The equation described below represents this strategy as a linear combination:

$$f_c = \sum_{i=1}^{M} w_i f_i$$

Where fc is the combined output, M is the number of basic methods, fi is the individual output each base method and wi is a positive weight associated with that output, and the sum of these weights equal to one. Some approaches to obtain these weights. When the weights are identical, the combination is referred to as the simple average and is expressed by

$$f_c = \frac{1}{M} \sum_{i=1}^{M} f_i$$

3.3.1.4. NON-LINEAR COMBINATION

In this strategy of combining the individual outputs of the methods are based on interrelated in a non-linear. This nonlinear mapping can be accomplished by means of remote implementations of artificial intelligence technologies, such as networks neural, fuzzy logic and hybrid approach. The equation below shows a representation adapted from

$$f_c = \psi(f_1, f_2, \ldots, f_n)$$

Where fc is the combined output (f_1, f_2, …, f_n) is the output of each individual method base and Ψ (.) is a nonlinear function.

3.3.1.5. BAGGING

Bagging is a method for generating multiple versions of forecasters and use them to obtain an aggregated forecast. This aggregation makes the average of those versions where it provides a numerical value and does a majority vote when a class provides. The multiple versions are formed by using bootstrap techniques [Efron and Tibshirani, 1993] that replicate the training set to form new training sets.

3.3.1.6. BOOSTING

Boosting [Schapire, 1990] is an approach different from the previous, where training sets are first generated from a uniform sampling with replacement. The committee members are trained sequentially and the training of a particular member is dependent on the training and performance of previously trained members. A practical limitation of this approach is that often requires a large sample of training. AdaBoost [Freund and Schapire, 1996] is a variant of boosting is probably the most widespread.

3.3.1.7. MIXTURE OF EXPERTS (ME)

The principle governing the architecture of Mixture of Experts is that various predictors (neural networks) will be able to "specialize" specific parts of the input space. A network of passage (gating network) receives the same inputs and is responsible for learning the proper combination of weights to modulate the outputs of each neural network expert.

3.3.1.8. HIERARCHICAL MIXTURE OF EXPERT (MHE)

The model of Hierarchical Mixture of Experts (MHE) is a natural extension of the approach Mixture of Experts (ME). This model is similar to a tree, where the network of passages are in various non-terminal points of the tree and the specialists are in the leaves of trees. This model differs in that the input space is divided into sets of nested subspaces, with information being combined and redistributed among experts under the control of multiple networks of passage arranged hierarchically.

3.4. THE ART OF NN TRAINING

As NN training is an art, many searchers and practitioners have worked in the field to work towards successful prediction and classification. Some principles for NN prediction and classification are of critical :

1. Clean the data prior to estimating the NN model.
2. Scale and deseasonalize data prior to estimating the model.
3. Use appropriate methods to choose the right starting point.
4. Use specialized methods to avoid local optima.
5. Expand the network until there is no significant improvement in fit.
6. Use pruning techniques when estimating NNs and use holdout samples when evaluating NNs.
7. Take care to obtain software that has in-built features to avoid NN disadvantages.
8. Build plausible NNs to gain model acceptance by reducing their size.
9. Use more approaches to ensure that the NN model is valid

3.4.1. STEPS OF NN FORECASTING:

3.4.1.1. STEP 1. DATA PREPROCESSING

A general format of data is prepared. Depending on the requirement, longer term data, e.g. weekly, monthly data may also be calculated from more frequently sampled time series. We may think that it makes sense to use as frequent data sampling as possible for experiments. However, researchers have found that increasing observation frequency does not always help to improve the accuracy of forecasting [28]. Inspection of data to find outliers is also important as outliers make it difficult for NNs and other forecasting models to model the true underlying functional. Although NNs have been shown to be universal approximators, it had been found that NNs had difficulty modeling seasonal patterns in time series [11].When a time series contains significant seasonality, the data need to be deseasonalized.

Before the data is analyzed, basic preprocessing of data is needed. In the case of days with no trading at all exist, the missing data need to be fill up manually. Heinkel and Kraus [9] stated that there are three possible ways dealing with days with no trading:

1) Ignore the days with no trading and use data for trading days.
2) Assign a zero value for the days which are no trading.
3) Build a linear model which can be used to estimate the data value for the day with no trading.

