Learning and Nonlinear Models (L&NLM) – Journal of the Brazilian Neural Network Society, Vol. 8, Iss. 1, pp. 3-22, 2010.
© Sociedade Brasileira de Redes Neurais (SBRN)

NEURAL NETWORKS APPLIED TO STOCK MARKET FORECASTING: AN EMPIRICAL ANALYSIS

LEANDRO S. MACIEL, ROSANGELA BALLINI
Economics Institute (IE), State University of Campinas (UNICAMP)
Pitágoras Street, 65 Cidade Universitária Zeferino Vaz
CEP 13083-857 Campinas, São Paulo, Brazil
E-mail: leandro_maciell@hotmail.com; ballini@eco.unicamp.br

Abstract

Neural networks are an artificial intelligence method for modeling complex target functions. For certain types of problems, such as learning to interpret complex real-world sensor data, artificial neural networks (ANNs) are among the most effective learning methods. During the last decade, they have been widely applied to the domain of financial time series prediction, and their importance in this field is growing. This paper aims to analyze neural networks for financial time series forecasting, specifically, their ability to predict future trends of North American, European, and Brazilian stock markets. Their accuracy is compared to that of a traditional forecasting method, generalized autoregressive conditional heteroskedasticity (GARCH). Furthermore, the best choice of network design is examined for each data sample. This paper concludes that ANNs do indeed have the capability to forecast the stock markets studied, and, if properly trained, robustness can be improved, depending on the network structure. In addition, the Ashley–Granger–Schmalancee and Morgan–Granger–Newbold tests indicate that ANNs outperform GARCH models in statistical terms.

Keywords: Artificial neural networks, finance forecasting, economic forecasting, stock markets.

1. Introduction

There is a long history of research in financial and economic modeling. Time series analysis is one of the most widely used traditional approaches in this field. There are two kinds of models to describe the behavior of time series. The first are the linear models. A linear approach to time series analysis is typically effected through one of the following methods: (a) Box–Jenkins techniques, (b) Kalman filters, (c) Brown's theory of exponential smoothing, and (d) piecewise regression. The second kinds are the nonlinear models, based on (a) Taken's theorem, (b) Markov switching models, (c) threshold autoregression, and (d) smooth transition autoregression, for example. These techniques attempt to reconstruct the time series based upon a sampling of the data to forecast future values. Although these techniques are statistically powerful, they have low success rates when used to forecast financial markets.

Recent evidence shows that financial markets are nonlinear; however, the linear methods mentioned are still able to well describe the nonlinear systems found in financial market time series analysis (Fang *et al.*, 1994). Bollerslev (1986) provides an excellent survey of the existence of nonlinearities in the financial data and develops a model to predict financial time series, called generalized autoregressive conditional heteroskedasticity (GARCH), that combines all the features observed in these series. But, the economy evolves (rather slowly) over time; this aspect cannot be easily captured by fixed specification linear models, however, and manifests itself in the form of an evolving coefficient estimate. Many factors interact in finance and economics, including political events, general economic conditions, and traders' expectations. Therefore, predicting financial and economic movements is quite difficult.

Artificial neural networks (ANNs) are a very powerful tool in modern quantitative finance and have emerged as a powerful statistical modeling technique. They provide an attractive alternative tool for both researches and practitioners. They can detect the underlying functional relations within a set of data and perform tasks such as pattern recognition, classification, evaluation, modeling, prediction, and control (Anderson and Rosenfeld, 1988; Hecht-Nielsen, 1990; Hertz *et al.*, 1991; Hiemstra and Jones, 1994). Several distinguishing features of ANNs make them valuable and attractive in forecasting. First, ANNs are nonlinear data-driven approaches. They are capable of modeling nonlinear systems without an *a priori* knowledge about the relations between the input and output variables. The nonparametric ANN model may be preferred over traditional parametric statistical models in situations where the input data do not meet the assumptions required by the parametric model, or when large outliers are evident in the dataset (Lawrence, 1991; Rumelhart and McClelland, 1986; Waite and Hardenbergh, 1989; Wasserman, 1993). Second, ANNs are universal functions approximations. It has been shown that a neural network can approximate any continuous function to any accuracy desired (Hornik, 1993; Hornik *et al.*, 1987). Third, ANNs are able to generalize. After learning the data presented, ANNs can often correctly infer the unseen part of a population, even if the sample data contain noisy information. Neural networks are able to capture the underlying pattern or autocorrelation structure within a time series even when the underlying law governing the system is unknown or too complex to describe.

Because of their pattern recognition abilities, ANNs have been applied successfully in many fields and are increasingly being used in economics, as well as in business research. Wong and Selvi (1998) classify pertinent articles by year of publication, application area, journal, various decision characteristics (problem domain, decision process phase, level of management, level of task interdependence), means of development, integration with other technologies, and major contribution. Zang *et al.* (1998) survey articles that address modeling issues when ANNs are applied to forecasting. The authors summarize the most frequently cited advantages and disadvantages of the ANN models. Chatterjee *et al.* (2000) provide an overview of the ANN system and its wide-ranging use in financial markets. Their work further discusses the superiority of ANNs over traditional methodologies. The study concludes with a description of the successful use of ANNs by various financial institutions. Edward Gately (1996), in his book *Neural Networks for Financial Forecasting*, describes the general methodology required to build, train, and test a neural network using commercially available software.In addition, Shapiro (2003) describes capital market applications of neural networks, fuzzy logic, and genetic algorithms.

Garcia and Gencay (2000), Gencay (1998), and Qi and Madala (1999) employ ANNs in stock market predictions. Qi and Wu (2003), who employ an ANN model with monetary fundamentals, find that their model cannot supass the random walk model. Alternatively, Kiani (2005) and Kiani *et al.* (2005) use ANN models with macroeconomic time series and find that these outperform the linear as well as other nonlinear models employed. O'Connor and Madden (2005) evaluate the effectiveness of using ANNs with external indicators, such as commodity prices and currency exchange rates, in predicting movements in the Dow Jones Industrial Average index. Their results show that there are a few benefits to using these indicators over traditional methods based on historical data output only.

Dutta *et al.* (2006) discuss modeling the Indian stock market (price index) using ANNs. The authors study the efficacy of ANNs in modeling the Bombay Stock Exchange Sensex weekly closing values. They use root mean squared error (RMSE) and mean absolute error (MAE) as indicators of performance for two kinds of ANN structures. They conclude that the ANN with more input values can improve the verified results.[1] Most recently, Faria et al. (2009) performed a predictive study of the Ibovespa through ANNs and an adaptive exponential smoothing method to compare the forecasting performances of both methods on this market index and evaluate their accuracy in predicting the sign of market returns. The authors show that both methods produce similar results regarding the prediction of index returns. They use two different metrics to evaluate forecasting accuracy: RMSE and the measure N(tend) that represents the correct tendencies number achieved by the model, that is, the number of times the predictions follow the real tendencies of the market. Finally, Lin and Yu (2009) investigate the profitability of using ANN predictions that are transformed into a simple trading strategy, whose profitability is evaluated against a simple buy–hold strategy. The authors adopt this approach to analyze the Taiwan Weighted Index and the Standard & Poor's (S&P) 500 and find that the trading rule based on ANNs generates higher returns than the buy–hold strategy.[2]

In addition, ANNs have been successfully applied to predict important financial and market indexes, the S&P 500 and the Nikkei 225 Index, among others (Chen, 1994; Enke and Thawornwong, 2005; Huang *et al.*, 2007; Huarng and Yu, 2006; Refenes *et al.*, 1994; Yu and Huarng, 2008), but these works focus only on special markets that conform to different kinds of ANN architectures and do not necessarily compare with traditional statistical methods such as autoregressive integrated moving average (ARIMA)–GARCH models. Besides, none of these works evaluate the differences between competitive methods in statistical terms, but only by using traditional metrics such as RMSE and MAE.

This paper aims to analyze and examine the use of neural networks to predict future trends of North American, European, and Brazilian stock market indexes, namely, the Dow Jones and S&P 500 (United States), the DAX (Germany), the CAC 40 (France), the FTSE (United Kingdom), the IBEX 35 (Spain), the PSI 20 (Portugal), and Ibovespa (Brazil). We provide a detailed discussion of the application of neural networks to forecasting stock market economic indicators. As a comparison, we analyze a GARCH model applied to each series to evaluate the accuracy of ANNs, according to traditional performance measurements and statistical tests such as the Ashley–Granger–Schmalancee (AGS) and Morgan–Granger–Newbold (MGN) tests. An exploration about how ANNs can incorporate the heteroskedasticity of financial time series is presented to verify the model's robustness.

This paper is organized as follows. Section 2 discusses neural network applications in stock market index price forecasting. Sections 3 and 4 describe GARCH and neural network models, respectively. Section 5 discusses the structure of the neural network applied. Section 6 compares the performances of the methods. Finally, Section 7 presents our conclusions

2. Applications in Stock Market Index Forecasting

The stock market is one of the most popular investments, owing to its high expected profit. However, the higher the expected profit, the higher the implied risk. The stock market, which has been investigated by various studies, is a rather complicated

[1] A number of attempts have been made to apply ANNs to the task of modeling security prices (see, e.g., Cao *et al.*, 2005; Jasic and Wood, 2004; Kaastra and Boyd, 1996; Lam, 2004; Nygren, 2004).

[2] Pan *et al.* (2004) present an application to predicting the Australian stock market using feedforward neural networks with the objective of developing an optimal architecture for this purpose.

environment. There are three degrees of market efficiency. The strong form of the efficient market hypothesis states that all information that is knowable is immediately factored into the market price as a security. If this is true, then all of those price predictors are wasting their time, even if they have access to private information. In the semi-strong form of the efficient market hypothesis, all public information is considered to have been reflected in the price immediately as it became known, but possessors of private information can use that information for profit. The weak form holds only that any information gained from examining past trading history is reflected in the price as a security. Of course, past trading history is public information, implying that the weak form is a specialization of the semi-strong form of the efficient market hypothesis, which itself is a specialization of the strong form.

