Università
Ca'Foscari
Venezia

Master's Degree programme. Second Cycle (*D.M. 270/2004*)
in Economics and Finance

Final Thesis

Ca' Foscari
Dorsoduro 3246
30123 Venezia

MultiLayer ANNs: predicting the S&P 500 index

Supervisor
Ch. Prof. Marco Corazza

Graduand
Giovanni Cogo
Matriculation Number 822860

Academic Year
2014 / 2015

To my parents, Oreste and Otella,
without whose loving support this
would not have happened.

INTRODUCTION

"Our greatest glory is not in never falling, but in rising every time we fall."
Confucius

In this thesis, the main objective is to develop an ANN to forecast the S&P 500 index price. First of all, an ANN-based variable selection model is presented. This model explores the relationship between some initial input variables and the closing price of the S&P 500 index. Furthermore, this research investigates how the training algorithm, as well as the number of neurons in the hidden layer and the distribution of the training data, affect the accuracy of the network.

The thesis consists in six chapters. In the first one both Efficient Market Hypothesis and Chaos theory are explained before the description of my objective. Moreover, a brief historical parenthesis on ANNs is presented. In the second chapter ANNs are widely explained in all their features and architectures. The third chapter presents a detailed literature review regarding the application of ANNs in the financial world, in particular to stock market prediction. In the fourth chapter the methodology that stands behind both the variable selection and price prediction models is presented. In particular, all the parameters used are widely explained.

In the fifth chapter all the results of my research are explained through the use of table and graphs. The sixth and final chapter presents the conclusions and some suggestions for future works.

CHAPTER 1 MARKET HYPOTHESIS

"That all our knowledge begins with experience there can be no doubt."

I. Kant

This first chapter presents two antithetical views of financial markets, Efficient Market Hypothesis and chaos theory. The former is defined as the main theory of market efficiency. The latter might be viewed as a rival theory that moves away from the underlying notions of the Efficient Market Hypothesis and introduces chaotic analysis in economics. Furthermore neural networks and my objective regarding financial time series forecasting are introduced. Finally the history of neural networks is told from the beginning of neural networks to modeling and computer simulation.

The financial industry has been a prime application area for many artificial intelligence techniques (e.g. ANNs[1] were widely used for financial time series forecasting).

Nevertheless, acceptance of the market as being exclusively governed by random forces leads to the efficient market hypothesis, and if strictly adhered precludes forecasting as a viable activity.

Traditionally, the Efficient Market Hypothesis and the chaos theory, have been seen as competing theories. The first one is based on linear mathematics and rational reasoning, the latter moves towards nonlinear systems and deterministic dynamics. Proponents of the two different theories disagree on the financial markets' design. Efficient Market Hypothesis supporters believe that prices in the market are inherently fair. Those defending chaos theory disagree. Peters Edgar E. said "Markets were never designed to insure fair pricing; markets were created simply to bring people together, to provide a forum for discussion and a place for trading" [Peters Edgar E. , 2001].

Whilst the debate in academic environment is far from being conclusive, ideas from nonlinear dynamics and chaos theory are emerging that point to possible deterministic mechanisms underlying market behaviuor, and which are not explicable by invoking market risk alone.

1.1 Efficient Market Hypothesis

The hypothesis of stochastic trends of stock prices in financial markets leads to the theory of efficient financial markets also called Efficient Market Hypothesis (EMH). EMH is an important concept in finance claiming that financial markets are "informationally efficient" [Zhang H., 2010].

The first theoretical developments of the EMH are due to Samuelson (1965). The proposal of an efficient market is mostly due to Eugene F. Fama's Ph.D. dissertation written in the early 60's entitled "The Behaviour of Stock Market Prices". It argued that securities will be appropriately priced and reflect all public information if the market includes investors who are rational. If the market is efficient an investor will be unable to outperform it consistently [Fama E. F., 1970].

The theory of the EMH, in its weak form, is based on the assumption that the current values of publicly traded securities "fully reflects" the information available in the financial market at any given time. Consequently potential changes in the future prices will occur exclusively in relation to the widespread of new information regarding events that may affect the future value of the stock.

[1] An artificial neural network is a computer software program that mimics the human brain's ability to classify patterns or to make forecasts or decision based on past experience.

According to the theory of the EMH there is no room for fooling investors, and as a result, all investments in efficient markets are fairly priced (i.e. on average investors get exactly what they pay for). Furthermore it implies that no person can outperform the market consistently because stock prices are random and cannot be anticipated or profited from. If someone does outperform the market, the accomplishment is purely based on luck or by statistical chance [Peters E. E., 1999].
In such a financial market, neither technical analysis nor fundamental analysis may allow an investor to achieve higher profits than those that other investors would obtain by holding a portfolio of securities selected at random with the same degree of risk [Zhang H., 2010].
The definitional statement that in an efficient markets prices "fully reflect" available information can be specified in the following stochastic process, that describes an asset price movement:

$$P_{t+1} = \mathrm{E}\left(\tilde{P}_{t+1}\middle|\Phi_t\right) + \tilde{\varepsilon}_{t+1}$$

where the time interval considered is between t and $t+1$, P_{t+1} is the price of an financial asset, $\mathrm{E}(\cdot)$ is the expectation operator, Φ_t is a general symbol for whatever set of information is assumed to be fully reflected in the price at time t and the tildes indicate that $\tilde{P}_{t+1}$ and $\tilde{\varepsilon}_{t+1}$ are random variables at $t+1$, respectively the financial asset price random variable and the financial asset prediction error random variable. It is important to note that the $\mathrm{E}(\tilde{\varepsilon}_{t+1})$ is equal to zero.
The conditional expectation notation $\mathrm{E}\left(\tilde{P}_{t+1}\middle|\Phi_t\right)$ is meant to imply that the asset price always fully reflect available information [Fama E. F., 1970].
Recalling that the EHM hypothesis, in its weak form, implies that no person can outperform the market consistently because stock prices are random and cannot be anticipated or profited from, it is possible to conclude that they follow a random walk [Peters E. E., 1999].

1.2 Random Walk Hypothesis

One of the first studies suggesting that stock prices follow a random walk was made by Louis Bachelier in his work of 1900 known as "Theorie de la Spèculation".
The Random walk hypothesis is based on the statement that the current price of a security "fully reflects" all the available information. Furthermore, it is defined as a process in which changes from one period to another are independent and identically distributed. If the prices of the financial

markets did follow a random walk, this would imply that the size and direction of the changes in the past price provides no insight in the future price changes of that financial asset [Floreano D., 1996]. The general formalization of the random walk hypothesis is:

$$\mathrm{E}\left(\tilde{P}_{t+1}\middle|\Phi_t\right) = P_t$$

This equation shows that the entire distribution of security prices is independent of the information available at time t, Φ_t.

1.3 Three subsets of the Efficient Market Hypothesis

Market efficiency can be more precisely defined with reference to the information set available to market participants. Stock prices are influenced by three different kinds of information: technical and quantitative, qualitative, private and public information. In relation to the different types of available information, it is possible to distinguish three versions of the Efficient Markets Hypothesis: Weak Form Efficiency, Semi-Strong Form Efficiency, Strong Form Efficiency.

The first kind of market efficiency states that prices efficiently reflect all the information contained in the past series of stock prices. That is, nobody can detect mispriced securities and outperform the market by analyzing historical security prices. In other words, no one should be able to earn using security prices, which are considered as easily available information. However, many investors attempt to obtain profits by using a particular technique called technical analysis, that is based on the study of historical security prices and trading volume data;

The second type of market efficiency asserts that current security prices fully incorporate all the publicly available information. The latter includes not only historical data but also data reported in a company, for example financial statements, earnings and dividend announcements, expectations about macroeconomics factors etc. Similarly to the weak form of market efficiency, the assertion behind semi-strong one is that nobody should be able to profit using public information. Nevertheless, this assumption is far stronger than the one of the weak market efficiency, because it requires the existence of investors able to comprehend both financial and macroeconomics information.

The last kind of market efficiency asserts that current price fully reflects all the available information, both public and private. The strong form of market efficiency is different than the semi-strong one,

because in the latter one nobody should be able to consistently generate profits even if trading on inside information. In other words, the strong form of market efficiency states that not even the company's management is able to systematically profit from inside information. The rationale in this last type of market efficiency is that the financial markets anticipates future developments and therefore stock prices might have already assimilated the information. Not surprisingly, though, empirical research in finance has found that the strong form of market efficiency is inconsistent [Ball R., 1994].

1.4 The sufficient conditions for the Efficient Market Hypothesis

There are four sufficient conditions that might help or hinder efficient adjustments of prices to information. First, there should be a large number of rational investors, who actively participate in the market. A rational investor is one who wishes to maximize his expected return for a given level of market risk [Peters E. E., 1999]. Second, information should be costless and available for all the actors in the financial market. Third, all the investors should agree on the implications of current information for the actual and future stock prices [Fama E. F., 1970]. Finally, there must be no transaction costs in trading securities and investors must react quickly and fully to new information [Jones C. P., 1998].

However, frictionless markets do not exists in the real world and in any market there will be a combination of transactional costs, costly information and disagreement concerning the information that is commonly held [Glen P. J.,2005].

1.5 Limitations and anomalies of the EMH

The Efficient Market Hypothesis was created so that probability theory and linear models could be applied to capital markets [Kar A.2010]. This theory is based on the Central Limit Theorem, which implies that security price changes plotted over a time period, should follow the normal curve.

Peters Edgar E. and other researchers applied the central limit theorem to different historical data sets of security prices in order to find a correlation between the EMH and a certain financial market from which the data were taken [Peters E. E., 1994]. They discovered that EMH does not correlate well with market movements. Given the previous finding and considering that EMH is based

exclusively on linear analysis, its descriptive power and normative implications have become questionable [Lorie J. H. et al., 1985].

Furthermore, the EMH has several limitations due to its sufficient conditions that are almost inapplicable in the real world:

- one of the weaknesses of the EMH is that it models individual investors as rational economic agents. The behaviuor of investors is not rational. Looking at the recent history of financial markets, it is possible to see that investors tend then to invest at the peak of a bubble right before a crash [Choen, 1997];
- there are no subjective beliefs in typical formulations of the EMH: information is assumed to be an objective commodity, with the same meaning for all investors. But investors have different information and beliefs. Therefore, the actions of individual investors are based not only on their own beliefs, but on their beliefs about the beliefs of others, and thus their trading is affected by their incomplete knowledge of others' motive for trading [Ball R., 2010];
- in the EMH information processing is assumed to be costless. This limitations implies that information is incorporated into prices immediately and exactly. The cost of acquiring public information may be negligible, but information processing (i.e. costs related to the interpretation of the data) costs are an entirely different matter [Ball R., 2010];
- the EMH assumes markets are costless to operate. Generally speaking, stock markets are paradigm examples of low-cost, high-volume markets, but they are not entirely without costs. The role of transactions costs in the EMH is unclear [Ball R., 1994];

Ray Ball defined three categories of anomalies to the Efficient Market Hypothesis. The first are empirical anomalies, which are related to problems in fitting the EHM theory to the available data. For example market overreaction and excessive volatility to different types of announcements. It is observed that financial market tends to overreact to extreme news. So, systematic price reversals can sometimes be predicted. The second category of anomalies is related to the efficiency defects of the EMH. The latter assumes that information are evenly distributed between financial markets' operators. In the real world, information are unequally distributed. Insiders have more information than the public does and banks have a large support staff and faster information flow than does the individual investor [Choen B., 1997].

The third category is related to problems in testing the efficiency of EHM as a model of the stock markets. One problem is seasonal changes in the risk component. Firms usually announce major events in December. Consequently in January, stock returns are inexplicably high or low depending on the announcements of the previous year [Ball R., 1994].

1.6 An alternative to EMH: Chaos Theory

In financial markets fundamental information about asset values is crowded out by extraneous information and noise. The presence of noise itself reveals the inadequacy of the linear testing models that led to the EMH and the random walk. Noise, defined more narrowly, refers to those pricing influences that are not associated with rational expectations about the underlying value of the asset. The fact that such expectation are not necessarily rational, should not lead to the conclusion that they are irrational. Investment strategies based on non-rational information may represent anything (for example loyalty to a friend). The noise theory studies the effects of investors' sub-optimal behaviours on financial markets.

The most important implication of noise theory is that it reveals markets to be nonlinear systems, to which the linear mathematics and reasoning that underlie the EMH are opposite [Cunningham L. A., 1994]. The presence of extraneous information and noise constitute a system in which individuals together overreact to information or withhold action in the face of information.

Such elements are hallmarks of nonlinear systems because they indicate a non-proportional relationship between a cause and its effect (e.g., between news and price changes).

Nonlinearity means the absence of proportionality - changes in one variable will produce a change in another variable but exponentially rather than proportionally [Baker G. L. & Gollub J. P., 1990].

An emerging field in the science of nonlinear dynamics is the chaos theory. It took a long time before economic theorists would start to study in depth the phenomenon of chaos. In the early '80s the researchers began to take seriously this phenomenon, studying the indicators of economic chaos which were applied to finance [Fabbri G. & Orsini R., 1993].

Chaos is mathematically defined as "randomness" generated by simple deterministic systems. What must at all times be kept in mind is that chaos does not imply randomness, it implies some underlying complex pattern or solution, not merely irregularity. Chaos theory shows that what appears to be random is not, and that patterns may be discerned [Cunningham L. A., 1994].

A chaotic system is one that seems random on the surface, but that turns out to be nonrandom upon more critical inspection [Roe M. J., 1996].

The importance of chaos theory is related to its ability to explain chaotic events, such as market crashes, and by its criticism of the EMH for its inability to explain such events [Cunningham L. A., 1994]. Using nonlinear techniques of chaos theory in empirical studies is possible to show that information in security prices is not immediately absorbed by market prices, as predicted by the EHM. Considering that information can be utilized by market participations, and that information is not immediately absorbed into the price of securities, the possession of such information becomes extremely important [Hazen T. L., 1991]. This fact undermines not only the semi-strong and strong versions of the EHM, but even the weak version, because of the newly acquired importance of available information. With the introduction of the chaos theory historical data become fundamental for any investment decision.

1.7 Brief introduction to Artificial Neural Networks (ANNs)

Prometheus speaks of the fruits of his transgression against the gods of Olympus: his purpose was not merely to steal fire for the human race but also to enlighten humanity through the gift of intelligence or nous: the rational mind. This intelligence forms the foundation for all of human technology and ultimately all human civilization [Luger G. F., 2009].

Intelligence is too complex to be described by any single theory; instead, researchers are constructing a hierarchy of theories that characterize it at multiple levels of abstraction. ANNs[2] (also called ANN or neural networks), genetic algorithms[3] and other forms of emergent computation have enabled us to understand the processes of adaptation, perception, embodiment and interaction with the physical world that must underlie any form of intelligent activity.

Artificial intelligence (AI) is a branch of computer science that studies the application of intelligent behavior to machines. This definition has a relevant shortcoming, researchers has not yet found a clear and acceptable definition of "intelligent behaviour". [Luger G. F., 2009].

Artificial intelligence is a discipline that has made a major contribution to the advancement of the entire computer science. It was influenced by a number of different disciplines including philosophy,

[2] An artificial neural network is a computer software program that mimics the human brain's ability to classify patterns and to make forecasts or decisions on past experience.

[3] Genetic Algorithms are mathematical adaptive heuristics which exploit a randomized search in order to solve mainly optimization problems. Based on an finite dimensional set (population) of individuals, i.e. solutions of the problem, GAs mimic the principles of natural selection and "survival of the fittest", adapting and evolving solutions to real world issues.

mathematics, psychology, cybernetics, cognitive science. AI is the study of the theoretical foundations, methodologies and techniques that allow scientists to design hardware and software systems capable of providing performances that once were exclusive domain of human intelligence. The purpose of AI is not to replicate human intelligence, but to reproduce or emulate some of its features. Nowadays certain human performances - for example, the ability to solve problems through inferential processes - can be supplied by a machine [Mercadal D., 1990].

AI was born as a discipline that studies the theoretical foundations, the principal methods, design criteria and construction of programs that allow the computer to create mechanisms and to perform activities that are commonly regarded as exclusive domain of human intelligence.

ANNs and genetic architecture shift the emphasis of AI away from the problems of symbolic representation and sound inference strategies to issues of learning and adaptation. The intelligence of a neural network does not require that the world be recast as an explicit symbolic model. Rather, the network is shaped by its interactions with the world, reflected to the implicit traces of experience. Neural networks can answer a number of questions that may be outside the expressive abilities of symbol based AI[4]. An important class of such questions concerns perception. Nature is not generous as to deliver our perceptions to a processing system as neat bundles of predicate calculus expressions. Neural networks offer a model of how we might recognize “meaningful” patterns in the chaos of sensory stimuli [Luger G. F., 2009].

1.8 My objective

From the beginning of time it has been man’s common goal to make his life easier. The prevailing notion in society is that wealth brings comfort and luxury, so it is not surprising that there has been so much work done on ways to predict financial markets [Patel M. B. et al., 2014].

A stock market is the centre of a network of transactions where securities buyers meets sellers at a certain price. It is essentially dynamic, non-linear, nonparametric and chaotic in nature.

Forecasting security prices is a difficult task from both theoretical and practical point of view because they are influenced by many factors, both economic and non-economic.

Many statistical and econometric models have been developed by researchers for the purpose of predicting stock prices but no technique or combination of techniques has been successful enough.

[4] Symbolic artificial intelligence is the collective name for all methods in artificial intelligence research that are based on high-level "symbolic" (human-readable) representations of problems, logic and search. Symbolic AI was the dominant paradigm of AI research from the middle fifties until the late 1980s.

Unlikely other classical techniques used for time series forecasting (e.g ARMA [5]), ANNs are able to approximate any nonlinear function without using a-priori information about the data series.
Recent studies have shown the classification and prediction power of the ANNs. The efficiency of neural networks in predicting stock market cannot be overlooked since an 96% accuracy of prediction has even been reached using self-learning neural networks [Kar A. 2010]. Considering recent developments in the study of neural networks, researchers and investors are hoping that the market mysteries can be unraveled [Patel M. B., 2014].
My objective is to implement, through the neural network toolbox of Matlab environment, an ANN for financial time series forecasting. In particular, the aim of this thesis is to determine whether an ANN model can predict the S&P 500 index price one-day ahead.
The process of developing a proper ANN for time series forecasting will be structured as follows:

- determining what is to be forecast;
- selecting the data;
- pre-processing the data;
- define the network architecture;
- training and testing.

1.9 History of ANNs

The attempts for understanding the human brain date back to ancient times. The first attempt was made by Hippocrates as Fischler and Firschen reported in their book "Intelligence: The Eye, the Brain and the Computer".
In 1890 William James tried to understand the workings of the brain but with poor results. The first analogy between the human brain and the computer was proposed in 1936 by Alan Turing, but only in 1943 Warren McCulloch and Walter Pitts, a neurophysiologist and a mathematician, reproduced a simple neural network employing electrical circuits connected together. They proposed an electronic model of the neuron, consisting of a summing amplifier and variable resistors. The model was activated and it sent an output a signal when the summation of the input signals exceeded a

[5] ARMA is statistical analysis model that uses time series data to predict future trends. It is composed by an autoregressive part (AR) and by a moving average model (MA). When estimating the next time series value, ARMA models take into account past estimations or forecasting errors.

certain threshold. McCulloch and Pitts also demonstrated that neural networks are similar to a Turing machine[6], so that any operation performed by a neural network could also be performed by a computer. The neural network developed by McCulloch and Pitts is the model upon which almost all modern neural systems are based [Fabbri G. & Orsini R., 1993].

After several years John Von Neumann used the researches of McCulloch and Pitts as a tool for his courses on computer theory. Wiener in 1949 published his book titled 'Cybernetics', based on a new interpretation about the universe and the behaviuor of its many elements.

Another fundamental book was published by Donald Hebb in 1949 entitled "The Organization of Behaviour", which explained the theory of reinforcement of standard connections. Many of the recent learning laws are based on the original interpretation introduced by Hebb.

In the '50s, at the beginning of the computer era, it became possible to test theories about the neural networks using computer simulations. Much of the researches in this period were carried out by Nat Rochester and other scholars at the research laboratories of IBM [M. Buscema, 1997].

Another milestone in the history of neural networks was written in 1956 at the seminar held at Dartmouth College, where the word artificial intelligence was created.

In 1956 John Von Neumann wrote the book "The Computer and The Brain", which introduced suggestions on the imitation of simple neural functions through the use of repeaters. In 1957 F. Rosenblatt began his work on the Percepton, a pattern of a neural system assembled into an hardware, which is the most dated neural systems that is still used today. He proposed a general class of neural networks defined as perceptons and invented a learning algorithm based on an iterative error correction that was much more powerful than the Hebbian rule.

In 1969 Marvin Minsky and Seymour Papert conducted a detailed study on the real possibilities of neural networks. Though in theory neural networks could artificially reproduce many complex human performances, in practice they did not know the techniques to determine the nature of synapses that were supposed to connect different neurons.

Consequently, the initial enthusiasm for neural networks faded and for more than ten years neural networks remained in the shadows [M. Buscema, 1997].

[6] A Turing machines is an abstract model of computation. It provides a precise, formal denition of what it means for a function to be computable. A Turing machine can be thought of as a finite state machine sitting on an infinitely long tape containing symbols from some finite alphabet Σ.

In 1982 there was a new interest in neural network technology. The credit for this change is due to John Hopfield of Caltech University. The key to success of Hopfield is represented by the practicality of his theories.

Hopfiled explained how, through the use of neural networks, it is possible to equip electronic computers with associative memories. Thanks to this stored memories, neural networks could correct any error in the data processed by a computer.

In 1986 D.E. Rumelhart, G. E. Hinton and R.J. Williams described an algorithm which could determine the nature of the connections of the neural networks architecture. This technique, called error backpropagation, iteratively compare the performance provided by a network with the desired one and modify the interconnections to obtain the desired performance.

CHAPTER 2 NEURAL NETWORKS

"Much memory, or memory of many things, is called experience."

Hobbes

This chapter briefly describe the biological neuron and presents the artificial neuron as the fundamental part of a neural network, including discussions on different activation functions (also referred to as transformation or transfer function), learning rules and neuron geometry. It also covers supervised and unsupervised learning, with an introduction to different types of supervised and unsupervised networks. They include feedforward neural networks, simple recurrent neural networks, radial basis function neural networks and self-organizing feature maps.

Neural networks are nonlinear models that can be trained to map past values of a time series, and thereby extract hidden structure and relationships governing the data.

The recent rise in neural network research activity has been a consequence of the realization that neural networks have a strong predictive power, greater than those of other techniques in many applications. Neural networks methodology can be considered as a multivariate nonlinear nonparametric inference technique that is data driven and model free [Michael A. E., 1994].

Multivariate refers to the neural network input comprising many different variables whose interdependencies and causative influences are exploited in predicting future behavior of a temporal sequence. Neural networks are nonparametric and model free because of the lack of any a priori assumption regarding the relation between the input variables and the output variables [Deboeck G. J., 1994]. Furthermore neural networks are trained to learn from experience without any constraint and they are shaped only by the input data. The study of neural networks is based on the basic unit of the human brain called biological neuron.

2.1 The Biological Neuron

The modern science of the central nervous system began in 1880 when Camillo Golgi invented a technique for coloring the nerve fibers, whose application allow to identify the blank space between dendrites[7] and axons[8], called synapses[9] [Fabbri G. & Orsini R., 1993].

The structure of the human brain is represented by a network of about one hundred billion neurons connected together. The number of these connections is estimated at about a million billion. The interaction between these nerve cells generate the brain activity (e.g. thoughts, actions and feelings).

The biological neuron (Fig. 1) comprises:

- The cell body and its extensions;
- The nerve fiber with ramifications.

[7] An axon is a single nerve cell that conducts impulses away from the cell body. Nerve cells are the primary cells in the nervous system. They are responsible for relaying electrical messages to cells and tissues in other organ systems.

[8] Dendrites are the segments of the neuron that receive stimulation in order for the cell to become active. They conduct electrical messages to the neuron cell body for the cell to function. Dendrites have various compartments known as functional units that are able to compute incoming stimuli. These functional units are involved in processing inputs and are composed of the subdomains of dendrites such as spines, branches, or groupings of branches.

[9] Synapses are junctions that allow a neuron to electrically or chemically transmit a signal to another cell. Synapses can either be excitatory or inhibitory. An excitatory synapse is defined as a synapse in which the nerve impulse (i.e. The movement of action potential along a nerve fiber in response to a stimulus such as touch, pain, heat or cold) in a presynaptic cell tend to increase the probability of the postsynaptic cell to fire an action potential. An inhibitory synapse is defined as a synapse in which the nerve impulse in a presynaptic cell results in a reduced likelihood for a postsynaptic cell to fire an action potential. A presynaptic cell is the cell that releases neurotransmitter that will stimulate the postsynaptic cell. A postsynaptic cell is a cell that receives a signal (binds neurotransmitter) from the presynaptic cell and respond with depolarization in an electrical synapse , that is a change in the charge of a cell. Usually cells are negatively charged relative to their surroundings. This negative internal charge of the cell shifts to a positive through the process of depolarization.

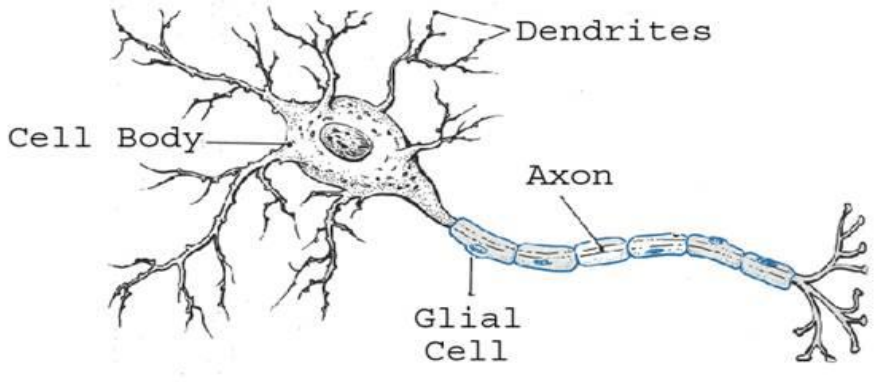

Fig. 1 - the biological neuron

The connections between neurons occurs through specific formations called synapses. A synapse is the point of contact between the axon terminal of a neuron (the presynaptic neuron) and the dendritic branch of another neuron (postsynaptic neuron). This description of a neuron is simplified, but it captures those features that are relevant to neural models of computation. Synapses have two important properties:

- they transmit the signal in one direction only;
- they are chemical neurotransmitters.

Usually artificial neural models tend to mimic the connection effect between all the synapses. Actually, the geometric configuration and the distinction between excitatory and inhibitory synapses result in interesting interactions that are able to perform basic operations. In general it is important to note that the inhibitory synapses tend to exercise a kind of veto on the signal that travels to the postsynaptic neuron.

The neuron generates electrical signals, nerve impulses, which are propagated as a result of chemical and physical conditions of the cell membrane.

The information content of the brain is represented by the activation values of all the neurons; information processing is determined by the flow of signals between the various neurons that excite or inhibit each other. The memory, that derives from the synaptic connections, is able to bind or

retrieve memories from incomplete information. The interesting plasticity[10] of synapses and the continuing association of concepts allow the individual to categorize and generalize the environmental stimuli and to adapt to changes in the outside world: in other words to learn from experience [Ferrari A., 1996].

Another important component of neural networks are the glial cells. This non-neuronal cells provide support and protection for neurons. Four main functions of glial cells:

- to surround neurons and hold them in place;
- to supply nutrients and oxygen to neurons;
- to insulate one neuron from another;
- to destroy pathogens and remove dead neurons.

If you want to make a comparison between the human brain and an ANN with the classical processing structure, it could be argued that in a computer the information is processed rapidly and serially, according to a strict schedule that is perfectly explicit. In the brain the processing speed is much lower, the information is distributed on a multitude of elementary neurons, and then can be treated in parallel. A computer works on the basis of a binary code while the brain is based on less precise communication methods but it can adjust the number of synapses in a complex manner in order to adapt its functioning to the experience.

The main difference between the human brain and an ANN is that humans can forget but neural networks cannot. Once fully trained, a neural net will not forget. Whatever a neural network learns is hard-coded[11] and becomes permanent. A human's knowledge is volatile and may not become permanent. There are several factors that cause our brain cells to die and if they do, the information that is stored in that part is lost and we start to forget.

Another key difference is accuracy. Once a particular application or process is automated through a neural network, the results are repeatable and accurate. Whether the process is replicated one thousand times or one million times, the results will be the same and will be as accurate as calculated the first time. Human beings are not like that. The first 10 processes may be accurate, but later we may start to make mistakes in the process.

[10] Plasticity is the ability of neurons to adapt their behavior to different needs arising over time. For example, when a neural network is damaged the other neurons of the network adjust their behavior to solve the problem.

[11] hard-coded means that the information is memorized into the architecture of the artificial neural network.

2.2 The Artificial Neural Network

Since the early 1990s when the first usable types of ANNs emerged, they have rapidly grown in popularity.

ANNs are based only superficially on the biological model and today we do not know actually how the brain works. Many researchers are working on the problem with the understanding of the brain and everyone is aware that the results of this research will be available only in a few decades [Zhang H., 2010].

An ANN is a computer software program that mimics the human brain's ability to classify patterns or to make forecasts or decision based on past experience. The human brain relies on neural stimuli while the ANN uses mathematical data sets, which take the form of specific series of information.

ANNs differ from human brain in several ways. Their capacity is very small, a few hundred neurons, versus the billions of neurons of the human brain. Another difference is that ANNs work only with numbers, while the brain get simultaneous inputs from the five senses. The interconnections of the neurons in an ANN are much more limited than in the case of the neurons in the human brain. A biological neuron is interconnected to a thousand or more other neurons, while an artificial neuron is interconnected to a limited number of hidden neurons incorporated in the network. Moreover, ANNs employ a learning algorithm[12] to autonomously extract the functional relationship between inputs and outputs and encode it in connection weights [Ray Tsaih et al., 1998]. The brain and the ANNs learn from their environment and encrypt this learning by altering the connections between their processing elements, neurons in the case of the human brain, nodes in the case of ANNs.

The complexity of the nervous system is based on two basic concepts, that are basis of the theory of ANNs:

- the capacity of the memory is proportional to the number of connections between neurons and not to the number of neurons;
- the ability to learn is made possible through the strength of the connections. In other words, the incidence of a neuron on another depends on how the connections amplifies the signal that passes through it (called "strength of the connection").

[12] A learning algorithm can be defined as a process that allow the memorisation of the patterns and the subsequent response of the ANN.

The previous concepts represent the basis of the theory of ANNs.

An ANN consists of a set of neurons connected together, also called neurons. Each unit receives multiple inputs from other neurons and produces an output. The input of a unit is equal to the weighted sum of the signals from all neurons connected to it.

A neural network is not programmed, but trained. Training a network simply means providing pairs of inputs and outputs, without explaining how the output is to be obtained from the input. The training process can be described as follows: at the beginning connections between neurons are rated at random and the network receive the desired output. The network, which is controlled by a learning algorithm, compares the actual output with the expected one, and gradually change the connections to minimize this difference. In this way the right synapses are strengthened while interconnections are inhibited. At the end of the process there will be a scheme of standard connections such that the network will always provide the correct output.

In an ANN the information is processed by different neurons in parallel processing and emerges as a collective phenomenon: a defect or a partial leakage information sinks into the collectivity and is neutralized.

This highlights two fundamental aspects of ANNs that link the elasticity of the human brain:

- noise tolerance and the ability to operate also with uncertain, incomplete or slightly incorrect data;
- robustness, or the ability to maintain its functionality even in the case of loss of a substantial proportion of neurons.

Note that the knowledge gained by the ANNs cannot be analyzed since it is hidden in the values assumed by the connections. In other words, the network is unable to provide an explanation about a certain output. Thanks to their adaptability[13], i.e. the ability to learn, to self-organize[14] and to generalize[15], ANNs can be used in those areas where inputs are ambiguous and imprecise and where recognition capabilities of classification or prediction are required.

[13] The adaptability is the ability of an ANN to change in response to changing conditions.

[14] Self-organization identifies the ability of neurons to comply with the learning law used to train the network. This involves changing the individual weights (i.e. synaptic weights) in response to input changes.

[15] ANNs are able to produce an answer to a problem that they have never seen before, using similar or related information. This ability is known as generalization.

2.3 The artificial neuron

The artificial neuron (AN) is a simple schematic of the biological neuron. It is characterized by a set of formal synapses that correspond to the terminals of other neurons, by a threshold and by an activation function (f_{AN}) (fig. 2). The AN is also called unit or node. An AN implements a nonlinear mapping from $\mathbb{R}^n$ usually to [0,1] or [-1 ,1], depending on the activation function used. That is,

$$f_{AN}: \mathbb{R}^I \rightarrow [0,1]$$

$$f_{AN}: \mathbb{R}^I \rightarrow [-1,1]$$

where I is the number of input signals to the AN. An AN receives a vector of I inputs signals,

$$X = (x_1\ x_2 \ldots x_I).$$

To each input signal, x_i, is associated a weight, w_i, to strengthen or deplete the input signal. The weights take into account the impact caused by the input signal.

Each weight can be seen as a measure of the strength of connection between neurons. This is something that takes into account the change in synaptic strength of biological neuron. Weights are adaptive coefficients within the network, which determine the intensity of the input signal. The initial weight for a node can be modified in response to various inputs, coherently with the rules provided for the learning process.

The AN computes the net input signal, and uses a activation function f_{AN} to compute the output signal, o_j, given the net input. The strength of the output signal is further influenced by a threshold value, θ, also reffered to as bias.

The value of the output of a node is given by the activation function which has the purpose to block the activation inside the node up to the threshold level, beyond that level the unit emits an output signal.