In most cases, the horizontal axis is marked by the market day instead of (or in addition to) the calendar date. Suppose we are forecasting the price of next time point. If it is on a Monday, the next time point is tomorrow or Tuesday. If it is on a Friday, the next day will be next Monday in fact it is two days later. In most of times, weekly closing price refers to each Fridays closing prices. In the event of Friday being a holiday, the most recently available closing price for the stock was used. Some researchers also pick any day as weekly prices.

Normalization is also conducted in this phase. The purpose of normalization is to modify the output levels to a reasonable value. Without such transformation, the value of the output may be too large for the network to handle, especially when several layers of nodes in the NN are involved. A transformation can occur at the output of each node, or it can be performed at the final output of the network. Original values, Y, along with the maximum and minimum values in the input file, is later entered into the equation below to scale the data

Ver paper duidelines…

3.4.1.2. STEP 2. SELECTION OF INPUT & OUTPUT VARIABLES

Select inputs from available information. Inputs and targets also need to be carefully selected. Traditionally, only changes are processed to predict targets as the return or changes are the main concerns of fund managers.

In addition, pure time series forecasting techniques require a stationary time series while most raw financial time series are not stationary. Here stationarity refers to a stochastic process whose mean, variances and covariance (first and second order moments) do not change with time.

However, after NNs have been introduced, we can use the original time series as our forecasting targets. We can let the networks to determine the units or patterns from the time series. In fact, the traditional returns are not the exact returns in real life. The inflation is not taken into account at least. These returns are named as nominal returns ignoring inflation as the inflation cannot be calculated so sensibly from daily series. After the aim has been fixed. The NN model will find out the relationship between inputs and the fixed targets. The relationship is discovered from the data rather than according to the human expectation.

3.4.1.3. STEP 3. SENSITIVITY ANALYSIS

Sensitivity Analysis is used to find out which indicator is more sensitive to the outputs. In other words, after a sensitivity analysis, we can easily eliminate the less sensitive variables from the input set. Usually, sensitivity analysis is used to

reduce the number of fundamental factors. NN or some other forecasting models are used in forecasting as the forecast target is believed to have relationship with many other series. Sometimes, the input variables may be correlated with each other. Simply using all the available information may not always enhance the forecasting abilities. This is the same as the observation that complex models do not always out perform simple ones.

The basic idea used here is that several trainings are conducted using different variables as inputs to a NN and the performance of them are then compared. If there is no much difference on the performance with or without a variable, this variable is said to be of less significance to the target and thus can be deleted from the inputs to the network. Instead of changing the number of input variables, another approach changes the values of a particular variable.

Again, several trainings are conducted using perturbed variables. Each time, a positive or negative change is introduced into the original value of a variable. If there is no much difference on the performance with or without changes of a variable, this variable is said to be of less significance.

Overfitting is another major concern in the design of a NN. When there is no enough data available to train the NNs and the structure of NNs is too complex, the NN tends to memorize the data rather than to generalize from it. Keeping the NN small is one way to avoid overfitting. One can prunes the network to a small size using the technical such as in [12].

3.4.1.4. STEP 4. DATA ORGANIZATION

The next step is Data Organization. In data preprocessing step, we have chosen the prediction goal and the inputs that should be used. The historical data may not necessarily contribute equally to the model building. We know that for certain periods the market is more volatile than others, while some periods are more stable than others. We can emphasize a certain period of data by feeding more times to the network or eliminate some data pattern from unimportant time periods.

With the assumption that volatile periods contribute more, we will sample more on volatile periods or vise versa. We can only conclude this from the experiments of particular data set. The basic assumption for time series forecasting is that the pattern found from historical data will hold in the future. Traditional regression forecasting model building uses all the data available. However, the model obtained may not be suitable for the future. When training NNs, we can hold out a set of data, out-of- sample set apart from training. After the network is confirmed, we use the out-of-sample data to test its performance. There are tradeoffs for testing and training.

One should not say it is the best model unless he has tested it, but once one has tested it one has not trained enough. In order to train NNs better, all the data available should be used. The problem is that we have no data to test the ``best'' model. In order to test the model, we partition the data into three parts. The first two parts are used to train (and validate) the NN while the third part of data is used to test the model. But the networks have not been trained enough as the third part is not used in training. The general partition rule for training, validation and testing set is 70%, 20% and 10% respectively according to the authors' experience.