Stock market fluctuations are the result of complex phenomena, whose effect translates into a blend of gains and losses that appear in a stock market time series that is usually predicted by extrapolation. The periodic variations follow either seasonal patterns or business cycles in the economy. Short-term and day-to-day variations appear at random and are difficult to predict, but they are often the source of stock trading gains and losses, especially in the case of day traders.

Numerous investigations have given rise to different decision support systems for the sake of providing investors with optional predictions. Since a long time, many stock markets experts have employed technical analysis for better predictions. Generally speaking, a technical analysis derives a stock's movements from the stock's own historical value. The historical data can be used directly to form support and resistance levels, or they can be plugged into many technical indicators for further investigation. Conventional research addressing this problem has generally employed time series analysis techniques— that is, mixed autoregression moving average (ARMA) methods—as well as multiple regression models (Huang *et al.*, 2005). Considerable evidence exists that shows that stock market price is to some extent predictable (Lo and MacKinlay, 1988).

2.1. Input Variables

There are two kinds of theoretical approaches to determine the input variables for stock market index forecasting with neural networks. The first one introduces the relations between the stock market index price and other macroeconomic indicators. The second one introduces nonlinearity in the relation between stock prices, dividends, and trading volume.

Chen (1991) studies the relation between changes in financial investment opportunities and changes in the economy. The author provides additional evidence that variables such as the default spread, term spread, one-month T-bill rate, lagged industrial production growth rate, and dividend–price ratio are important determinants of the future stock market index. This study interprets the ability of these variables to forecast the future stock market index in terms of their correlations with changes in the macroeconomic environment. Fama and French (1993) identify the overall market factor, factors related to firm size, and book-to-market equity as three common risk factors that seem to explain average returns on stocks and bonds. Ferson and Schadt (1996) show that the omission of variables such as the lagged stock index and previous interest rates can lead to erroneous results. Sitte and Sitte (2000) discuss the predictive ability of time delay neural networks for the S&P 500 index time series.

The vector autoregression (VAR) method is mainly used to investigate the relations between variables. Its advantage is that multiple variables can be investigated at the same time and their interdependence can be tested automatically with sophisticated statistically significant levels. Ao (2003a, b) find that (1) HK depends on its past prices, JP, NASDAQ, S&P, and DJ; (2) AU depends on its past prices, S&P, and DJ; (3) depends on its past prices, HK, NASDAQ, S&P, and DJ; (4) JP depends on its past prices, NASDAQ, S&P, and DJ; (5) DJ depends on its past prices and NASDAQ; and (6) S&P depends on its past prices and NASDAQ. The results from VAR modeling suggest that, for Asian markets, the relevant information is the stock's own historical values as well as the stock's movements from the U.S. markets. It is also positive to know the extent and time-dependent nature of market dynamics when we draw the correlation diagram of the local market with U.S. markets. Further investigation tells us that, at time of low correlation, such as in the late '90s during the Asian financial crisis, the Hong Kong market (and, similarly, other Asian markets) is dominated by local events, such as currency problems. At other periods, the local market is greatly correlated with the U.S. markets.

In summary, the set of potential macroeconomic indicators is as follows: the term structure of interest rates (TS), the short-term interest rate (ST), the long-term interest rate (LT), the consumer price index (CPI), industrial production (IP), government consumption (GC), private consumption (PC), the gross national product (GNP), and the gross domestic product. These are the most easily available input variables that are observable to a forecaster. Though other macroeconomic variables can be used, the general consensus in the literature is that the majority of useful information for forecasting is subsumed by interest rates and lagged predictive variables. The term structure of interest rates, that is, the spread of long-term bond yields over short-term bond yields, may have some power in forecasting the stock index.

This paper utilizes as input variables the historical data of each series studied,[3] shown in Table 1. The goal is to analyze the influence of the ANN structure on the forecast results. After determining the best structure, we compare the performances of the ANN and GARCH models.

Table 1. Stock market indexes that form the sample.

Country	Index	Variable
United States	Dow Jones	DOW
United States	S&P 500	S&P
Germany	DAX	DAX
France	CAC 40	CAC
United Kingston	FTSE	FTSE
Spain	IBEX 35	IBEX
Portugal	PSI 20	PSI
Brazil	Ibovespa	IBOV

3. GARCH Models

The ARIMA models have one severe drawback: They assume that the volatility[4] of the variable being modeled (e.g., stock price) is constant over time. In many cases this is not true. Large differences (of either sign) tend to be followed by large differences. In other words, the volatility of asset returns appears to be serially correlated (Campbell *et al.*, 1997). The autoregressive conditional heteroskedasticity (ARCH) model was developed to capture this property of financial time series. The ARCH[5] process is defined as

$$\text{ARCH (q): } y_t = a + \sigma_t \varepsilon_t \tag{1}$$

$$\sigma_t = \sqrt{\alpha_0 + \sum_{i=1}^{q} \alpha_i y_{t-i}^2} \tag{2}$$

where σ_t is the conditional standard deviation of y_t, given the past values of this process, and a is a constant. The ARCH(q) process is uncorrelated and has a constant mean and a constant unconditional variance (α_0), but its conditional variance is nonconstant. This model has a simple intuitive interpretation as a model for volatility clustering: Large values of past squared returns (y_{t-i}^2) give rise to large current volatility (Martin, 1998).

The ARCH(q) model is a special case of the more general GARCH(p,q) model:

$$\text{GARCH(p,q): } y_t = a + \sigma_t \varepsilon_t \tag{3}$$

$$\sigma_t = \sqrt{\alpha_0 + \sum_{i=1}^{p} \alpha_i y_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2} \tag{4}$$

In this model, current volatility depends upon the volatilities for the previous *q* days and the squared returns for the previous *p* days.

A long and vigorous line of research has followed the basic contributions of Engle and Bollerslev (developers of the ARCH and GARCH models, respectively), leading to a number of variants of the GARCH(p,q) model, including power GARCH (PGARCH) models, exponential GARCH (EGARCH) models, threshold GARCH (TGARCH) models, and other models that incorporate so-called *leverage effects*. Leverage terms allow for a more realistic modeling of the observed asymmetric behavior of returns, according to which a "good news" price increase leads to lower subsequent volatility, while "bad news" decreases in price lead to a subsequence increase in volatility. It is also worth mentioning two-component GARCH models, which reflect differing short- and long-term volatility dynamics, and GARCH-in-the-mean (GARCH-M) models, which allow the mean value of returns to depend upon volatility (Martin, 1998).[6]

[3] The data were obtained from http://finance.yahoo.com, accessed on August 17, 2008.
[4] Volatility is synonymous of standard deviation.
[5] This section is based upon the work of Ruppert (2001).
[6] In this work, the GARCH(1,1) model was utilized as a result of correlation and autocorrelation analysis.

4. Neural Networks

Neural networks learning methods provide a robust approach to approximating real-, discrete-, and vector-valued target functions. For certain types of problems, such as learning to interpret complex real-world sensor data, ANNs are among the most effective learning methods known (Mitchell, 1997). One motivation for ANN systems is to capture this kind of highly parallel computation based on distributed representations. Most ANN software runs on sequential machines emulating distributed processes, although faster versions of the algorithms have also been implemented on highly parallel machines and specialized.

4.1. Basic Definitions

The multilayer perceptron (MLP) is the most commonly used type of artificial network structure. It consists of several layers of processing units (also termed neurons or nodes). The input values (input data) are fed to the neurons in the so-called *input layer,* which processes the input values, and the output values of these neurons are then forwarded to the neurons in the *hidden layer*. Each connection has an associated parameter indicating its strength, the so-called *weight*. By changing the weights in a specific manner, the network can “learn” to map patterns presented at the input layer to target values on the output layer. The procedure by means of which this weight adaptation is performed is called the *learning* or *training algorithm.*

Usually, the data available for training the network are divided into (at least) two non-overlapping parts: the so-called *training* and *testing sets*. The commonly large training set is used to “teach” the network the desired target function. Then the network is applied to the data in the test set to determine its *generalization ability*, that is, its ability to derive correct conclusions about the population properties of the data from the sample properties of the training set (e.g., if a network has to learn a sine function, it should produce correct results for all real numbers and not only for those in the training set). If the network is not able to generalize but, instead, learns the individual properties of the training patterns without recognizing the general features of the data (i.e., produces correct results for training patterns but has a high error rate in the test set), it is said to be *overfitted* or to be subject to *overfitting.*

4.2. Neural Network Properties

ANN learning is well suited to problems in which the training data correspond to noisy, complex sensor data, such as input from cameras and microphones. It is also applicable to problems for which more symbolic representations are often used, such as decision tree learning tasks. In this case ANNs and decision tree learning produce results of comparable accuracy (Haykin, 2001).

The backpropagation algorithm is the most commonly used ANN learning technique. It is appropriate for problems with the following characteristics (Mitchell, 1997).

- *Instances are represented by many value pairs.* The target function to be learned is defined over instances that can be described by a vector of predefined features, such as pixel values. These input attributes can be highly correlated or independent of one another. The input values can be any real values.
- *The target function output can be discrete valued, real valued, or a vector of several real- or discrete-valued attributes.*
- *The training examples may contain errors.* ANN learning methods are quite robust to noise in the training data.
- *Long training times are acceptable.* Network training algorithms typically require longer training times than, say, decision tree learning algorithms. Training times can range from a few seconds to many hours, depending on factors such as the number of weights in the network, the number of training examples considered, and the settings of various learning algorithm parameters.
- *Fast evaluation of the learning target function can be required.* Although ANN learning times are relatively long, evaluating the learning network, to apply it to a subsequent instance, is typically very fast.
- *The ability of humans to understand the learning target function is not important.* The weights learned by neural networks are often difficult for humans to interpret. Learned neural networks are less easily communicated to humans than learned rules.