The net input signal of an AN is computed as the weighted sum of all input signals,

$$net\ input = \sum_{i=1}^{n} x_i\ w_i$$

Each input is multiplied by its relative weight and all the products are added together inside the neuron. The value of the sum obtained is compared with the predetermined threshold: if it is greater the neuron generates a signal, otherwise no signal is emitted by the neuron. Equivalently, the threshold value, θ, can be subtracted from the net input. Depending on whether the difference yields a positive or negative result, the network will respectively generate or not a signal.

$$A_i = \sum_{i=1}^{n} x_i \, w_i - \theta_i$$

Consequently

$$f_{AN}(A_i) = f_{AN}\left(\sum_{i=1}^{n} x_i \, w_i - \theta_i\right)$$

That is the activation function of A_i .

The basic functions of an artificial neuron are:

- determination of the strength of each signal;
- calculation of the total combination of input signals;
- comparison between the total value and the threshold value;
- determination of potential output values.

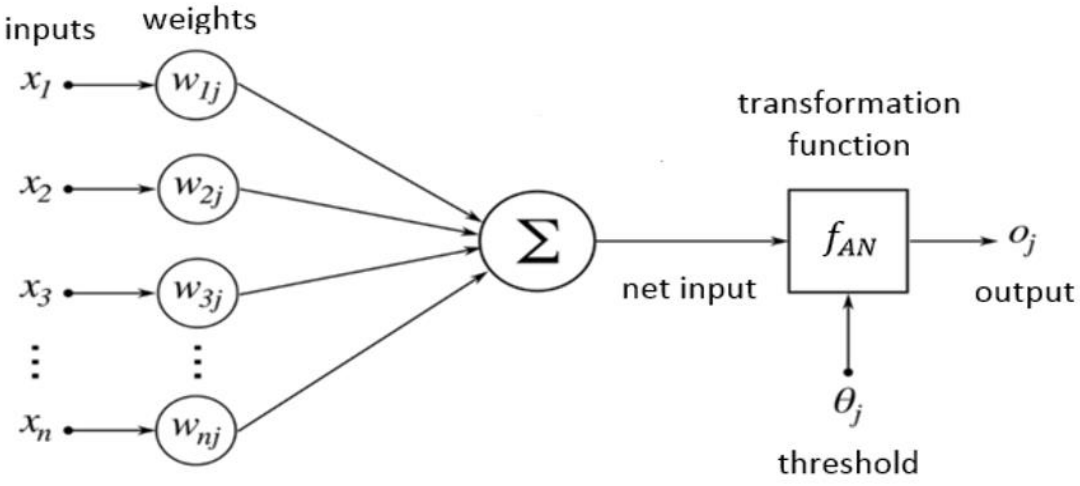

Fig. 2 - the artificial neuron

For each node there may be several inputs but only one output. In response to the signals, which must arrive simultaneously, the unit may be activated or not according to threshold value.
The earliest example of neural computing is the McCulloch-Pitts model. It is based on the following assumptions:

- each neuron has a fixed threshold, θ;
- the inputs to a McCulloch-Pitts neuron are either excitatory (+1) of inhibitory (-1);
- at each time step the activation function multiplies each input, x_i, by its corresponding weight and sums the results; if the sum is greater than or equal to the threshold and the neuron do not receive any inhibitory input, the output of the neuron (V_i) is equal to 1, otherwise to -1.

It is possible to summarize these results with the McCollugh-Pitts output rule:

$$V_i = \begin{cases} 1 : \sum_{i=1}^{n} x_i \, w_i \geq 0 \; AND \; no \; inhibition \\ 0 : otherwise \end{cases}$$

McCulloch and Pitts showed how the neurons could be constructed to compute any logical function, demonstrating that systems of neurons provide a complete computational model.
Figure 3 shows the McCulloch-Pitts neurons for computing logical functions. The “and neuron” has three inputs x, y and the bias. They have weights of +1, +1, and -2, respectively. Thus, for any values of x, the neuron computes the value of $x + y - 2$: if this value is less than 0, it returns -1, otherwise a +1. The table 1 illustrates the neuron computing $x \wedge y - 2$, where " $\wedge$ " represents the and logic function “and”. The “or” logic function is represented by " $\vee$ ". In a similar fashion, the weighted sum of input data for the “or” neuron, see figure 3, is greater than or equal to 0 for both x and y equal -1.

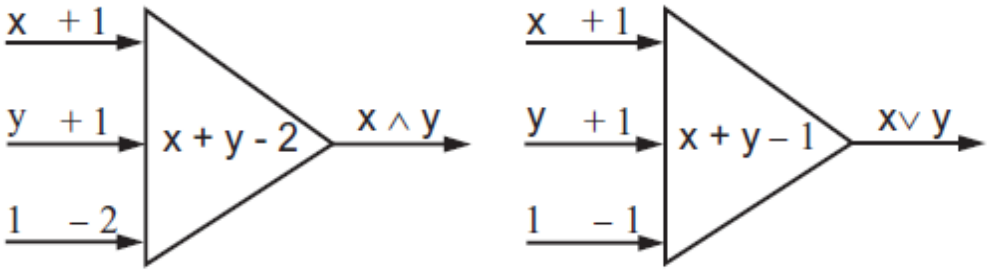

Fig. 3 - McCulloch-Pitts neurons to calculate the logic functions “and” and “or”.

x	y	x + y – 2	Output
1	1	0	1
1	0	–1	–1
0	1	–1	–1
0	0	–2	–1

Table 1 - McCulloch and Pitts model for logical “and”.

2.4 Activation functions

The activation function (also called transfer function) determines the type of response that a neuron is able to emit. The activation function, f_{AN}, receives the net input signal and the bias and determines the output of the neuron. Activation functions are often uniformly monotonic,[16] with

[16] In order to understand why activations functions are often uniformly monotonic, first consider what it means for an input to a unit to have an excitatory influence on the output of the unit. Such an input must have the property that an increase in its value must result in an increase in the output of the unit, as long as all other inputs are held costant. Furthermore, this should be true regardless of the value of other inputs. A similar property should hold for an inhibitory input, where the output of the unit must decrease as the value of the input is increased in this case. This is the basic idea behind the notion of uniform monotonicity. Therefore, an activation function is uniformly monotonic if and only if each input maybe classified as solely excitatory or solely inhibitory, independently of the values actually taken on by any other input [Rumelhart et al., 1986].
During the training phase, the learning algorithm link the weights to each neuron, meaning that it defines how much each input neuron should influence the neurons in the hidden layer. If the activation function is not monotonic then increasing the neuron's weight might cause it to have less influence, instead of a greater influence. The result would be a chaotic behavior during training, with the ANN unable to converge to the global minimum.

$$f_{AN}(-\infty) = 0 \; or \; f_{AN}(-\infty) = -1$$

and

$$f_{AN}(\infty) = 1$$

There are different types of activation functions:

1. The threshold function, or McCulloch.Pitts (MCP) is generally used in situations where the output is either 0 or 1. The model of McCulloch and Pitts refers to the actual morphology of the neuron and reproduces its behavior. In this model the neuron generates a signal when the sum of the input received exceeds a certain threshold value [Floreano D. , 1996]. Kuan and White (1994) describe this threshold feature as the fundamental characteristic of nonlinear response in the neural network paradigm. They describe it as "the tendency of neurons to be quiescent to modest levels of input activity, and to become active only after the input activity passes a certain threshold".

   The threshold function produces one of two scalar output values, depending on the threshold value. The equation of the threshold function is:

$$f_{AN}(x) = \begin{cases} 1 \;\; if \; x \geq \theta \\ 0 \;\; if \; x < \theta \end{cases}$$

   when the net input is less than the threshold value the output is 0, otherwise the output value is 1 as it is possible to see in figure 4.

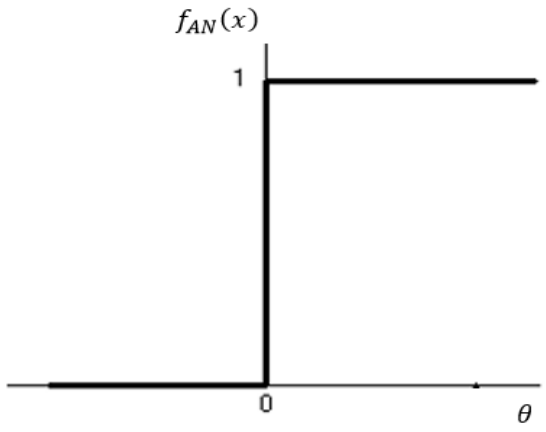

Fig. 4 - threshold function

A variant of the threshold function is the Signum function (fig. 5), that has an output equal to -1 or 1.

The equation of the Signum function is:

$$(x) = \begin{cases} 1 \ if \ x \geq \theta \\ -1 \ if \ x < \theta \end{cases}$$

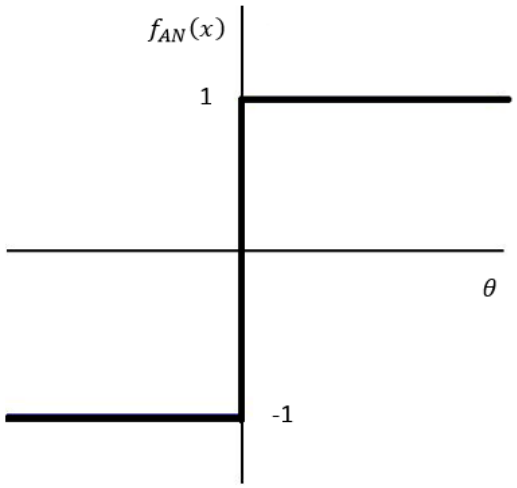

Fig. 5 - signum function

2. The linear function is characterized by the equation:

$$f_{AN}(x) = \alpha x$$

where α is the slope of the function. The linear function produces a linearly modulated output, where α is a costant (fig. 6).

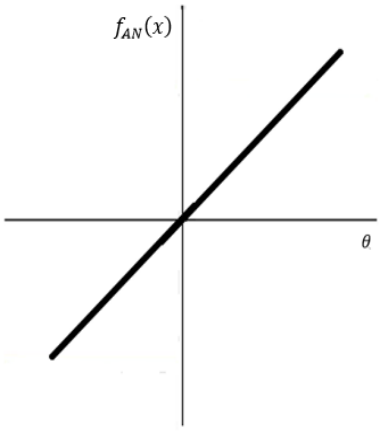

Fig. 6 - linear function

3. The ramp function is a combination of the linear function and the threshold one (fig.7). It has values of either between -1 and 1. The equation of the ramp function is:

$$f_{AN}(x) = \begin{cases} 1 & x \geq \theta \\ x & 0 < x < \theta \\ -1 & x < 0 \end{cases}$$

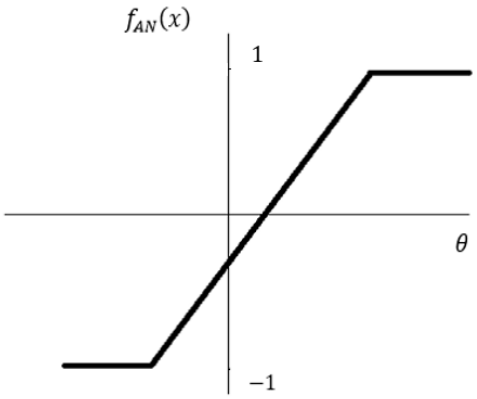

Fig.7 - ramp function

4. The sigmoid or logistic function is one of the most used and it is a sort of meeting point between the Boolean functions[17] and the linear one. The equation of the sigmoid function is:

$$f_{AN}(x) = \frac{1}{1 + e^{-kx}}$$

where k is a constant that controls the steepness of the function (fig.8); for $k \to \infty$ the sigmoid function tends to the threshold function. The sigmoid function is a continuous version of the ramp function.

[17] Boolean functions are functions that return either TRUE or FALSE. They are based on the operations AND, OR and NOT, and their elements are from the domain of Boolean algebra.

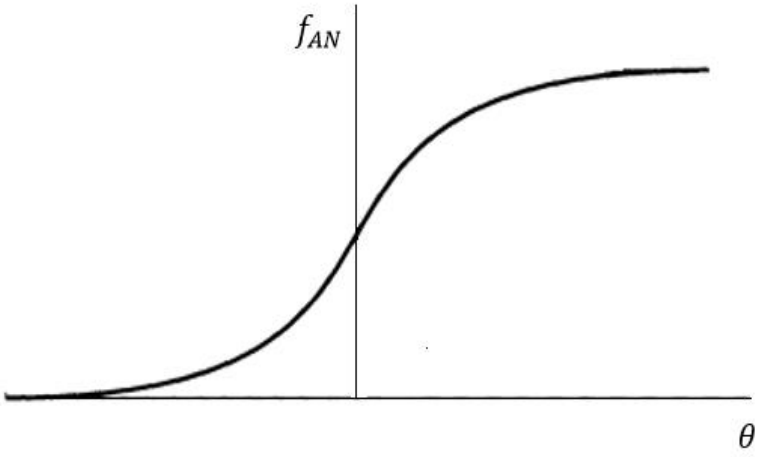

Fig. 8 - sigmoid function

Furthermore, the sigmoid transfer function takes the input, which can have any value between plus and minus infinity, and squashes the output into the range 0 to 1.

5. The tan-sigmoid or hyperbolic tangent transfer function is the modification to the sigmoid function, resulting in transfer function symmetrical with respect to the origin (fig. 9). Tan-sigmoid transfer function produces an output in the range of -1 to 1. This function is described the following formula:

$$T(S) = \tanh(S) = \frac{e^{s} - e^{-s}}{e^{s} + e^{-s}}$$

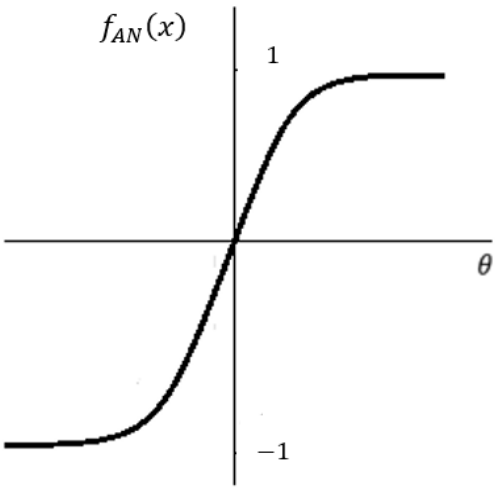

Fig. 9 - tan-sigmoid function

2.5 Neural Network learning

Given a dataset and the learning algorithm to be used, an ANN has the capacity to learn the optimal values for the weights and the threshold. Learning consists in adjusting weights and threshold values until a certain criterion is satisfied [Engelbrecht A. P., 2007].

The learning process takes place when the weights are changed in relation to certain inputs. To obtain a proper and efficient network, several learning methods that allow the modification of the weights in response to different inputs and outputs have been developed. In particular, there are three main learning methods:

- Supervised learning: It is characterized by the presence of an "external teacher[18]" which provides the information used for correction of the response of a neural network. The learning process provides the input data and the desired outputs to the network. This data set is called the "training set". The training phase may require a long time period and it is considered complete when the network produces the desired outputs for a given input sequence. When the learning process ends, the weights are frozen, meaning that they are kept at their final values. Using this technique, a network can perform tasks such as making decisions, generalize etc.

  Supervised learning has two different phases: (a) training also known as learning, (b) generalization also known as recall. In the first phase of the training process the neural network learn the information contained in the training set. During the generalization process the neural network uses the model to analyze new inputs.

- Unsupervised learning: the output is not taken into account and the network does not contemplate the presence of an "external teacher" to update its weights; this is rather an internal monitoring of performance based on local information and internal control. The network searches the possible trends in input signals and performs adjustments in accordance with the specifications of the considered network. Currently, unsupervised learning is subject to innumerable researches.

[18] "External" means extraneous to the neural network, i.e. not generated by the neural network.

- Reinforcement learning: there are some situations where the information that comes from the environment is only a signal that indicates the goodness of an action or a sequence of actions. The feedback signal that comes from the environment is interpreted as a positive reinforcement or negative response by the neural network.

  Reinforcement learning models have a long tradition that goes beyond the implementation of connectionists[19]. The first ideas of reinforcement learning, related to the development of human behavior, were based on the researches of psychologists. These ideas were later formalized and applied in engineering for control systems and artificial intelligence.

  Reinforcement learning is a form of supervised learning because it is characterized by a signal that comes from the environment, but this signal does not provide a precise information that can be used by the system to identify the correct answer. The environment that generates the signal does not indicate the correct answer, but it provides a valuation about the response of the system. This means that the system must be able to generate random responses and observe the reinforcement value provided by the environment. The learning process consists in changing the inner parameters of the system to increase the probability of emitting responses that receive positive reinforcements and to decrease the probability of emitting responses that receive negative reinforcements. The periodic generation of random responses can be seen as an exploration process that plays a central role in reinforcement learning. When the space of the possible pairs of input-output is too large to be thoroughly explored, the system must be able to guess, on the basis of the previous experience, which is the most appropriate response.

2.6 Neural Network Architecture

An ANN may consist of an input layer, a certain number of hidden layers and an output layer. Artificial neurons in one layer are connected to the artificial neurons in the next layer. The number of neurons to be included in each input and output layer depends on the number of available inputs and on the desired output. The process that leads to the right number of neurons for the hidden layers is quite complex and requires several attempts to get to the optimal solution.

[19] Connectionism is an approach to the study of human cognition that utilizes mathematical models, known as connectionist networks or artificial neural networks. The term was introduced by Donald Hebb in 1940s [Elman, Jeffrey L. et al., 1996].

Some neural networks use only two layers, as inputs are directly transformed into outputs. This is possible when there is a good similarity between the inputs and outputs. Otherwise, as in most cases, it is necessary to use hidden layers useful to create an internal representation of the input signal. The ability of processing information grows in relation to the number of layers used. This is a very important feature of the ANNs [McNelis P. D., 2005].
The first ANNs were composed only by two layers, the input layer and the output one. These types of ANNs were the basis for the development of two fundamental learning laws, the Hebbian rule and the Widrow-Hoff one. Another important learning law that was developed later than the other two is the backpropagation learning algorithm.

2.7 Hebbian rule

Most of the existing learning laws derive from the Hebbian learning rule. In 1949, D. Hebb published an article to describe the changes that take place at the cellular level when an animal learns. In this article he stated that " When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes place in firing it, some growth process or metabolic change occurs in one or both cells such that A's efficiency, as one of the cell firing B is increased".
This concept of learning can be formalized by the following Hebbian formula:

$$\Delta w_{i,j} = \varepsilon X_i X_j$$

where

- $\Delta w_{i,j}$ is the change in the weight $w_{i,j}$, that connect neuron j to neuron i;
- ε is the constant of proportionality called the learning rate;
- and X_j indicate the value assumed by two generic neurons that form the network in the equilibrium state.

You may notice that the higher the value of the neurons i and j, the higher the weight change ($\Delta w_{i,j}$) between these two neurons.

The formula above, while giving good results in some practical situations, it is not able to guarantee the ability to learn for any function $Y = f(x)$, because of the limitations of the Hebbian rule. Infact, it is possible to prove that this rule is not able to describe:

- how to calculate the activity of neurons;
- how to decrease the activity of neurons;
- under what conditions the connections should be strengthened.

2.8 Widrow-Hoff rule

Widrow-Hoff rule, also known as delta rule, is based on the idea of changing continuously over time (t) the weights of the connections to reduce the difference between the desired output and the actual value of the output neuron. The scope of this rule is limited to simple problems of classification and the process to reduce the error can last a long time. In its simplest formulation the rule can be written as:

$$\Delta w_{i,j}(t) = \alpha e_i a_j$$

where

- α is the constant of proportionality called the learning rate,
- $e_i = t_i - o_i$ is the error for the unit i and it is defined as the difference between the desired output t_i and the actual output o_i;
- a_j is the activation function of a given input vector.

A variant of this rule is defined as the mean squared error rule, often called MSE rule. In this case, the total error is calculated as follows:

$$E(x_k, w_n) = \frac{1}{P} \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2$$

where

- p is the index of training patterns, from 1 to P, where P is the total number of patterns;

- m is the index of network outputs, from 1 to M, where M is the total number of output neurons;
- x is the input vector, from 1 to k;
- w is the weight vector, from 1 to n;
- i and j are the indices of weights, from 1 to N, where N is the number of weights;
- $e_{p,m}$ is the training error at output m when applying pattern p and it is defined as

$$e_{p,m} = t_{p,m} - o_{p,m},$$

in which

- t is the desired output vector;
- o is the actual output vector.

The objective of Widrow-Hoff rule is to find a set of weights that minimizes the error function through the gradient descent method (also known steepest descent method) [Yu H. & Wilamowski B. M., 2010]. This method is completely founded on a mathematical basis. At any iteration, the weights are changed by an amount proportional to the negative value of the first derivative of the total error with respect to the weights.

$$\Delta w_{ij}(t) = -\alpha \frac{E(x_k, w_n)}{\partial\, w_{ij}}$$

where is $E(x_k, w_n)$ is the total error. The application of this rule allows to reach an equilibrium point, but in a very slow descent. Graphically, the total error of the network can be represented in a multi-dimensional space, where there are as many axes as the number of the weights (in our case $w_1\ and\ w_2$), plus one axes for the error term. The error reaches its minimum value in the lowest point of the curve also called the equilibrium point. Usually, when there is a long and narrow valley in the error function surface, the steepest descent method is particularly slow, so an additional parameter called momentum term is needed. In this situation, the direction of the gradient is almost perpendicular to the long axis (x) of the valley (fig. 10). The system thus oscillates back and forth in the direction of the short axis (y), and only moves very slowly along the long axis of the valley (fig. 11). With momentum m, the weight update at a given time t becomes

$$\Delta w_{ij}(t) = -\alpha \frac{E(x_k, w_n)}{\partial\, w_{ij}} + m\, \Delta w_{ij}(t-1)$$

The momentum term helps average out the oscillation along the short axis while at the same time adds up contributions along the long axis (fig. 12) [Rumelhart et al., 1986].

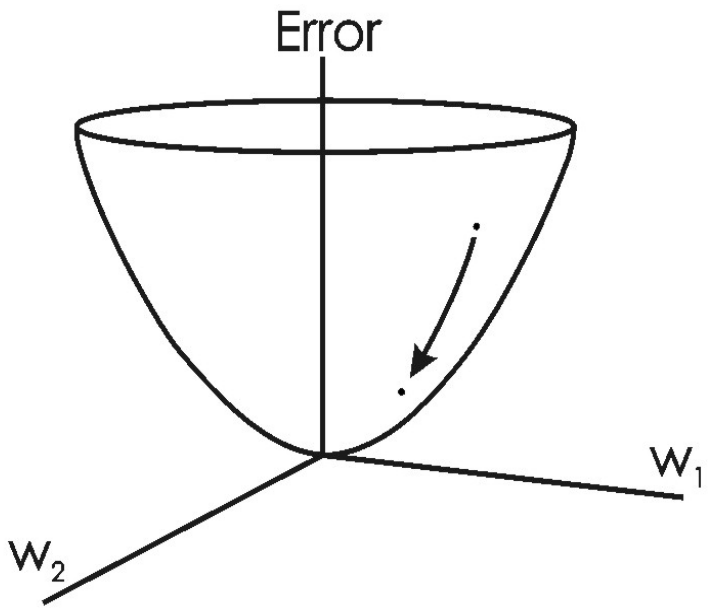

Fig. 10 - gradient descent method

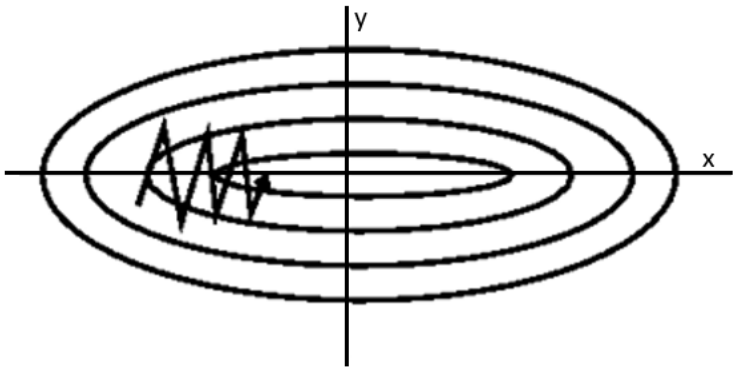

Fig. 11 - gradient descent without momentum term

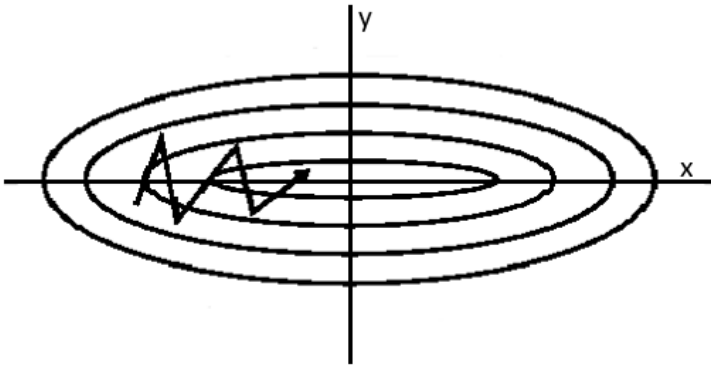

Fig. 12 - gradient descent with momentum term

2.9 Backpropagation algorithm

Backpropagation algorithm for error correction is not attributed to a specific researcher because it made its appearance almost simultaneously in several scientific articles. The first implementation of this rule is due to Amari (1967), which introduced a single neuron hidden for solving problems of nonlinear classification. This approach was refined by Bryson & Ho (1969), but the final contribution was given by the studies of Rumelhart, Hinton and Williams (1968), who explored the potential applications of this learning algorithm, restoring enthusiasm to the scientific community.

The backpropagation algorithm is based on an extension of the Widrow-Hoff rule, which calculates the difference between the actual output value and the desired one.

The procedure is based on the gradient descent method and can be divided in two phases, feedforward phase and feedback phase. In the first phase the input of the network is propagated forwards and then it is possible to compute the output for each neuron. For each neuron, the desired output is compared to the actual one and then the difference between these two values is computed. In the second phase the error is sent backwards to update the weights. Only when these two steps are completed a new input is fed to the network.

The Back-propagation method is applicable to neural networks with any number of layers of connections and with very different architectures[20].

Consider a two-layer feedforward network with input neurons x_k hidden neurons h_j and output neurons y_i. The value of the response for each unit is t_i^u , where u indicates a specific pattern input.

[20] The description of the Back-propagation algorithm is similar to the one presented by Hertz Krogh and Palmer, 1991

Each output unit receives synaptic weights w_{ij} of all the hidden neurons and each unit receives hidden synaptic weights v_{jk} of all the input neurons. All the neurons have a continuous and nonlinear activation function, that can be for example the logistic function:

$$f_{AN}(A) = \frac{1}{1+e^{-kA}},$$

where $A_i = \sum_{i=1}^{n} x_i \, w_i - \theta_i$.

Given an input pattern x^u, the activation of all the hidden neurons h_j is computed:

$$h_j^u = f_{AN}\left(\sum_{k=0} v_{jk} x_k^u\right)$$

and subsequently the activation of all the output neurons of y_i

$$y_j^u = f_{AN}\left(\sum_{j=0} w_{ij} h_j^u\right).$$

In both cases the neurons have a connection with a bias unit that has a constant value equal to -1 . Given that we wish to reduce the difference between the desired response t_i^u and the output of the network y_j^u, the error function E_w is:

$$E_w = \frac{1}{P}\sum_u \sum_i (t_i^u - y_i^u)^2$$

which, if expanded, becomes

$$E_w = \frac{1}{P}\sum_u \sum_i \left[t_i^u - f_{AN}\left(\sum_j w_{ij} \, f_{AN}\left(\sum_k v_{jk} x_k^u\right)\right)\right]^2$$

E_w depends on two groups of synaptic weights (w_{ij} and v_{jk}) and we can use the method of gradient descent function to get the changes in the synaptic weights. The matrix of connections w_{ij} between the hidden neurons and the output neurons is shown in the following formula:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \sum_u \delta_i^u \, h_j^u$$

where $\delta_i^u = (t_i^u - y_i^u) \, f_{AN}^{\cdot}(A_i^u)$

To calculate the delta of the weights of the lower layer v_{jk}, we must consider the variation of the activation of the inner neurons with respect to the change of synaptic weights v_{jk} that transmit signal from a constant input [Fabbri G. & Orsini R., 1993].

$$\Delta v_{jk} = -\eta \frac{\partial E}{\partial v_{jk}} = -\eta \sum_u \frac{\partial E}{\partial h_j^u} \frac{\partial h_j^u}{\partial v_{jk}}$$

which develops as follows

$$\Delta v_{jk} = -\eta \sum_u \sum_i (t_i^u - y_i^u) \, f_{AN}^{\cdot}(A_i^u) \, w_{ij} \, f_{AN}^{\cdot}(A_j^u) x_k^u.$$

Now we can substitute δ_i^u into the previous formula, obtaining

$$\Delta v_{jk} = -\eta \sum_u \sum_i \delta_i^u \, w_{ij} \, f_{AN}^{\cdot}(A_j^u) x_k^u.$$

For each hidden unit h_j the synaptic weights of the lower layer require the sum of the products between the delta δ_i^u of the upper layer of neurons to which the hidden unit is connected and the corresponding synaptic weights w_{ij}. In other words errors (delta, δ_i^u) of the upper layer are propagated backwards through the same synaptic connections and added to each unit from which they received the signal. The deltas for the hidden neurons δ_j^u are given by

$$\delta_j^u = f_{AN}^{\cdot}\left(A_j^u\right) \sum_i w_{ij}\ \delta_i^u$$

the modifications of synaptic weights v_{jk} take the same form of the modifications of synaptic weights w_{ij} , in particular

$$\Delta v_{jk} = \eta \sum_u \delta_i^u\ x_k^u$$

Considering that the delta of layer neurons are obtained according to the delta of the upper layer, the rule of Back-propagation is applicable to neural networks with any number of layers each consisting of any number of nodes [Fabbri G. & Orsini R., 1993].

In brief, the operation of the algorithm is to present the input pattern and propagate the activation layer by layer to the output unit; therefore the modification of the synaptic weights w_{mn} (where m is the index of the unit of the upper layer and n is the index of the unit of the lower layer) is given by

$$\Delta w_{mn} = \eta \sum_u \delta_m^u\ x_n^u$$

where

$$\delta_m^u = (t\ {}_m^u - \ y_m^u)\ f_{AN}^{\cdot}\ (A_m^u)$$

if m corresponds to the output unit, or differently

$$\delta_m^u = f_{AN}^{\cdot}\ (A_m^u) \sum_s w_{sm}\ \delta_s^u$$

where m corresponds to the hidden neurons and s indicates the unit of the higher layer with respect to m.

Most supervised neural learning applications use backpropagation as the learning algorithm. Backpropagation as a learning method has two limitations. One limitation is that the network

training may easily stop because of being stuck in one of the locally optimal solutions. A second limitation occurs in the degree of difficulty of the training data itself [Pendharkar P. C., 2002]. Besides the standard backpropagation learning algorithm that uses gradient descent method there are several different types of backpropagation learning algorithms.

2.9.1 Steepest descent learning algorithm

The steepest descent algorithm (also called gradient descent algorithm) is a first-order backpropagation algorithm. It uses the first-order derivative of the total error function to find the minima in the error space. Remember that the total error function is the sum of the squares of the individual errors:

$$E(x_k, w_n) = \frac{1}{P}\sum_{p=1}^{P}\sum_{m=1}^{M} e_{p,m}^2$$

Moreover, the gradient, g, can be defined as the first-order derivative of the total error function:

$$g = \frac{E(x_k, w_n)}{\partial w} = \left[\frac{\partial E}{\partial w_1}\frac{\partial E}{\partial w_2}\dots\frac{\partial E}{\partial w_N}\right]^T$$

With the definition of gradient g, the update rule of the steepest descent algorithm could be written as

$$w_{k+1} = w_k - \alpha \mathrm{g}_k$$

where α is the learning constant and k is the index of iterations [Yu H. & Wilamowski B. M., 2010].

2.9.2 Gauss-Newton learning algorithm

Levenberg-Marquardt learning algorithm can be thought of as a combination of steepest descent and the Gauss-Newton algorithms. The Gauss-Newton learning algorithm can be explained starting from the update rule for the Newton's method, that is

$$w_{k+1} = w_k - H_k^{-1} \mathrm{g}_k$$

where H is the Hessian matrix of the second order derivatives of the total error function. The calculation of the Hessian matrix that could be very complicated. In order to simply the calculating process, Jacobian matrix J is introduced as

$$J = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \cdots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & \cdots & \frac{\partial e_{1,2}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & \cdots & \frac{\partial e_{1,M}}{\partial w_N} \\ \frac{\partial e_{P,1}}{\partial w_1} & \frac{\partial e_{P,1}}{\partial w_2} & \cdots & \frac{\partial e_{P,1}}{\partial w_N} \\ \frac{\partial e_{P,2}}{\partial w_1} & \frac{\partial e_{P,2}}{\partial w_2} & \cdots & \frac{\partial e_{P,2}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{P,M}}{\partial w_1} & \frac{\partial e_{P,M}}{\partial w_2} & \cdots & \frac{\partial e_{P,M}}{\partial w_N} \end{bmatrix}$$

where e is the individual error for to the different indexes of patterns and outputs, respectively p and m [Yu H. & Wilamowski B. M., 2010]. By integrating the formula of the gradient and the total error function, the gradient vector can be calculated as

$$g_i = \frac{\partial E}{\partial w_i} = \frac{\partial \left(\frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2 \right)}{\partial w_i} = \sum_{p=1}^{P} \sum_{m=1}^{M} \left(\frac{\partial e_{p,m}}{\partial w_i} \, \partial e_{p,m} \right)$$

Combining the Jacobian matrix J and the previous equation, the relationship between J and the gradient vector is defined as

$$g = Je$$

where the error vector e has the form

$$e = \begin{bmatrix} e_{1,1} \\ e_{1,2} \\ \dots \\ e_{1,M} \\ \dots \\ e_{P,2} \\ e_{P,1} \\ \dots \\ e_{P,M} \end{bmatrix}$$

Furthermore, the Hessian matrix can be approximated as follows

$$H \approx J^T J$$

Finally, the update rule of Gauss-Newton algorithm is presented as

$$w_{k+1} = w_k - (J_k J_k^T)^{-1} J_k e_k$$

The Gauss-Newton has a relevant problem, the Hessian matrix $(H \approx J^T J)$ may not be invertible.