3.4.1.5. STEP 5. MODEL CONSTRUCTION

Model Construction step deals with NN architecture, hidden layers and activation function. A backpropagation NN is decided by many factors, number of layers, number of nodes in each layer, weights between nodes and the activation function. In our study, a hyperbolic tangent function is used as the activation function for a backpropagation network. Similar to the situation of conventional forecasting models, it is not necessarily true that a complex NN, in terms of more nodes and more hidden layers, gives a better prediction. It is important not to have too many nodes in the hidden layer because this may allow the NN to learn by example only and not to generalize.

.When building a suitable NN for the financial application we have to balance between convergence and generalization. We use a one hidden layer network for our experiment. We adopt a simple procedure of deciding the number of hidden nodes which is also determined by the number of nodes in the input or preceding layer. For a single hidden layer NN, the number of nodes in the hidden layer being experimented are in the order of n_2, $n_2 \pm 1$, $n_2 \pm 2$, …, where n_2 stands for half of the input number. The minimum number is 1 and the maximum number is the number of inputs, n, plus 1.

In the case where a single hidden layer is not satisfactory, an additional hidden layer is added. Then another round of similar experiments for each of the single layer networks are conducted and now the new n_2 stands for half of the number of nodes in the preceding layer. Besides the architecture itself the weight change is also quite important. The learning rate and momentum rate can lead to different models. The crucial point is the choice of the sigmoid activation function of the processing neuron.

There are several variations from the standard backpropagation algorithm which aim at speeding up its relatively slow convergence, avoiding local minima or improving its generalization ability. e.g. the use of different activation functions other than the usual sigmoid function, the addition of a small positive offset to

the derivative of the sigmoid function to avoid saturation at the extremes, the use of a momentum term in the equation for the weight change.

The accuracy of approximation for NNs depends on the selection of proper architecture and weights, however, backpropagation is only a local search algorithm and thus tends to become trapped in local optima. Random selection of initial weights is a common approach. If these initial weights are located on local grades, the algorithm will likely become trapped at a local optimum. Some researchers have tried to solve this problem by imposing constraints on the search space or by restructuring the architecture of the NNs.

For example, parameters of the algorithm can be adjusted to affect the momentum of the search so that the search will break out of local optima and move toward the global solution. Another common method for finding the best (perhaps global) solution using backpropagation is to restart the training at many random points. A “fix” for certain classification problems by constraining the NN to only approximate monotonic functions. Another issue with the backpropagation network is the choice of the number of hidden nodes in the network.

While trial-and-error is a common method to determine the number of hidden nodes in a network, genetic algorithms are also often used to find the optimum number. No matter how sophisticated the NN technology, the design of a neural trading system remains an art. This art, especially in terms of training and configuring NNs for trading, and be simplified through the use of genetic algorithms. Traditional backpropagation NNs training criterion is based on goodness-of-fit which is also the most popular criterion for forecasting. However, in the context of financial time series forecasting, we are not only concerned at how good the forecasts fit their target.

3.4.1.6. STEP 6. POST ANALYSIS

In Post Analysis step, experiment results will be analysized to find out the possible relationship such as the relations between higher profit and data characters. According to the performance of each segment, we can decide how long this model can be used. In other words, how long we should retrain the NN model.

The knowledge gained from experiment will be used in future practices. A major disadvantage of NNs is that their forecasts seem to come from a black box. One cannot explain why the model made good predictions by examining the model parameters and structures. This makes NN models hard to understand and difficult for some managers to accept.

3.4.1.7. STEP 7. MODEL RECOMMENDATION

As we know, certainty can be produced through a large number of uncertainties. The behavior of an individual could not be forecast with any degree of certainty, but on the other hand, the behavior of a group of individuals could be forecast with a higher degree of certainty. With only one case of success does not mean it will be successes in the future. In our approach, we do not just train the network once using one data set.

The final NN model we suggested in using of forecasting is not a single network but a group of networks. The networks are amongst the best model we have found using the same data set but different samples, segments, and architectures.

The Model Recommendation could be either Best-so-far or Committee. Best-so-far is the best model for the testing data and in the hope of that it is also the best model for the future new data. As we cannot guarantee that the only one model is suitable for the future, we recommend a group of models as a committee in our final model. When forecast is made, instead of basing on one model, it can conclude from the majority of the committee.