4.3. MLPs

A network consists of a set of nodes that constitute the input layer, one or more hidden layers of nodes, and an output layer of nodes. The input propagates through the network in a forward direction, on a layer-by-layer basis. These neural networks are referred to as MLPs. In the mid-1980s, ANNs were mostly studied by employment of the *error backpropagation* (EBP) learning algorithm in combination with multilayer networks (Rumelhart and McClelland, 1986). Basically, the EBP process consists of two phases through the different layers of the network: a forward pass and a backward pass. In the forward pass, an input vector is applied to the nodes of the network, and its effect propagates through the network, layer by layer. Finally, a set of outputs is produced as the actual network response. During this phase, the weights are all fixed. During the backward pass, the weights are all adjusted in accordance with the error correction rule. Specifically, the actual response of the network is subtracted from a desired response, producing an error signal. This error is propagated backward through the network, against the direction of synaptic connections—hence the name EBP. The synaptic weights are adjusted so as to make the actual network response closer to the desired response (Haykin, 2001).

An MLP network consists of at least three layers: an input layer, one or more hidden layers, and an output layer. The nodes are connected by links associated with real numbers, named *weights*. Each node takes on multiple input values, processes them, and produces an output, which can be "forwarded" to other nodes. Given a node *j*, its output is equal to

$$o_j = transfer(\sum x_{ji} w_{ji}) \tag{5}$$

where o_j is the output of node *j*, x_{ji} is the *i*th input to unit *j*, w_{ji} is the weight associated with the *i*th input to *j*, and *transfer* is the nonlinear transfer function responsible for transferring the weighted sum of the inputs to some value that is given to the next node.[7] A neuron can have an arbitrary number of inputs, but only one output. By changing the weights of the links connecting the nodes, the ANN can be adjusted to approximate a particular function.

4.4. Learning Algorithms

Usually, the weights of an ANN must be adjusted using some learning algorithm so that the ANN is able to approximate the target function with sufficient precision. This section presents a stochastic gradient descent backpropagation learning algorithm, as follows.[8] The term *neural network* refers to an MLP trained with this learning algorithm, often called backpropagation or EBP. Assume that an ANN uses the error function

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2 \tag{6}$$

where o_{kd} is the output value produced by output neuron *k*, t_{kd} is the desired (correct) value this neuron should produce, and *D* denotes the set of all training patterns, that is, $E(\vec{w})$ is the sum of the prediction errors for all training examples. The prediction errors of the individual training examples are, in turn, equal to the sum of the differences between the output values produced by the ANN and the desired (correct) values, where $\vec{w}$ is the vector of the weights of the ANN.

The goal of a learning algorithm is to minimize $E(\vec{w})$ for a particular set of training examples. There are several ways to achieve this, one of them being the so-called *gradient descent* method, which basically works as follows (Schraudolph and Cummins, 2002):

1. Choose some (random) initial values for the model parameters.
2. Calculate the gradient *G* of the error function with respect to each model parameter.
3. Change the model parameters so that we move a short distance in the direction of the greatest rate of decrease of the error, that is, in the direction of –*G*.
4. Repeat steps 2 an 3 until *G* gets close to zero.

Let $G = \nabla f(x)$, the gradient of the function *f*, be the vector of first partial derivatives,

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n} \right) \tag{7}$$

In our case, $G = \nabla E(\vec{w})$ (i.e., the derivative of the error function *E* with respect to the weight vector $\vec{w}$). With this in mind, we now explore the *gradient descent backpropagation* (EBP) learning algorithm. First, a neural network is created and the parameters are initialized (the weights are set to small random numbers). Then, until the termination condition (e.g., the mean squared error of the ANN is less than a certain error threshold) is met, all training examples are taught the ANN. The inputs of each training example are fed to the ANN and processed from the input layer, over the hidden layer(s), to the output layer. In this way, a vector *o* of output values produced by the ANN is obtained.

[7] There are a several types of transfer functions, discussed in Haykin (2001).
[8] See more learning algorithms in Haykin (2001).

In the next step, the weights of the ANN must be adjusted. Basically, this is accomplished by moving the weight in the direction of the steepest descent of the error function. This happens by adding to each individual weight the value

$$\Delta w = \eta \delta_j x_{ji} \tag{8}$$

where η is the *learning rate* that determines the size of the step that we use to move toward the minimum of *E*, and δ_j represents the error term of the neuron *j*.[9] The learning rate can be thought of as the lengths of the arrows.[10]

Many improvements of this algorithm, such as momentum terms and weight decay, are described in the appropriate literature (Bishop, 1996). Nevertheless, MLPs in combination with the stochastic gradient descent learning algorithm are the most popular ANNs in practice.[11] Another important feature of this learning algorithm is that it assumes a quadratic error function, and therefore assumes there is only one minimum. In practice, the error function can have—apart from the global minimum—multiple local minima. There is a danger the algorithm will land in one of the local minima and thus not be able to reduce the error to the highest extent possible by reaching a global minimum. The next section describes an ANN design for our data and a step-by-step comparison with a GARCH model.

5. ANN Design in Stock Market Forecasting

The methodology described in this section is based upon Kaastra and Boyd (1996). The design of a neural network that successfully predicts a financial time series is a complex task. The individual steps of this process are as follows:

1. Variable selection.
2. Data collection.
3. Data preprocessing.
4. Data partitioning.
5. Neural network design.
6. Training the ANN.

A detailed description of each step is presented below.

5.1. Variable Selection

Success in designing a neural network depends on a clear understanding of the problem (Gately, 1996). Knowing which input variables are important in the market being forecast is critical. This is easier said than done, because the very reason for relying on a neural network is its powerful ability to detect complex nonlinear relations among a number of different variables. However, economic theory can help choose variables that are likely important predictors. At this point in the design process, the concern is about the raw data, from which a variety of indicators will be developed. These indicators will be derived from the actual inputs to the neural networks (Kaastra and Boyd, 1996).

The financial researcher interested in forecasting market prices must decide whether to use both technical and fundamental economic inputs from one or more markets. Technical inputs are defined as lagged[12] values of the dependent variable[13] or as indicators calculated from the lagged values. The model applied in this paper uses the lagged values of the dependent variables as a result of correlation and autocorrelation analysis.[14] For each series we plot the sample autocorrelation functions (ACF) and partial autocorrelation functions (PACF), as in Box *et al.* (1994). Then, according to the lags and their respective ACF and PACF values, we select those lags that indicate significant correlation. In the next step, we apply all the selected lags to the network, as described above. Nevertheless, we decrease the number of lags for each series, and apply the standard Bayesian information criterion (BIC) model selection procedure (Schwartz, 1978), according to the RMSE metric. That is, the choice of inputs is based on correlation analysis and the BIC procedure. Table 2 shows the input structures performed for the data utilized.

[9] Stochastic gradient descent backpropagation learning algorithm derivatives are discussed in Haykin (2001).

[10] Usually, $\eta \in \Re$, $0 < \eta \leq 0.9$. Note that too large an η leads to oscillation around the minimum, whereas too small an η can lead to the ANN's slow convergence.

[11] This structure was utilized in the present work.

[12] *Lagged* means pertaining to an element of the time series in the past. For example, at time *t*, the values $y_{t-1}, y_{t-2}, y_{t-p}$ are said to be lagged values of the time series *y*.

[13] The dependent variable is the variable whose behavior is being modeled or predicted (Dougherty, 1992).

[14] Such models have outperformed traditional ARIMA-based models in price forecasting, although not in all studies (Sharda and Patil, 1994; Tang *et al.*, 1991).

Table 2. Variable selection.

Variables	Input Past Closing Values
DOW	$DOW_{t-1}, DOW_{t-2}, DOW_{t-3}$
S&P	$S\&P_{t-1}, S\&P_{t-2}$
DAX	DAX_{t-1}, DAX_{t-2}
CAC	$CAC_{t-1}, CAC_{t-2}, CAC_{t-3}, CAC_{t-4}$
FTSE	$FTSE_{t-1}, FTSE_{t-2}, FTSE_{t-3}$
IBEX	$IBEX_{t-1}, IBEX_{t-2}$
PSI	$PSI_{t-1}, PSI_{t-2}, PSI_{t-3}, PSI_{t-4}$
IBOV	$IBOV_{t-1}, IBOV_{t-2}, IBOV_{t-3}$

The frequency of the data depends on the researcher's objectives. A typical off-floor trader in the stock or commodity futures markets would likely use daily data if designing a neural network as a component of an overall trading system. An investor with a longer horizon may use weekly or monthly data as inputs to the neural network, rather than a passive buy and hold strategy (Kaastra and Boyd, 1996), to formulate the best asset mix.