2.9.3 Levenberg-Marquardt learning algorithm

Levenberg-Marquardt learning algorithm can be thought of as a combination of steepest descent and the Gauss-Newton algorithms. In order to make sure that the approximated Hessian matrix is invertible, Levenberg-Marquardt algorithm introduces another approximation to the Hessian matrix:

$$H \approx J^T J + \mu I$$

where

- μ is always positive and it is called the combination coefficient;
- I is the identity matrix.

By combining the previous equation and update rule of Gauss-Newton algorithm, the update rule of Levenberg-Marquartd algorithm can be written as

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k$$

Levenberg-Marquardt learning algorithm can be considered as a combination of steepest descent and the Gauss-Newton algorithms. Thus, it switches between the two algorithm during the training process. When the combination coefficient μ is near zero, the Levenberg-Marquardt learning algorithm is just the Gauss-Newton algorithm. When the combination coefficient μ is very large, the Levenberg-Marquardt learning algorithm becomes the steepest descent algorithm [Yu H. & Wilamowski B. M., 2010].

2.10 Types of ANNs

As with biological neural networks, ANNs can be structured in several different ways. In particular, they can be divided in five fundamental classes:

- Single-layer feedforward networks;
- Multilayer feedforward networks;
- Simple recurrent networks, such as the Elman simple recurrent neural networks;
- Radial basis function networks;
- Self-organizing maps.

Single-layer feedforward networks, multilayer feedforward networks, simple recurrent networks are supervised learning neural networks meaning that their learning is supervised. Radial basis function networks and self-organizing maps are unsupervised learning neural networks meaning that their learning is unsupervised.

2.11 Single-layer feedforward networks

An ANN is typically structured into layers of processing neurons. The most basic form is the single layer feedforward network, which consists of an input layer and an output layer. The input stimulus is fed directly into the output layer (fig. 13).

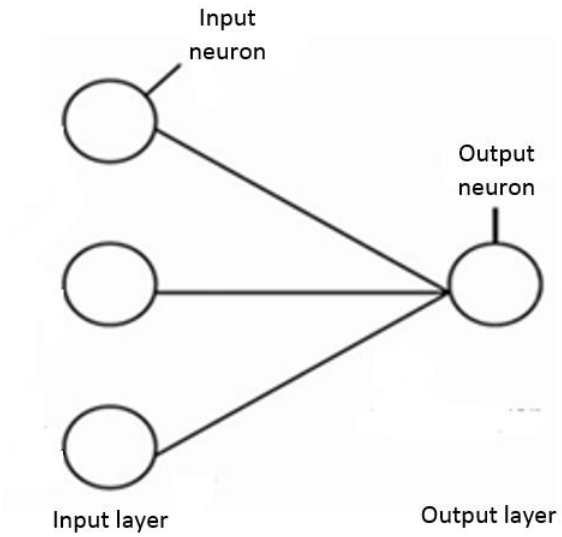

Fig. 13 - single-layer feedforward network

2.12 Multilayer feedforward networks

The multilayer feedforward network is more complex than the single-layer feedforward network because it has one or more hidden layers in its network (fig. 14). The added complexity is due to the higher number of parameters to estimate, which use up a valuable degree of freedom and cost a greater amount of training time. With more parameters there is also the possibility that the parameters estimates may converge to a local, rather than global, optimum. The feedforward network coupled with the sigmoid activation function is called multilayer percepton or MLP network.

The MLP network was born in relation to the need for a model of the real world, where problems are not always linearly separable and where it is not possible to apply network architectures formed by one or two levels of nodes. The development of the MLP network has allowed the expansion of the applications of neural networks in different subjects. The MLP network is the first architecture used for the development of a neural system not only for the simplicity of its implementation, but also for its ability to solve many of the typical problems of a neural application.

An MLP network is a feedforward neural network with one or more layers between input and output layer. It maps sets of input data onto a set of appropriate outputs [M. B. Patel, 2014]. It is also called backpropagation architecture, by association with the learning law that governs these networks (i.e. backpropagation algorithm). An MLP network operates in two distinct phases during the learning process:

- In the first phase a set of data is submitted to the network via the input layer. This set of data travels from one layer to another until it reaches the output layer, where the output is revealed;
- In the second phase, the output calculated by the network is compared to the desired output for that particular input. If the output generated is not the one desired, the network generates an error signal that travels backwards from the output layer to the input layer through the hidden layers, changing the weights to avoid any further error signal.

The MLP network id described by the following equations:

$$n_{k,t} = w_{k,0} + \sum_{i=1}^{i^*} w_{k,i} x_{i,t}$$

$$N_{k,t} = L(n_{k,t}) = 1/(1 + e^{-n_{k,t}})$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^*} \gamma_k N_{k,t}$$

where $L(n_{k,t})$ represents the logistic function. In the network there are i^* input variables x and k^* neurons. The net input is represented by $n_{k,t}$. This variable is transformed by the logistic function and becomes a neuron $N_{k,t}$ at time t. y_t is the forecast of the neural network (i.e. the output).

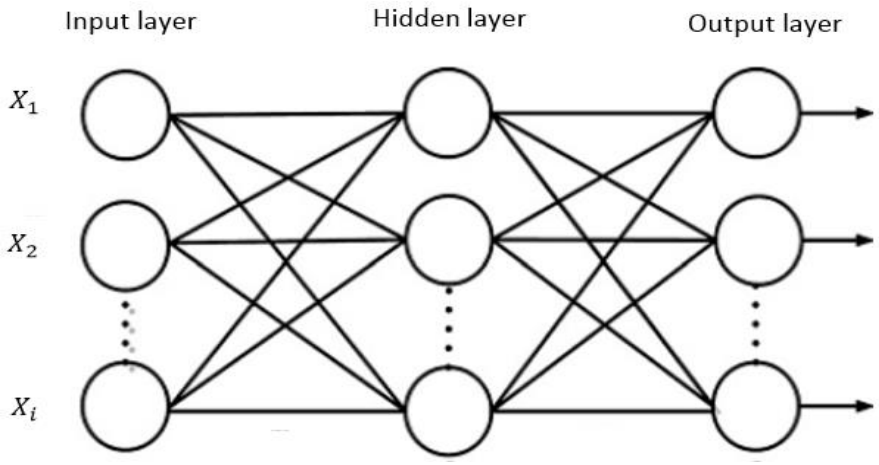

Fig. 14 - multi-layer feedforward network

2.13 Simple recurrent networks

Recurrent networks were inspired by the observation that the human brain is actually a recurrent network. The activation of a particular neuron can initiate a flow of activations in other neurons which in turn feed back into the neurons which initially fired [Anthony Brabazon, 2006].

Simple recurrent networks have feedback connections which add the ability to also learn the temporal characteristic of the data set. An example of a simple recurrent network is the Elman network (fig. 15). This network allows the neurons to depend not only on the input variables X_i, but also on their own lagged values. Thus, the Elman network builds "memory" in the evolution of the neurons. This type of network is similar to the commonly used moving average (MA) process in time-series analysis. In the MA process, the dependent variable is a function of observed inputs as well as current and lagged values of unobserved disturbance term or random shock [McNelis P. D., 2005]. The following equations represents the Elman recurrent network:

$$n_{k,t} = w_{k,0} + \sum_{i=1}^{i^*} w_{k,i} x_{i,t} + \sum_{k=1}^{k^*} \emptyset_k \, n_{k,t-1}$$

$$N_{k,t} = 1/(1 + e^{-n_{i,t}})$$

$$y_t = \gamma_0 + \sum_{k=1}^{k^*} \gamma_k N_{k,t}$$

The Elman recurrent network is a way of capturing memory in financial markets, particularly for forecasting high-frequency data such as daily or intra-daily data returns. While the use of lags certainly is one way to capture memory. Memory may also show up in the way the nonlinear structure changes through time. The use of the Elman network, in which the lagged neurons feed back into the current neurons, is a very good way to model this type of memory structure, in which the hidden layer itself changes through time, due to the feedback from past neurons.

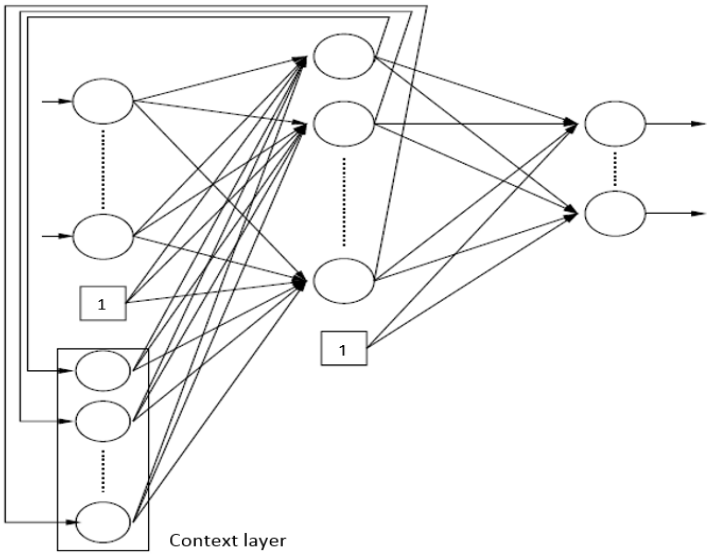

Fig. 15 - Elman recurrent network

The Elman recurrent network makes a copy of the hidden layer, which is referred to as the context layer. The purpose of the context layer is to store a previous state of the hidden layer, i.e. the state of the hidden layer at the previous pattern presentation. The context layer serves as an extension of the input layer, feeding signals representing previous network states, to hidden layer. The input vector is therefore equal to

$$x = (x_1, \dots x_{n+2}\ , x_{n+2,} \dots x_{n+1+k})$$

The Elman network is an explicit dynamic network. The feedforward network is usually regarded as a static network, in which a given set up of input variables at time t are used to forecast a target output variable at time t.

2.14 Radial basis function networks

Radial basis function network is a type of neural network that has been developed for both the supervised and the unsupervised learning paradigms.

A radial basis function neural network (RBFNN) is a standard feedforward neural network where hidden neurons do not implement an activation function, but represents a radial basis function. An RBFNN approximates a desired function by superposition of non-orthogonal, radially symmetric functions [Engelbrecht A. P., 2007].

The radial basis network function neural network makes use of the radial basis or Gaussian density function as the activation function, but the structure of the network is different from the feedforward or MLP networks. The input neuron may be a linear combination of regressors, as in the other networks, but there is only one input signal, only one set of coefficients of the input variables [McNelis P. D., 2005].

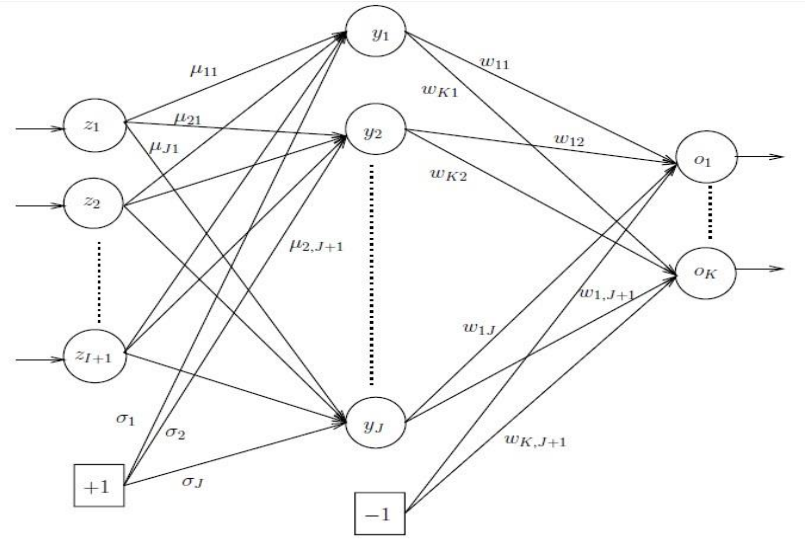

Fig. 16 - Radial basis function network

Figure 16 illustrates a general architecture of the RBFNN. The architecture is very similar to the one of a standard feedforward neural network, with the following differences:

- Hidden neurons implement a radial basis function, Φ. The output of each hidden unit is calculated as

$$y_{j,p}(z_p) = \phi\left(\left\|z_p - u_j\right\|_2\right)$$

where u_j represents the center of the basis function, and $\|.\|_2$ is the Euclidean norm[21].

- Weights from the input neurons to a hidden unit, referred to as $u_{i,j}$, represent the center of the radial basis function of hidden unit j.

[21] The Euclidean distance or Euclidean metric is the "ordinary" (i.e. straight-line) distance between two points in Euclidean space. With this distance, Euclidean space becomes a metric space. The associated norm is called the Euclidean norm. Older literature refers to the metric as Pythagorean metric. A generalized term for the Euclidean norm is the L^2 norm or L^2 distance.

- Some radial basis functions are characterized by a width, σ_j. For such functions, the weight from the basis unit in the input layer to each hidden unit represents the width of the function.

The output of an RBFNN is calculated as

$$o_{k,p} = \sum_{j=1}^{J+1} w_{k,j} y_{j,p}$$

Note that the output neurons of an RBFNN implement linear activation. Each hidden unit of the network implements a radial basis function. These functions, also referred to as kernel functions, are strictly positive and radially symmetric. A radial basis function (RBF) has a unique maximum at its center, u_j, and the function usually drops off to zero rapidly further away from the center. The output of a hidden unit indicates the closeness of the input vector, z_p, to the center of the basis function. In addition to the center of the function, some RBFs are characterized by a width, σ_j, which specifies the width of the receptive field of the RBF in the input space for hidden unit j [Engelbrecht A. P., 2007].

A number of different methods have been developed to train RBFNNs. The most widely used method is the one developed by Moody and Darken. They used gradient descent method to adjust weights, centers, and widths [Moody J.E. & Darken C., 1989].

2.15 Self-organizing feature maps

Kohonen developed the self-organizing feature map (SOM), as motivated by the self-organization characteristics of the human cerebral cortex. Studies of the cerebral cortex showed that the motor cortex, somatosensory cortex, visual cortex and auditory cortex are represented by topologically ordered maps. These topological maps form to represent the structures sensed in the sensory input signals. They are a type of unsupervised learning neural networks.

A self-organizing feature map is a multidimensional scaling method that projects an I-dimensional input space to a discrete output space, effectively performing a compression of input space onto a

set of codebook vectors[22]. The output space is usually a two-dimensional grid. The SOM uses the grid to approximate the probability density function of the input space, while still maintaining the topological structure of input space. That is, if two vectors are close to one another in the input space, so is the case for the map representation.

In a SOM neurons are usually organized on a rectangular grid, and neighbors are updated to also perform an ordering of the neurons. In the process, SOMs effectively cluster the input vectors through a competitive learning process, while maintaining the topological structure of the input space.

SOM training is based on a competitive learning strategy. Assume I-dimensional input vectors z_p, where the subscript p denotes a single training pattern. The first step of the training process is to define a map structure, usually a two-dimensional grid (refer to fig. 17). The map is usually square, but can be of any rectangular shape. The number of elements (neurons) in the map is less than the number of training patterns. Ideally, the number of neurons should be equal to the number of independent training patterns [Engelbrecht A. P., 2007].

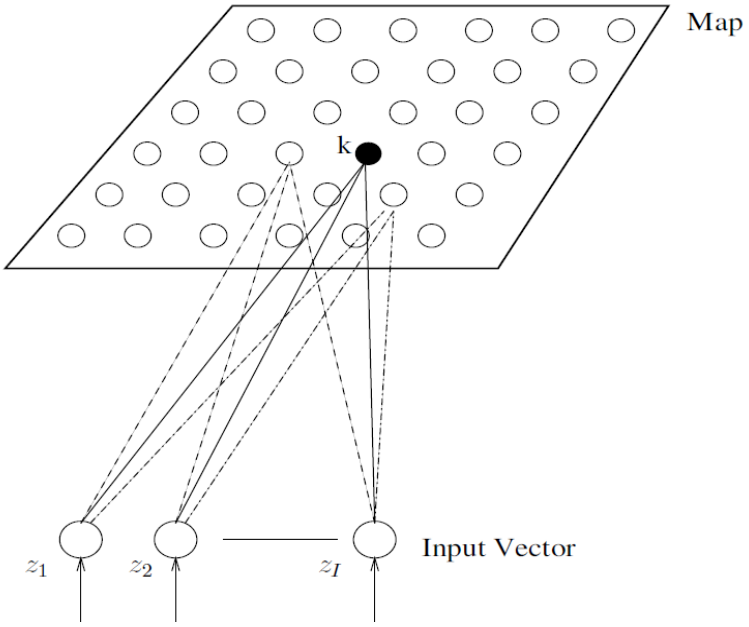

Fig. 17 - Self-Organizing Feature Maps

[22] A self-organizing network consists of a number of codebook vectors, which constitute cluster centers. The codebook vectors are of the same dimension as the input space, and their components are the parameters of the self-organizing network. The codebook vectors are called the neurons of the self-organizing network.

Each neuron on the map is associated with and I-dimensional weight vector that forms the centroid of one cluster. Larger cluster groupings are formed by grouping together "similar" neighboring neurons.

Initialization of the codebook vectors can occur in various ways:

- Assign random values to each weight $w_{kj} = (w_{kj\,1}, w_{kj2}, w_{KJI})$, with K the number of rows and J the number of columns of the map. The initial values are bounded by the range of the corresponding input parameters. While random initialization of weight vectors is simple to implement, this form of initialization introduces large variance components into the map which increases training time.
- Assign to the codebook vectors randomly selected input patterns. That is,

  $$w_{kj} = z_p$$

  with $p \sim U(1, P_t)$

  This approach may lead to premature convergence[23] of the ANN, unless weights are perturbed with small random values.
- Find the principal components of the input space, and initialize the codebook vectors to reflect these principal components.
- A different technique of weight initialization is due to Su et al., where the objective is to define a large enough hyper cube to cover all the training patterns [Su et al, 2010].

The algorithm starts by finding the four extreme points of the map by determining the four extreme training patterns. Firstly, two patterns are found with the largest inter-pattern Euclidean distance. A third pattern is located at the furthest point from these two patterns, and the fourth pattern with largest Euclidean distance from these three patterns. Weight values of the remaining neurons are found through interpolation of the four selected patterns, in the following way:

- Weights of boundary neurons are initialized as

  $$w_{1j} = \frac{w_{1j} - w_{11}}{J - 1}(j - 1) + w_{11}$$

[23] The convergence characteristic of an ANN can be described as the ability of the network to converge to specified error levels (usually considering the generalization error) The ability of the network to converge to a specific error is expressed as the number of times, out of a fixed number of simulations , that the network succeeded in reaching that error.

$$w_{Kj} = \frac{w_{Kj} - w_{K1}}{J-1}(j-1) + w_{K1}$$

$$w_{k1} = \frac{w_{K1} - w_{11}}{K-1}(k-1) + w_{11}$$

$$w_{kj} = \frac{w_{KJ} - w_{1J}}{K-1}(k-1) + w_{1J}$$

for all j = 2,…, J − 1 and k = 2,…,K − 1.

- The remaining codebook vectors are initialized as

$$w_{kj} = \frac{w_{kj} - w_{k1}}{J-1}(j-1) + w_{k1}$$

for all j = 2,…, J − 1 and k = 2,…,K − 1.

The standard training algorithm for SOMs is stochastic, where codebook vectors are updated after each pattern is presented to the network. For each neuron, the associated codebook vector is updated as

$$w_{kj}(t+1) = w_{kj}(t) + h_{mn,kj}(t)\left[z_p - w_{kj}(t)\right]$$

where mn is the row and column index of the winning neuron. The winning neuron is found by computing the Euclidean distance from each codebook vector to the input vector, and selecting the neuron closest to the input vector. That is,

$$\left\|w_{mn} - z_p\right\|_2 = \min_{\forall kj}\left\{\left\|w_{mn} - z_p\right\|_2^2\right\}$$

The function $h_{mn,kj}$ is referred to as the neighborhood function. Thus, only those neurons within the neighborhood of the winning neuron mn have their codebook vectors updated. For convergence, it is necessary that $h_{mn,kj} \to 0$ when $t \to \infty$.

The neighborhood function is usually a function of the distance between the coordinates of the neurons as represented on the map, i.e.

$$h_{mn,kj}(t) = H\left(\left\|c_{mn} - c_{kj}\right\|_2^2, t\right)$$

with the coordinates c_{mn}, $c_{kj} \in \mathbb{R}^2$. With increasing value of $\left\|c_{mn} - c_{kj}\right\|_2^2$ (that is, neuron kj is further away from the winning neuron mn), $h_{mn,kj} \to 0$. The neighborhood can be defined as a square or hexagon. However, the smooth Gaussian kernel is mostly used:

$$h_{mn,kj}(t) = \eta(t) e^{-\frac{\left\|c_{mn} - c_{kj}\right\|_2^2}{2\sigma^2(t)}}$$

where $\eta(t)$ is the learning rate and $\sigma(t)$ is the width of the kernel. Both $\eta(t)$ and $\sigma(t)$ are monotonically decreasing functions.

The learning process is iterative, continuing until a "good" enough map has been found. The quantization error is usually used as an indication of map accuracy, defined as the sum of Euclidean distances of all patterns to the codebook vector of the winning neuron, i.e.

$$\varepsilon_t = \sum_{p=1}^{P_t} \left\|z_p - w_{mn}\right\|_2^2$$

Training stops when ε_t is sufficiently small.

CHAPTER 3 ANNs IN FINANCE

"Any sufficiently advanced technology is indistinguishable from magic."
Arthur C. Clarke

Because of their ability to deal with uncertain and insufficient data which fluctuate rapidly over short time periods, ANNs have become very important in the field of finance. In particular, during the last decade they have been widely applied to the domain of financial time series forecasting. Furthermore, ANN techniques can be used extensively in the financial markets to help investors make qualitative decisions. In this chapter the importance of ANNs for stock market prediction is presented through a detailed literature review regarding this field.

3.1 Why ANNs?

In the last 20 years, neural networks have been widely applied in finance area. In particular, it is possible to identify three large areas of research for which neural networks seem to be particularly promising:

- function approximation;
- classification;
- time series forecasting.

In the case of functions approximation problems, ANNs have been applied in advanced pricing and risk management models which usually lack of a precise functional form for the evaluation of complex instruments such as American and Exotic options.

In classification problems, ANNs have the task of assigning the inputs to a number of pre-defined categories corresponding to the outputs. An example of classification problem can be predicting the odds that a house will be sold within the next six months, considering as initial inputs the features of a house (e.g. market price, number of bedrooms etc.).

Finally, ANNs are well suited to applications in financial time series forecasting. The purpose of most of these applications is to make profits through speculative trading.

There are several reasons that make ANNs valuable for time series forecasting. First, they are data-driven self-adaptive methods with few a priori assumptions. ANNs learn from examples and capture subtle functional relationship among the data even if the underlying relationship are unknown or hard to describe (for example in the case of nonlinear statistics). Second, they can deduce the unseen parts of a sample after analyzing the input data. Finally, they can approximate any continuous function to any desired accuracy [Zhang G. et al.,1998].

However, in each of the possible applications of an ANN, it is necessary to divide the data into in-sample and out-of-sample observations. The initial historical data on which the idea is tested and optimized is referred to as the in-sample data. The out-of-sample data provide a way to test the idea on data that has not been a component in the optimization model, in particular they are important to validate the initial data. ANNs can be used for both short term and long term financial time series forecasting.

In order to develop an efficient ANN it is important to evaluate the following elements:

- input variables: in choosing these variables it is necessary take into account that whichever type of input we feed to an ANN will be processed even non-significant variables. Consequently it's essential to feed to the network only explanatory variables;
- time horizon: this is an important element in time series forecasting since very short time horizons increase the number of correct forecasts; whereas, long term horizons are on average less correct;
- optimal level of learning: an excessively short training process does not allow the network to grasp the relationship between the different variables, while a too long training process could make the network unable to generalize [Michael A. E., 1994].

3.2 Time Series Forecasting with ANNs

Financial markets modeling is complex for several reasons. Financial markets are influenced by many elements including interest rates, exchange rates, and the rate of economic growth. There is no exact theory about how these factors affect the prices of financial assets, because these effects are based on complex processes. For example, a change in interest rates of a given country may have an impact on the exchange rate of a currency, changing the level of imports and exports to and from that country. Another problem of financial modeling is that, unlike other types of research fields, it is not possible to conduct controlled experiments since we only have a history of market events. In addition, some events that may affect the financial markets are unpredictable [Malkiel B. G., 2003].

Taken together, these problems imply that our ability to predict market movements will always be imperfect [Michael A. E., 1994].

Nonlinear modeling and forecasting of time series data has a relatively recent history. The statistics community has contstructed stochastic nonlinear models since about 1980.

Time series are a special form of data where past values in the series may influence future values, depending on the presence of underlying deterministic forces. These forces may be characterized by trends cycles and nonstationary behaviuor in the time series and predictive models attempts to recognize the recurring patterns and nonlinear relationships [Gately E., 1996].

Forecasting simply means understanding which variables lead to predict other variables [D. Mcnelis, 2005]. This means a clear understanding of the timing of lead-lag relations among many variables, understanding the statistical significance of these lead-lag relations and learning which variables are the more important ones to watch as signals for predicting market movements. Better forecasting is the key element for better financial decision-making, in the increasing financial market volatility and internationalized capital flows.

Stock market forecasting has always been a topic of great interest, both in the academic area as well as the real-life markets. The increased attention towards this area of study can be attributed to the fact that if the direction of the stock markets can be successfully predicted, the investors could then be better guided. The payoff from any investment decision and trading activity greatly depends on the market predictability.

The central idea to successful stock market prediction is achieving best results using minimum required input data and the least complex stock market model.

Forecasting stock returns or stock prices involves an assumption that fundamental information publicly available in the past has some predictive relationship to the future stock prices. The samples of such information include economic variables such as interest rates and exchange rates, industry specific information such as growth rates of industrial production and consumer price, and company specific information such as income statements and dividend yields. This is opposed to the general perception of market efficiency, previously discussed [Enke D. et al., 2005].

Approaches to forecasting the future price of a stock fall broadly into two categories. those that rely on technical analysis[24] and those that rely on fundamental analysis[25]. While technical analysis uses only historical data (past prices, volume of trading etc.) to determine the price of an asset, fundamental analysis is based on external information, that comes from the economic system surrounding the market. Such information includes interest rates, prices and returns of other assets and other macro-economic or micro-economic variables. The use of technical analysis goes against the grain of conservative academic opinion, which regards this behaviour as irrational given the efficient market hypothesis [Malkiel B. G., 2003].

[24] Technical analysis is the study of market action, primarily through the use of charts, for the purpose of forecasting future price trends. It makes the assumption that everything known about a stock is reflected in its market data, primarily its price but also volume etc. The intrinsic, or fundamental value if the stock is not considered. Technical analysis is considered by many to be the original form of investment analysis, dating back to 1800s.

[25] Fundamental analysis is a method of evaluating a security that attempts to investigate the intrinsic value of a stock in relation to an in-depth analysis of various economic, financial, qualitative and quantitative factors.

Financial forecasting or stock market prediction is one of the hottest fields of research lately due to its commercial applications owing to the high stakes and the kinds of attractive benefits that it has to offer. Generally there are three schools of thoughts regarding stock market prediction:

- The first school believes that no investor can achieve above average trading advantages based on historical and present information. This thought is based on the random walk hypothesis and the efficient market hypothesis;
- the second view is based on fundamental analysis. It encourages analysts to undertake in-depth studies into the various macro-economic factors and look into the financial conditions and results of the industry concerned to discover the extent of correlation that may exist with the changes in the stock prices;
- the third view is recently sprung up to develop ANN models for forecasting financial data. Most ANN based models use historical and present stock index data to predict future prices [Zhang Y. & Wu L., 2009].

ANNs have earned themselves an excellent reputation as non-linear approximators. Like other form of soft computing ANNs exhibit a high tolerance to imprecision and perform well in noisy data environments. Furthermore, they allow deep analysis of large sets of data especially those that have the tendency to fluctuate within a short time period.
For these reasons ANNs are considered as good candidates for financial problem domains, in particular the area of stock market prediction [Vanstone B. & Finnie G., 2008].

3.3 Literature review

A variety of papers within the category of time series prediction have been reviewed, and provide evidence and conclusions consistent with the above:

- Yao and Poh used technical indicators along with price information to predict future prices values. They achieved good returns, and found their models performed better using daily rather than weekly data [Yao and Poh,1995];
- Austin et al. developed a neural network that predicted the proper time to move money into and out of the stock market. They used two valuation indicators, two monetary policy

indicators, and four technical indicators to predict the four-week forward excess return on the dividend adjusted S&P 500 index. The results significantly outperformed the buy and hold strategy[26] [Austin et al., 1997];

- Kim et al. used backpropagation ANNs to predict future elements in the price time series in the KOSPI, that is the Korea Stock Exchange index [Kim et al., 2003];
- Mingo et al. Used time delay connections in enhanced neural networks to forecast IBEX_35 (Spanish stock index) close prices of one-day ahead [Mingo et al., 2002];
- Jaruszewicz and Mandziuk trained ANNs using both technical analysis variables and intermarket data, to predict one-day changes in the NIKKEY index. They achieved good results using MACD, Williams, and two averages, along with related market data from the NASDAQ and DAX [Jaruszewicz and Mandziuk, 2004];
- Chang and Foo used neural networks to predict the value of the future time series data and then used these predictions to compute various technical indicators used in technical analysis. Using the DEM/USD daily data, they predicted the high, the low and close prices three days into future.

Seyed Taghi et al. forecasted the daily direction of Standard & Poor's 500 (S&P 500) index using an ANN (ANN). In order to select the most influential features (factors) for their ANN, they conducted different experiments using 27 potential financial and economical variables along with a feature defined as the number of nodes of the ANN [Seyed Taghi et al., 2013].
The variables used by Seyed Taghi et al. were:

1. SPY_{t-1} , the return of the S&P 500 index in day t-1 ;
2. SPY_{t-2}, the return of the S&P 500 index in day t-2;
3. SPY_{t-3}, the return of the S&P 500 index in day t-3;
4. Oil, relative change in the price of the crude oil;
5. Gold, relative change in the gold price;
6. CTB3M, change in the market yield on US Treasury securities at 3-month constant maturity;

[26] Buy and hold strategy is an investment strategy in which stocks are bought and then held for a long period, regardless of the market's fluctuations. The buy and hold approach to investing in stocks rests upon the assumption that in the very long term (over the course of, say, 10 or 20 years) stock prices will go up, but the average investor doesn't know what will happen tomorrow.

7. CTB6M, change in the market yield on US Treasury securities at 6-month constant maturity;
8. CTB1Y, change in the market yield on US Treasury securities at 1-year constant maturity;
9. CTB5Y, change in the market yield on US Treasurysecurities at 5-year constant maturity;
10. CTB10Y, change in the market yield on US Treasury securities at 10-year constant maturity;
11. AAA, change in the Moody's yield on seasoned corporate bonds - all industries, Aaa;
12. BBB, change in the Moody's yield on seasoned corporate bonds - all industries, Baa;
13. USD-Y, relative change in the exchange rate between US dollar and Japanese yen;
14. USD-GBP, relative change in the exchange rate between US dollar and British pound;
15. USD-CAD, relative change in the exchange rate between US dollar and Canadian dollar;
16. HIS,Hang Seng index return in day t-1 ;
17. FCHI, CAC 40 index return in day t-1 ;
18. FTSE, FTSE 100 index return in day t-1 ;
19. GDAXI, DAX index return in day t-1 ;
20. V, Relative change in the trading volume of S&P 500 index;
21. XOM, Exxon Mobil stock return in day t-1 ;
22. GE, General Electric stock return in day t-1 ;
23. MSFT, Microsoft stock return in day t-1 ;
24. PG, Procter and Gamble stock return in day t-1 ;
25. JNJ, Johnson and Johnson stock return in day t-1 ;
26. DJI, Dow Jones Industrial Average index return in day t-1 ;
27. IXIC, NASDAQ composite index return in day t-1 .

Each of these factors had two levels, level 1 (also called 1) and level 0 (also called 0). Level 1 of a factor indicates that the corresponding factor was used as a feature, and level 0 representes the nonexistence of the factor. In addition to the 27 features, the number of hidden nodes was also considered as a potential factor in designing the experiments. The experiments with fewer input factors needed fewer hidden nodes and vice versa.

To make the total number of experiments practical, they used two approaches. In the first approach, for each pair of the features that were highly correlated, they kept one and the other was excluded. This was based on the fact that if two features were highly correlated (i.e. absolute value of the correlation between these two features is equal at least to 0.5), then both were inferred to contain

similar information. Hence, using one of them as the input was almost sufficient for an ANN to extract the desired pattern. The application of this approach led to exclude 7 of the 27 initial financial and economical variables. Thanks to this approach they omitted the following variables: DJI,IXIC, CTB6M, CTB1Y, CTB5Y, CTB10Y, and BAA.

The second approach was based on the concept of grouping. In this approach, they placed the remaining features of the first approach with almost similar nature in different groups. They employed the second approach to organize the remaining 20 features in six groups:

- The first group, denoted by G1, represents the S&P 500 index return in three previous days. The input variables included in this group are SPYt-1 , SPYt-2, SPYt-3;
- The second group, denoted by G2, represents the financial and economical indicators. The input variables included in this group are Oil, Gold, CTB3M, AAA ;
- The third group, denoted by G3, represents the return of the five biggest companies in S&P 500 financial and economical indicators. The input variables included in this group are XOM, GE, MSFT, PG, JNJ;
- The fourth group, denoted by G4, represents the exchange rate between USD and three other currencies. The input variables included in this group are USD-Y, USD-GBP, USD-CAD;
- The fifth group, denoted by G5, represents the return of the four world major indices. The input variables included in this group are HIS, FCHI, FTSE, GDAXI;
- The sixth group, denoted by G6, represents S&P 500 trading volume. The input variables included in this group is the volume of the S&P 500.