As on the average, the committee suggestion for the historical data is superior to a single model. Therefore the possibility for future correctness is greater.

3.5. NN METHODOLOGY IN RELATION TO PROBLEM DOMAIN

Analysis of the problem domains of NN applications in previous research has shown that there are three main groups of problems that NN applications frequently deal with. First group consists of predicting stock performance by trying to classify stocks into the classes such as: stocks with either positive or negative returns [4, 8, 9] and stocks that perform well, neutrally, or poorly. Such NN applications give valuable support to making investment decisions, but do not specify the amount of expected price and expected profit.

NNs for stock price predictions [3, 7]. try to predict stock prices for one or more days in advance, based on previous stock prices and on related financial ratios. The third important group of NN applications in stock markets is 4 concerned with modeling stock performance and forecasting [6, 12]. Such applications are not only focused on the prediction of future values, but also on the factor significance estimation, sensitivity analysis among the variables that could impact the result, and other analyses of mutual dependencies (including portfolio models, and arbitrage pricing models).

The last group of applications frequently exists in NN research recently, although there are other, not so frequent problem domains. In order to discover

the connections between the problem domain and NN methodology used in the experiments, NN architectures are compared in relation to problem domains. Table 1 shows the NN algorithms and structures used in different applications in relation to problem domain.

As can be seen in the table, the Backpropagation algorithm is the most common NN architecture, although other algorithms are used in some applications. The three-layer structure seems to be more effective according to many authors, with the exception of two applications [6,7] where the four-layer structure outperforms other structures.

	NN architecture	
Problem domain	**Algorithm**	**NN structure**
Predicting stock performance (classification)	Backpropagation [8] Backpropagation [9[Boltzman machine [4]	2,3, and 4 layers (9-3-3-2)i [8] 6 feedforward networks [9] 2 layers (88-1) [4]
Stock price predictions	**Backpropagation [3]** **Backpropagation [7]** **Perceptron [7],** **ADALINE / MADALINE [7]**	**3 layers (24-24-1) [3]** **4 layers (10-10-10-1) [7]** **2 layers (40-1) [7]** **2 layers (40-1) [7]**
Modeling the stock performance (ANN combined forecasts)	Backpropagation [6] Hybrid approach (Backpropagation NN + expert system) [12]	4 layers (3-32-16-1) [6] 3 layers (4-7-2) [12]

Table 1

Furthermore, the Table 1 also shows that a hybrid approach is used in modeling the stock performance, while individual NN algorithms are used in other problem domains. The next characteristic of NN methodology that should be compared with the problem domains is NN's learning function. It is found that majority of applications use the sigmoid transfer function, with changeable learning parameters a and h, that are optimized in the experiments.

An important trend in the applications is combining two or more NNs into a single NN system, or incorporating other artificial intelligence methods into a NN system, such as expert systems, genetic algorithms, natural language processing. The number of Kohonen's, Hopfiled's, and other algorithms is relatively small in the stock market NN applications.

4. RESULTS

4.1. PRE-PROCESSING DATA

The pre-processed data (logarithmical and with a 2th lag differencing) was subjected to analysis in order to evaluate its condition for NN forecast analysis. The first step was to evaluate the auto dependency. The auto-correlation and partial autocorrelation functions are

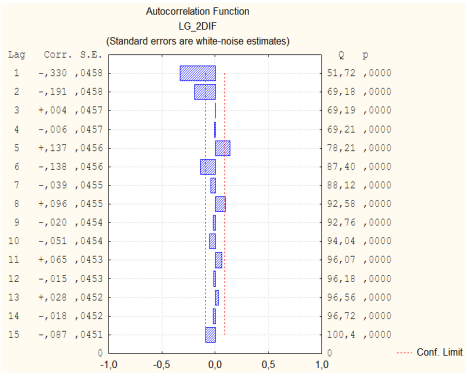


Chart 12 – Autocorrelation function

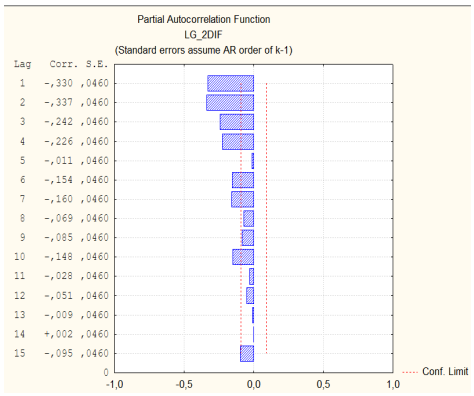


Chart 13 – Partial autocorrelation function

Charts 12 and 13 shows the inexistence of seasonality and within the limits, as the partial autocorrelation function shows lags within confidence limits with no serious depend and rejectable non normality function.