5.2. Data Collection

The research must consider cost and availability when collecting data for the variables chosen in the previous step. Technical data are readily available from many vendors at a reasonable cost, whereas fundamental information is more difficult to obtain. Time spend collecting data cannot be used for preprocessing, training, or evaluating network performance. The vendor should have a reputation for providing high-quality data; however, all data should still be checked for errors by examining day-to-day changes, ranges, logical consistency, and missing observations (Kaastra and Boyd, 1996). Missing observations, which are common, can be handled in a number of ways. All missing observations can be dropped, or a second option is to assume that the missing observations remain the same by interpolating or averaging from nearby values. In this work, we assume that there are no missing observations in the sample and that some values, which can be viewed as *outliers*, are present in the data, because we aim to model stock markets mainly in turbulent scenarios, characterized by low losses.[15]

5.3. Data Processing

As in most other neural network applications, data processing is crucial in achieving good predictive performance when applying neural networks to the prediction of financial time series. The input and output variables for which the data are collected are rarely fed into the network in raw form. At the very least, the raw data must be scaled between the upper and lower bounds of the transfer functions (usually between zero and one minus one and one). Two of the most common data transformations in both traditional and neural network forecasting are first differencing and taking the logarithm of a variable. First differencing, or using changes in a variable, can be used to remove a linear trend of data. Logarithmic transformation is useful for data that can take on both small and large values. Logarithmic transformations also convert multiplicative or ratio relations to additive ones, which is believed to simplify and improve network training (Masters, 1993).[16] In this work we use the logarithmic transformation of the return,

$$R_t = \ln\left(\frac{Index_t}{Index_{t-1}}\right) \tag{9}$$

where R_t represents the normal logarithm of the returns. This approach is especially useful in financial time series analysis, and produce good results, according to the literature (see Fama, 1965; Granger and Morgenstern, 1970). In addition, the returns behavior is more closely approximated by a normal probability distribution, but, as will be shown here, this is a very hardly hypothesis.

5.4. Data Partitioning

Common practice is to divide the time series into three distinct sets, the training, testing, and validation[17] (out-of-sample) sets. The training set is the largest and is used by neural networks to learn the patterns present in the data. The testing set, ranging in size from 10% to 30% of the training set, is used to evaluate the generalization ability of a supposedly trained network. A final check on the validation set chosen must strike a balance between obtaining a sufficient sample size to evaluate a trained network and having sufficient remaining observations for both training and testing. The validation set should consist of the most recent contiguous observations, because the past values was applied to test the neural network and, for evaluate the

[15] The sample ranges from January 12, 2000, to July 27, 2008, with daily data.

[16] Another popular data transformation is to use the ratios of the input variables (see Topek and Querin, 1984).

[17] Some studies call the validation set the *testing set* .

generalization capability of the model, the most recent observations consists as a validation sample as well as in standard time series models. This work breaks down the sets as follows:

1. Training set: 80%.
2. Testing set: 15%.
3. Validation set: 5%.

5.5. Neural Network Design

There are an infinite number of ways to construct a neural network. *Neurodynamics* and *architecture* are two terms used to describe the way in which a neural network is organized. The number of input neurons is one of the easiest parameters to select once the independent variables have been reprocessed, because each independent variable is represented by its own input neuron.[18] The tasks of selecting the number of hidden layers, the number of neurons in the hidden layers, and the number of input neurons, as well as the transfer functions, are much more difficult.

5.5.1. Hidden Layers

Hidden layers provide the network with its ability to generalize. In practice, neural networks with one and occasionally two hidden layers are widely used and have performed very well. Increasing the number of hidden layers also increases the computation time and the danger of overfitting, which leads to poor out-of-sample forecasting performance. In the case of neural networks, the number of weights, which is inexorably linked to the number of hidden layers and neurons, and the size of the training set (number of observations) determine the likelihood of overfitting (Baum and Haussler, 1989). Here we analyzed neural networks structure with one and two hidden layers to a comparison.

5.5.2. Hidden Neurons

Despite its importance, there is no "magic" formula for selecting the optimum number of hidden neurons, and therefore researchers fall back on experimentation. However, some rules of thumb have been advanced. A rough approximation of the optimum number of hidden neurons can be obtained by the geometric pyramid rule proposed by Masters (1993). For a three-layer network with n input neurons and m output neurons, the hidden layer would have $\sqrt{n \cdot m}$ neurons. Baily and Thompson (1990) suggest that the number of hidden layer neurons in a three-layer neural network should be 75% of the number of input neurons. Katz (1992) indicates that the optimal number of hidden neurons will generally be found between one-half to three times the number of input neurons. Ersoy (1990) proposes doubling the number of hidden neurons until the network's performance on the testing set deteriorates. Klimasauskas (1993) suggests that there should be at least five times as many training facts as weights, which sets an upper limit on the number of input and neurons. Because of these features, this work applies different structures to all the data, chosen randomly, with 2, 3, 4, 5, and 6 neurons in the hidden layer to describe the best structure according to the index.

5.5.3. Output Neurons

Deciding on the number of neurons is somewhat more straightforward, since there are compelling reasons to always use only one output neuron. Neural networks with multiple outputs, especially if these outputs are widely spaced, produce inferior results compared to networks with a single output (Masters, 1993). In this works, we applied a network with the output layer composed by one neuron and it means the closed price one step ahead.

5.5.4. Transfer Function

The majority of current neural network models use the sigmoid transfer function, but others, such as the hyperbolic tangent, arc tangent, and linear transfer functions, have also been proposed (Haykin, 2001). Linear transfer functions are useful for nonlinear mapping and classification. Levich and Thomas (1993) and Kao and Ma (1992) find that financial markets are nonlinear and have memory, suggesting that nonlinear transfer functions are more appropriate. Transfer functions such as the sigmoid are commonly used for time series data, because they are nonlinear and continuously differentiable, desirable properties for network learning. In this study, the sigmoid transfer function is applied in the proposed network.[19]

5.6. Training the ANN

Training a neural network to learn patterns in the data involves iteratively presenting it with examples of the correct known answers. The objective of training is to find the set of weights between the neurons that determine the global minimum of the error function. Unless the model is overfitted, this set of weights should provide a good generalization. The backpropagation network applied in this work uses the gradient descent training algorithm, which adjusts the weights to move down the steepest slope of the error surface. Finding the global minimum is not guaranteed, since the error surface can include many local

[18] Each set of data has its specific input variables, as described in Table 1.
[19] The sigmoid transfer function is the default in the Neural Network Toolbox in MATLAB®.

minima, in which the algorithm can become "struck." This section discusses when to stop training a neural network and the selection of learning rates and momentum values.

5.6.1. Training Iterations

Many studies that mention the number of training iterations report convergence from 85 to 5,000 iterations (Deboeck, 1994; Klaussen and Uhrig, 1994). However, the range is very wide, since 50,000 and 191,400 iterations (Klimasauskas, 1993; Odom and Sharda, 1992) and training times of 60 hours have also been reported. Training is affected by many parameters—the choice of learning rate and momentum values and proprietary improvements to the backpropagation algorithm, among others—which differ between studies, and it is therefore difficult to determine a general value for the maximum number of runs.

In addition, the numerical precision of the neural network software can affect training, because the slope of the error derivative can become very small, causing some neural networks programs to move in the wrong direction due to round-off errors, which can quickly build up in the highly iterative training algorithm. It is recommended that studies determine their own particular problem and test as many random starting weights as computational constraints will allow (Kaastra and Boyd, 1996). We utilize 100, 250, 500, 800, and 1,200 randomly selected iterations to choose the best performance for each index.

5.6.2. Learning Rate

During training, a learning rate that is too high is revealed when the error function changes drastically without showing continued improvement. A very low learning rate also requires more training time. In either case, the researcher must adjust the learning rate during training, or "brainwash" the network by randomizing all weights and changing the learning rate for the new run through the training set. The initial learning rates used in this work vary widely, from 0.1 to 0.9. Most neural network software programs provide default values for learning rates that typically work well. Common practice is to start training with a higher learning rate, such as 0.7, and decrease as training proceeds. Many network programs will automatically decrease the learning rate as convergence is reached (Haykin, 2001).

6. Comparison Analysis

This section presents the neural network structure implemented for the data that results in a minimum number of errors. In addition, the results of the ANN and GARCH models are described for comparison. Table 3 shows the best neural network structure performed for each index studied.

Table 3. Neural network design.

Index	**Inputs**	**Hidden Layer(s)**	**Hidden Neurons**	**Iterations**	**Learning Rate**
DOW	3	2	4	800	0.4
S&P	2	2	6	500	0.6
DAX	2	1	2	800	0.4
CAC	4	2	3	1200	0.5
FTSE	3	1	2	500	0.7
IBEX	2	1	5	800	0.5
PSI	4	2	2	250	0.6
IBOV	3	2	3	800	0.5

The results show that the choice of structure is different, depending on the data. There is no magic formula to describe a structure that minimizes errors and leads to the best result. The best choice must be sought through random alternatives, according to the data.

The experimental results reveal that the proposed algorithm provides a promising alternative to stock market predictions, resulting in low errors in comparison with the GARCH model (see Table 4). Table 4 compares the ranked coefficients of multiple determination for each model. The R-squared value represents the proportion of variation in the dependent variable that is explained by the independent variables. The better the model explains variation in the dependent variable, the higher the R-squared value. Without further comparison, the neural network best explains variation in the dependent variable, followed by the regression model. The ranked error statistics are provided for comparison. These statistics are all based on return errors between the desired value and the neural network output value.

Table 4. Error comparison.

Index	**R Squared**	**Percentage Mean Error**	**RMSE**	**POCID**

	ANN	GARCH	ANN	GARCH	ANN	GARCH	ANN	GARCH
DOW	0.97326	0.86327	3.89323	7.73428	0.62152	2.83222	74.65%	63.74%
S&P	0.95432	0.73429	2.73273	6.89329	0.87323	3.83282	76.23%	57.76%
DAX	0.98732	0.87364	4.98321	8.78383	0.63263	2.71327	72.98%	58.90%
CAC	0.94327	0.83272	3.03933	7.32653	0.93289	4.02391	74.52%	61.13%
FTSE	0.95342	0.79322	4.32187	6.63733	0.73732	3.93811	77.03%	55.24%
IBEX	0.89763	0.86342	3.09323	7.63723	0.83221	2.83917	78.65%	59.99%
PSI	0.93721	0.78873	2.67327	6.98430	1.83283	5.63261	69.03%	49.64%
IBOV	0.96390	0.80323	2.03115	9.83921	0.63282	3.63783	78.11%	62.11%

In Table 4 it is relatively easy to visually verify that the neural network model performs better than the GARCH process. This differs from the model ranking, due to R-squared values. The neural network model predicts the closing value relatively accurately.