These six groups of factors were combined together in all the possible ways. Each of the possible combinations of these groups corresponds to a different ANN. Seyed Taghi et al. conducted the experiments in a way that for each observation of each possible combination, the corresponding ANN was initialized with a random value. After that, the initialized ANN was trained with the training dataset until the early stopping rule[27] terminates the training process. After simulating the trained network with the test dataset the forecasts were gathered. The response variable was the number of correct forecasts that had been produced in the test period.

[27] Early stopping rule is a method to avoid over-fitting, but it improves the generalization ability of ANNs. This method halts the training process when the performance with validation data stops improving.

After conducting the experiments and obtaining the results, they used the analysis of variance approach (ANOVA)[28] to determine the factors that had significant effects on the response variable. The main findings of this research are as follows:

1. A factor that significantly influences the daily direction of S&P 500 is the exchange rates between the US dollar and three main currencies including the British pound, Canadian dollar, and Japanese yen;
2. Using some of the variables including return of S&P 500 in previous days and relative change in the trading volume of S&P 500 not only is unhelpful, but also confuses the ANN and decreases the performance of ANNs for forecasting the daily direction of S&P 500;
3. The results obtained in this research are statistically supportable;
4. The factorial design of experiments helped to examine not only the main effects of the factors, but also the mutual effects between them. The combination of the network features is also important for an ANN to produce good results;
5. By statistical analysis, it was proven that the designed ANN had a good predictive power [Seyed Taghi et al., 2013].

Edward Gately developed an ANN to forecast the S&P 500 ten days into the future. The dataset for this model consisted of five years historical data (from March 1, 1991, to April 25, 1994) of the following initial inputs:

- S&P 500-C (close price);
- S&P 500-H (high price);
- S&P 500-L (low price);
- Nasdaq PHLX Gold/Silver Sector Index (XAU);
- Dow Jones Industrial Average Index (INDU);
- volume of Dow Jones Industrial Average Index (VOL. INDU);

[28] The one-way analysis of variance approach (ANOVA) is used to determine whether there are any significant differences between the means of three or more independent groups. Specifically, it tests the null hypothesis:

$$H_0: \mu_1 = \mu_2 = \mu_3 = \cdots = \mu_k$$

where μ = group mean and k = number of groups. If the one-way ANOVA returns a significant result, the alternative hypothesis (H_0) is accepted, which means that at least two group are significantly different from each other

- Dow Jones Transportation Average Index (TRAN);
- Dow Jones Utilities Average Index (UTIL)
- Thomson Reuters Core Commodities Index (CRB);
- S&P GSCI Crude Oil Index (SPGCCLP);
- US Dollar Index;
- 3-Month Treasury Bill Interest Rate;
- 1-Year Tresury Bill Interest Rate;
- S&P 100 index (OEX);
- Advance-Decline index (ADV.-DEC.) ;
- TRIN[29].

To find out which inputs were making the most significant contribution, Gatley created an ANN variable selection model. At each step of this model some variables were deleted and others added to the network due to their contribution when used to train a backpropagation network [Gately E., 1996]. This variable selection model can be divided in the following steps:

- determining which of the initial inputs are making contributions to the neural network;
- determining a suitable moving average for the volume input;
- determining which inputs to lag and the number of days to lag;
- determining which moving average period makes the largest contribution;
- determining the contribution of several technical indicators;
- selecting the definitive inputs.

Prior to training the ANN, initial data series were pre-processed. 3-Month Treasury Bill Interest Rate, 1-Year Treasury Bill Interest Rate and US Dollar Index were deleted in the pre-processing stage because they lacked date integrity. Also the variable S&P 100 index was deleted because it had essentially the same information of the S&P 500 index.

In the first step, a 3-layer backpropagation network (also called MLP) was trained using the initial input variables. The aim of this first step was to find out which inputs were making the most significant contribution to the target. The relative contribution to the target was determined by

[29] TRIN is a short-term technical analysis stock market trading indicator based on the Advance-Decline data.

summing the strength of connections (i.e. sum of the weights) between the input neurons and the neurons in the hidden layer. Inputs that had little effect on the output had weak connections to hidden layer and conversely, inputs that had a large effect on the output had strong connections to the hidden layer [Gately E., 1996].

In the second step, a 3-day, 5-day and 10-day moving averages of the volume of Dow Jones Industrial Average Index were added and the network was trained again to find out which inputs contribute the most to the solution.

In the third step, all the relevant inputs were lagged (10-day lag) and then the network was trained again using these lagged variables to see which inputs might be relevant lagged inputs.

In the fourth step, to find out which moving average period applied to the inputs has the greater effect, Gately trained a network using the 10-day moving average of all the inputs, then a 20-day moving average of the inputs and so on. In this case the inputs of the network were the moving average of original inputs.

Finally, the contribution to the target of several technical indicators was tested.

Testing process created a large number of possible indicators from moving averages to lagged inputs. The final definitive input variables chosen by Gately were:

- S&P 500-C (close price);
- S&P 500-H (high price);
- S&P 500-L (low price);
- Nasdaq PHLX Gold/Silver Sector Index (XAU);
- Dow Jones Transportation Average Index (TRAN);
- 10-day lag of Dow Jones Transportation Average Index;
- Dow Jones Utilities Average Index (UTIL);
- 30-day moving average of volume of Dow Jones Industrial Average Index;
- Thomson Reuters Core Commodities Index (CRB);
- 10-day lag of Thomson Reuters Core Commodities Index;
- S&P GSCI Crude Oil Index (SPGCCLP);
- 10-day lag of S&P GSCI Crude Oil Index.

Final results of Gatley's model showed a great relative contribution for all lagged inputs and a low contribution value for 30-day moving average of volume of Dow Jones Industrial Average, TRAN Index and S&P 500-L.

CHAPTER 4 RESEARCH

"Study the past if you would divine the future"
Confucius

The main purpose of this research is to determine whether an ANN model can accurately predict the price of the S&P 500 index one day forward. First of all, an ANN-based variable selection model is presented. This model explores the relationship between some initial input variables and the closing price of the S&P 500 index.

Another important question of this thesis is how the training algorithm, as well as the number of neurons in the hidden layer and the distribution of the training data, affect the accuracy of the network.

4.1 Methodology

In this first paragraph a step by step method to design an ANN for financial time series forecasting is described. Among the most widely used methods to develop an ANN there is the one of G. Deboeck. In 1994, he developed a seven-step methodology for constructing neural networks:

1. Data collection
2. Data analysis and transformation
3. Variable selection
4. Model development and optimization
5. Model verification
6. Model deployment

This process is not a single pass procedure but an iterative one. At any time, it may be necessary to re-examine the information from a priori step or even to collect new or different data [Deboeck G. J., 1994].

Following the work of Deboeck, Kaastra and Boyd proposed an eight step procedure to design a neural network forecasting model for financial and economic time series forecasting. The following framework is based on the previous work by Deboeck [Kaastra I. & Boyd M., 1996].

Step 1: Variable selection

Step 2: Data collection

Step 3: Data pre-processing

Step 4: Training, validation and testing sets

Step 5: Neural network paradigms

number of hidden layers

number of hidden neurons

number of output neurons

transfer function

Step 6: Evaluation criteria

Step 7: Neural network training

number of training iterations

learning rate and momentum

Step 8: Implementation

This method is the most widely used to design an ANN forecasting model for financial time series forecasting.

4.2 Variable selection

Success in designing a neural network depends on a clear understanding of the problem. Knowing which input variables are important in the market being forecasted is critical [Kaastra I. & Boyd M., 1996]. The selection of inputs having a predictive relationship to the target is as important as the selection of the best possible target to be forecasted. Selecting inputs to forecast the price of S&P 500 index one day in advance is a very difficult job because both the inputs and the target (i.e. output) used are noisy. For this reason, a variable selection model, that provides appropriate inputs by a trial and error process, is needed.

There are two approaches to the selection of which inputs to use and how often inputs should be sampled [Deboeck G. J., 1994].

The first approach, known as sensitivity analysis, can be used to determine which inputs are important to the neural network. This approach consists in ranking each input based on the contribution to the neural network and eliminating the bottom of the list.

The essential process is to select a set of data for input, train a network until an acceptable level of performance is achieved, apply sensitivity analysis and eliminate all the variables below a predefined threshold.

An alternative to sensitivity analysis is to determine the impact that setting an input to its average value has on the overall performance of the ANN. If the network is not using an input to discriminate between possible outcomes, fixing that input to its average value should actually improve the overall performance [Deboeck G. J., 1994].

Inputs for financial time series forecasting can be divided in two broad categories. The first category comprise inputs based on fundamental factors. These include the price of gold, oil, 3-month Treasury bill, gross national product (GNP).

The second category contains technical indicators, and include items such as volatility, relative strenght index (RSI), daily high low, and closing price, moving averages etc. [Gately E., 1996].

Technical indicators are underpinned by the concept of technical analysis. Advocates of technical analysis consider that it can be used to pre-process historical prices/volumes information to uncover patterns, which they can recur [Brabazon A. & O'Neill M., 2006].

Input selection technique of this thesis is based on an ANN-based variable selection model developed by Edward Gately. The first part of my research consists in three tests on Gatley's variable selection model:

- first test: final inputs of Gatley's variable selection model are tested using recent data;
- second test: Gatley's variable selection model as a whole is replicated and tested using recent data;
- third test: an improvement of Gatley's variable selection model is tested with recent data.

In all these tests, a 3-layer backpropagation network with a Levenberg-Marquardt learning algorithm was used to train the data. In order to test the contribution of the different input variables and to find a good measure of generalization of the network's predictive power, each ANN was trained one hundred times. The results of the ANN's performance were then expressed as averages of the R-squared and RMSE over all the simulations. Furthermore, all the ANNs were trained starting from different initial weights and biases [Bale M. H. et al., 2015].

The data collected for these three tests consist of five years daily historical data (2010-2015) of Gatley's input variables. The output variable is the one-day ahead closing price of S&P 500 index. The first four years of collected data were used to train, validate and test the ANN (i.e. in-sample analysis), while the last year of the collected data was used to make the one-day ahead predictions (i.e. out-of-sample analysis). The reason for choosing to predict the one-day ahead price instead of a longer look-ahead time interval is that the actual price of a stock market index tends to exponentially diverge from the actual one, as the prediction interval increases [Abhisher Kar, 2010].

Prior to training the ANN, initial data were pre-processed. This process consists in manipulating the original data to extract more information. First of all, the presence of outliers were checked. The latter were removed and replaced with their local median reference value.
After that, the data were normalized by standardizing the possible numerical range that each input vector elements can take. Moreover, all the given inputs were compared the target to determine whether a predictive relationship existed. Therefore, Pearson correlation coefficient was computed to find out the correlation between the designated inputs and the variable to be forecast. All inputs variable had a large correlation coefficient (above 0,7) implying a strong association between inputs and target.

4.3 Data collection

Data collection involves the acquisition of high-quality data from reputable sources that provide complete and consistent information. In this research, the dataset used for the analysis was obtained from Bloomberg platform. The dataset comprises five years of daily historical data (2010-2015) of the input variables provided by an improved version of Gatley's variables selection model. The output variable is the next day's closing price. The input variable selected are as follows:

- S&P 500-C (close price);
- S&P 500-H (high price);
- S&P 500-L (low price);
- Nasdaq PHLX Gold/Silver Sector Index (XAU);
- Dow Jones Transportation Average Index (TRAN);
- Dow Jones Utilities Average (UTIL);
- 3-day moving average of volume of Dow Jones Industrial Average Index (INDU);
- 10-day moving average of Thomson Reuters Core Commodities Index (CRB);
- S&P GSCI Crude Oil Index (SPGCCLP);
- 12-day Wilder RSI of S&P500-C;

This selection comprises both fundamental and technical inputs. Most of these variables are widely held US stock market indices. Let us give a detailed description of each input variable.

S&P 500 Index

The S&P 500 index is widely regarded as the best single gauge of large-cap U.S. equities. There is over USD 7.8 trillion benchmarked to the index, with index assets comprising approximately USD 2.2 trillion of this total. The index includes 500 leading companies and captures approximately 80% coverage of available market capitalization.

On March 4, 1957 the S&P 500 came into existence. It holds constituents that are primarily U.S. based. The common myth is that the S&P 500 represents the 500 largest companies in the United States. In reality, the S&P 500 is chosen by a select committee that uses the criteria of "market size, liquidity, and group representation". The majority of the S&P 500 constituents are Large Capitalization companies, however the minimum capitalization needed to get into the index is $3 billion. Moreover, there are certainly at least 20 companies included in the S&P 500 that are small and mid capitalization in nature.

The constituents are chosen from these major industry categories: Energy, Materials, Industrials, Consumer Discretionary, Consumer Staples, Health Care, Financials, Information Technology, Telecommunications Services, and Utilities". In addition, there are over 100 subgroups within these industries that determine the criteria.

Thomson Reuters Core Commodities Index

Founded in 1957, the CRB Index has a long history as the most widely followed index of commodities futures. As markets have evolved, the CRB has undergone periodic updates. the latest occurring in 2005. These updates include:

- Component Changes:
  Over the years, commodities including eggs, oats, lard, rubber, potatoes and wool have been dropped and replaced by more liquid and significant contracts.
- Weighting:
  The 10th revision (2005) included a weighting change from equal weighted components to a 4-tiered grouping system designed to reflect the significance of each commodity:

Energy: 39%

Agriculture: 41%

Precious Metals: 7%

Base/Industrial Metals: 13%

This index is comprised of 19 commodities: Aluminum, Cocoa, Coffee, Copper, Corn, Cotton, Crude Oil, Gold, Heating Oil, Lean Hogs, Live Cattle, Natural Gas, Nickel, Orange Juice, RBOB Gasoline, Silver, Soybeans, Sugar and Wheat. The index trades on the ICE Futures Exchange.

Dow Jones Transportation Average Index

The Dow Jones Transportation Average the oldest is U.S. stock index. The first Dow Jones stock index, assembled in 1884 by Charles Dow, co-founder of Dow Jones & Company, was composed of the stocks of nine railroad companies, including the New York Central and Union.

Today, the Dow Jones Transportation Average is composed of 20 stocks that are chosen to represent the transportation industry. Like other Dow Jones Averages, it is a price-weighted index, meaning that firm's relative weight in the index is determined by its share price, not by its market capitalization. As of 2015, some well-known companies tracked by this index include Southwest Airlines Co., Union Pacific Corporation and Continental Airlines, Inc.

Dow Jones Industrial Average Index

Charles H. Dow first unveiled his industrial stock average on May 26, 1896. In that period the stock market was not highly regarded. Prudent investors bought bonds, which paid predictable amounts of interest and were backed by real machinery, factory buildings and other hard assets.

Two-thirds of the actual DJIA's 30 component companies are manufacturers of industrial and consumer goods. The others represent industries as diverse as financial services, entertainment and information technology. Even so, the DJIA today serves the same purpose for which it was created. to provide a clear, straightforward view of the stock market and, by extension, the U.S. economy.

Nasdaq PHLX Gold/Silver Sector Index

The PHLX Gold/Silver sector index is a capitalization-weighted index composed by companies in the gold and silver mining industry. This index was set to an initial value of 100 in January 1979.

S&P GSCI Crude Oil Index

The S&P GSCI Crude Oil Index is designed as a benchmark for investment in the commodity markets and as a measure of commodity market performance over time. It is also designed as a tradable index that is readily accessible to market participants. In order to accomplish these objectives, the S&P GSCI is calculated primarily on a world production-weighted basis and comprises the principal physical commodities that are the subject of active, liquid futures markets.

Moving Average

The 3-day moving average of volume of Dow Jones Industrial Average Index and the 10-day moving average of Thomson Reuters Core Commodities Index are simple moving averages of the closing price. The n-day simple moving average is calculated simply by adding the closing prices of the last n days and divide by n. Here the notation n refers to the number of days.

Wilder RSI

Relative Strenght Index as defined by Wilder is a technical indicator that provides a measure of the size and direction of price movements. The RSI is obtained from the relative strenght (RS), that is computed as follows:

$$RS = \frac{EMA(Price\ increases, n)}{EMA(Price\ decreases, n)}$$

Where EMA is the exponential moving average and n refers to the number of days considered.
The relative strength is converted into the RSI as follows:

$$RSI = 100 - \frac{100}{1 * RS}$$

4.4 Data pre-processing

Data pre-processing consists in analyzing and transforming the input and output variables to minimize noise, highlight important relationships, detect trends and flatten the distribution of the variable to assist the neural network in its learning process the raw volume values [Kaastra I. & Boyd M., 1996].

The collected data was imported and organized in four matrices, two input matrices and two output matrices. The first pair of inputs and targets were used to train the ANN and the other pair was used to test the ANN.

All the inputs and the output of the ANN were pre-processed. Pre-processing permits to extract information that have real value to the ANN. The most widely used ways to pre-process the data are: telescoping the inputs[30], introducing moving averages instead of raw data and normalizing the data.

Introducing moving averages of the input variables is a useful pre-processing technique. It is often substituted to the raw data because it smooths out noise and improve the forecasting ability of the ANN. In this work, the moving average of the raw volume of an index used to train the ANN instead of the raw volume data. Furthermore, the rows of the data that show a constant value (i.e. their max. value minus their minimum value is equal to zero), are deleted because they do not provide any information to the ANN.

Normalizing the data is another important pre-processing technique. Normalization process consists in standardizing the possible numerical range that the data can take. It is generally desirable in order to remove the possibility that ANN's parameters are 'tuned' for a given range of input-output data and also to bring the inputs of the activation function inside the function's normal operating range [Refenes Apostolos-Paul N., 1995].

In this study the collected data can take only positive values, so input variables are normalized to lie within the range [0,1].

[30] Telescoping inputs is the practice of presenting data from previous time periods. This process consists in lagging the input variables to obtain more information and improve the performance of the ANN.

4.5 Training, validation and testing sets

Before implementing an ANN, the data needs to be divided into training, validation and testing sets. The training set is used to develop the forecasting model. In particular, it is used to compute the gradient and to update the ANN's weights and biases. The validation set measures how well the model interpolates over the training set. This dataset is used for monitoring the error during training. Usually during training, the error on the validation dataset decreases until over-fitting to occur at which point the error starts to increase again. ANN's learning parameters are set to ensure that over-fitting does not occur. A final check on the performance of the trained network is made using the testing set. This dataset is essentially used in an out-of-sample test between actual and predicted data [Kaastra I. & Boyd M., 1996].
In this research, the distribution of the training dataset was changed for each test to study its influence to the predicted prices.

4.6 Neural networks paradigms

This paragraph provides further details about the architecture of the ANN used to predict the price of the S&P 500 index. The structure adopted consisted of a 3 layer network which comprises the input layer, a hidden layer, and the output layer.

Number of hidden layers

The hidden layers provide the ANN with its ability to generalize. From a theoretical point of view an ANN with one hidden layer and a sufficient number of hidden neurons is able to accurately model any continuous function. Increasing the number of hidden layers can improve the performance of the ANN but it also increases computation time and the danger of overfitting.
Considering that there is not any established method for selecting the number of hidden layers, the heuristic of using one or at most two hidden layers is generally considered appropriate [Kaastra I. & Boyd M., 1996]. For the aims of this thesis, a single hidden layer has been used.

Number of hidden neurons

There is no proven theory for selecting the optimum number of hidden neurons. Thus, researchers fall back on experimentation. Masters suggests that for a three-layer ANN with *n* input neurons and *m* output neurons, the hidden layer should have sqrt($n \times m$) neurons [Masters T., 1993]. Baily and Thompson suggest that the number of hidden nodes in a three-layer neural network should be 75% of the number of input neurons [Baily D. & Thompson D.M., 1990]. Katz indicates that an optimal number of hidden neurons will generally be found between one-half to three times the number of input neurons [Katz J.O., 1992].

In order to investigate how the number of neurons in the hidden layer influence the ANN, the hidden layer size was changed for each test. In particular, ANNs with 10, 50 and 150 hidden neurons were tested. The number of hidden neurons for the tests were selected on the basis Anhishek Kar's research on stock prediction, where the performance of an ANN was investigated for a single hidden layer, varying the number of its nodes [Abhisher Kar, 2010].

Number of output neurons

Choosing the number of ouput neurons is a relatively simple matter compared to the determination of the optimum number of hidden neurons. The ANN was with a single output neuron to provide the predicted price of S&P 500 index.

Transfer function

Transfer functions are mathematical formulas that determine the output of a processing neuron. They are also referred to as transformation, activation or threshold functions. The majority of current neural networks models use the sigmoid function [Kaastra I. & Boyd M., 1996].

For the purposes of this research, the transfer function used in the hidden and output layer are respectively the tan-sigmoid and the linear transfer functions. The transfer function used in the hidden layer can process input values in the range $[-\infty, +\infty]$, so there won't be any problem for processing negative inputs and negative weights. The linear transfer function in the output layer is used for function fitting purposes.

Evaluation criteria

There are several ways to measure the error of an ANN. In this research the root mean squared error function (RMSE) was used as the error function to be minimized.

Number of training iterations

There are two schools of thought regarding the point at which training should be stopped. The first stresses the danger of getting trapped in a local minimum and the difficulty of reaching a global minimum. The researcher should only stop training until there is no improvement in the error function based on a reasonable number of randomly selected starting weights. The point at which the network does not improve is called convergence. The second view advocates a series of train test interruptions. Training is stopped after a predetermined number of iterations (i.e. epochs) and the network's ability to generalize on testing set is evaluated and training is resumed. Generalization is the idea that a model based on a sample of the data is suitable for forecasting the general population [Kaastra I. & Boyd M., 1996].

This second approach has been criticized because it can cause the error on the validation set to fall further before rising again or it could even fall asymptotically [Kaastra I. & Boyd M., 1996]. Thus, researches cannot know if further training could improve the generalization ability of the ANN [Kaastra I. & Boyd M., 1996].

The main objective of the convergence method is to reach a global minimum. Unfortunately, even if an ANN is trained for a sufficient number of epochs and using a reasonable number of randomly selected starting weights, there is no guarantee with a backpropagation network that a global minimum is reached since it may be trapped in a local minimum [Kaastra I. & Boyd M., 1996].

In order to maximise the probability of finding a global minimum, the maximum number of training epochs was set at 1000. Moreover, a validation stop rule was set at a maximum of 50 validation failures. These conditions ensure that the backpropagation algorithm will not run forever. The backpropagation algorithm stops if:

- 1000 training iterations are completed; or,
- there are 50 iterations with no reduction in the validation set error.

In the standard Levenberg-Marquardt backpropagation algorithm the validation stop is set at a maximum of six validation failures. In my work, the number of sequential iterations with no reduction in the validation set was set at 50 to increase the likelihood that the ANN would locate a global minimum. The maximum number of epochs and the maximum number of validation failures was selected on the basis of many studies. These studied assert that the maximum number of training iterations in a backpropagation network should be from 85 to 5000 and the maximum number of validation failures should be among 20 and 100. [Deboeck G. J., 1994; Kaastra I. & Boyd M., 1996, Klaussen K.L. & Uhrig J.W., 1992].

Learning rate and momentum

A backpropagation network is usually trained using gradient descent algorithm which follows the contours of the error surface by always moving down the steepest slope [Kaastra I. & Boyd M., 1996].

In this work, the ANN was trained using the Levenberg-Marquartdt algorithm which combines the gradient descent method and the Gauss-Newton method. This algorithm appears to be the fastest method for training moderate-sized feedforward neural networks (up to several hundred weights). Moreover, the ANN was trained using two other learning algorithms, the gradient descent and the gradient descent with momentum to study how changing the type of algorithm influence the predicted prices. The gradient descent had a learning rate of 0.1 and a momentum equal to 0.5.

4.7 Implementation

Implementation of the ANN as specified was a complex process. The main objective of the research, that is predicting the one-day ahead price of S&P 500, can be divided into three tests. The first one consists in studying how the type of training algorithm influence the output of the network. The second test investigates how the number of neurons influence the predicted prices. The third one examines how the distribution of the training dataset influence the same predictions. The configuration of the ANN was changed for each test, in relation to different objectives. For each configuration the ANN was trained one hundred times. Furthermore, the minimum, maximum and average accuracy and average RMSE of the ANNs were computed. The accuracy of the ANN is

defined as the quotient between its output and its target respectively the predicted price and the actual one. The following tables describe the configuration of the ANN for each test.

Parameter	**Setting**
Training algorithm	Levenberg-Marquardt (LV-MA) Steepest Descent (GD) Steepest Descent with Momentum (GDM)
Inputs	output of the variable selection model
Hidden layer size	10 neurons
Time horizon	data from 2010 to 2015
training period	data from 2010 to 2014
testing period	data from 2014
training data distribution	70% for training, 15% for validation, 15% for testing

Table 2 - ANN's configuration of the first test

Parameter	**Setting**
Training algorithm	Levenberg-Marquardt
Inputs	output of the variable selection model
Hidden layer size	10 , 50, 100 neurons
Time horizon	data from 2010 to 2015
training period	data from 2010 to 2014
testing period	data from 2014
training data distribution	70% for training, 15% for validation, 15% for testing

Table 3 - ANN's configuration of the second test

Parameter	**Setting**
Training algorithm	Levenberg-Marquardt
Inputs	output of the variable selection model
Hidden layer size	10 neurons
Time horizon	data from 2010 to 2015
training period	data from 2010 to 2014
testing period	data from 2014
training data distribution	34% for training, 33% for validation, 33% for testing 90% for training, 5% for validation, 5% for testing 5% for training, 90% for validation, 5% for testing 5% for training, 5% for validation, 90% for testing 4% for training, 48% for validation, 48% for testing 48% for training, 4% for validation, 48% for testing 48% for training, 48% for validation, 4% for testing

Table 4 - ANN's configuration of the third test

CHAPTER 5 RESULTS

"The key to make money in stocks is not to get scared out of them."

P. Lynch

The results can be divided in two main parts. In the first one the results of an ANN-based variable selection model are presented. In the second one the outputs of the variable selection model are used as initial input variables to develop an ANN based system for predicting the one-day ahead price of the S&P 500 index. Furthermore, each of these two parts is divided into three test. The tests of the first part examine the performance of three different input variables groups. The tests of the second part investigate how the accuracy of the ANN varies by changing different parameters, namely the training algorithm, the number of neurons in the hidden layer and the distribution of the training dataset. The results are mainly represented with tables and figures.

5.1 Variable selection model: first test

In the first test, a 3-layer backpropagation network with a Levenberg-Marquardt learning algorithm was trained one hundred times using all Gatley's final inputs to determine the forecasting power of these inputs. The results of this test can be summarized as follows:

R-squared				
Min.	**Max.**	**Mean**	**Standard deviation**	**Median**
0.393	0.968	0.671	0.241	0.586

Table 5 - Performance of the ANN in terms of the R-squared

RMSE				
Min.	**Max.**	**Mean**	**Standard deviation**	**Median**
11.48	85.30	43.54	26.57	46.85

Table 6 - Performance of the ANN in terms of the RMSE

The performance of the ANN (in terms of RMSE and R-squared) is relatively poor. Gatley's final inputs were selected twenty years ago in order to accurately predict S&P 500 index price. Nowadays, these variables have less influence to the target (i.e. S&P 500 index price) due to obvious economic changes over such a long time period.

5.2 Variable selection model: second test

In this second test, every step of Gatley's variable selection model was replicated to test its robustness. Moreover, each ANN was trained one hundred times and the average value of each input's relative contribution was computed. Remember that, from the previous chapter, Gatley's variable selection model can be divided in the following steps:

1. determining which of the initial inputs are making contributions to the neural network;
2. determining a suitable moving average for the volume input;
3. determining which inputs to lag and the number of days to lag;
4. determining which moving average period makes the largest contribution;

5. determining the contribution of several technical indicators;
6. selecting the definitive inputs.

First of all, to find out the relative contribution of each input to the target, a 3-layer backpropagation network with a Levenberg-Marquardt learning algorithm was trained 100 times using all Gatley's initial inputs. The relative contribution to the target was determined by summing the strength of connections (i.e. sum of the weights) between the input neurons and the neurons in the hidden layer. Inputs that had little effect on the output had weak connections to hidden layer and conversely, inputs that had a large effect on the output had strong connections to the hidden layer [Gately E., 1996]. In the following tests the value of total weights (i.e. weights' sum value) is normalized to lie in the range from -1 to 1.

Table 7 and figure 18 show the relative contribution of each initial input of Gatley's variable selection model. In all the following tables ranking is in reverse order. Inputs making the most contribution are at the bottom of the ranking.

Relative contribution	**Input**
-0.294	ADV.-DEC.
-0.233	XAU
-0.21	TRIN
-0.13	TRAN
-0.0846	S&P 500-H
-0.0782	VOL. INDU
0.0135	S&P 500-L
0.0211	UTIL
0.067	INDU
0.43	CRB
0.167	S&P 500-C
0.405	SPGCCLP

Table 7 - Relative contribution of each input to the target

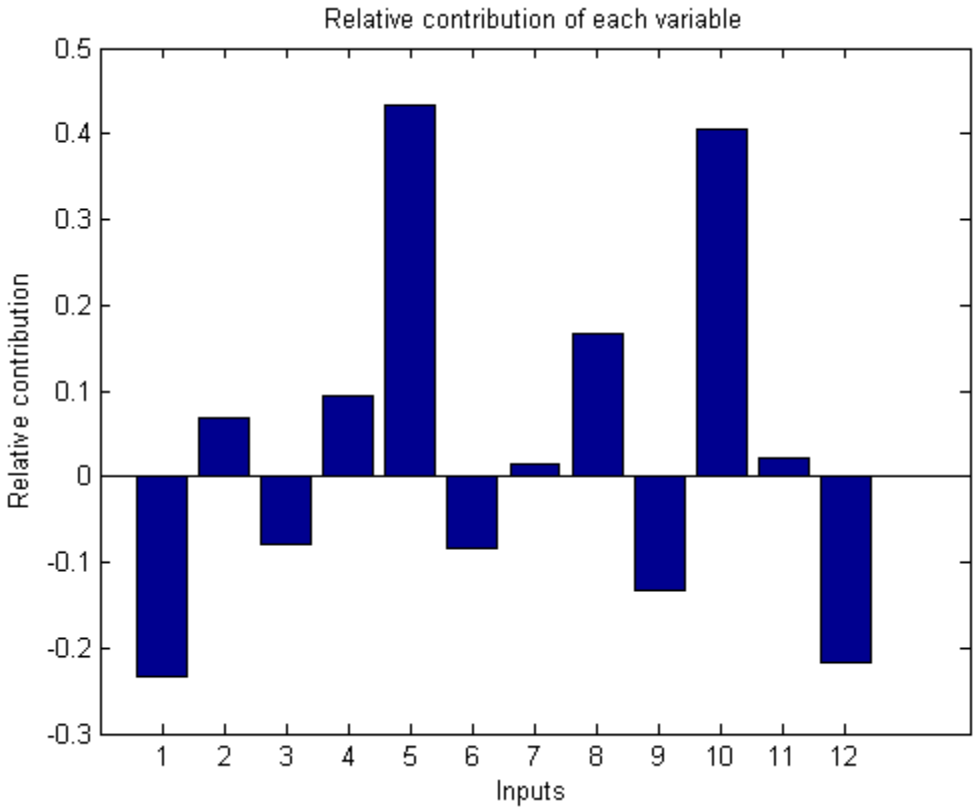

Fig. 18 - Relative contribution of each input to the target

ADV. - DECL., XAU, TRIN and TRAN indices had a poor performance, meaning that their relative contribution to the target is not relevant. VOL. INDU is another input that made a small contribution to the target. Usually raw volume data is not used with ANNs because it is very noisy and volatile. Moving averages of volume data are often used instead of raw volume data.

The next step will be to train the ANN again after deleting the input variables that made a poor contribution to the target (i.e. ADV.-DECL., XAU, TRIN, TRAN indices and Volume of INDU). Furthermore, a 3-day, 5-day and 10-day moving average (MA (3) VOL. INDU, MA (5) VOL. INDU, MA (10) VOL. INDU) of INDU's volume will be added, and the ANN trained again to find out which inputs contribute the most to the target. Table 8 and fig. 19 show the contribution of this set of input variables.

Relative contribution	Input
-0.5941	UTIL
-0.3797	S&P 500-C
-0.3255	MA (5) VOL. INDU
-0.0703	S&P 500-L
-0.0570	S&P 500-H
-0.0056	MA (10) VOL. INDU
0.1075	CRB
0.1164	XAU
0.1997	SPGCCLP
0.2049	TRAN
0.4911	MA (3) VOL. INDU

Table 8 - Relative contribution of each input to the target

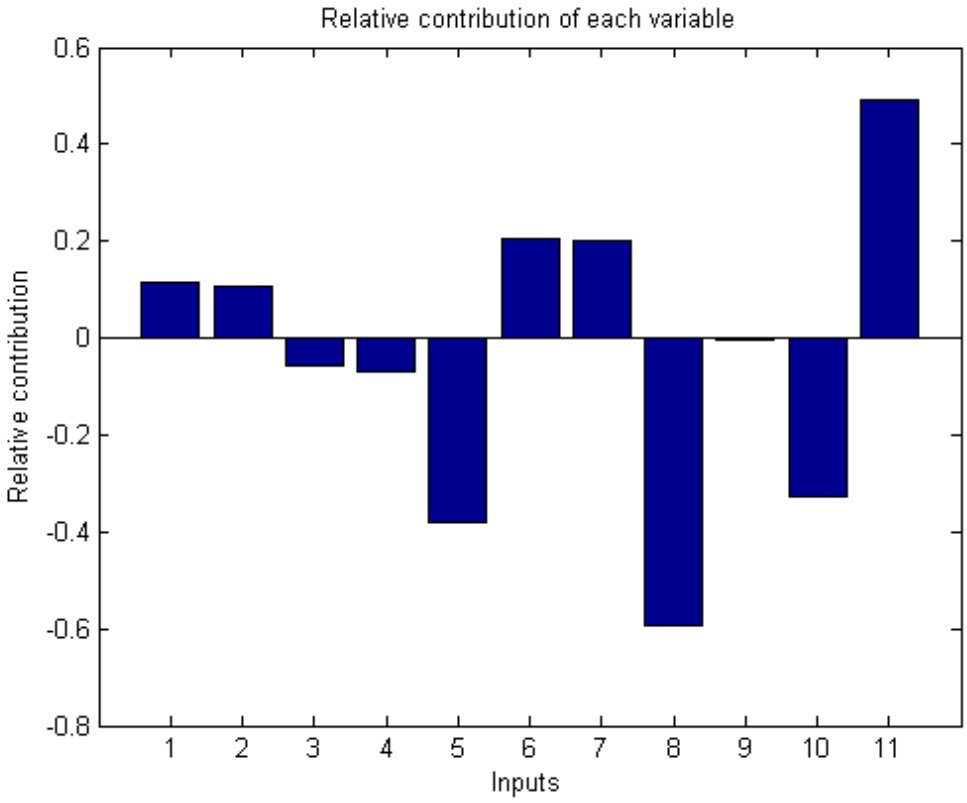

Fig. 19 - Relative contribution of each input to the target

Note that the 10-day moving average of INDU's volume (MA (10) VOL. INDU) and the 3-day moving average of INDU's volume (MA (3) VOL. INDU) contribute more than the raw volume data of the

previous ANN. In particular, the MA (3) VOL. INDU is the first most relevant input, far ahead of MA (10) VOL. INDU and MA (5) VOL. INDU. Since the MA (10) VOL. INDU is less relevant than the MA (3) VOL. INDU, there is no need to train another ANN with a 20-day or 30-day moving average. Therefore, it is possible to conclude that the 3-day moving average is the best choice.