Spectral analysis show no seasonality and two relevant modes on the pre-processed series, meaning a not regular behaviour on the series. The periodogram shows no seasonal pattern and peaks on the series modes,

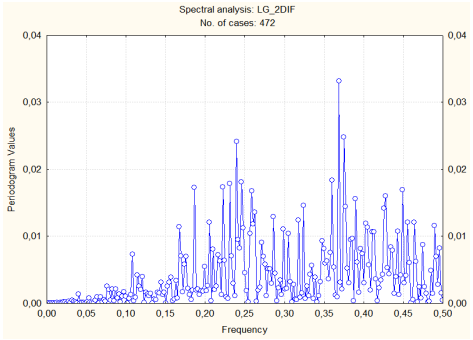


Chart 14 – Periodogram

The spectral density also shows a two mode non regular series,

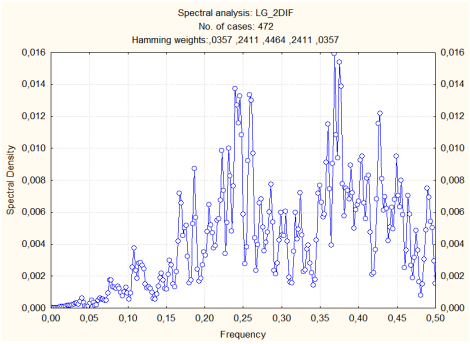


Chart 15 – Spectral density

4.2. RESIDUAL ANALYSIS

The NN prediction was made and the resulting residual was analyzed. The residual against predicted values scar plot is the following,

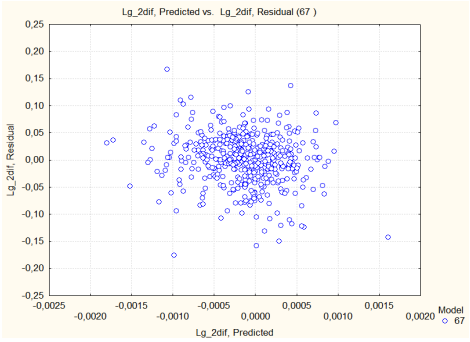


Chart 16 – Residual vs Forecasts

This charts shows the complete absense of patterns on it and normal variance. The QQ linear plot shows a clear normal approximation. The resulting plot is

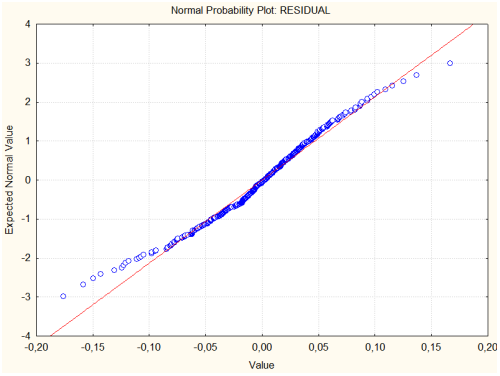


Chart 17 – QQ Plot

The previous chart is very close to perfect normality plot. The residual histogram follows,

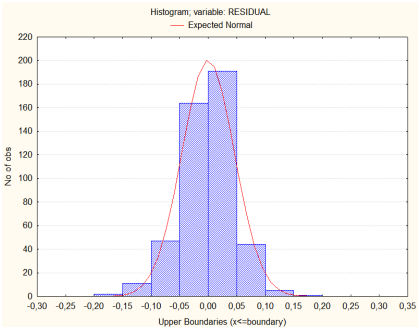


Chart 18 – Histogram

Showing a normal distribution of the residual; the residual time series is the following