In an attempt to evaluate the robustness of the ANN model applied, we analyze the error dimension in the sets performed (training, test, and validation). The results are measured by the maximum percent error (MPE) and the RMSE:

$$MPE = \max\left\{\frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i}\right\}$$

(10)

$$RMSE = \sqrt{\frac{1}{n}(y_i - \hat{y}_i)^2} \qquad (11)$$

where y_i denotes the desire value i, and $\hat{y}_i$ the neural network output.

The financial market is a complex, evolutionary, and nonlinear dynamic system. Many factors interact in finance, including political events, general economic conditions, and traders' expectations. Therefore, predicting financial price movements is quite difficult but of extreme importance. In this case, we employ another relevant evaluation measure, the correctness of the prediction of change in direction (POCID), defined as

$$POCID = 100\frac{\sum_{i=1}^{N} D_i}{n} \qquad (12)$$

where $D_i = 1$ if $(y_i - y_{i-1})(\hat{y}_i - \hat{y}_{i-1}) > 0$, or $D_i = 0$, otherwise.

In this case, the model with the higher POCID more accurately predicts the market's movements. Table 5 compares the network sets.

Table 5. Network set comparison.

Index	Sets								
	Training			Test			Validation		
	MPE	RMSE	POCID	MPE	RMSE	POCID	MPE	RMSE	POCID
DOW	4.32712	2.12521	74.16%	4.87126	2.87521	73.13%	4.91274	3.13134	70.14%
S&P	7.53272	3.42513	73.92%	7.92177	3.87532	76.53%	8.08643	4.05235	73.98%
DAX	3.34741	2.36282	76.88%	3.72362	2.76512	73.28%	4.29347	3.32712	71.41%
CAC	6.83212	3.78236	75.59%	7.53132	4.13263	70.19%	7.35124	4.73512	69.19%
FTSE	5.97272	3.08221	79.01%	6.02183	4.02138	78.63%	6.68724	4.53289	76.23%
IBEX	6.74162	2.21342	72.11%	7.21633	3.42156	72.01%	7.53281	4.02571	70.86%
PSI	6.26324	4.76532	68.94%	6.83635	5.73512	67.99%	7.01963	6.00012	68.17%
IBOV	4.53127	3.09217	72.13%	5.02746	4.03821	70.12%	5.21456	4.63218	71.08%

We can see in Table 5 that a neural network structure applied to all the indexes has good prediction capability and, as can be seen in the low rate of validation errors compared with the GARCH model (see Table 4), a neural network is capable of learning from the data and can process good results for forecasting. The good performance of neural networks in this case is verified compared with GARCH model, which presents a higher rate of errors than the ANN model. Finally, the results in the test and validation sets confirm the neural network's generalization ability. Nevertheless, the differences between POCID measures for the ANN and GARCH models reveal that the ANN model employed predicts the market's movements more precisely than the GARCH process (see Tables 4 and 5). In all samples of data, we observe (see Table 4) that the GARCH model predicts a mean of 60% in market price movements (up or down), whereas the neural network model has the capability of predicting this measure to around 75%. As we well know, the ANN provides low POCID values in the validation set, compared with the other sets.

We measure the forecast performance for all the indexes in this study based on traditional statistical loss, a method employed by many, including Mathews (1994), but these tests do not reveal whether the forecast by one model is statistically superior to that of the other. Therefore, another commonly used test is needed that can help compare competitive models in terms of forecast accuracy. In the literature, there are few works that incorporate statistical tests to verify competitive models, and then mainly only when one of them is a neural network model. For this reason, in addition to the traditional measures described above, we perform two parametric tests of pairwise forecast evaluations. These tests are described as follows.

One of the parametric tests employed in this work is the AGS test, due to Ashley *et al.* (1980), and it tests for the statistical significance of the difference between the RMSEs associated with two competing model forecasts. This test is based on the equation

$$d_t = \beta_1 + \beta_2 (s_t - s_{mean}) + \zeta_t \tag{13}$$

where d_t is the difference between the forecast errors, as obtained by the ANN and GARCH models, s_t is the sum of these forecast errors, s_{mean} is the sample mean of s_t, and ζ_t is a white noise process. The AGS test is a test of the equality of the mean squared errors (MSEs) of two competing models against the alternative that the second model's MSE is lower (more accurate) than the first model's. The test is performed by jointly testing the significance of the parameters β_1 and β_2 in regression (13).[20] The test statistics for the AGS test are calculated from the residuals obtained from estimating an unrestricted model represented by equation (13) and a model that restricts $\beta_1 = \beta_2 = 0$ in equation (13). The test statistics for this test are calculated using the equation as follows:

$$TS = ((SSE_R - SSE_{UR})/(k-1))/(SSE_{UR}/(n-k)) \tag{14}$$

where SSE_R is the sum of the squared residuals from the restricted model, SSE_{UR} is the sum of the squared residuals from the unrestricted model, and *k* is the number of variables in the regression model. In this test, we assume that the forecast errors are not contemporaneously correlated and are free from serial correlations. The test statistic for this test is distributed *F* with 2 and *n* – 2 degrees of freedom under the assumption of normality.

Another parametric test of equal forecast accuracy is the MGN test. This test is recommended by Diebold and Mariano (1995) and is often employed when the assumption of contemporaneous correlation of errors is relaxed. The test statistics for this test can be computed using the equation

$$MGN = \frac{\hat{\rho}_{sd}}{(1-\hat{\rho}_{sd})^{12}}(n-1)^{12} \tag{15}$$

where $\hat{\rho}_{sd}$ is the estimated correlation coefficient between $s = e_1 + e_2$, and $d = e_1 - e_2$, with e_1 and e_2 the errors of models 1 and 2, respectively. In this work, 1 represents the GARCH model and 2 the ANN model. The test statistics for the MGN test are distributed *t* with *n* – 1 degrees of freedom. For this test, if the forecasts are equally accurate, then the correlation between *s* and *d* will be zero. Forecast accuracy measures based on the AGS and MGN tests of the forecast equivalence of two competing models are shown in Table 6 for all the series.

The results from the AGS and MGN tests shown in Table 6, which examines pairwise comparisons of the forecasts of the two competitive models (with ANNs and GARCH as benchmarks), reveal that the ANN shows statistically significant evidence of superior performance in predicting all the indexes evaluated in this study. These results confirm what was already indicated by traditional measures, that is, that the ANN outperforms the GARCH model in predicting the future trends of all the stock market indexes considered in this paper.

Graphically, the results are shown in Figures 1-8. For each series we plot (a) the real value of the index with GARCH and ANN predictions and (b) the difference between the index values, with each model's result depicted as an error measure. All the results correspond to the validation set.

[20] **The procedure for the AGS test is also described in Bradshaw and Orden (1990) and Kiani *et al.* (2005).**

Table 6. AGS and MGN tests for pairwise forecast evaluations for all the indexes.

Index	**Tests**			
	AGS Test		**MGN Test**	
	Statistic	**p-Value**	**Statistic**	**p-Value**
DOW	246.8763	(0.0001)	1.8923	(0.0001)
S&P	566.7255	(0.0001)	-2.2397	(0.0001)
DAX	411.1202	(0.0001)	-1.2863	(0.0001)
CAC	398.7256	(0.0001)	3.8726	(0.0002)
FTSE	137.7235	(0.0001)	-1.9304	(0.0001)
IBEX	509.8363	(0.0001)	-0.1976	(0.0003)
PSI	256.0982	(0.0001)	2.3223	(0.0000)
IBOV	145.8726	(0.0001)	-2.6398	(0.0001)

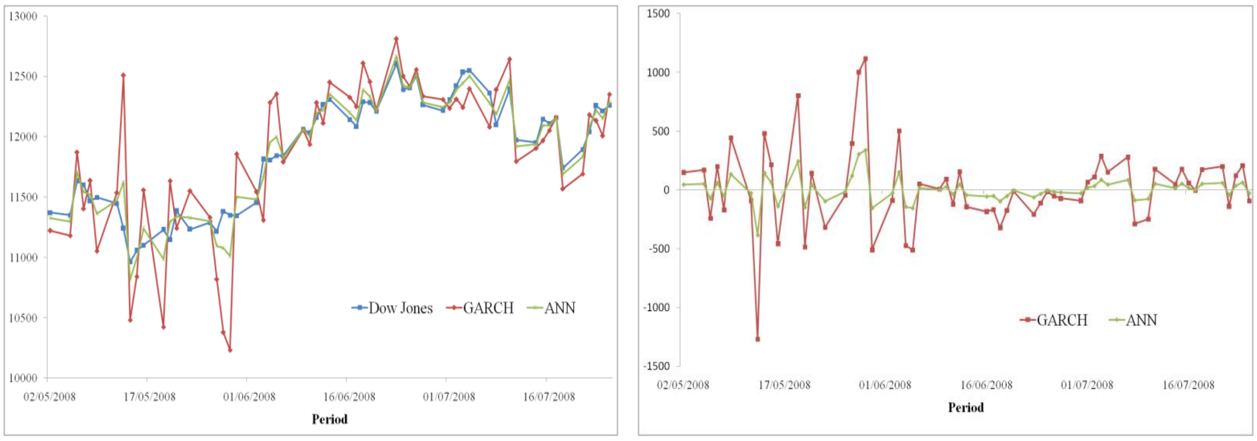

Figure 1. Dow Jones: (a) Index and model predictions. (b) Differences between real and predicted values.