Another step of the second test consists in lagging the inputs because inputs with a 5-day, a 10-day or even a 50-day lag often have a greater forecasting ability. Therefore, in an effort to find which inputs might have value when lagged, an ANN will be trained using just these lagged inputs:

- 10-day lag of S&P 500-C (close price);
- 10-day lag of S&P 500-H (high price);
- 10-day lag of S&P 500-L (low price);
- 10-day lag of Nasdaq PHLX Gold/Silver Sector Index;
- 10-day lag of Dow Jones Transportation Average Index;
- 10-day lag of Dow Jones Utilities Average Index;
- 10-day lag of 3-day moving average of volume of Dow Jones Industrial Average Index;
- 10-day lag of Thomson Reuters Core Commodities Index;
- 10-day lag of S&P GSCI Crude Oil Index.

Table 9 and fig. 20 show the contribution of this set of input variables.

Relative contribution	Input
-0.4888	10-day lag of S&P 500-C
-0.0786	10-day lag of TRAN
-0.0323	10-day lag of S&P 500-H
-0.0150	10-day lag of CRB
0.1663	10-day lag of SPGCCLP
0.1726	10-day lag of UTIL
0.1831	10-day lag of XAU
0.3363	10-day lag of MA (3) VOL. INDU

Table 9 - Relative contribution of each input to the target

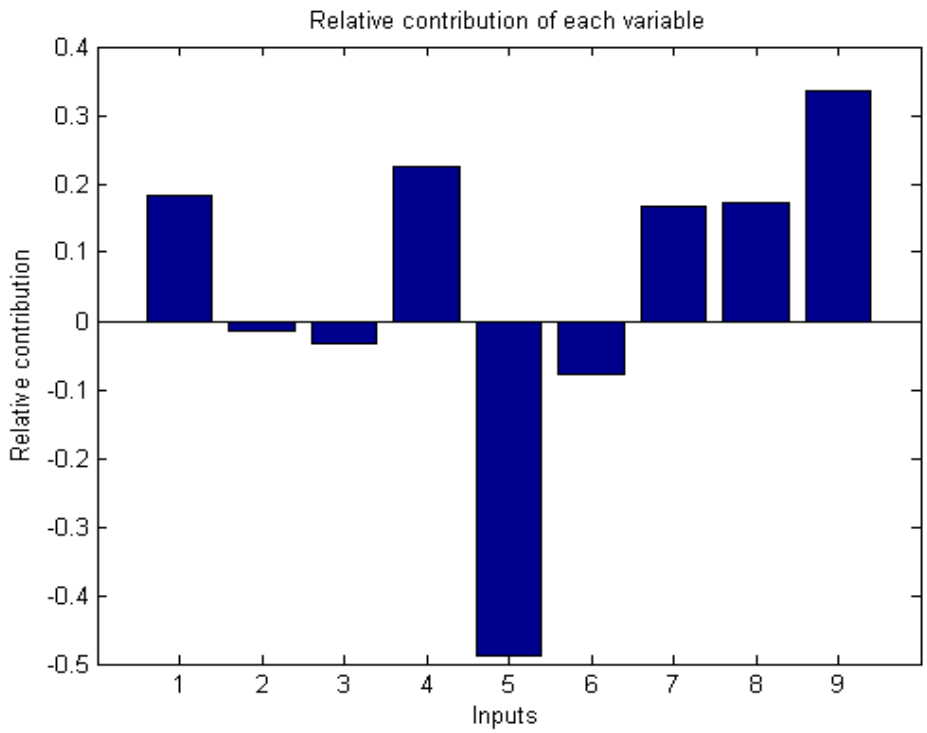

Fig. 20 - Relative contribution of each input to the target

Note that input 9, 10-day lag of 3-day moving average of volume of Dow Jones Industrial Average, makes the most relevant contribution, far ahead of the other inputs. To determine the best lag for the 3-day moving average of volume of Dow Jones Industrial Average input, a 5-day, 15-day and 20-day lag of the previous input were added to the input variables, and then the ANN was trained again. The inputs for the ANN were:

- S&P 500-C (close price);
- S&P 500-H (high price);
- S&P 500-L (low price);
- Nasdaq PHLX Gold/Silver Sector Index (XAU);
- Dow Jones Transportation Average Index (TRAN);
- Dow Jones Utilities Average Index (UTIL);
- 3-day moving average of volume of Dow Jones Industrial Average Index;
- Thomson Reuters Core Commodities Index (CRB);
- S&P GSCI Crude Oil Index (SPGCCLP);
- 5-day lag of 3-day moving average of volume of Dow Jones Industrial Average Index;
- 10-day lag of 3-day moving average of volume of Dow Jones Industrial Average Index;
- 15-day lag of 3-day moving average of volume of Dow Jones Industrial Average Index;

- 20-day lag of 3-day moving average of volume of Dow Jones Industrial Average Index.

Table 10 and fig. 21 show the contribution of this set of input variables. The 10-day lag of 3-day moving average of volume of Dow Jones Industrial Average Index was the best of the four lags tested. It was the second best input. Thus, the 10-day lag is the optimum lag for the 3-day moving average of volume of Dow Jones Industrial Average Index.

Relative contribution	Input
-0.4055	XAU
-0.3850	S&P 500-L
-0.3564	5-day lag of MA (3) VOL. INDU
-0.3026	CRB
-0.0891	MA (3) VOL. INDU
-0.0476	20-day lag of MA (3) VOL. INDU
-0.0448	15-day lag of MA (3) VOL. INDU
0.0028	UTIL
0.0915	S&P 500-C
0.1474	S&P 500-H
0.1910	TRAN
0.2278	10-day lag of MA (3) VOL. INDU
0.3875	SPGCCLP

Table 10 - Relative contribution of each input to the target

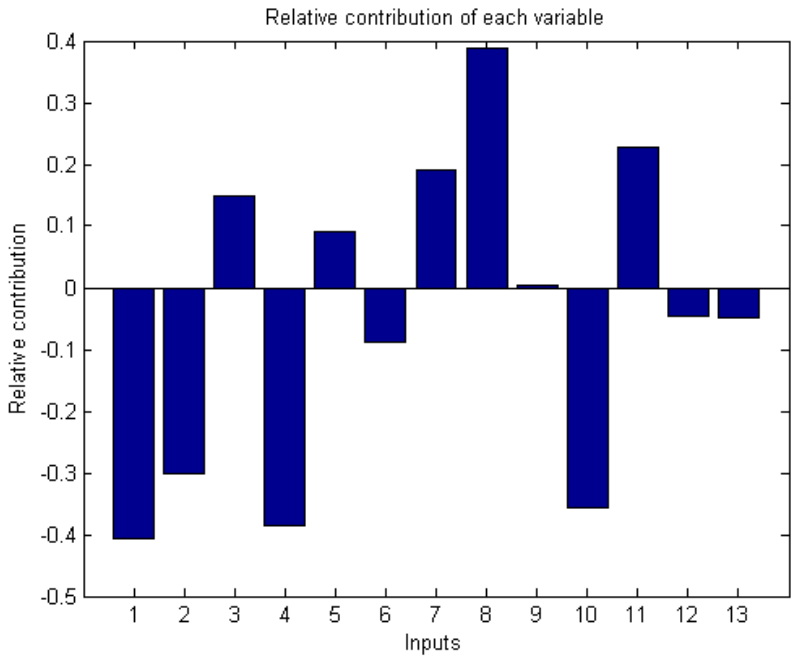

Fig. 21 - Relative contribution of each input to the target

After that, XAU index is lagged by 5,10,15,20 days to determine its optimum lag. The inputs for the ANN were:

- 5-day lag of Nasdaq PHLX Gold/Silver Sector Index (XAU);
- 10-day lag of Nasdaq PHLX Gold/Silver Sector Index (XAU);
- 15-day lag of Nasdaq PHLX Gold/Silver Sector Index (XAU);
- 20-day lag of Nasdaq PHLX Gold/Silver Sector Index (XAU);
- Nasdaq PHLX Gold/Silver Sector Index (XAU);
- S&P 500-C (close price);
- S&P 500-H (high price);
- S&P 500-L (low price);
- Dow Jones Transportation Average Index (TRAN);
- Dow Jones Utilities Average Index (UTIL);
- 3-day moving average of volume of Dow Jones Industrial Average Index;
- Thomson Reuters Core Commodities Index (CRB);
- S&P GSCI Crude Oil Index (SPGCCLP);

Table 11 and fig. 22 show that inputs 10,11,12,13, which corresponds to the 5-day, 10-day, 15-day and 20-day lags of XAU index, do not reach a clearly defined peak. This means that their contributions to the target are not relevant.

Relative contribution	Input
-0.4904	15-day lag of XAU
-0.3808	S&P 500-H
-0.2444	10-day lag of XAU
-0.2122	20-day lag of XAU
-0.1409	5-day lag of XAU
-0.1061	S&P 500-C
-0.1026	UTIL
-0.1005	CRB
-0.0780	SPGCCLP
0.1430	TRAN
0.1461	XAU
0.2075	S&P 500-L
0.3481	MA (3) VOL. INDU

Table 11 - Relative contribution of each input to the target

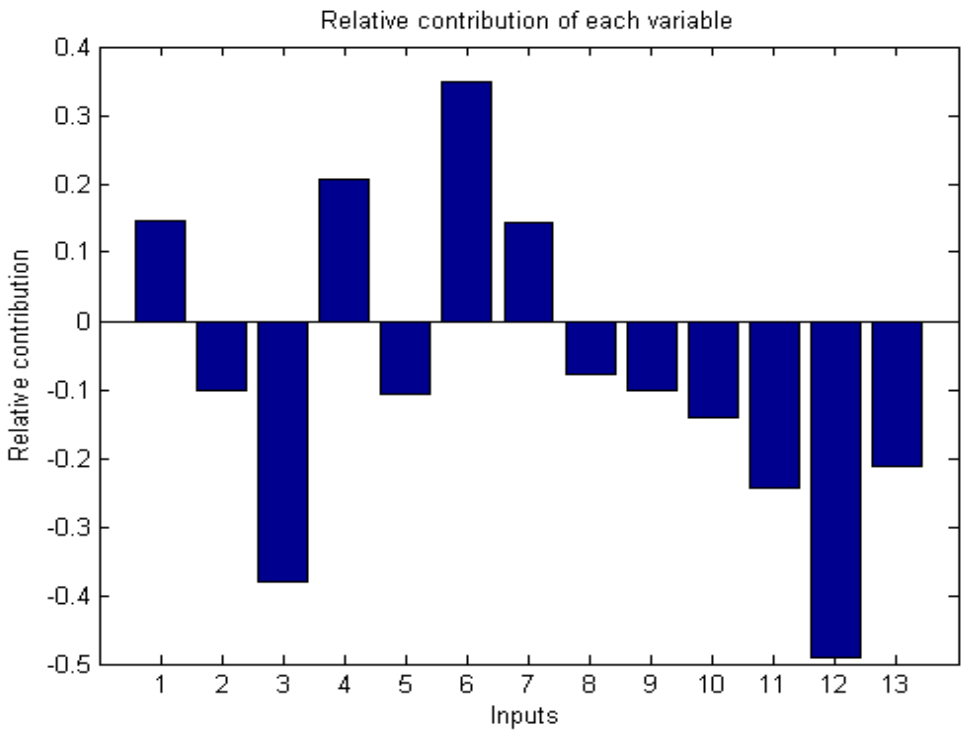

Fig. 22 - Relative contribution of each input to the target

At this point, it is not possible to know whether a 20-day lag or a 100 day-lag is the optimum one. Thus, after adding a 10-day, 20-day, 30-day, 40-day lags of the XAU index, the ANN is trained again. Running into problems like this is normal in developing an ANN. Only by trying and testing it is possible to find out what works and what does not.

Relative contribution	**Input**
-0.3431	S&P 500-L
-0.2724	40-day lag of XAU
-0.1809	MA (3) VOL. INDU
-0.1760	UTIL
-0.1160	20-day lag of XAU
-0.1039	10-day lag of XAU
-0.0390	30-day lag of XAU
0.0601	SPGCCLP
0.0647	S&P 500-C
0.0793	XAU
0.1708	S&P 500-H
0.2468	CRB
0.2815	TRAN

Table 12 - Relative contribution of each input to the target

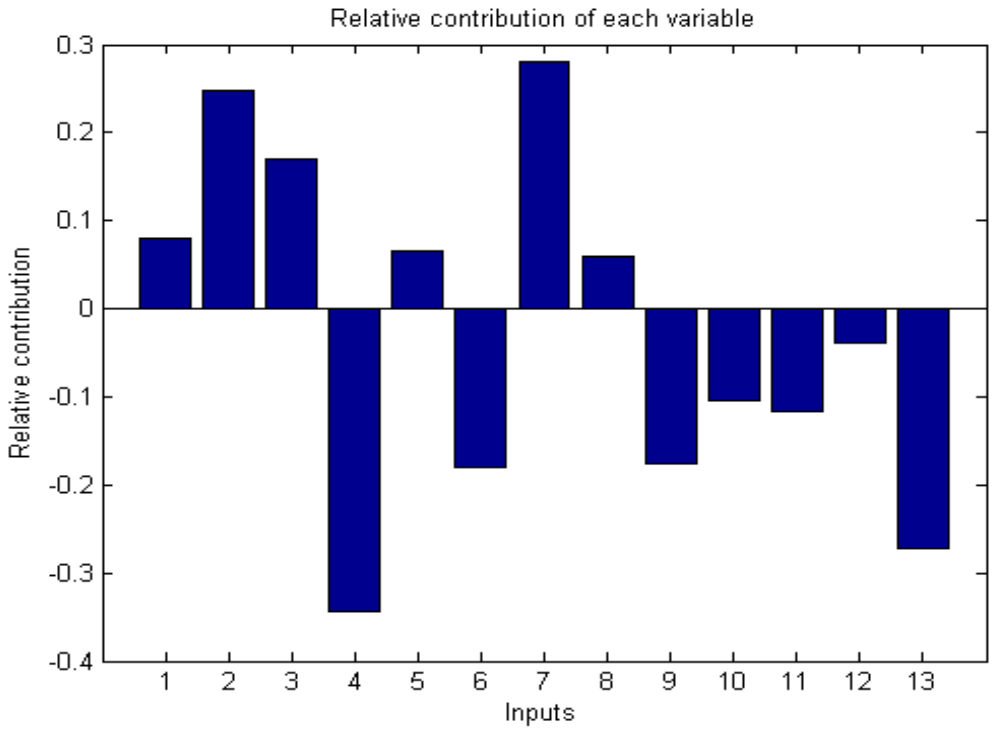

Fig. 23 - Relative contribution of each input to the target

Table 12 and fig. 23 show that inputs 10, 11, 12, 13, which corresponds to the 10-day, 20-day, 30-day and 40-day lags of XAU index, do not reach a clearly defined peak. A solution to this problem would be to train the ANN four times, where each test uses only one of the four lagged inputs, then compare the root mean squared error (RMSE) of the four tests.

n° of days lagged	**RMSE**
0	118.74
10	64.36
20	62.71
30	44.81
40	80.05

Table 13 - RMSE of the four ANNs

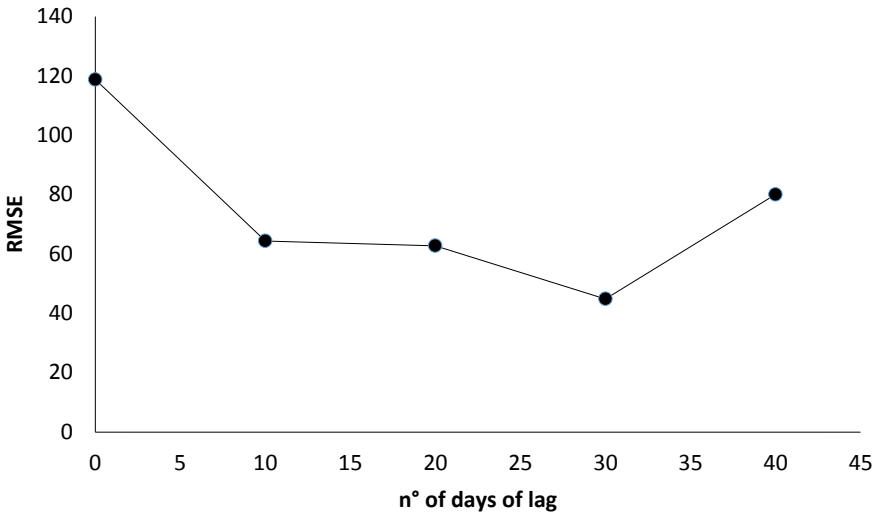

Fig. 24 - RMSE of the four ANNs

Table 13 and Fig. 24 show that the minimum error was reached with a 30-day lag. Thus, the optimum lag of XAU index is 30 days.

The next step consists in finding which moving average period makes the largest contribution to the target. Therefore, an ANN was trained using a 10-day moving average of all the basic inputs (i.e. TRAN, CRB, UTIL, XAU, SPGCCLP, S&P 500-H, S&P 500-L, S&P 500-C, VOL. INDU) except volume of Dow Jones Industrial Average index, then a 20-day moving average of the inputs, and finally a 30-day moving average. The inputs to the network included only the moving average of original inputs. Table 14 and fig. 25 show the results of training the ANN using a 10-day moving average of each input. The 10-day moving average of CRB, TRAN, S&P 500-H, UTIL made a major contribution to the solution of the ANN (i.e. the target).

Relative contribution	Input
-0.6368	MA(10) of S&P 500-L
-0.1712	MA(10) of S&P 500-C
-0.1322	MA(10) of XAU
-0.0634	MA(10) of SPGCCLP
0.1151	MA(10) of UTIL
0.2485	MA(10) of S&P 500-H
0.4928	MA(10) of TRAN
0.6421	MA(10) of CRB

Table 14 - Relative contribution of each input to the target

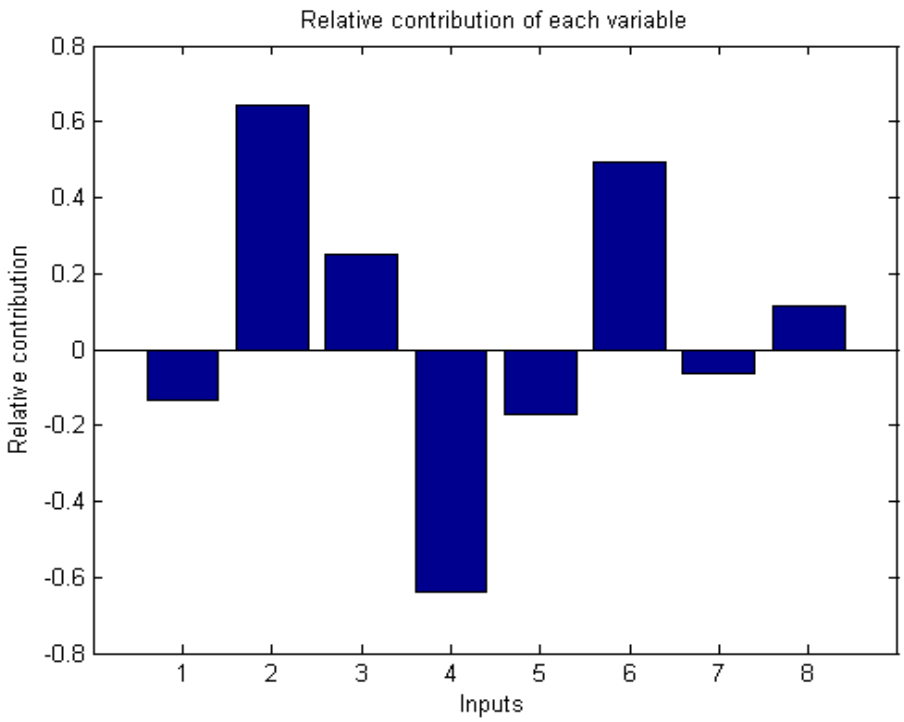

Fig. 25 - Relative contribution of each input to the target

Table 15 and fig. 26 show the relative contributions of the 20-day moving average of the inputs. The 20-day moving average of SPGCCLP, S&P 500-L, CRB, S&P 500-C made a major contribution to the solution of the ANN.

Relative contribution	Input
-0.5031	MA(20) of XAU
-0.4527	MA(20) of UTIL
-0.1148	MA(20) of TRAN
-0.0063	MA(20) of S&P 500-H
0.1241	MA(20) of S&P 500-C
0.1427	MA(20) of CRB
0.1569	MA(20) of S&P 500-L
0.2502	MA(20) of SPGCCLP

Table 15 - Relative contribution of each input to the target

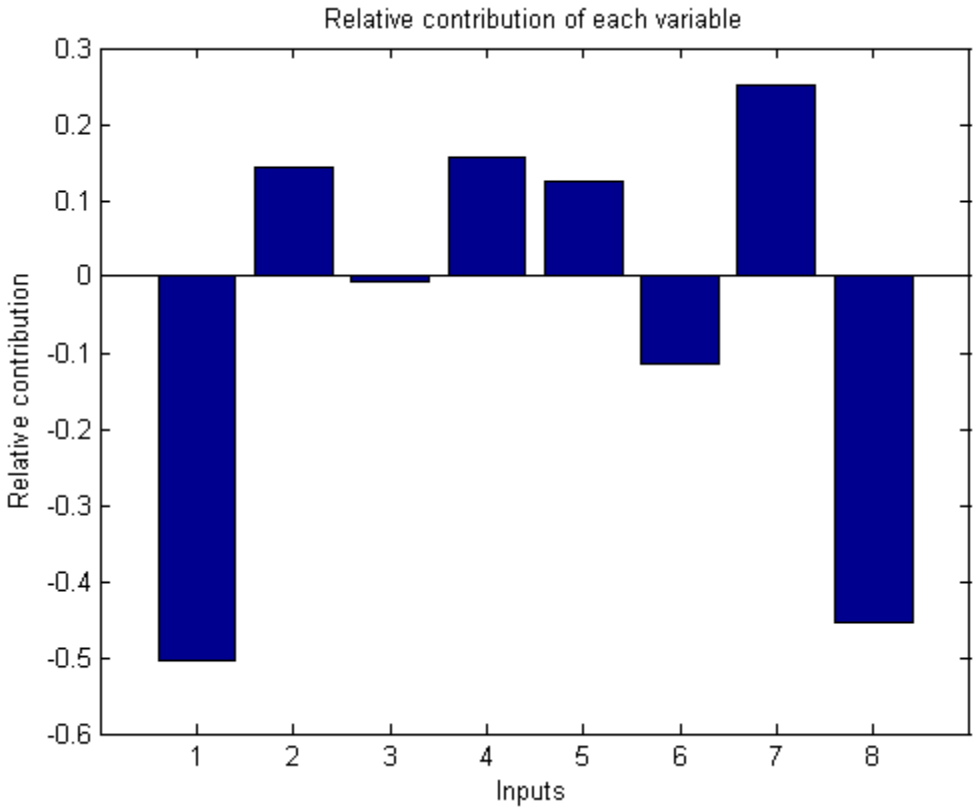

Fig. 26 - Relative contribution of each input to the target

Table 16 and fig. 27 show the relative contributions of the 30-day moving average of the inputs. The 30-day moving average of TRAN, CRB, S&P 500-C and SPGCCLP made a major contribution to the solution of the ANN. In these tests on the moving average, the input that made the most relevant contribution is the 10-day moving average of CRB.

Relative contribution	Input
-0.4969	MA(30) of S&P 500-L
-0.4969	MA(30) of XAU
-0.1294	MA(30) of S&P 500-H
0.0171	MA(30) of UTIL
0.0744	MA(30) of SPGCCLP
0.1375	MA(30) of S&P 500-C
0.1630	MA(30) of CRB
0.5425	MA(30) of TRAN

Table 16 - Relative contribution of each input to the target

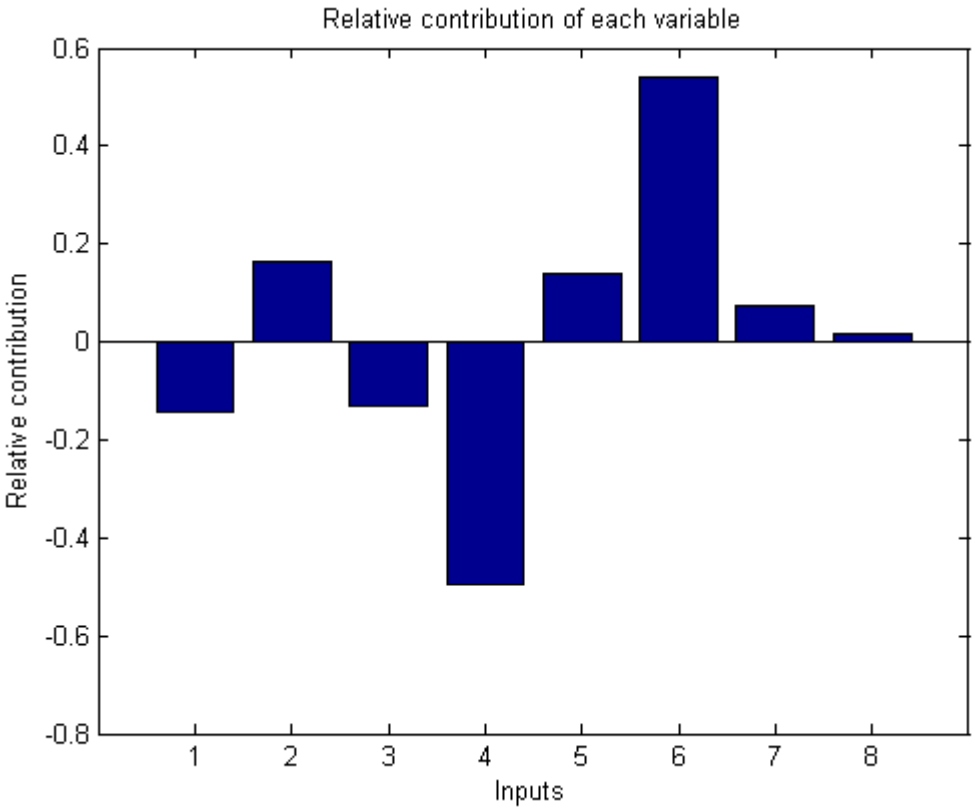

Fig. 27 - Relative contribution of each input to the target

Determining the contribution of several technical indicators is another important step of this variable selection model. The number of technical indicators that can added to the input variables of the ANN is almost limitless. In order to illustrate the use of technical indicators as inputs to an ANN, the contribution of several technical indicators is measured [Gately E., 1996]. The first technical indicator to be tested is a 12-day RSI as defined by Wilder[31]. Furthermore, the 12-day moving average of the Wilder RSI, that often is a signal of directional changes, will be tested. The inputs of this test are all the basic inputs (i.e. TRAN, CRB, UTIL, XAU, SPGCCLP, S&P 500-H, S&P 500-L, S&P 500-C, VOL. INDU) except volume of Dow Jones Industrial Average index and plus a 12-day Wilder RSI of S&P 500-C and its 12-day moving average.

[31] Developed J. Welles Wilder, the Relative Strength Index (RSI) is a momentum oscillator that measures the speed and the rate-of-change of a security's price. As the price of a security rises, the value of this indicator increases. Once the price of a security begins to decrease, the value of this momentum oscillator will do the same.

Table 17 and fig. 28 show the contributions of these two indicators. They performed poorly, probably because they contain a very similar information and so they confused the ANN. Thus, the latter did not know which of them to use.

Relative contribution	Input
-0.5612	CRB
-0.3965	UTIL
-0.159969818	WilderRSI(12) of S&P 500-C
-0.1197	TRAN
-0.0655	SPGCCLP
-0.0379	S&P 500-C
-0.008599313	MA (12) of WilderRSI(12) of S&P 500-C
0.0107	XAU
0.2287	S&P 500-L
0.4199	S&P 500-H

Table 17 - Relative contribution of each input to the target

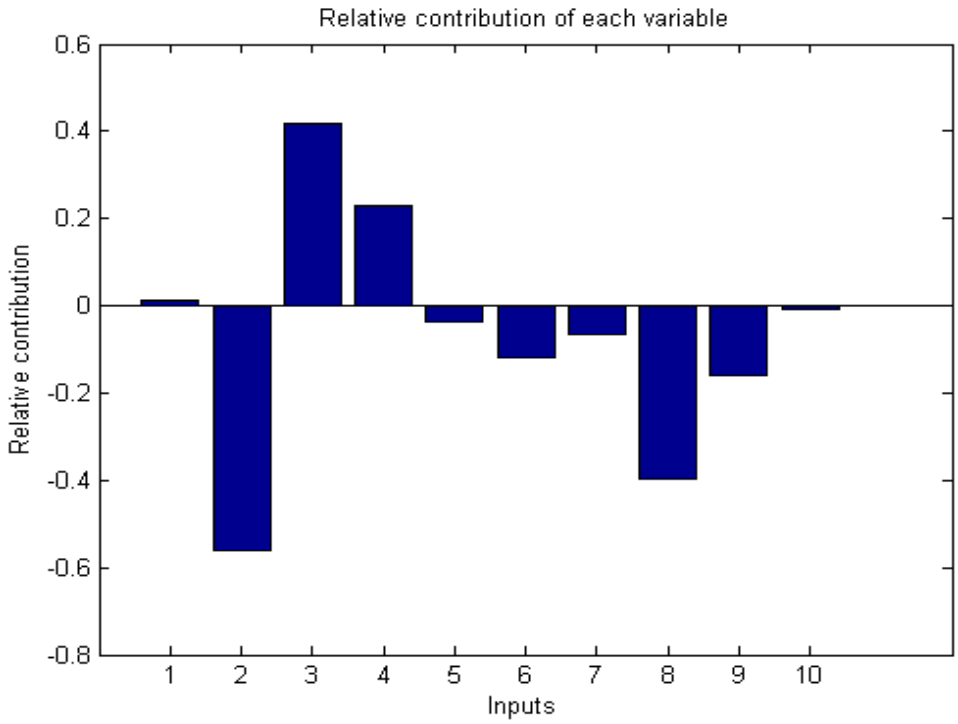

Fig. 28 - Relative contribution of each input to the target

A solution to the previous problem is to test the two indicators individually. The test using only the 12-day RSI is presented in Table 18 and fig. 29. The RSI performed much better than in the previous test and seems to help the other inputs to make significant contribution. This can be seen by comparing Fig. 23 to Fig. 22.

Relative contribution	Input
-0.7137	S&P 500-C
-0.5899	XAU
-0.281265225	SPGCCLP
-0.1102	TRAN
0.0829	S&P 500-L
0.270675334	S&P 500-H
0.3279	UTIL
0.5560	CRB
0.6002	WilderRSI(12) of S&P 500-C

Table 18. Relative contribution of each input to the target

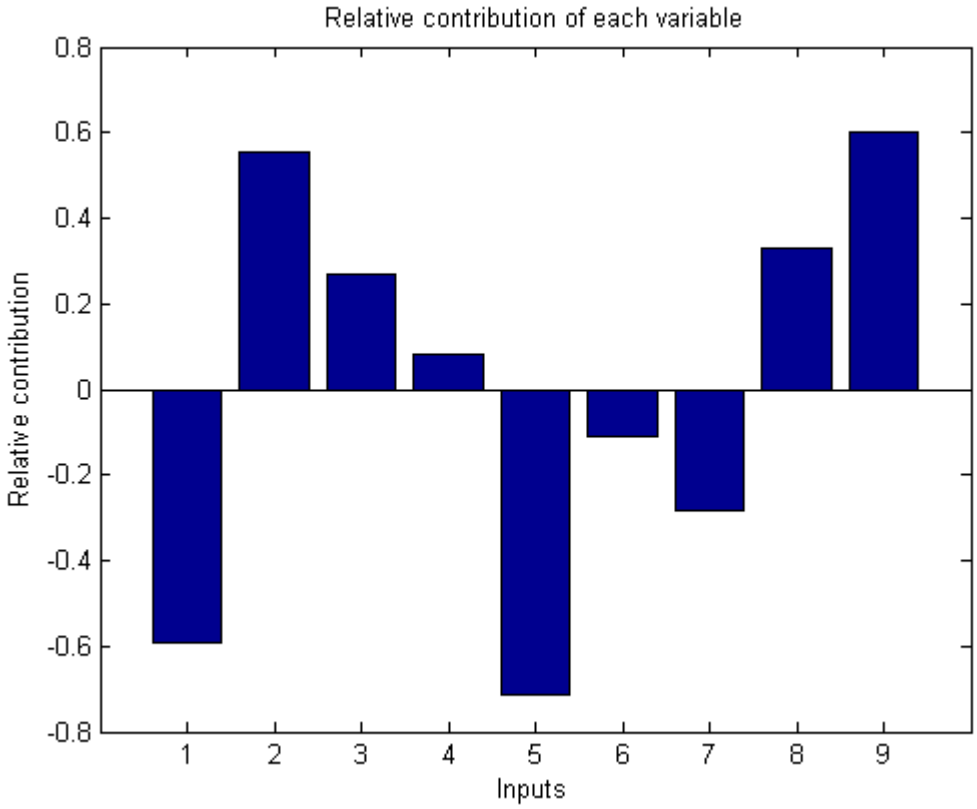

Fig. 29 - Relative contribution of each input to the target

Table 19 and fig. 30 present the results of the test using the 12-day moving average of the 12-day RSI. The results are much better than in the ANN which used both the RSI and its moving average but not as good as the previuos ANN which used just the RSI. Therefore, the latter can be considered as the best choice.