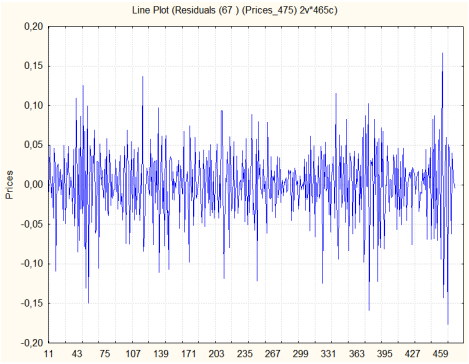


Chart 19 – Residual time plot

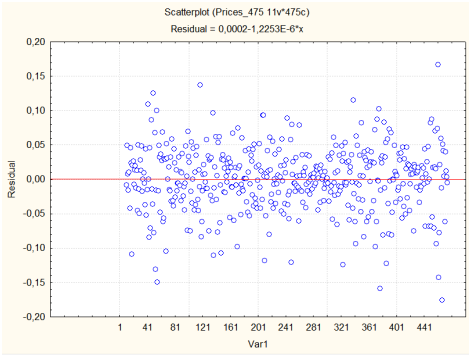


Chart 20 – Residual scatter plot

The previous charts, represents the absence of mean change and variable variance.

In order to have reliable estimations the analysis residual must be analysed in search of normal distribution and the absence of data patterns on the residuals, for it, several analysis where made and concluded that the residual does not have normal distribution and not variance patterns in it. The residual analysis concludes a god time series residual for forecasting purposes.

4.3. OVERVIEW OF TRIALS

The summary of the model trials is represented on the following table

Profile	Training performance	Select pref	Test perf	Train error	Select error	Test error	Training/
MLP s10 1:10-8-1:1	0,875760	0,860426	0,845609	0,10483	0,11856	0,14583	BP100,CG14b
MLP s10 1:10-1-1-1:1	1,000000	1,000000	1,000000	0,11971	0,13779	0,17241	BP100,CG22b
RBF s10 1:10-28-1:1	0,861976	0,876935	0,890282	20,05866	23,52110	29,83743	KM,KN,PI
Linear s10 1:10-1:1	0,811309	0,795926	0,803244	0,12140	0,13715	0,17339	PI
GRNN s10 1:10-316-2-1:1	0,985447	0,987742	0,985977	22,93223	26,45795	33,04816	SS
MLP s10 1:10-3-1:1	0,802580	0,782442	0,793616	0,09608	0,10787	0,13700	BP100,CG59b
MLP s10 1:10-8-5-1:1	0,805734	0,778880	0,795086	0,09645	0,10742	0,13717	BP100,CG59b
GRNN s5 1:5-316-2-1:1	0,987706	0,986858	0,989035	22,98456	26,43403	33,14943	SS

Table 2 – Summary of trials

4.4. MODEL RECOMMENDATION

Several models where tested and with several trials each model. The best resulting model and the best network found an acceptable performance (regression ratio=0,778880 and error=0,107415). The model summary report is the following,

Profile	Training performance	Select pref	Test perf	Train error	Select error	Test error	Training/	Training performance	Select pref	Test perf
MLP s12 1:12-6-1:1	0,083624	0,347166	0,307800	0,016022	0,068224	0,080902	BP100,CG217b	1	6	0