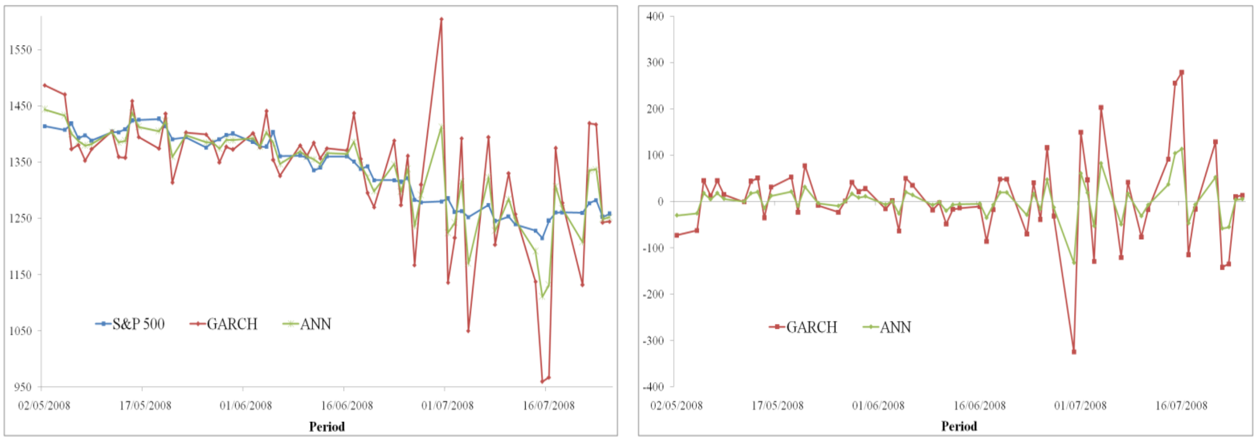

Figure 2. S&P 500: (a) Index and model predictions. (b) Differences between real and predicted values.

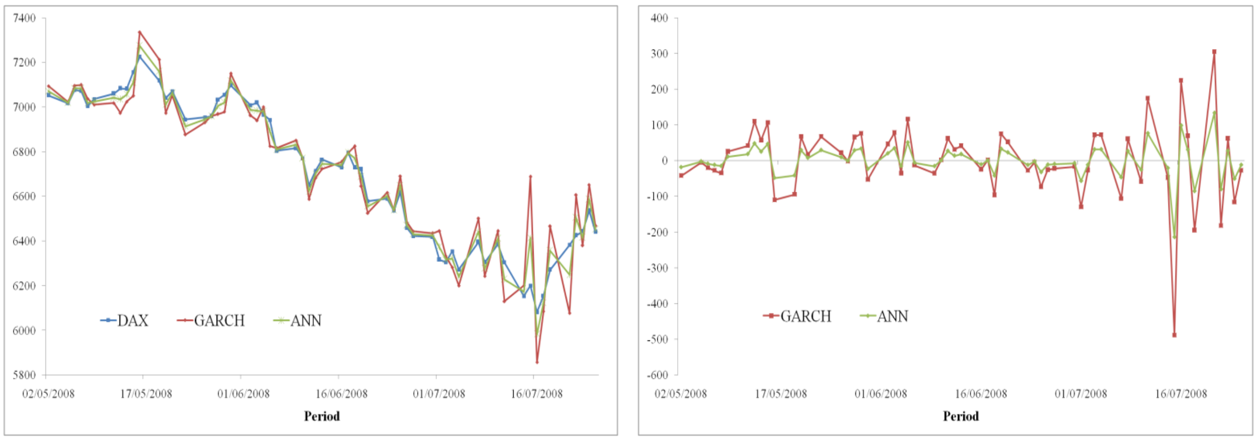

Figure 3. DAX: (a) Index and model predictions. (b) Differences between real and predicted values.

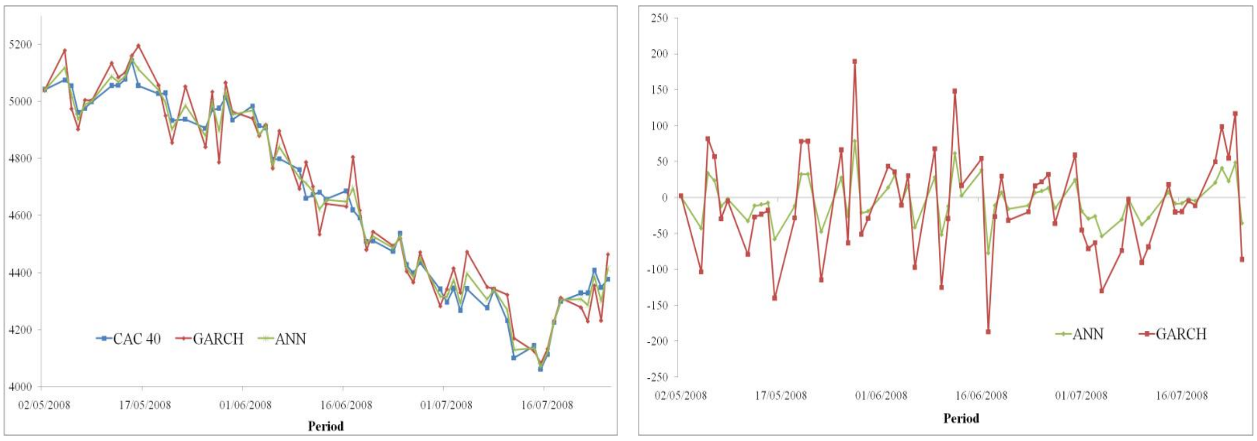

Figure 4. CAC 40: (a) Index and model predictions. (b) Differences between real and predicted values.

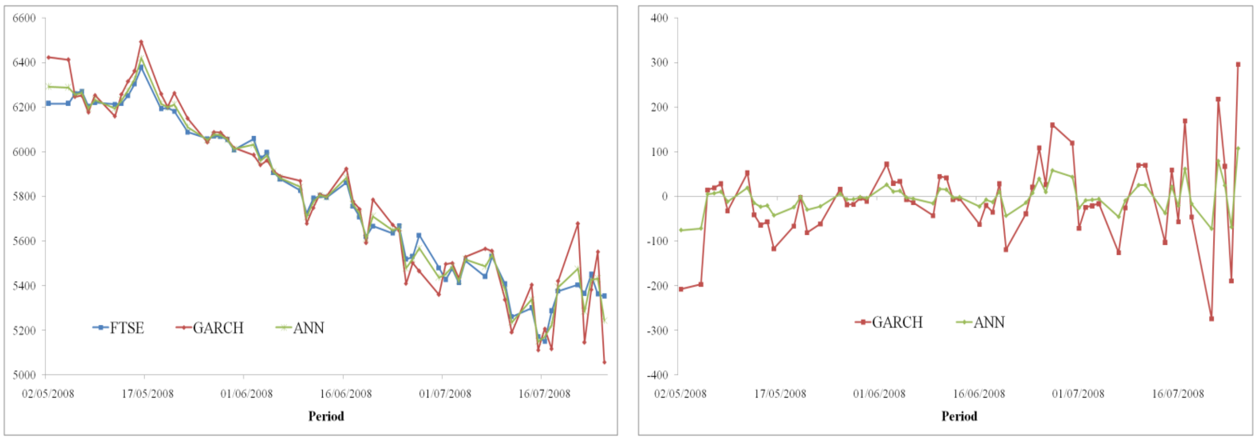

Figure 5. FTSE: (a) Index and model predictions. (b) Differences between real and predicted values.

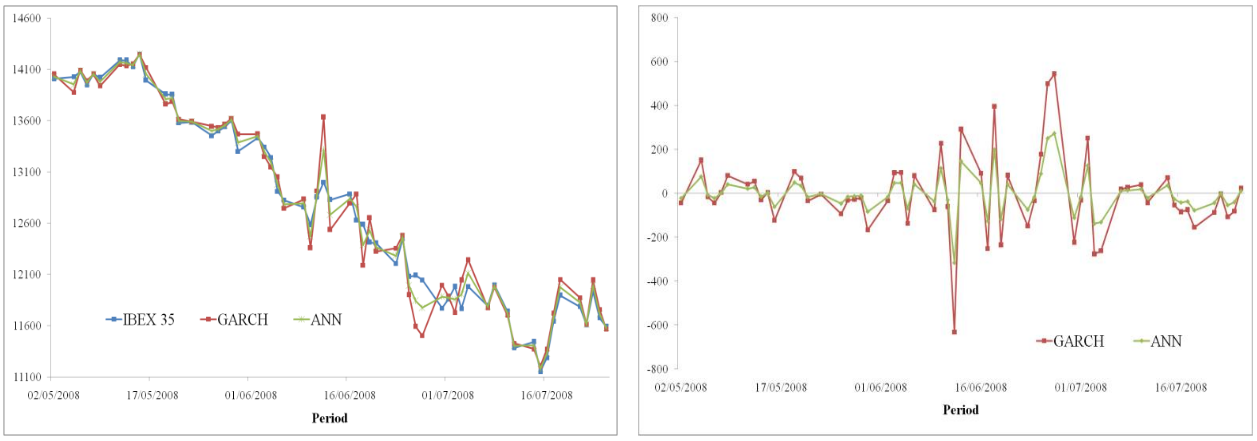

Figure 6. IBEX 35: (a) Index and model predictions. (b) Differences between real and predicted values.