Relative contribution	Input
-0.9772	UTIL
-0.1716	CRB
-0.151735113	SPGCCLP
0.0172	S&P 500-L
0.1656	XAU
0.186524182	S&P 500-H
0.2538	MA (12) of WilderRSI(12) of S&P 500-C
0.2550	TRAN
0.3110	S&P 500-C

Table 19 - Relative contribution of each input to the target

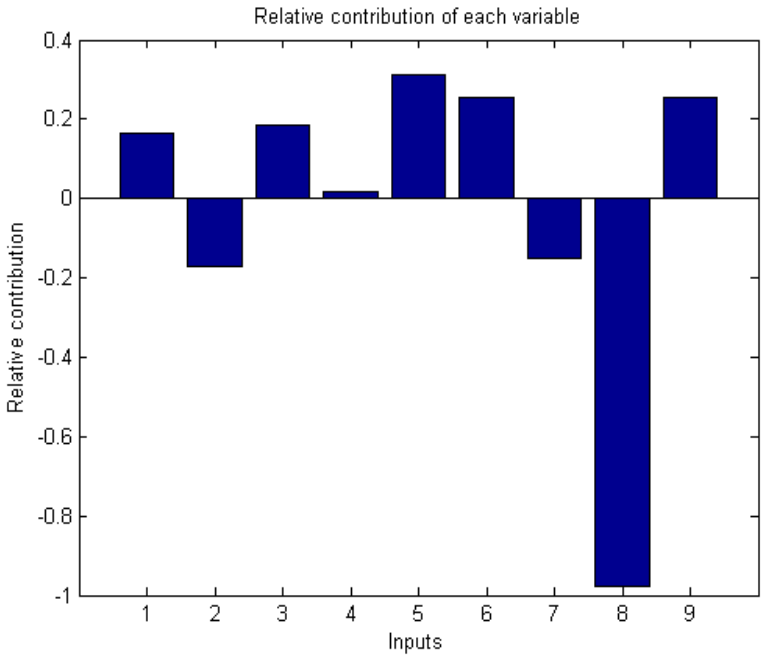

Fig. 30 - Relative contribution of each input to the target

The second technical indicator to be tested is the Bollinger Bands. This indicator is composed by two lines, upper band and lower band, that are drawn above and below the data. The bands are set at 2 standard deviation of a 20-day moving average above and below the data. The inputs of this test

are all the basic inputs except volume of Dow Jones Industrial Average index and plus the data for the Upper and Lower Bollinger Bands. These bands were placed two standard deviations from a 20-day moving average of the S&P 500-C. Table 20 and fig. 31 show the performance of this indicator. Note that the Upper Bollinger Band made a more relevant contribution than the XAU and TRAN indices.

Relative contribution	Input
-0.5604	S&P 500-L
-0.4098	BBandLower(20,2) of S&P 500-C
-0.391628319	S&P 500-H
-0.2894	CRB
-0.2445	UTIL
-0.0335	S&P 500-C
0.1035	TRAN
0.2513	XAU
0.3917	BBandUpper(20,2) of S&P 500-C
0.404867726	SPGCCLP

Table 20 - Relative contribution of each input to the target

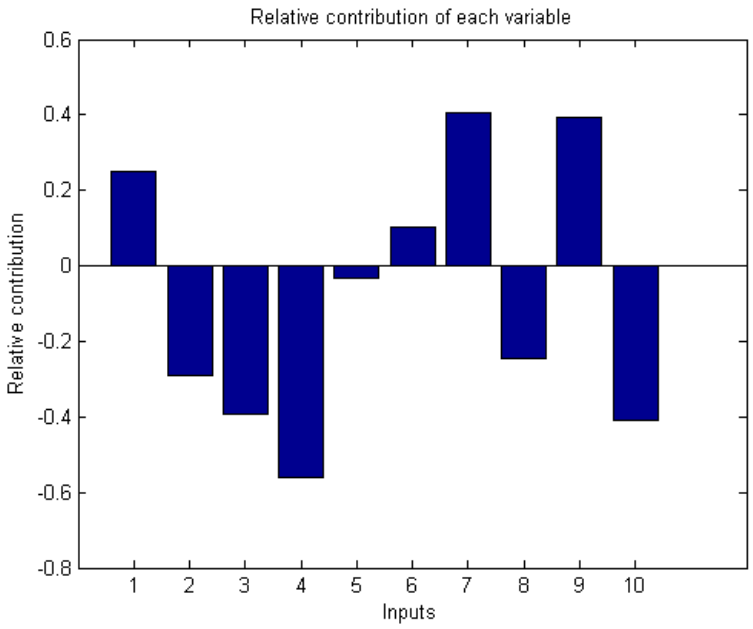

Fig. 31 - Relative contribution of each input to the target

After that, a 12-day momentum indicator is tested. This technical indicator measures the change in price over the last 12 days, that is, today's price less the price of 12 days ago. Table 21 and fig. 32 show the results of this test. Note that momentum indicator makes a fairly large contribution. It is the fourth of the list ahead of XAU and UTIL indices.

Relative contribution	Input
-0.7100	S&P 500-C
-0.4800	CRB
-0.381929593	S&P 500-H
0.0064	XAU
0.0304	UTIL
0.1694	Momentum(12) of S&P 500-C
0.281714545	SPGCCLP
0.5692	S&P 500-L
0.6788	TRAN

Table 21 - Relative contribution of each input to the target

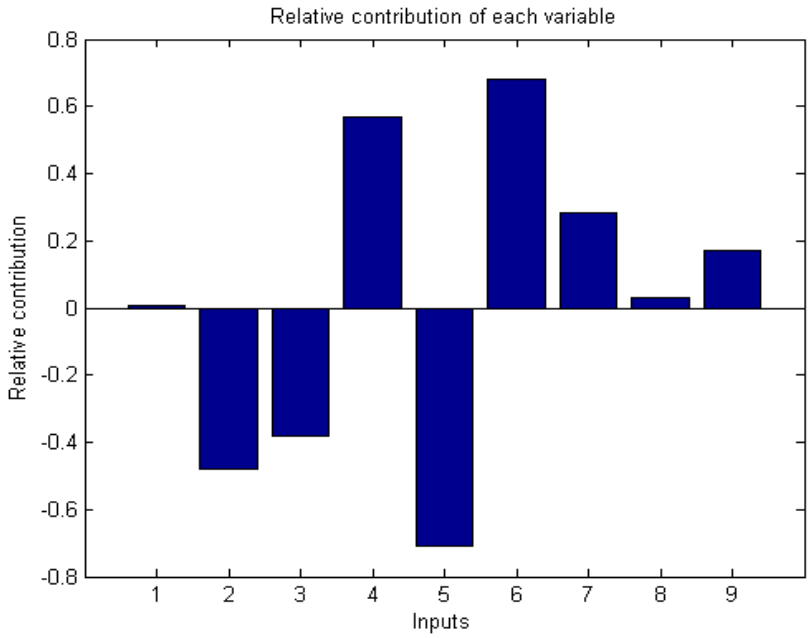

Fig. 32 - Relative contribution of each input to the target

Testing has created a large number of possible indicators from moving averages and lagged inputs to technical indicators. The final step of this variable selection model is selecting the definitive inputs that will help the solution of the problem, that is predicting the one-day ahead price of the S&P 500 index. The basic inputs will be the following:

- S&P 500 High price;
- S&P 500 Low price;
- S&P 500 Close price;
- UTIL;
- SPGCCLP;
- TRAN;
- MA (3) VOL. INDU.

As it is possible to see from table 22, these inputs made a relevant contribution in one of the previous tests and for this reason they were selected.

Input	**Relative contribution**
XAU	0.2056
S&P 500- L	0.2075
S&P 500- C	0.3110
UTIL	0.3279
SPGCCLP	0.4048
S&P 500-H	0.4199
MA (3) VOL. INDU	0.4911
TRAN	0.6788

Table 22 - Highest contribution of the basic inputs through all the tests.

Furthermore, WilderRSI(12) of S&P 500-C and MA(10) of CRB index were added to the definitive inputs because they made the highest contribution when tested with the basic inputs. CRB index was not included in the basic inputs because its 10-day moving average made a greater contribution

to the target, so the latter was chose as an input instead of CRB index. Furthermore, MA(10) of CRB index and CRB index contain the same information and they can confuse the ANN.

These inputs were tested as a group using a 3-layer backpropagation network with a Levenberg-Marquardt learning algorithm to determine their contribution. The results of this test are shown in Table 23 and fig. 33.

Relative contribution	Input
-0.2810	MA(3) VOL. INDU
-0.075057486	SPGCCLP
-0.015903642	S&P 500-H
-0.0121	XAU
0.1470	TRAN
0.1936	UTIL
0.2405	S&P 500- C
0.3216	MA(10) CRB
0.4904	WilderRSI(12) of S&P 500-C
0.5207	S&P 500- L

Table 23 - Relative contribution of each input to the target

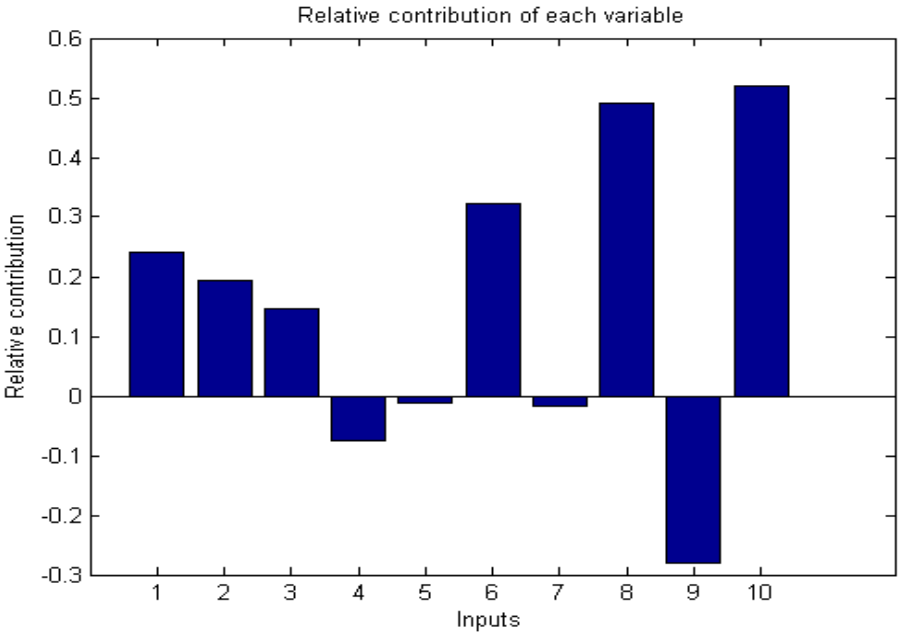

Fig. 33 - Relative contribution of each input to the target

In order to determine the forecasting power of the definitive inputs, a 3-layer backpropagation network using these inputs was trained one hundred times.

R-squared				
Min.	**Max.**	**Mean**	**Standard deviation**	**Median**
0.713	0.984	0.885	0.073	0.890

Table 24 - Performance of the ANN in terms of the R-squared

RMSE				
Min.	**Max.**	**Mean**	**Standard deviation**	**Median**
8.03	45.00	20.60	8.63	20.29

Table 25 - Performance of the ANN in terms of the RMSE

Table 24 and table 25 show the performance of the ANN which is relatively good. Thus, it is possible to conclude that even after 20 years and considering all the huge changes in financial markets, Gatley's ANN-based variable selection model keeps its robustness. Therefore, it is still able to provide a set of input variables to predict the price of the S&P 500 index.

5.3 Variable selection model: third test

Variable selection is the process of reducing the number of inputs used to train an ANN. Often, models developed with fewer inputs are more effective than those constructed with many inputs [Deboeck G. J., 1994].

In the previous test the MA(3) VOL. INDU, SPGCCLP, S&P 500-H and XAU had a poor performance. Their relative contribution to the target was very low.

The main objective of this last test is to determine whether it is possible to find an ANN that has a greater forecasting power, compared with the one of the Benchmark ANN, by deleting the input variables with a low-contribution value. Therefore, a 3-layer backpropagation network with a Levenberg-Marquardt learning algorithm was trained one hundred times using all the final inputs obtained from the second test except the MA(3) VOL. INDU, SPGCCLP, S&P 500-H and XAU. This ANN was called Benchmark ANN. Moreover, several other ANNs with the same architecture and learning algorithm were trained one hundred times using:

- all final inputs of the second test except MA(3) VOL. INDU, SPGCCLP, S&P 500-H and XAU (1° ANN);
- all final inputs of the second test except MA(3) VOL. INDU (2° ANN);

- all final inputs of the second test except SPGCCLP (3° ANN) ;
- all final inputs of the second test except S&P 500-H (4° ANN);
- all final inputs of the second test except XAU (5° ANN);
- all final inputs of the second test except MA(3) VOL. INDU and SPGCCLP (6° ANN);
- all final inputs of the second test except MA(3) VOL. INDU and S&P 500-H (7° ANN);
- all final inputs of the second test except MA(3) VOL. INDU and XAU (8° ANN);
- all final inputs of the second test except SPGCCLP and S&P 500-H (9° ANN).
- all final inputs of the second test except SPGCCLP and XAU (10° ANN).
- all final inputs of the second test except S&P 500-H and XAU (11° ANN);
- all final inputs of the second test except MA(3) VOL. INDU, SPGCCLP and S&P 500-H (12° ANN);
- all final inputs of the second test except MA(3) VOL. INDU, S&P 500-H and XAU (13° ANN);
- all final inputs of the second test except SPGCCLP, S&P 500-H and XAU (14° ANN);
- all final inputs of the second test except SPGCCLP, MA(3) VOL. INDU and XAU (15° ANN);

In this test, the performance of the alternative ANNs is compared to the one of the benchmark ANN. Remember that, goodness-of-fit criteria chose for all the tests of the variable selection model are the root mean squared error (RMSE) and the R-squared. Tables 26 and 27 show the results obtained by training one hundred times each ANN of the variable selection model. For all the different ANNs, minimum and maximum values of the R-squared do not differ significantly. Moreover, the RMSE lies below 50$ and the R-squared is concentrated in the range from 0.75 to 0.98 as shown respectively in figures 34 and 35. Therefore, all the ANNs have a relatively good performance. The ANN with the best performance is the 5° ANN, its mean and standard deviation of the R-squared reached the highest value among all the tests and its mean and standard deviation of the RMSE reached the lowest one. Furthermore, it shows the best values of “maximum R-squared” and “minimum RMSE”, respectively the highest and lowest one. Finally, its range is relatively small, meaning that the minimum and maximum value of the predicted price do not differ excessively. The 5° ANN shows a better performance that the Benchmark network, which contains the input variables provided by Gatley’s variable selection model. Thus, an ANN with a better generalization than the Benchmark network is found.

R-squared					
ANN	**Min.**	**Max.**	**Mean**	**Standard deviation**	**Median**
Bench.	0.839	0.984	0.936	0.042	0.948
1° ANN	0.776	0.984	0.909	0.055	0.917
2° ANN	0.787	0.983	0.909	0.061	0.925
3° ANN	0.787	0.984	0.921	0.049	0.933
4° ANN	0.839	0.984	0.931	0.041	0.940
5° ANN	0.896	0.984	0.949	0.026	0.956
6° ANN	0.839	0.984	0.932	0.040	0.944
7° ANN	0.786	0.983	0.912	0.058	0.925
8° ANN	0.777	0.984	0.914	0.062	0.938
9° ANN	0.781	0.983	0.915	0.055	0.924
10° ANN	0.789	0.985	0.920	0.055	0.936
11° ANN	0.786	0.983	0.916	0.061	0.938
12° ANN	0.776	0.984	0.900	0.063	0.921
13° ANN	0.791	0.984	0.912	0.053	0.923
14° ANN	0.778	0.984	0.913	0.055	0.923
15° ANN	0.794	0.981	0.926	0.045	0.935

Table 26 - R-squared of the ANNs tested

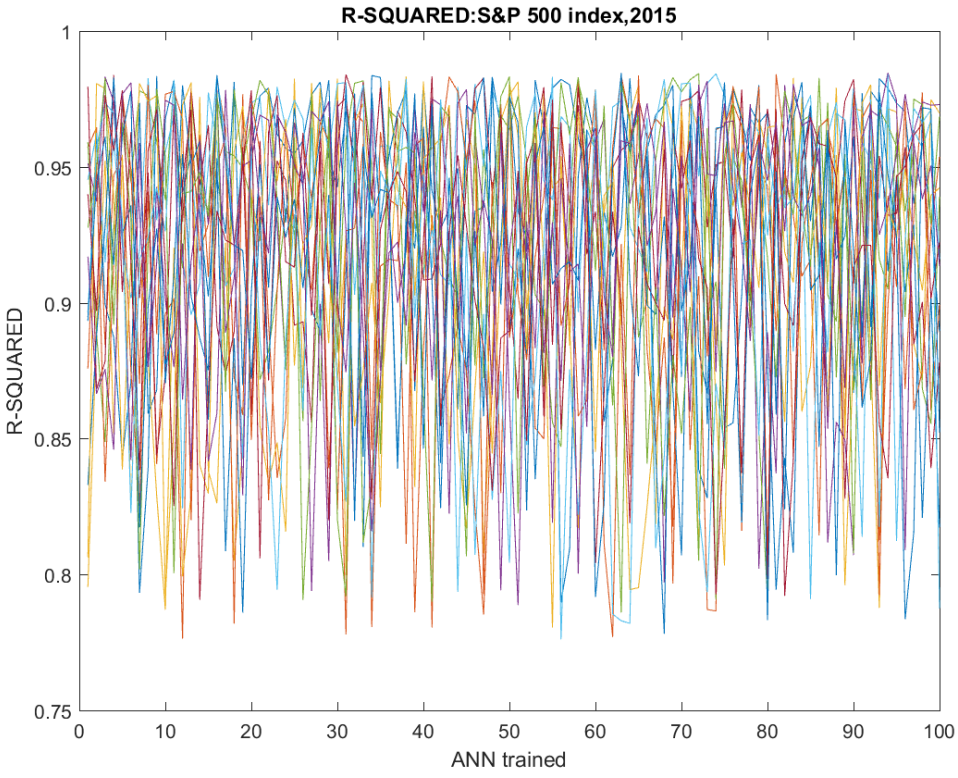

Fig. 34 - R-squared of the ANNs tested

RMSE					
ANN	**Min.**	**Max.**	**Mean**	**Standard deviation**	**Median**
Bench.	8.03	30.36	15.41	5.82	14.53
1° ANN	8.77	33.49	18.29	6.29	17.43
2° ANN	8.72	46.70	17.98	8.07	15.88
3° ANN	7.98	42.86	17.68	6.85	16.50
4° ANN	7.96	30.06	15.63	5.07	14.85
5° ANN	7.41	27.59	13.99	4.38	13.41
6° ANN	7.80	30.32	16.01	5.38	14.68
7° ANN	8.00	39.70	17.38	7.24	15.79
8° ANN	8.12	48.50	18.49	9.21	15.19
9° ANN	8.00	35.43	17.61	6.73	16.60
10° ANN	8.19	39.79	16.98	7.03	14.90
11° ANN	8.33	42.79	17.79	7.94	15.31
12° ANN	7.88	43.28	19.25	7.78	18.11
13° ANN	8.28	38.04	17.64	6.44	16.99
14° ANN	8.35	47.52	18.44	8.00	16.58
15° ANN	8.95	33.05	16.60	5.85	15.61

Table 27 - RMSE of the ANNs tested

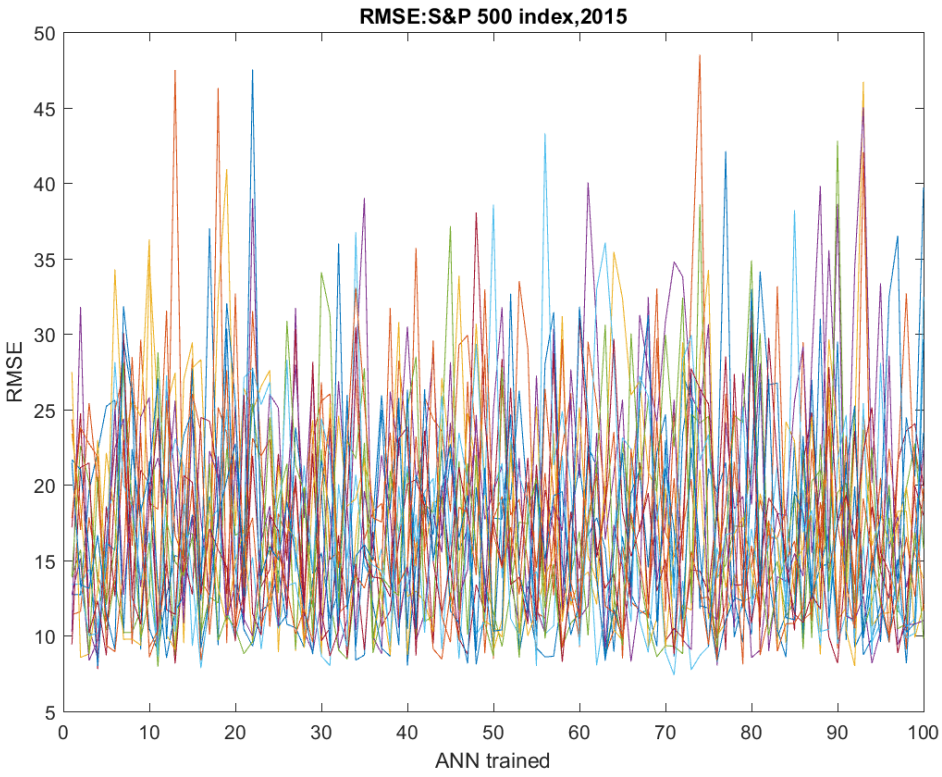

Table 35 - RMSE of the ANNs tested

Considering the previous results, the input variables of the 5° NN will be used as input variables for the price prediction model.

5.4 Price prediction model: first test

The first test consists in investigating how the type of training algorithm used influence the output of the ANN. In this test, the hidden layer size was set at ten neurons and the distribution of training dataset was divided into 70% for training, 15% for validation and 15% for testing. All the other parameters were kept on their default value. In order to find a good measure of generalization the ANN was trained one hundred times. In this research, figures and tables are based on the average values of the one hundred ANNs that were trained, meaning that they represent the average network predictions for each configuration. Furthermore, in all the results the accuracy and RMSE are indicated. The use of the RMSE rather than the MSE has the advantage that it uses the same dimensions as the stock index price. Tables 28-29 and figures 36-44 show the results of the whole test. This test is divided into three steps according to the training algorithm that was used.

	Accuracy				
Training algorithm	**Min.**	**Max.**	**Mean**	**Standard deviation**	**Median**
LV-MA	0.970	1	0.989	0.008	0.990
GD	0.296	0.954	0.786	0.395	0.821
GDM	0.122	0.923	0.743	0.404	0.723

Table 28 - Accuracy of the ANNs trained with different algorithms

	RMSE				
Training algorithm	**Min.**	**Max.**	**Mean**	**Standard deviation**	**Median**
LV-MA	7.01	25.12	13.75	4.15	13.47
GD	17.91	152.18	71.02	31.42	66.96
GDM	21.53	181.55	72.62	30.60	67.72

Table 29 - RMSE of the ANNs trained with different algorithms

The first training algorithm tested is the Levenberg-Marquardt training algorithm. Tables 28-29 and figures 36-38 show the results of this test, that are expressed in terms of ANN's accuracy in predicting one-day ahead price of S&P 500 index and RMSE.

Fig. 36 - One-day ahead price prediction of S&P 500 index vs. actual data

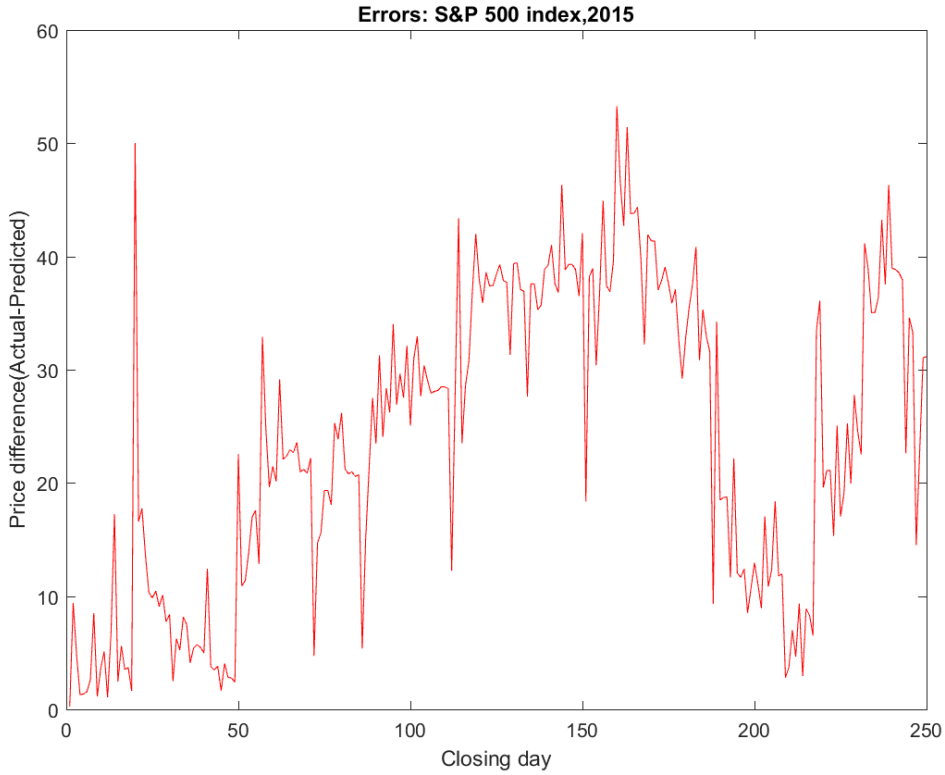

Fig. 37 - Difference between actual and predicted prices ($)

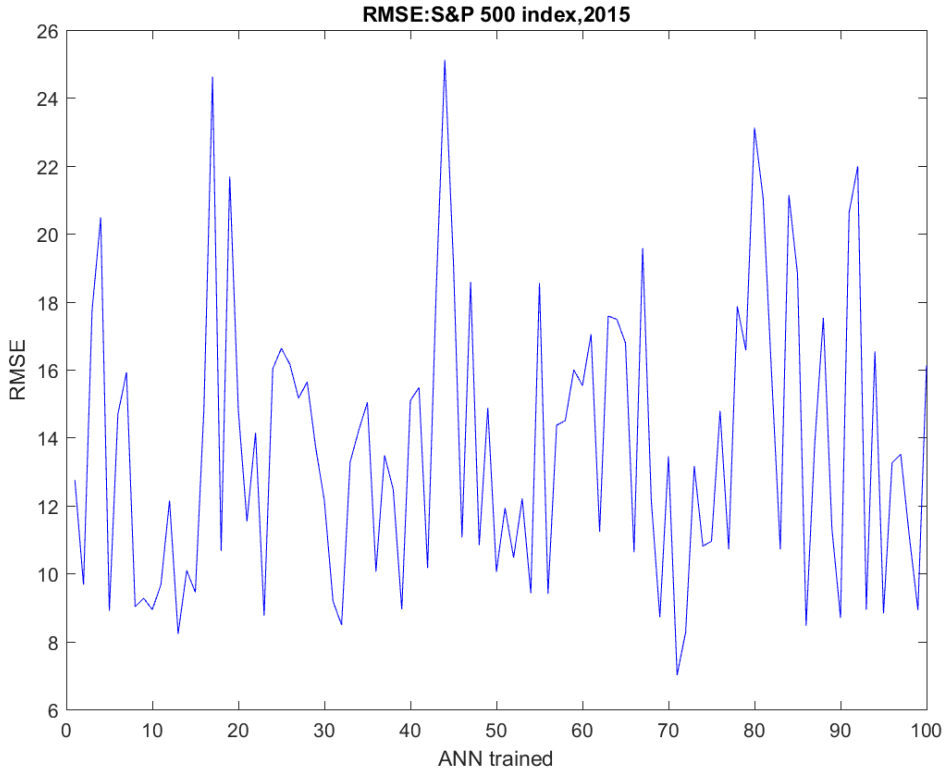

Fig. 38 - RMSE of the 100 ANNs trained

Fig. 36 shows a fairly aligned prediction curve. The predicted price, compared to the actual price, has a difference of at most 55$. However, almost all the errors lie below a difference of 20$. Considering that actual stock index prices usually do not fluctuate more than 2% of the index value, the predicted price mimic the patterns of the actual one fairly well for the first 50 days. After this time period, even though the errors are greater, the curve of the actual price is similar to the one of the predicted price, at least with respect to the prices rising and decreasing. However, the errors over the last 200 days lie below a difference of 60$, that corresponds to a 2.8% change in the actual price of S&P 500 index (i.e. the mean of the actual price over the period considered). This finding is confirmed by the average RMSE, which is equal to 13.75. Furthermore, the standard deviation of the RMSE is relatively low as shown in figure 38. These results indicate that on average the predicted prices differ from the actual ones by less than 2%. Finally, considering all the previous results and that the accuracy varies within a 3% range, it is possible to conclude that the price predicted by this test is reliable.

The second training algorithm tested is the steepest descent (gradient descent) training algorithm. Tables 28-29 and Figures 39-41 show the results of this test.