Table 3 - Model Summary Report

The best model selected has the following neural network,

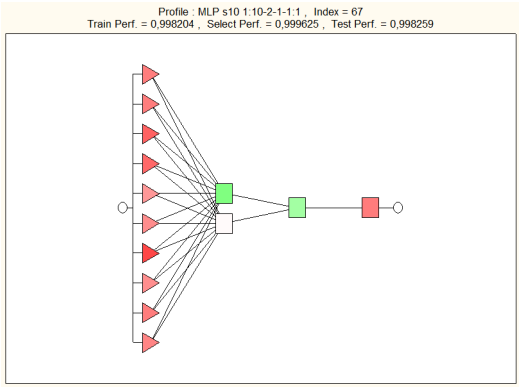


Chart 21 – Neural network

And produces the following 12 case forecast illustrated on the following graph,

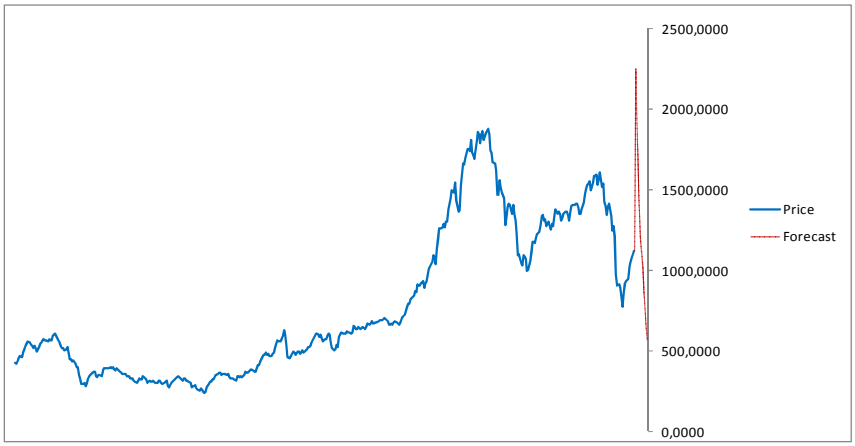


Chart 22 – NN forecasts

5. CONCLUSIONS

On the S&P500 stock series, one or two days is acceptable for forecasting, for long horizon forecast the results in forecasts with very high volatility.

The Statistica software model summary report shows an acceptable training error of 0,080902. The results show a very good performing NN model and consequently **the H1 was accepted**. However the forecasts provided by the model where just acceptable, and not good on a long term forecast. So **the H2 was rejected**.

Even though bibliography recommends one to two forecast cases on NN stock price data, a 12 case forecast where made with the selected model. The statistical numerical results resulted on an acceptable model, however the substantive analysis on the number and time series plot, shows an unreasonable long term stock price 12 forecasts; the series is severely volatile.

Authors (Nakamura, 2005) defend poor forecast proprieties for not short terms forecast with NN, this data confirm the poor long term forecasts. Other authors claim excellent results for NN forecast with S&P500 data until 1996, however the subsequent data shows a huge bi-modal crash suggesting bad forecasting results when compared with actual data subsequent to 1996 (Kulkani, 1996).

The just acceptable results, may be explained by the Input variable be only the series; (Coupelon 2008) refers a few more variables as indicators should be used as inputs in order to improve forecast capability: Relative Strength Index, Money Flow Index, Moving Average, Stochastic Oscillator and Moving Average Convergence/Divergence; as minimum indicators.

Bhardwaj et al (2010) refers NN as a potential for Stock prices with better results than traditional techniques, however, the explicit result demonstration of NN for stock forecast purposes was not found.

The results on this working paper confirmed that NN has the potential for stock prediction, with acceptable results, however confirmed the existing bibliography conclusions for a need to further development. A possible origin for the poor long term (12 cases) forecast e acceptable short term forecast may be the lack of input indicators, like as the traditional OLS regression prediction methods need good explanatory variables for better model description, the NN approach also needs not only the time series input variables but more explanatory stock phenomena indicators.

NNs, when are effectively implemented and validated, show potential for forecasting and prediction, however NNs prediction methods lacks validity; further developments are needed on validity field.

6. References

1. ADYA, M. and F. COLLOPY, *How Effective are Neural Networks at Forecasting and Prediction? A Review and Evaluation.* Journal of Forecasting, 1998. **17**: p. 481-495.
2. Bhardwaj, S. and R. Puri, *Stock Prediction: A Backpropagation Neural Network Approach.* 2010.
3. Coupelon, O., *Neural network modeling for stock movement prediction A state of the art*, in *Blaise Pascal University*. 2008.
4. Gryc, W., *Neural Network Predictions of Stock Price Fluctuations.* 2006.
5. Hamid, S.A., *Primer on using neural networks for forecasting market variables*, S.N.H. University, Editor. 2004.
6. Kulkarni, A., *Application of Neural Networks to Stock Market Prediction.* 1996.
7. Lakshminarayanam, S., *An integrated stock market forecasting model using neuralnetworks*. 2005, College of Engineering and Technology of Ohio University.
8. Lawrence, R. *Using Neural Networks to Forecast Stock Market Prices*. 1997. University of Manitoba.
9. Nakamura, E., *Inflation Forecasting using a Neural Network*, H. University, Editor. 2005.
10. Samur, Z.I. and G.T. Temur, *The Use of Artificial Neural Network in Option Pricing: The Case of S&P 100 Index Options.* World Academy of Science, Engineering and Technology, 2009.
11. Shapiro, A., *Capital Market applications of Neural Networks, Fuzzy Logic and Genetic algorithms*, U. Park, Editor. 2003.
12. YAO, J. and C.L. TAN, *Guidelines for Financial Forecasting with Neural Networks.*