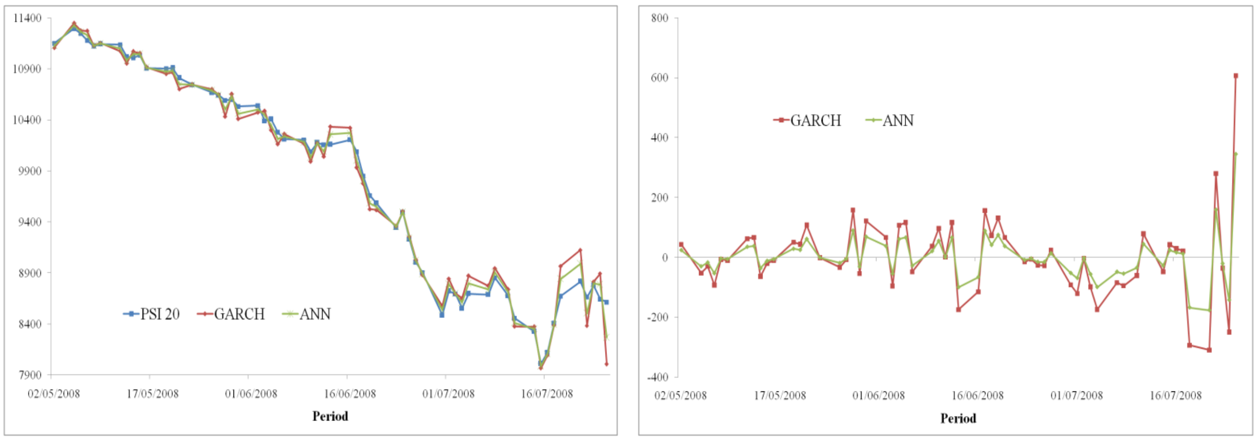

Figure 7. PSI 20: (a) Index and model predictions. (b) Differences between real and predicted values.

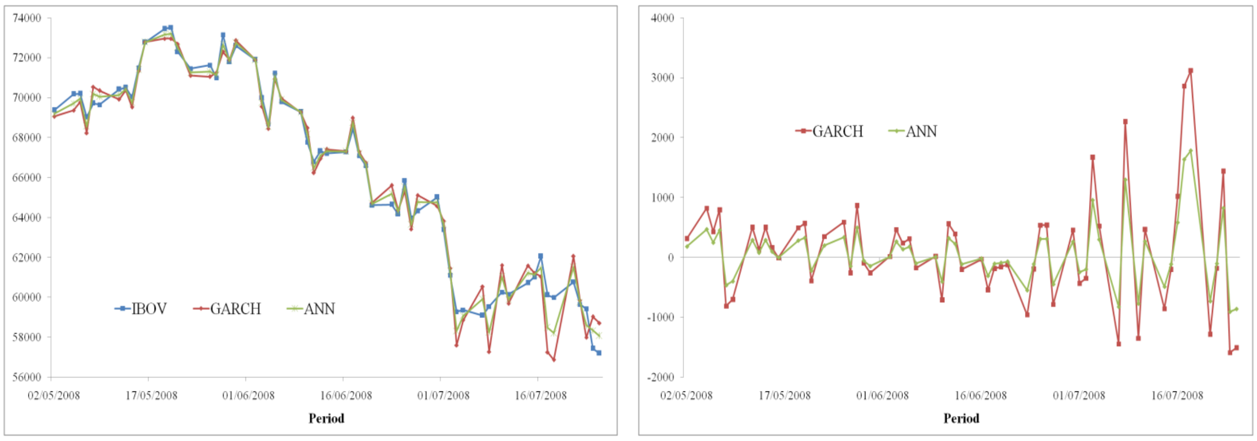

Figure 8. IBOV: (a) Index and model predictions. (b) Differences between real and predicted values.

Figures 1-8 confirm the results described above. We can see that both methods are capable of predicting all the indexes evaluated, but when we see the difference between the real and predicted values obtained for each model, it is clear that the ANN outperforms the GARCH method. The ANN presents errors closer to zero, while the traditional statistical model results in larger errors. In terms of error measures, the intelligence artificial method has been shown to be a good way to predict financial time series.

One question proposed in this work is whether or not neural networks can incorporate heteroskedastic phenomena. Table 7 shows the residual analysis of each series studied here. It includes a mean test, a test of normality (Jarque–Bera), and a test of correlations present in the residuals (Ljung–Box–Pierce Q-test), and it verifies heteroskedasticity in the residuals (Engle´s ARCH test)[21,22].

Table 7. Residuals analysis.

Index	**Tests**							
	Mean		**Jarque–Bera**		**Ljung–Box–Pierce**		**Engle´s ARCH**	
	Value	**Statistically zero?**	**Value**	**Normal?**	**Value**	**Correlation?**	**Value**	**Homoskedasticity ?**
DOW	0.002	Yes	18.972	Yes	28.921	No	12.872	Yes
S&P	0.008	Yes	16.982	Yes	29.990	No	9.8721	Yes
DAX	0.092	Yes	57.923	Yes	30.912	No	14.765	Yes
CAC	0.061	Yes	61.982	Yes	25.821	No	8.8216	Yes
FTSE	0.076	Yes	25.897	Yes	26.732	No	12.872	Yes
IBEX	0.072	Yes	56.932	Yes	33.812	No	15.876	Yes
PSI	0.086	Yes	22.372	Yes	27.978	No	9.991	Yes
IBOV	0.053	Yes	54.862	Yes	31.982	No	13.721	Yes

Analyzing Table 7, we can see that the neural network residuals for all the indexes studied have a mean statistically equal to zero and a normal distribution, there is no correlation between the residuals, and, finally, the residuals are homoskedastic. The results show that the neural network structure proposed is capable of series modeling and forecasting, capturing heteroskedastic phenomena, and confirming the method's robustness.

7. Conclusions

This research analyzes the use of neural networks as a forecasting tool; specifically, it tests their ability to predict future trends of stock market indexes. North American, European, and Brazilian stock market indexes were studied and accuracy compared against a traditional forecasting method (GARCH). While only briefly discussing neural network theory, this study determines the feasibility and practicality of the individual investor using neural networks as a forecasting tool. It concludes that neural networks do have a powerful capacity to forecast all the stock market indexes studied, and, if properly trained, the individual investor could benefit from the use of this forecasting tool over current techniques for the following reasons.

- When using multiple linear regressions, the governing regression assumptions must be true. The linearity assumption itself and normal distribution may not hold in most financial time series. Neural networks can model nonlinear systems and do not make any assumptions about the input probability distribution.
- ANNs are universal function approximations. It has been shown that a neural network can approximate any continuous function to any desired accuracy.
- ANNs are able to generalize. After learning the data presented to them, ANNs can often correctly infer the unseen part of a population, even if the sample data contain noisy information.
- Compared with the GARCH model, neural networks are significantly more accurate, according to traditional measurements tests, and ANNs outperform GARCH models in statistical terms, as the AGS and MGN tests indicate.
- ANNs can capture heteroskedastic phenomena.

The next step in future work is to integrate neural networks and other techniques—such as genetic techniques, wavelet analysis, fuzzy inference, pattern recognition, and traditional time series models—for financial and economic forecasting. The advantages of genetic techniques include adaptiveness and robustness, and they avoid the problem of neural networks getting

[21] **For more about these tests, see Brockwell and Davis (1991).**

[22] **Chebyshev's inequality test is applied to confirm the results about the residuals' probability distribution. For all the indexes, this was confirmed to be a normal distribution.**

stuck at a local optimum. Once a network is trained, tested, and identified as being "good," a genetic algorithm can be applied to optimize its performance. The process of genetic evolution works on the neuron connection of a trained network by applying two procedures: mutation and crossover. The application of hybrid systems seems to be well suited for the forecasting of financial data. On the other hand, the discussion about input variables can be interpreted according to each dataset studied.

References

AO, S. I. Analysis of the interaction of Asian Pacific indices and forecasting opening prices by hybrid VAR and neural network procedures. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation 2003, Vienna, Austria, 2003a.

AO, S. I. Incorporating correlated markets' prices into stock modeling with neural network. In Proceedings of the IASTED International Conference on Modelling and Simulation 2003, Palm Springs, CA, pp. 353–358, 2003b.

ANDERSON, J. A., and ROSENFELD, E. Neurocomputing: Foundations of Research. MIT Press, Cambridge, MA, 1998.

ASHLEY, R., GRANGER, C. W. J., and SCHMALANCEE, R. Advertising and aggregate consumption: An analysis of causality. Econometrica, 48, 1149–1168, 1980.

BAILY, D., and THOMPSON, D. M. Developing neural network applications. AI Expert, 12, 33–41, 1990.

BAUM, E. B., and HAUSSLER, D. What size net gives valid generalization? Neural Computation, 6, 151–160, 1989.

BISHOP, C. Neural Networks for Speech and Sequence Recognition. Thompson, London, 1996.

BOLLERSLEV, T. R. Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 51, 307–327, 1986.

BOX, G. E. P., JENKINS, G. M., and REINSEL, G. C. Time Series Analysis: Forecasting and Control, 3rd ed. Prentice Hall, NJ, 1994.

BRADSHAW, G. W., and ORDEN, D. Granger causality from the exchange rates to agricultural prices and export sales. West Journal of Agricultural Economics, 100–110, 1990.

BROCKWELL, P. J., and DAVIS, R. A. Time Series: Theory and Methods, 2nd ed. Springer, New York, 1991.

CAMPBELL, J. Y., LO, A. W., and MACLINKAY, A. C. The Econometrics of Financial Markets. Princeton University Press, 1997.

CAO, Q., LEGGIO, K. B., and SCHNIEDERJANS, M. J. A Comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market. Computers and Operations Research, 32, 2499–2512, 2005.

CHATTERJEE, A., AYADI, O. F., and BOONE, B. E. Artificial neural networks and the financial markets: A survey. Managerial Finance, 26, 32–45, 2000.

CHEN, C. H. Neural networks for financial market prediction. Proceedings of the IEEE International Conference on Neural Networks, 2, 1199–1202, 1994.

CHEN, N. Financial investment opportunities and the macroeconomy. Journal of Finance, 46, 529–554, 1991.

DEBOECK, G. J. Trading on the Edge: Neural, Genetic and Fuzzy Systems for Chaotic Financial Markets. Wiley, New York 1994.