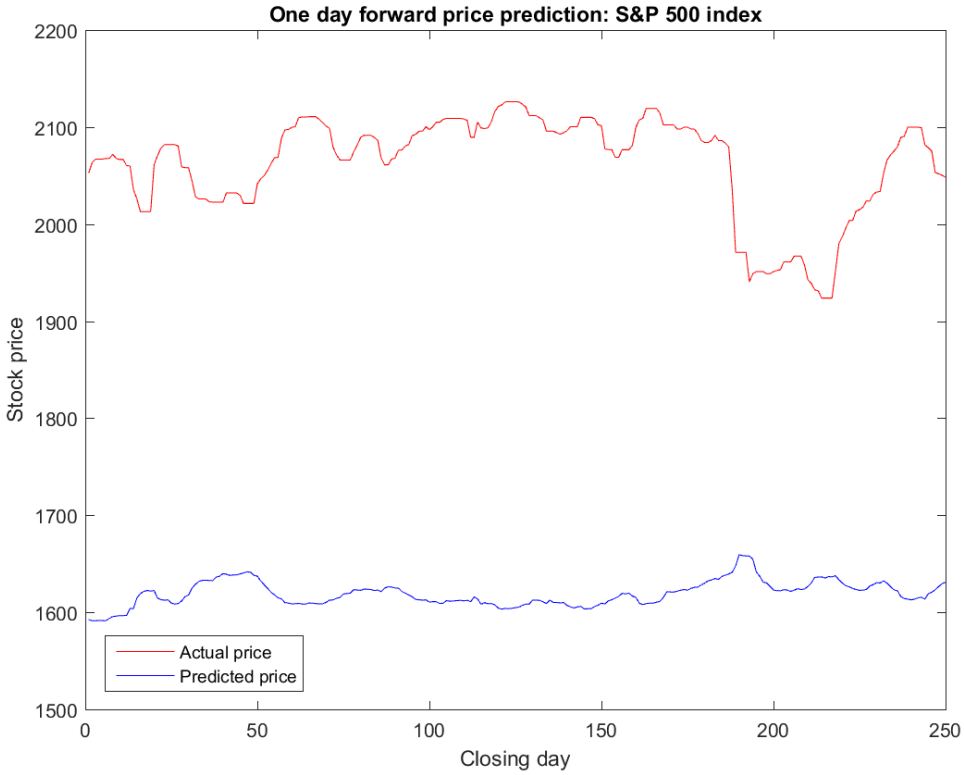

Fig. 39 - One-day ahead price prediction of S&P 500 index vs. actual data

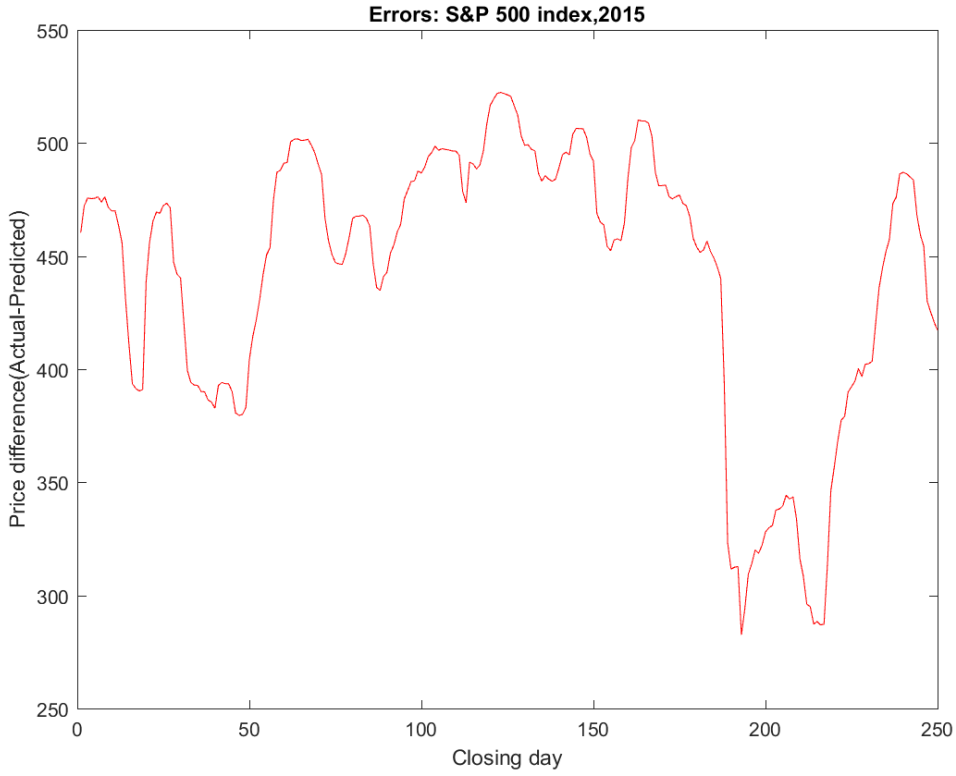

Fig. 40 - Difference between actual and predicted prices ($)

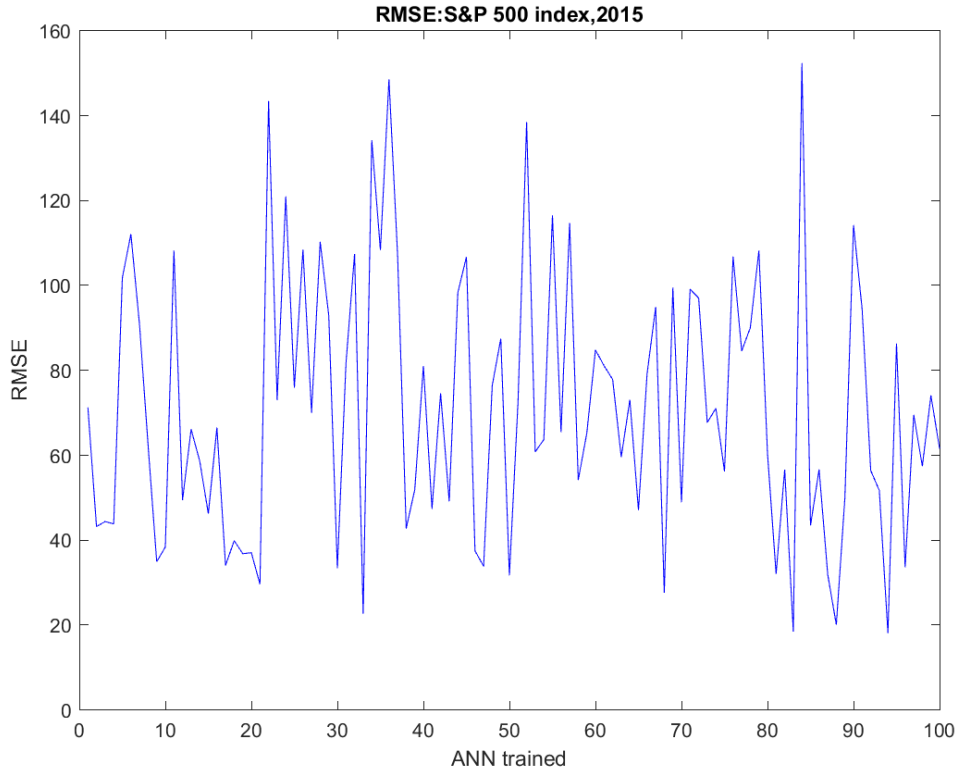

Fig. 41 - RMSE of the 100 ANNs trained

Fig. 39 shows a poorly aligned prediction curve for all the 250 days. The predicted price is not able to mimic the patterns of the actual one. In this case, the average RMSE and its standard deviation are respectively equal to 71.02 and 31.42, meaning that on average there is a large difference between actual and predicted prices. The ANN trained with gradient descent training algorithm has a poorer performance compared to one of the ANN trained with Levenberg-Marquardt training algorithm. Furthermore, the average accuracy and its standard deviation for one hundred retrained networks are respectively equal to 0.786 and 0.395. This result means that on average the predicted prices differ from the actual ones by more than 30% and that the accuracy presents a huge variation among the one hundred retrained networks. Finally, considering all the previous results and that the accuracy varies within a 66% range, it is possible to conclude that the price predicted by this test is unreliable.

The third training algorithm tested is the gradient descent training algorithm with momentum. Tables 28-29 and Figures 42-44 show the results of this test.

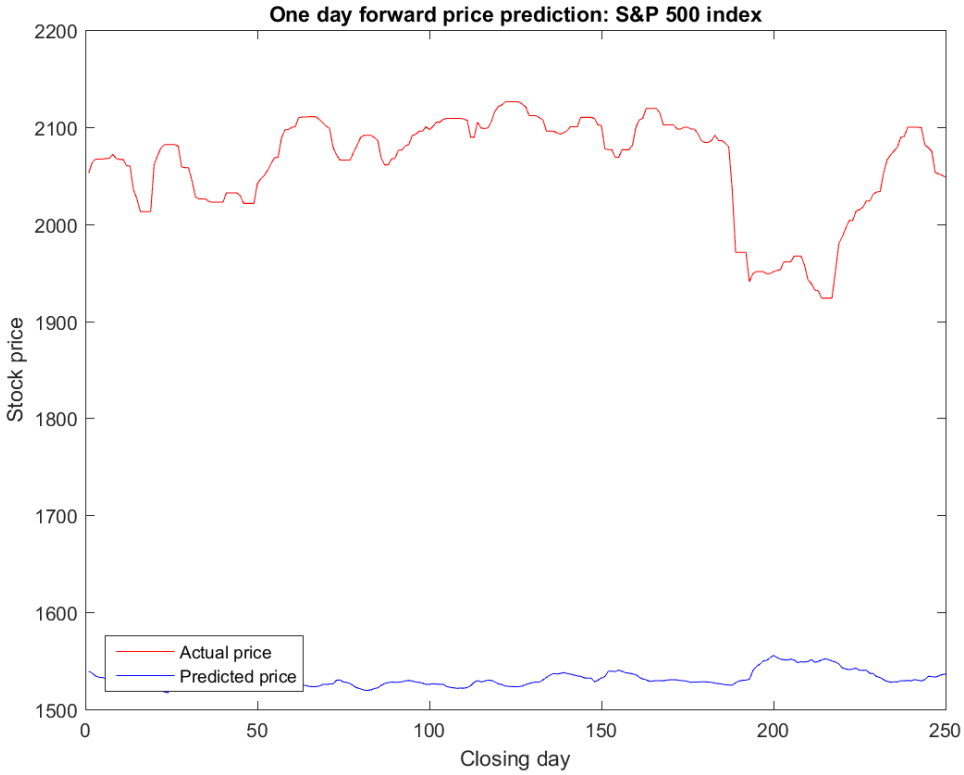

Fig. 42 - One-day ahead price prediction of S&P 500 index vs. actual data

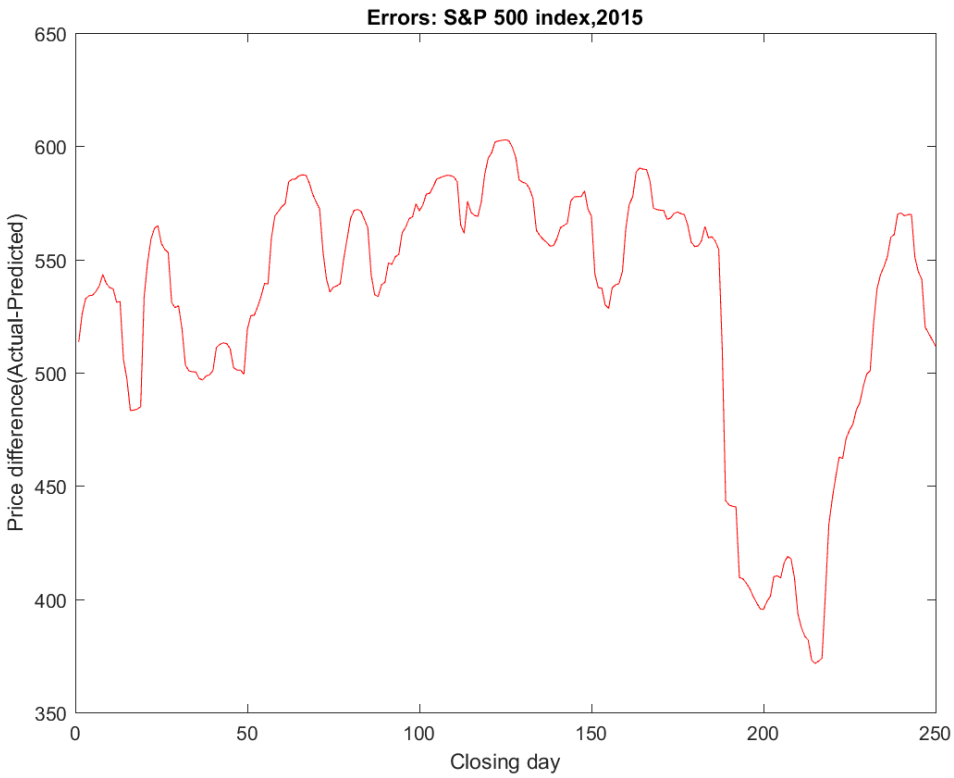

Fig. 43 - Difference between actual and predicted prices ($)

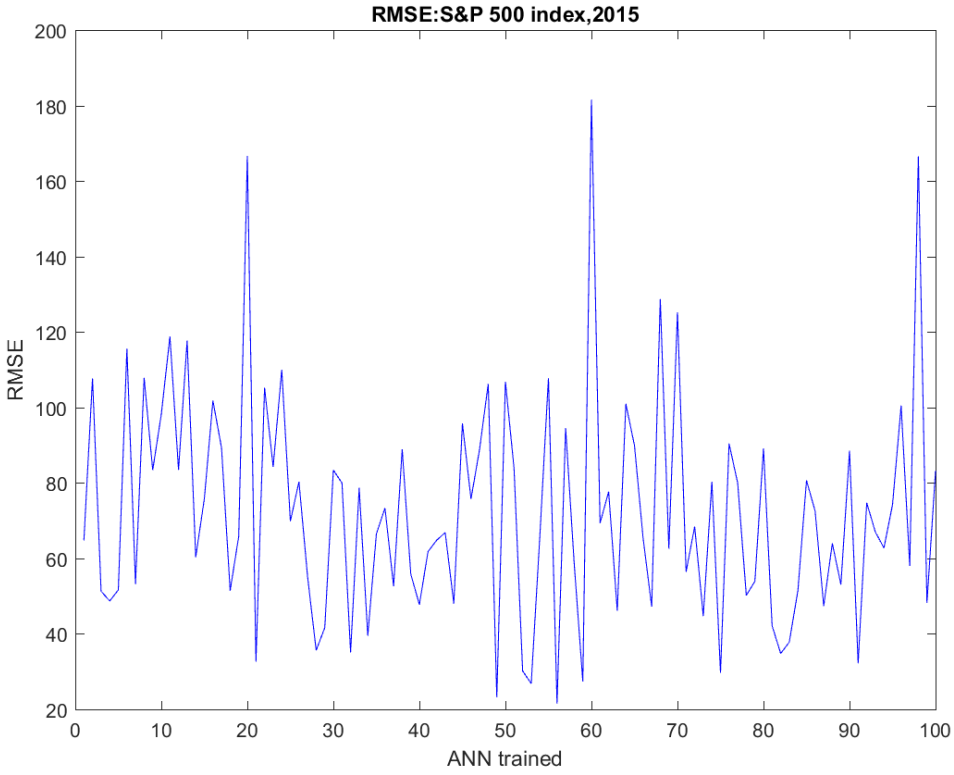

Fig. 44 - RMSE of the 100 ANNs trained

Fig. 42 shows a prediction curve that is not aligned at all with the one of the actual prices for all the testing period. Almost all the errors are greater than 400$ and they reach a peak of 600$. This finding is confirmed by the average RMSE which is equal to 72.62. The ANN trained with gradient descent training algorithm has the worse predictive power among all the others training algorithms used. Furthermore, the average accuracy and its standard deviation for one hundred retrained networks are respectively equal 0.743 and 0.404. This results means that the predicted prices differ from the actual ones by 35%. Finally, since the accuracy varies within a 80% range, the price predicted by this test is can be considered as unreliable.

Considering the results obtained by training the ANN with three different training algorithms, it possible to conclude that the Levenberg-Marquardt training algorithm is the most suitable algorithm to predict the price of the S&P 500 index. For this reason, it will be used to train the ANNs in all the tests of the price prediction model.

5.5 Price prediction model: second test

The second test consists in investigating how the number of neurons in the hidden layer affect the predictions of the ANN. In this test, the hidden layer size was set at 20, 50 and 150 neurons. The distribution of training dataset was divided into 70% for training, 15% for validation and 15% for testing. All the other parameters were kept on their default value. Furthermore, the results were based on 100 retrained ANNs. Tables 30-31 and figures 45-53 show the results of the whole test. This test is divided into three steps according to the hidden layer size that was selected.

	Accuracy				
Hidden layer size	**Min.**	**Max.**	**Mean**	**Standard deviation**	**Median**
20 neurons	0.928	1	0.983	0.020	0.986
50 neurons	0.812	1	0.979	0.037	0.983
150 neurons	0.028	0.782	0.655	0.410	1.128

Table 30 - Accuracy of the ANNs with different layer size

	RMSE				
Hidden layer size	**Min.**	**Max.**	**Mean**	**Standard deviation**	**Median**
20 neurons	8.18	54.22	22.11	10.56	20.08
50 neurons	8.50	211.38	43.42	36.50	31.62
150 neurons	128.71	1037.41	358.31	157.54	325.46

Table 31 - RMSE of the ANNs with different layer size

In the first step, the hidden layer size was set at 20 neurons. Tables 30-31 and figures 45-47 show the results of this test, that are expressed in terms of ANN's accuracy in predicting the one-day ahead price of S&P 500 index and RMSE.

Fig. 45 - One-day ahead price prediction of S&P 500 index vs. actual data

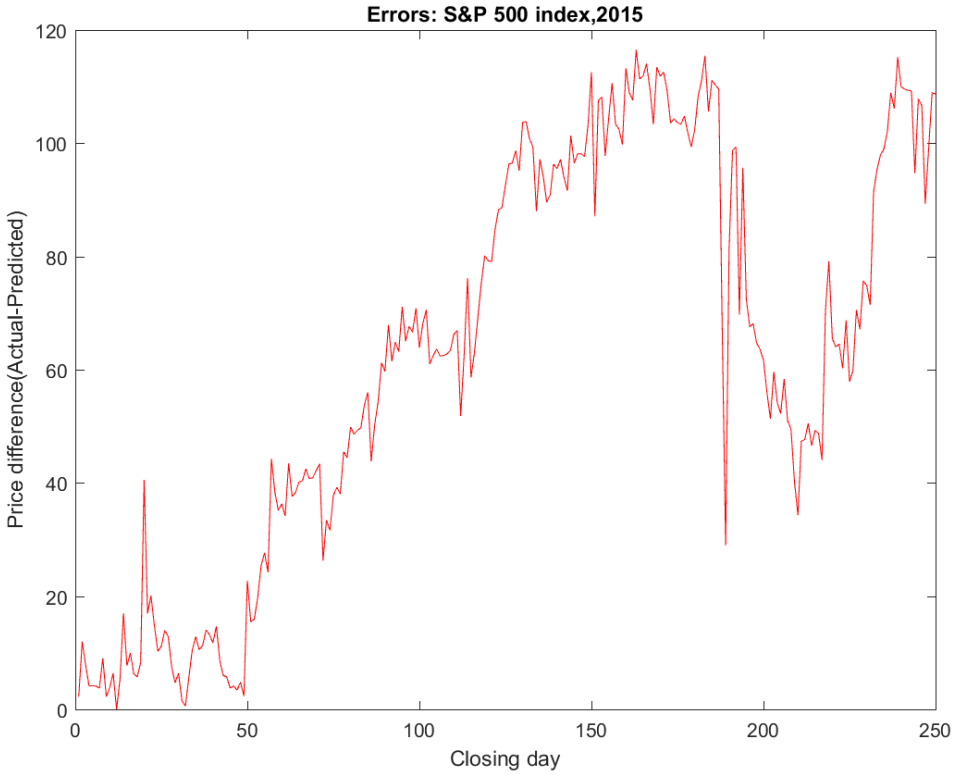

Fig. 46 - Difference between actual and predicted prices ($)

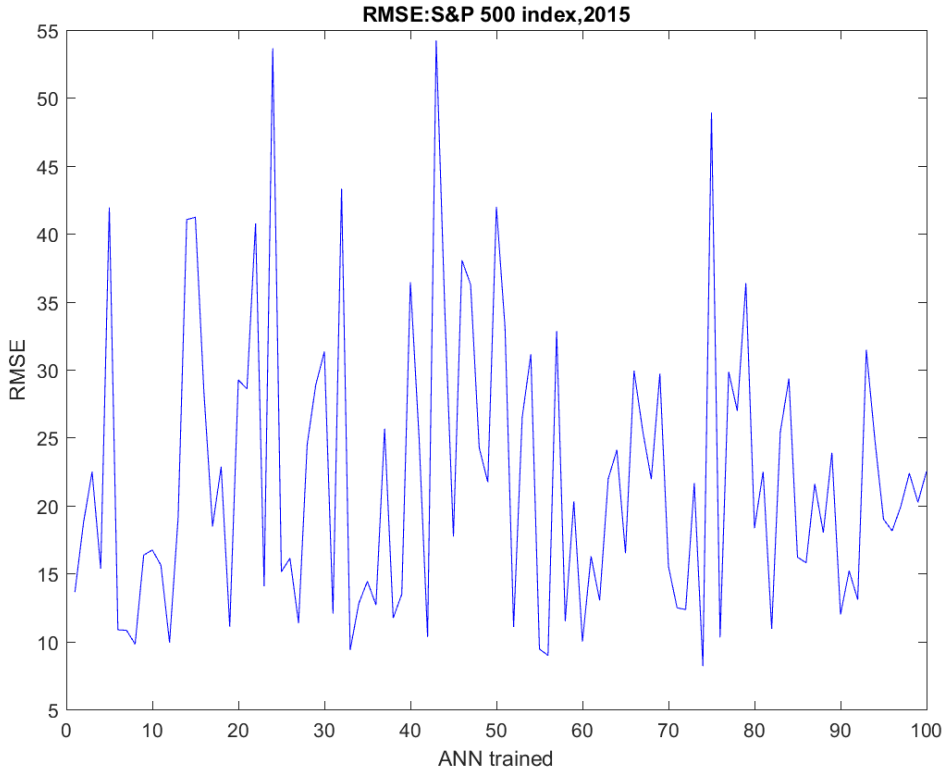

Fig. 47 - RMSE of the 100 ANNs trained

Fig. 45 shows a fairly aligned prediction curve. The predicted price, compared to the actual price, has a difference of at most 120$. Even though the errors tends to increase over time, the curve of the actual price is similar to the one of predicted price, at least with respect to the prices rising and decreasing. The behaviuor of the predictive curve is almost similar to the one of the previous test were the hidden layer size was set at its default value that is 10 neurons. This finding is confirmed by the average RMSE, which is equal to 22.11. Moreover, the standard deviation of the RMSE is relatively low as shown in figure 39. The average accuracy is equal 0.983, meaning that the predicted prices differ from the actual ones by less than 2%. Finally, considering all the previous results and that the accuracy varies within a 7.3% range, it is possible to conclude that the price predicted by this test is reliable.

In the second step, the hidden layer size was set at 50 neurons. Tables 30-31 and figures 48-50 show the results of this test.

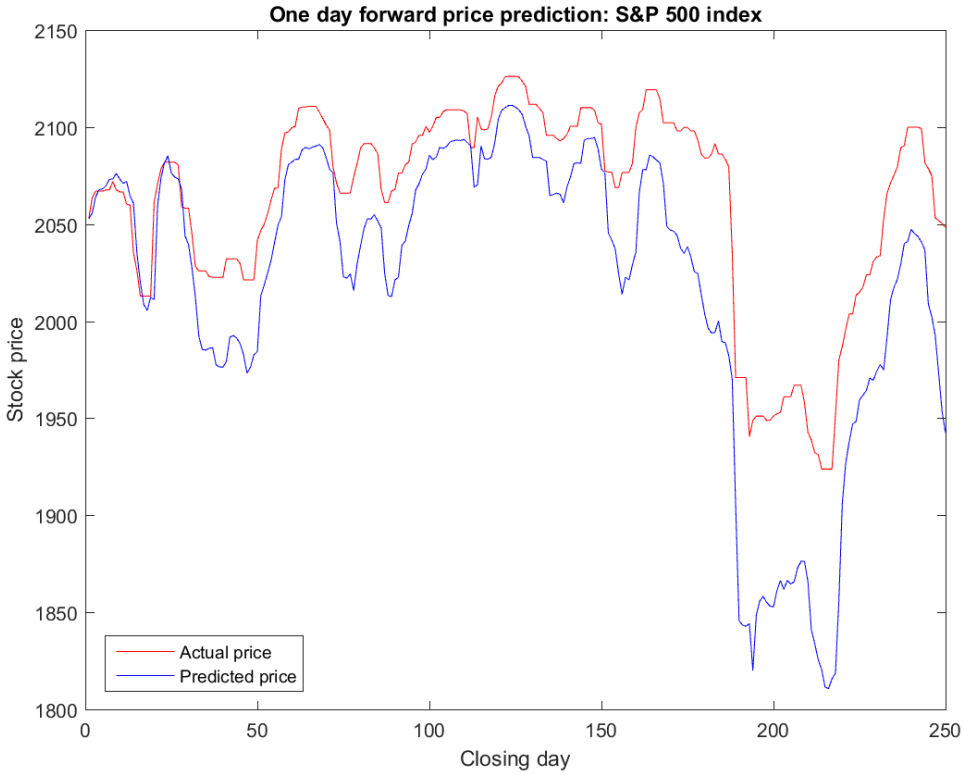

Fig. 48 - One-day ahead price prediction of S&P 500 index vs. actual data

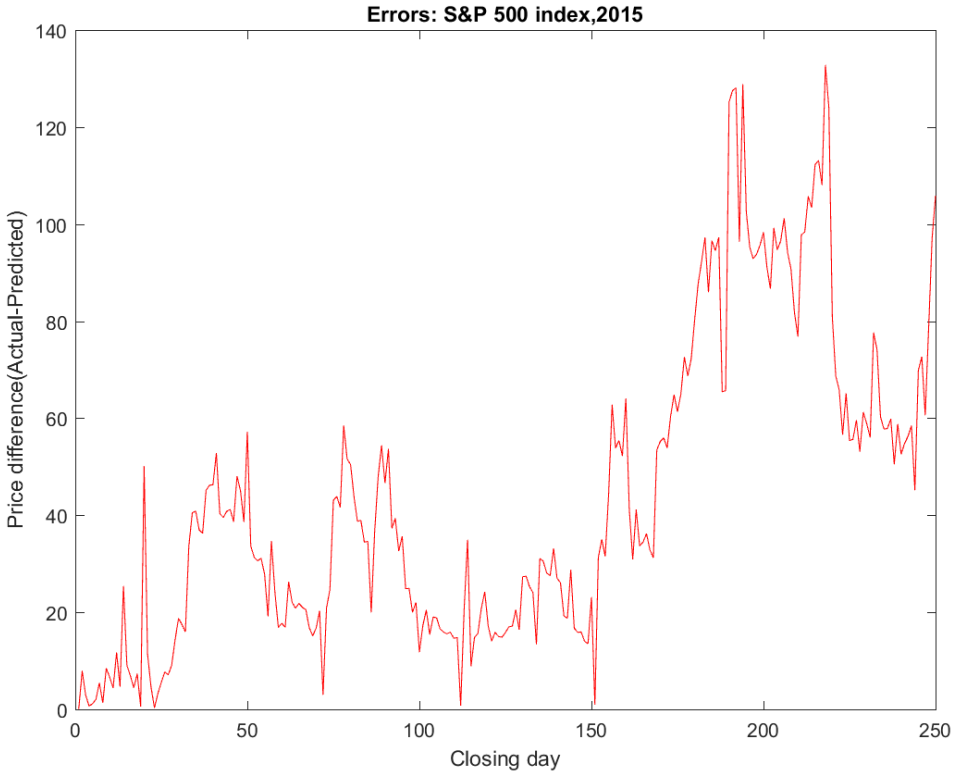

Fig. 49. Difference between actual and predicted prices ($)

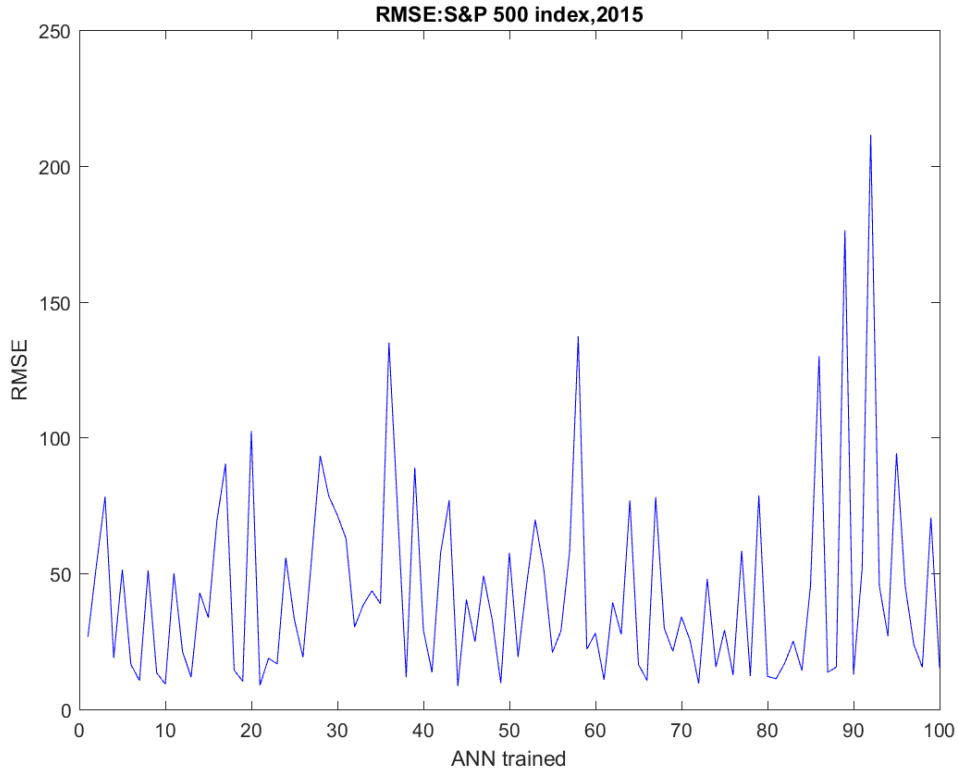

Fig. 50 - RMSE of the 100 ANNs trained

Fig. 48 shows a fairly aligned prediction curve. The predicted price, compared to the actual price, has a difference of at most 136$. Even though the errors tends to increase over time, the curve of the actual price is similar to the one of predicted price, at least with respect to the prices rising and decreasing. The difference between the results obtained using a hidden layer size set at 20 and 50 neurons is significant. In the case of 50 hidden neurons, the average RMSE is equal to 43.42. Furthermore, the standard deviation of the RMSE is equal to 36.50 as shown in figure 50. Both these results indicate that the ANN with 50 hidden neurons has a lower predictive power than the one with 20 hidden neurons. Moreover, the average accuracy is equal to 0.979 with a standard deviation that varies within a 19% range. Finally, considering all the previous results and that the accuracy varies within a 18.8% range, it is possible to conclude that the price predicted by this test is unreliable.

In the third step, the hidden layer size was set at 150 neurons. Tables 30-31 and figures 51-53 show the results of this test.

Fig. 51 - One-day ahead price prediction of S&P 500 index vs. actual data

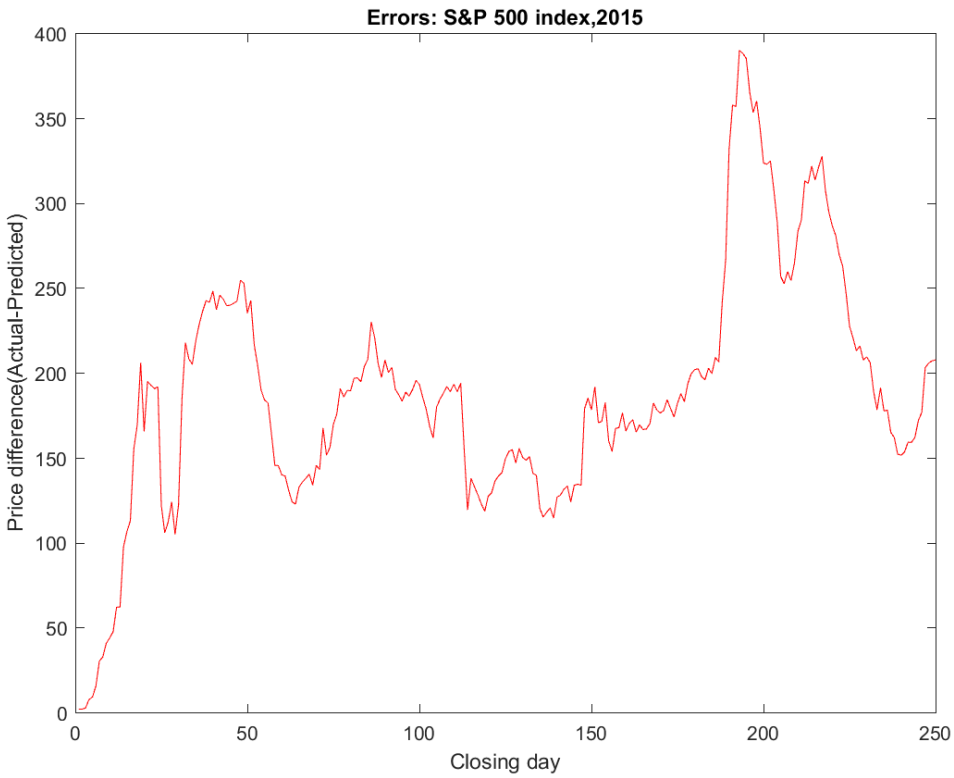

Fig. 52 - Difference between actual and predicted prices ($)

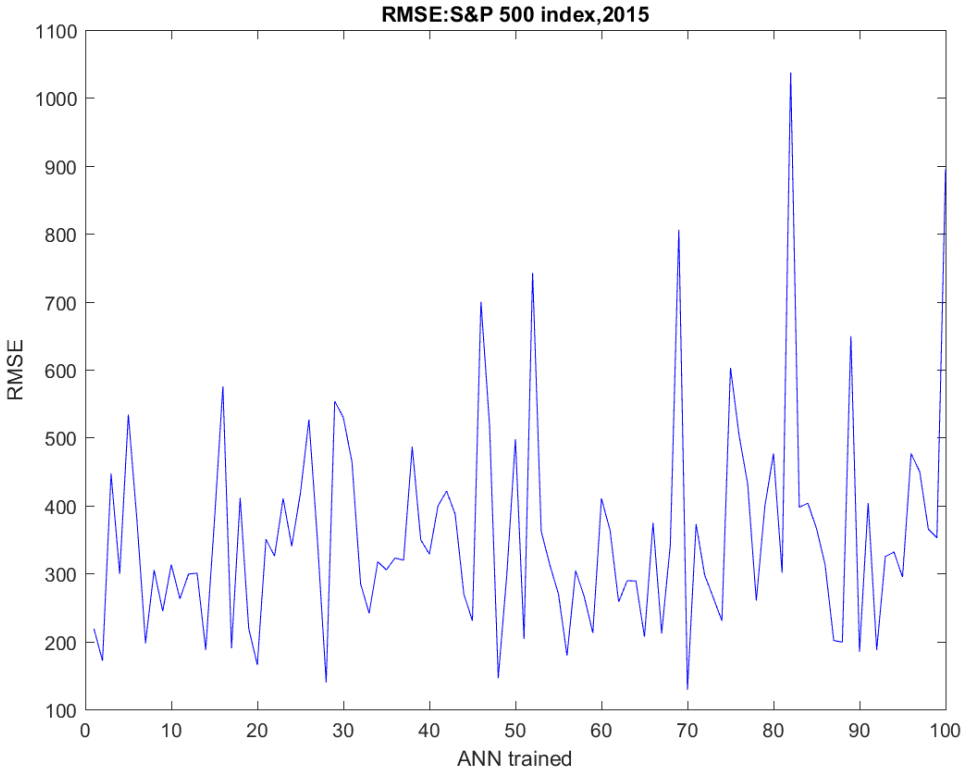

Fig. 53 - RMSE of the 100 ANNs trained

The difference between the results obtained using a hidden layer size set at 150 and the other two cases (i.e. 20 and 50 hidden neurons) is huge. The predicted price, compared to the actual one, reached a difference of almost 400$ as shown in figure 52. Furthermore, the average RMSE is equal to 358.31 with a large standard deviation of 157.54 as shown in figure 53. Both these results indicate that the ANN with 150 hidden neurons has the lowest predictive power. Since the range of the accuracy for predictions varies within a 75% range, the price predicted by the ANN with 150 neurons can be considered as unreliable.

Considering the results obtained in this test, it is possible to conclude that an increased number of neurons in the hidden layer results in a lower predictive power.

5.6 Price prediction model: third test

The third test consists in examining how the distribution of the training dataset affect the predictions of the ANN. In this test, the distribution of the training dataset was set as follows:

1. Equal distribution: 34% for training, 33% for validation, 33% for testing;
2. High training :90% for training, 5% for validation, 5% for testing;
3. High validation :5% for training, 90% for validation, 5% for testing;
4. High testing: 5% for training, 5% for validation, 90% for testing;
5. Low training: 4% for training, 48% for validation, 48% for testing;
6. Low validation: 48% for training, 4% for validation, 48% for testing;
7. Low testing: 48% for training, 48% for validation, 4% for testing;

All the other parameters were kept on their default value. Furthermore, the results were based on 100 retrained ANNs. Tables 32-33 and figures 46-62 show the results of the whole test. This test is divided into seven steps according to the different distribution of the training dataset.