DIEBOLD, F. X., and MARIANO, R. S. Comparing predictive accuracy. Journal of Business and Economics Statistics, 13, 253–263, 1995.

DOUGHERTY, C. Introduction to Econometrics. Oxford University Press, New York, 1992.

DUTTA, G., JHA, P., LAHA, A. K., and MOHAN, N. Artificial neural network models for forecasting stock price index in the Bombay stock exchange. Journal of Emerging Market Finance, 5, 283–295, 2006.

ENKE, D., and THAWORNWONG, S. The use of data mining and neural networks for forecasting stock market returns. Expert Systems with Applications, 29, 927–940, 2005.

ERSOY, O. Tutorial at the Hawaii International Conference on Systems Sciences, January 1990, Hawaii, 1990.

FAMA, E. F. The behavior of stock market prices. Journal of Business, 14, 34–105, 1965.

FAMA, E. F. and FRENCH, K. Common risk factors in the returns on stocks and bonds. Journal of Financial Economics, 33, 3–56, 1993.

FANG, H., LAI, S., and LAI, M. Fractal structure in currency futures price dynamics. Journal of Futures Markets, 14, 169–181, 1994.

FARIA, E. L., ALBUQUERQUE, M. P., GONZALEZ, J. L., CAVALCANTE, J. T. P., and ALBUQUERQUE, M. P. Predicting the Brazilian stock market through neural networks and adaptive exponential smoothing methods. Expert Systems with Applications, 36, 12506–12509, 2009.

FERSON, W., and SCHADT, R. Measuring fund strategy and performance in changing economic conditions. Journal of Finance, 51, 425–461, 1996.

GARCIA, R., and GENCAY, R. Pricing and hedging derivative securities with neural networks and a homogeneity hint. Journal of Econometrics, 94, 93–115, 2000.

GATELY, E. J. Neural Networks for Financial Forecasting. Wiley, New York, 1996.

GENCAY, R. The predictability of security returns with simple technical trading rules. Journal of Empirical Finance, 5, 347–359, 1998.

GRANGER, C., and MORGENSTERN, O. Predictability of stock market prices. Health, Lexington, MA, 1970.

HAYKIN, S. Neural Networks – A Comprehensive Foundation. IEEE Press, New York, 2001.

HECHT-NIELSEN, R. Neurocomputing. Addison-Wesley, Reading, MA, 1990.

HERTZ, J., KROGH, A., and PALMER, R. G. Introduction to the Theory of Neurocomputation. Addison-Wesley, Reading, MA, 1991.

HIEMSTRA, C., and JONES, J. D. Testing for linear and nonlinear Granger causality in the stock price–volume relation. Journal of Finance, 49, May, 1639–1664, 1994.

HORNIK, K. Some new results on neural network approximation. Neural Networks, 6, 1069–1072, 1993.

HORNIK, K., STINCHCOMBER, M., and WHITE, H. Multilayer feedforward networks are universal approximations. Neural Networks, 2, 359–366, 1987.

HUANG, W., LAI, K. K., NAKAMORI, Y., WANG, S. Y., and YU, L. Neural networks in finance and economics forecasting. International Journal of Information Technology and Decision Making, 6, 113–140, 2007.

HUANG, W., NAKAMORI, Y., and WANG, S. Y. Forecasting stock market movement direction with support vector machine. Computers and Operations Research, 32, 2513–2522, 2005.

HUARNG, K., and YU, T. H. The application of neural networks to forecast fuzzy time series. Physica A, 363, 481–491, 2006.

JASIC, T., and WOOD, D. The profitability of daily stock market indices trades based on neural network predictions: Case study for the S&P 500, the DAX, the TOPIX and the FTSE in the period 1965–1999. Applied Financial Economics, 14, 285–297, 2004.

KAASTRA, I., and BOYD, M. Designing a neural network for forecasting financial and economic time series. Neurocomputing, 10, 215–236, 1996.

KAO, G. W., and MA, C. K. Memories, heteroscedasticity and price limit in currency future markets. Journal of Future Markets, 12, 672–692, 1992.

KATZ, J. O. Developing neural network forecasters for trading. Technical Analysis of Stocks and Commodities, 8, 58–70, 1992.

KIANI, K. Detecting business cycle asymmetries using artificial neural networks and time series models. Computational Economics, 26, 65–89, 2005.

KIANI, K., BIDARKOTA, P., and KASTENS, T. Forecast performance of neural networks and business cycle asymmetries. Applied Financial Economics Letters, 1, 205–210, 2005.

KLAUSSEN, K. L., and UHRIG, J. W. Cash soybean price prediction with neural networks. In Conference on Applied Commodity Analysis, Price, Forecasting and Market Risk Management Proceedings, Chicago, 1994, pp. 56-65.

KLIMASAUSKAS, C. C. Applying Neural Networks. In R. R. Trippi and E. Turban, eds., Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance. Probus, Chicago, 1993, pp. 64–65

LAM, M. Neural network techniques for financial performance prediction: Integrating fundamental and technical analysis. Decision Support Systems, 37, 567–581, 2004.

LAWRENCE, J. Introduction to Neural Networks. California Scientific Software, Grass Valley, CA, 1991.

LEVICH, R. M., and THOMAS, L. R. The significance of technical trading rule profits in the foreign exchange market: A bootstrap approach. In Strategic Currency Investing – Trading and Hedge in the Foreign Exchange Market. Probus, Chicago, 1993, pp. 336–365.

LIN, T., and YU, C. Forecasting stock markets with neural networks. Working Paper Series, available at SSRN: http://ssrn.com/abstract=1327544, accessed April 28, 2009, 2009.

LO, A. W., and MACKINLAY, A. C. Stock market prices do not follow random walks: Evidence from a simple specification test. Review of Financial Studies, 1, 41–66, 1988.

MARTIN, R. D. GARCH modeling of time-varying volatilities and correlations. http://fenews.com/1998/Issue4/059802.htm, accessed July 5, 2008, 1998.

MASTERS, T. Practical Neural Network Recipes in C++. Academic, New York, 1993.

MATHEWS, B. P. Towards a taxonomy of forecast error measures. Journal of Forecasting, 13, 409–416, 1994.

MITCHELL, T. M. Machine Learning. McGraw-Hill, USA, 1997.

NELSON, M. M., and ILLINGWORTH, T. A Practical Guide to Neural Nets. Addison-Wesley, Reading, MA, 1991.

NYGREN, K. Stock Prediction: A neural network approach. Master’s thesis, Royal Institute of Technology, KTH, Sweden, 2004.

O’CONNOR, N., and MADDEN, M. G. A neural network approach to predicting stock exchange movements using external factors. Proceedings of 25th International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, 2005.

ODOM, M. D., and SHARDA, R. A neural network for bankruptcy prediction. In Proceedings of the IEEE International Conference on Neural Networks, San Diego, CA, 1992, pp. II163–II168.

PAN, H., TILAKARATNE, C., and YEARWOOD, J. Predicting the Australian stock market index using neural networks exploiting dynamical swings and intermarket influences. Lectures Notes in Computer Science (2003), pp. 327–338, 2004.

QI, M., and MADALA, G. S. Economic factors and the stock market: A new perspective. Journal of Forecasting, 18, 151–166, 1999.

QI, M., and WU, Y. Nonlinear prediction of exchange rates with monetary fundamentals. Journal of Empirical Finance, 10, 623–640, 2003.

REFENES, A. N., ZAPRANIS, A., and FRANCIS, G. Stock performance modeling using neural networks: A comparative study with regression models. Neural Networks, 7, 375–388, 1994.

RUMELHART, D. E., and MCCLELLAND, J. L. Parallel Distributed Processing, Explorations in the Microstructure of Cognition. MIT Press, Cambridge, MA, 1986.

RUPPERT, D. GARCH models. http://www.orie.cornell.edu/~davidr/or473/LectNotes/notes/node139.html, accessed April 25, 2008, 2001.

SCHRAUDOLPH, N., and CUMMINS, F. Introductions to Neural Networks. https://www.icos.ethz.ch./teaching/NNcourse/backprop.html#top, accessed September 13, 2008, 2002.

SCHWARTZ, G. Estimating the dimension of a model. Annals of Statistics, 6, 461–468, 1978.

SHAPIRO, A. F. Capital market applications of neural networks, fuzzy logic and genetic algorithms. http://www.actuaries.org/AFIR/colloquia/Maastricht/Shapiro.pdf, accessed August 13, 2008, 2003.

SHARDA, R., and PATIL, R. B. A connectionist approach to time series prediction: An empirical test. In G. J. Deboeck, ed., Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets. Wiley, New York, 1994, pp. 451–464.

SITTE, R., and SITTE, J. Analysis of the predictive ability of time delay neural networks applied to the S&P 500 time series. IEEE Transactions on Systems, Man and Cybernetics, 30, November, 568–572, 2000.

TANG, Z, ALMEIDA, C., and FISHWICK, P. A. Time series forecasting using neural networks vs. Box–Jenkins methodology. Simulation, vol. 57, no. 5, 303-310, 1991.

TOPEK, W. G., and QUERIN, S. F. Random process in prices and technical analysis. Journal of Future Markets, 4, 15–23, 1984.

WAITE, T., and HARDENBERGH, H. Neural nets. Programmer´s Journal, 7, 10–22, 1989.

WASSERMAN, P. D. Advanced Methods in Neural Computing. Van Nostrand Reinhold, New York, 1993.

WONG, B. K., and SELVI, Y. Neural network applications in business: A review and analysis of the literature. Information and Management, 34, 129–139, 1998.

YU, T. H. K., and HUARNG, K. H. A bivariate fuzzy time series model to forecast the TAIEX. Expert Systems with Applications, 34, 2945–2952, 2008.

ZANG, G., PATUWO, B. E., and HU, M. Y. Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting, 14**,** 35–62, 1998.