	Accuracy				
Training dataset distribution	**Min.**	**Max.**	**Mean**	**Standard deviation**	**Median**
Equal	0.927	1	0.984	0.014	0.986
high training	0.930	1	0.982	0.013	0.984
high validation	0.895	1	0.976	0.031	0.978
high testing	0.889	1	0.980	0.043	0.979
low training	0.856	1	0.972	0.041	0.974
low validation	0.926	1	0.985	0.013	0.988
low testing	0.958	1	0.983	0.011	0.984

Table 32 - Accuracy of the ANNs with different training dataset distribution

	RMSE				
Training dataset distribution	**Min.**	**Max.**	**Mean**	**Standard deviation**	**Median**
Equal	8.43	62.84	18.36	8.86	16.13
high training	8.24	54.89	19.63	8.47	19.18
high validation	7.32	85.52	28.83	16.61	24.12
high testing	4.54	86.31	30.22	14.82	27.41
low training	5.56	110.78	31.52	17.02	27.60
low validation	8.35	104.20	17.86	11.55	15.71
low testing	7.76	41.37	18.25	7.67	16.83

Table 33 - RMSE of the ANNs with different training dataset distribution

In the first step, the distribution of the training dataset is divided equally between training, validation and testing sets. Tables 32-33 and figures 54-56 show the results of this test, that are expressed in terms of ANN's accuracy in predicting one-day ahead price of S&P 500 index and RMSE.

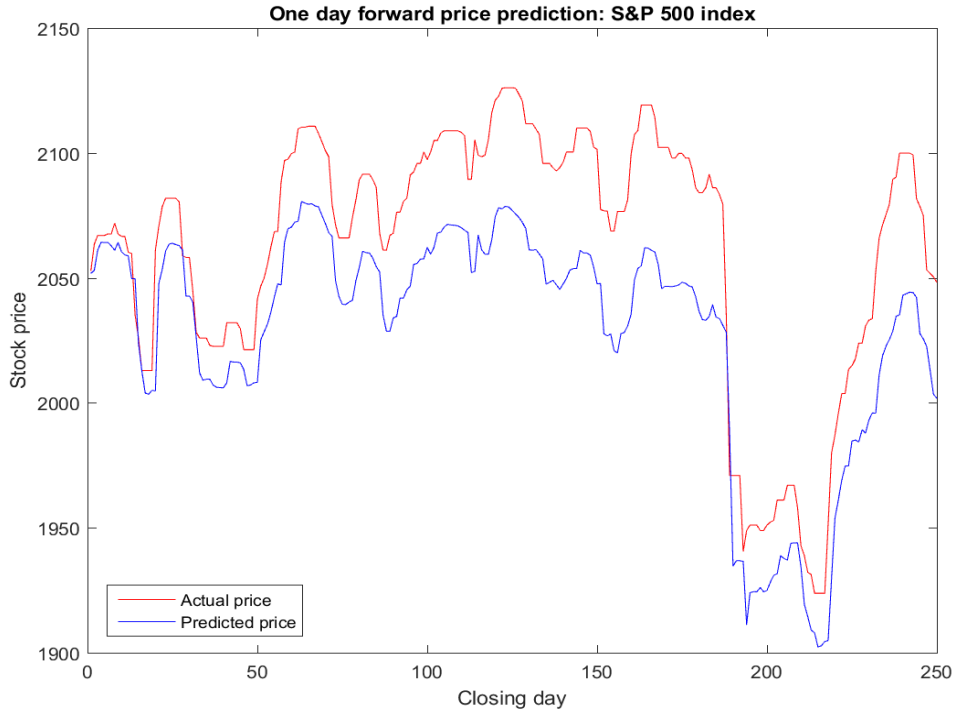

Fig. 54 - One-day ahead price prediction of S&P 500 index vs. actual data

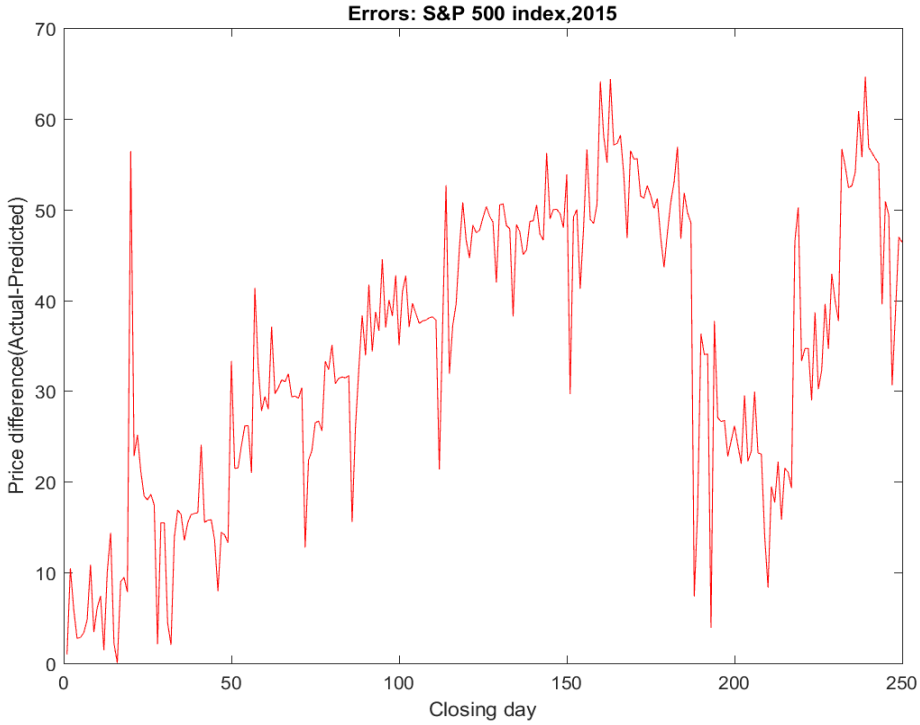

Fig. 55 - Difference between actual and predicted prices ($)

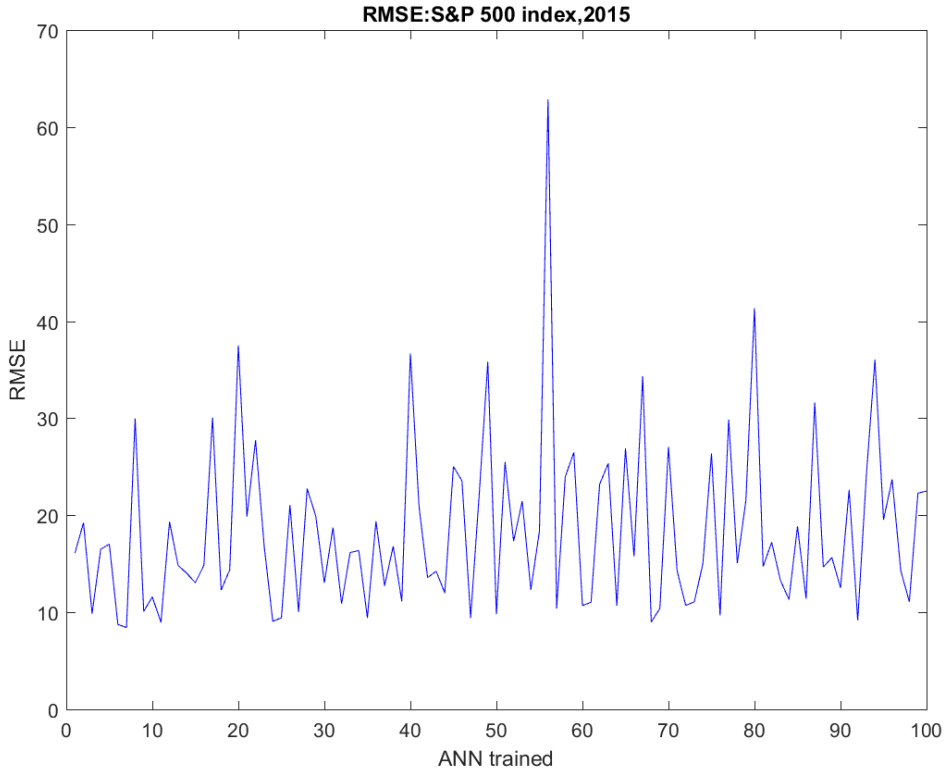

Fig. 56 - RMSE of the 100 ANNs trained

Fig. 54 shows a fairly aligned prediction curve. The predicted price, compared to the actual price, has a difference of at most 65$. Even though the errors start to increase after the first 50 days, the curve of the actual price is similar to the one of predicted price, at least with respect to the prices rising and decreasing. Furthermore, the average RMSE is equal to 18.36, meaning that on average the errors are really low. The RMSE has a relatively low standard deviation as shown in figure 47. Moreover, the average accuracy is equal to 0.984 with a standard deviation of 0.014, meaning that on average the predicted prices differ from the actual ones by less than 2%. Finally, considering all the previous results and that the accuracy varies within a 7.3% range, it is possible to conclude that the price predicted by this test is reliable.

In the second step, the distribution of the training dataset is divided as follows:

- 90% of the training dataset for training;
- 5% of the training dataset for validation;
- 5% of the training dataset for testing.

Tables 32-33 and figures 57-59 show the results of this test.

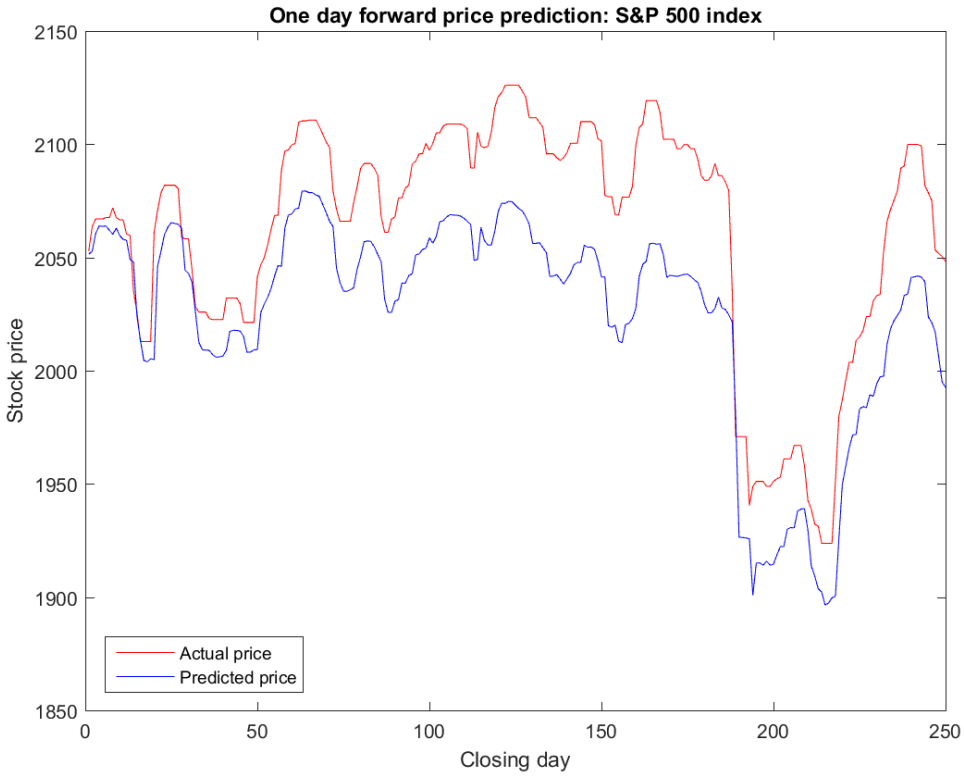

Fig. 57 - One-day ahead price prediction of S&P 500 index vs. actual data

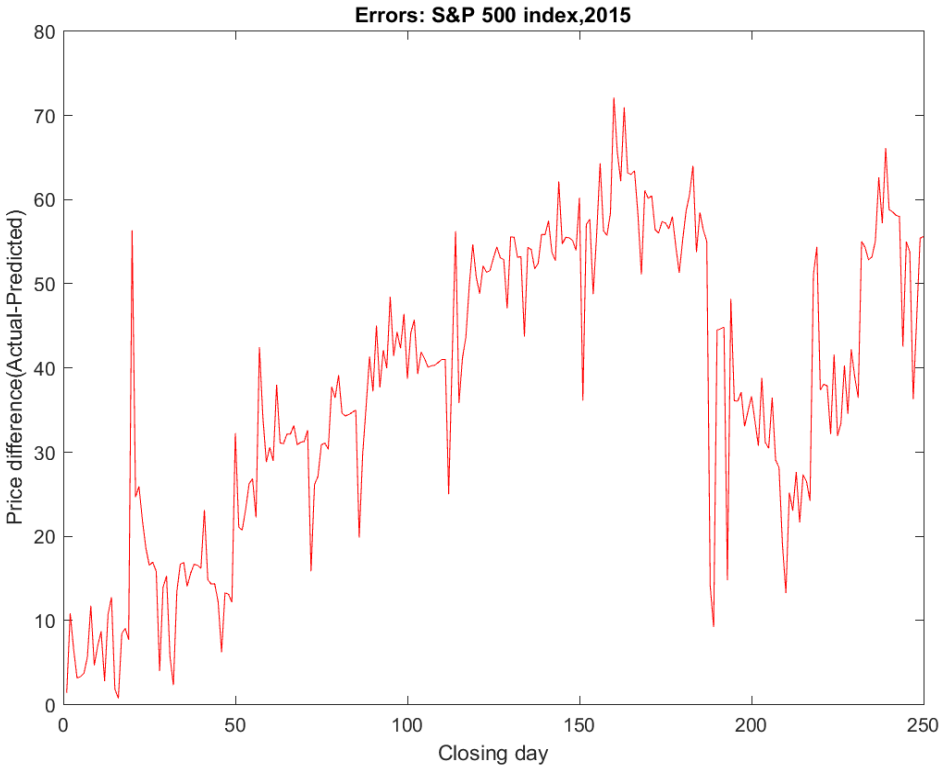

Fig. 58 - Difference between actual and predicted prices ($)

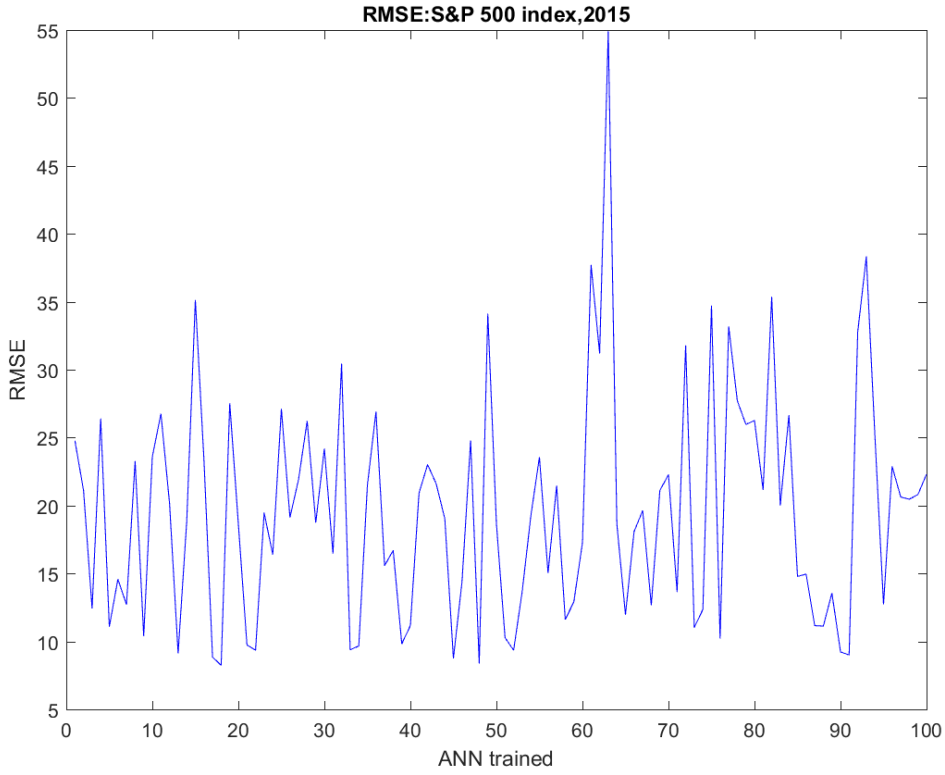

Fig. 59 - RMSE of the 100 ANNs trained

Fig. 57 shows a fairly aligned prediction curve. The predicted price, compared to the actual price, has a difference of at most 72$. Even though the errors for the first 150 days show an upward trend, the curve of the actual price is similar to the one of predicted price, at least with respect to the prices rising and decreasing. Furthermore, the average RMSE is equal to 19.63, meaning that on average the errors are really low. The RMSE has a relatively low standard deviation as shown in figure 50. Moreover, the average accuracy is equal to 0.982 with a standard deviation of 0.013, meaning that on average the predicted prices differ from the actual ones by less than 2%. Finally, considering all the previous results and that the accuracy varies within a 7% range, it is possible to conclude that the price predicted by this test is reliable.

In the third step, the distribution of the training dataset is divided as follows:

- 5% of the training dataset for training;
- 90% of the training dataset for validation;
- 5% of the training dataset for testing.

Tables 32-33 and figures 60-62 show the results of this test.

Fig. 60 - One-day ahead price prediction of S&P 500 index vs. actual data

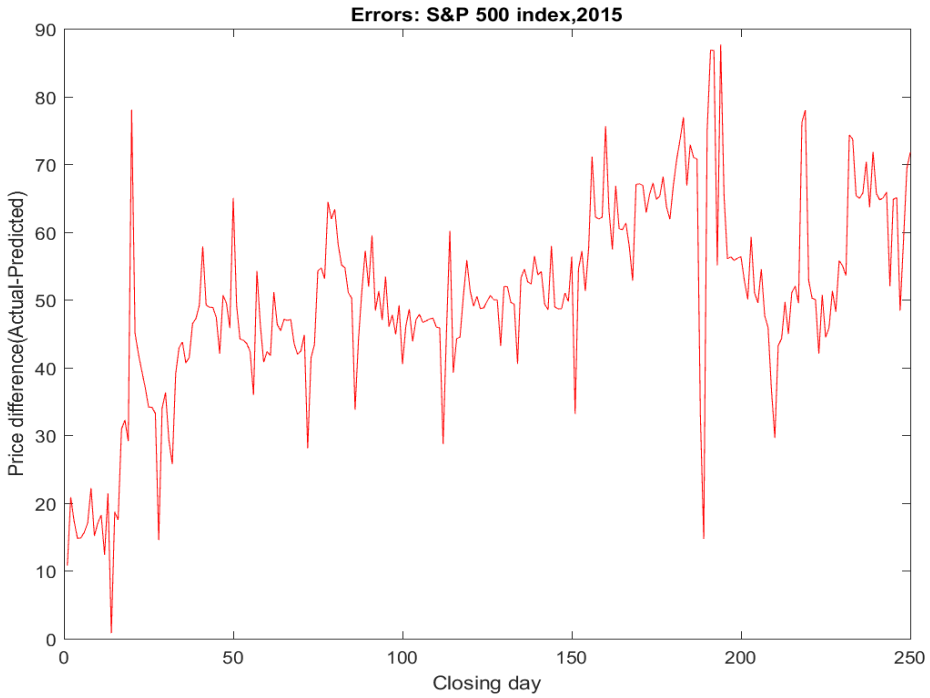

Fig. 61 - Difference between actual and predicted prices ($)

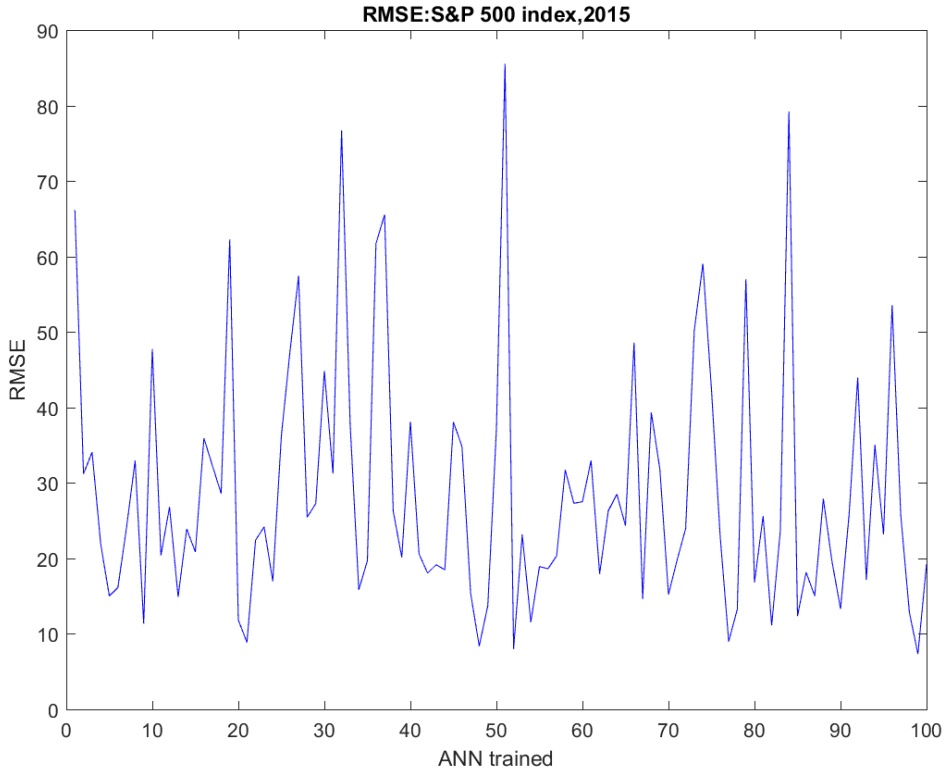

Fig. 62 - RMSE of the 100 ANNs trained

Fig. 60 shows a fairly aligned prediction curve. The predicted price, compared to the actual price, has a difference of at most 89$. Even though the errors increases over time, the curve of the actual price is similar to the one of predicted price, at least with respect to the prices rising and decreasing. Furthermore, the average RMSE is equal to 28.83, meaning that on average the errors are really low. The RMSE has a standard deviation equal to 16.61. Moreover, the average accuracy is equal to 0.976 with a standard deviation of 0.031, meaning that on average the predicted prices differ from the actual ones by less than 3%. Finally, considering all the previous results and that the accuracy varies within a 10% range, it is possible to conclude that the price predicted by this test is reliable.

In the fourth step, the distribution of the training dataset is divided as follows:

- 5% of the training dataset for training;
- 5% of the training dataset for validation;
- 90% of the training dataset for testing.

Tables 32-33 and figures 63-65 show the results of this test.

Fig. 63 - One-day ahead price prediction of S&P 500 index vs. actual data

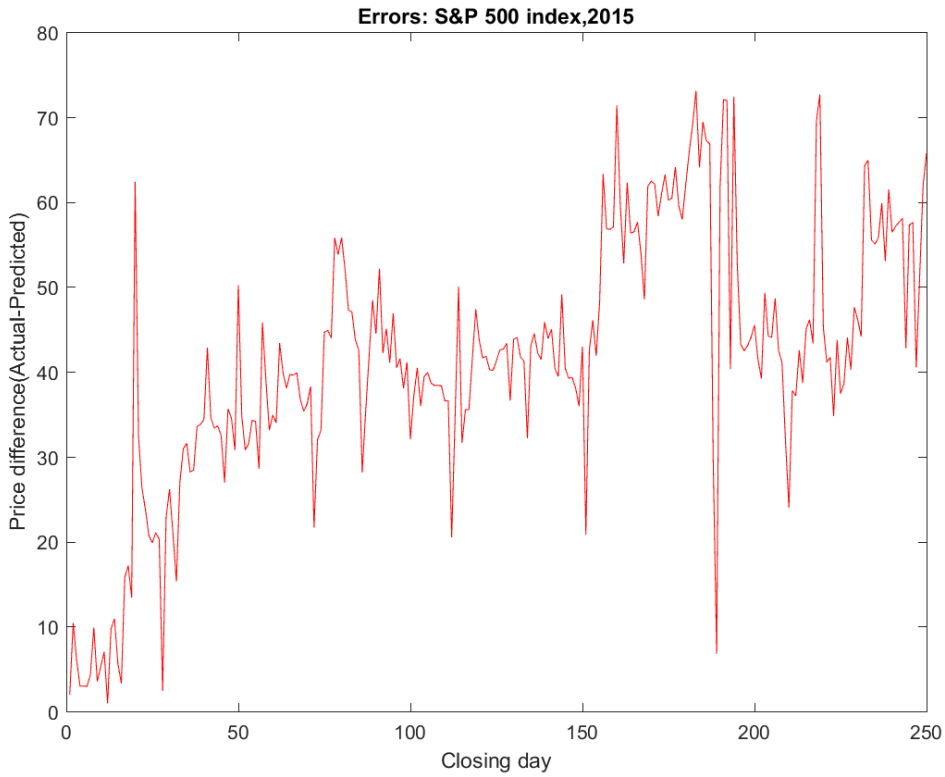

Fig. 64 - Difference between actual and predicted prices ($)

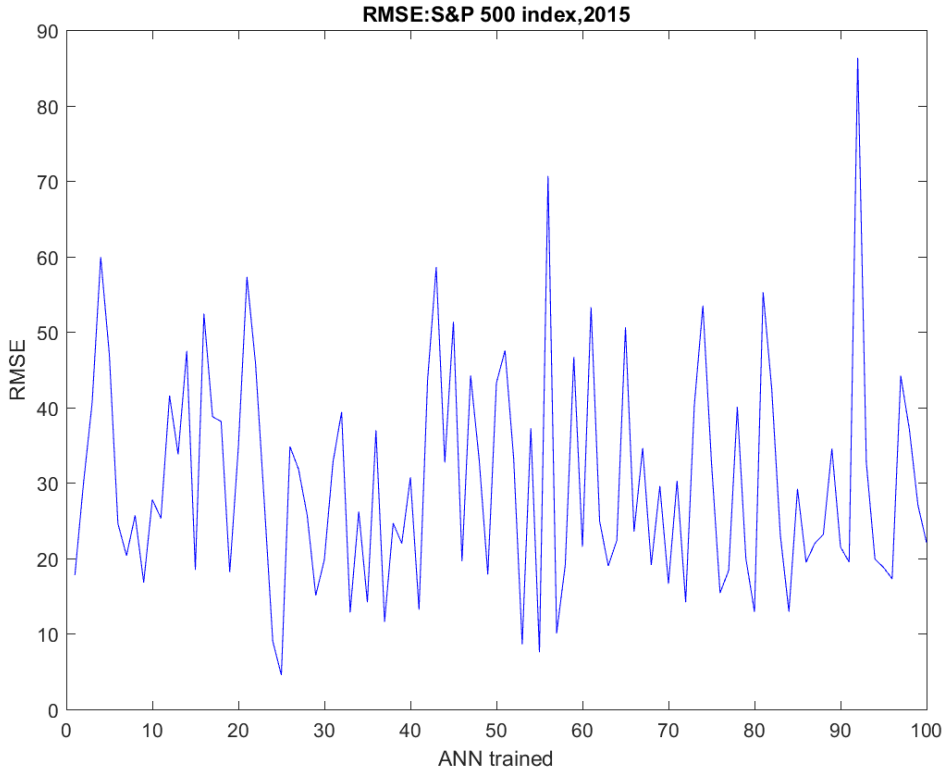

Fig. 65 - RMSE of the 100 ANNs trained

Fig. 63 shows the prediction curve vs the curve of actual prices. The predicted price, compared to the actual price, has a difference of at most 73$. The average RMSE is equal to 30.22 with a standard deviation of 14.82, meaning that on average the errors are relatively low. Moreover, the average accuracy is equal to 0.980 with a standard deviation of 0.043, meaning that on average the predicted prices differ from the actual ones by exactly 2%. Finally, considering all the previous results and that the accuracy varies within a 11.1% range, it is possible to conclude that the price predicted by this test is reliable.

In the fifth step, the distribution of the training dataset is divided as follows:

- 4% of the training dataset for training;
- 48% of the training dataset for validation;
- 48% of the training dataset for testing.

Tables 32-33 and figures 66-68 show the results of this test.

Fig. 66 - One-day ahead price prediction of S&P 500 index vs. actual data

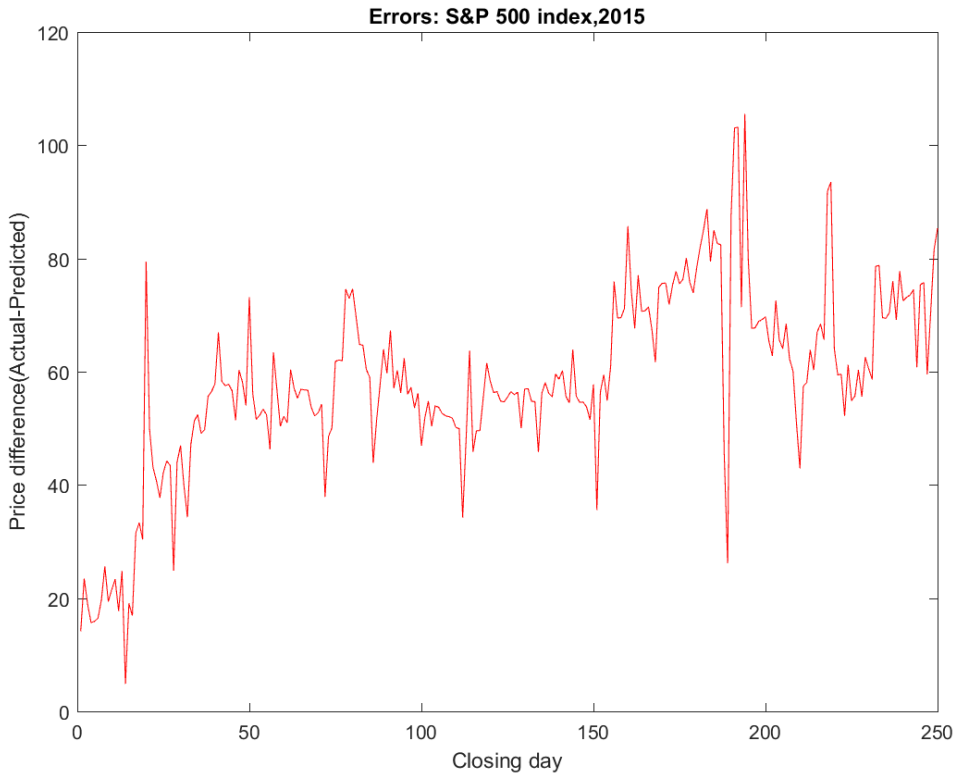

Fig. 67 - Difference between actual and predicted prices ($)

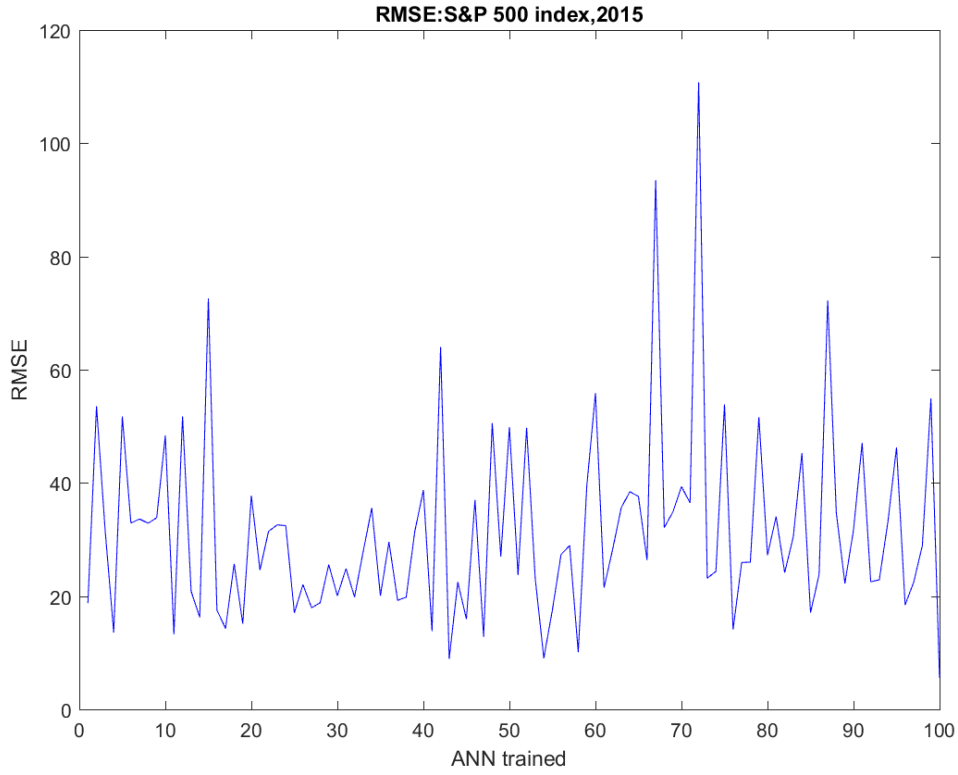

Fig. 68 - RMSE of the 100 ANNs trained

In fig. 66 the curve of the predicted prices is compared to the one of actual prices. The predicted price, compared to the actual price, has a difference of at most 105$. The average RMSE is equal to 31.52 with a standard deviation of 17.02, meaning that on average the errors are realtively low. Moreover, the average accuracy is equal to 0.972 with a standard deviation of 0.041, meaning that on average the predicted prices differ from the actual ones by less than 3%. Finally, considering all the previous results and that the accuracy varies within a 14.4% range, it is possible to conclude that the price predicted by this test is unreliable.

In the sixth step, the distribution of the training dataset is divided as follows:

- 48% of the training dataset for training;
- 4% of the training dataset for validation;
- 48% of the training dataset for testing.

Tables 32-33 and figures 69-71 show the results of this test.

Fig. 69 - One-day ahead price prediction of S&P 500 index vs. actual data

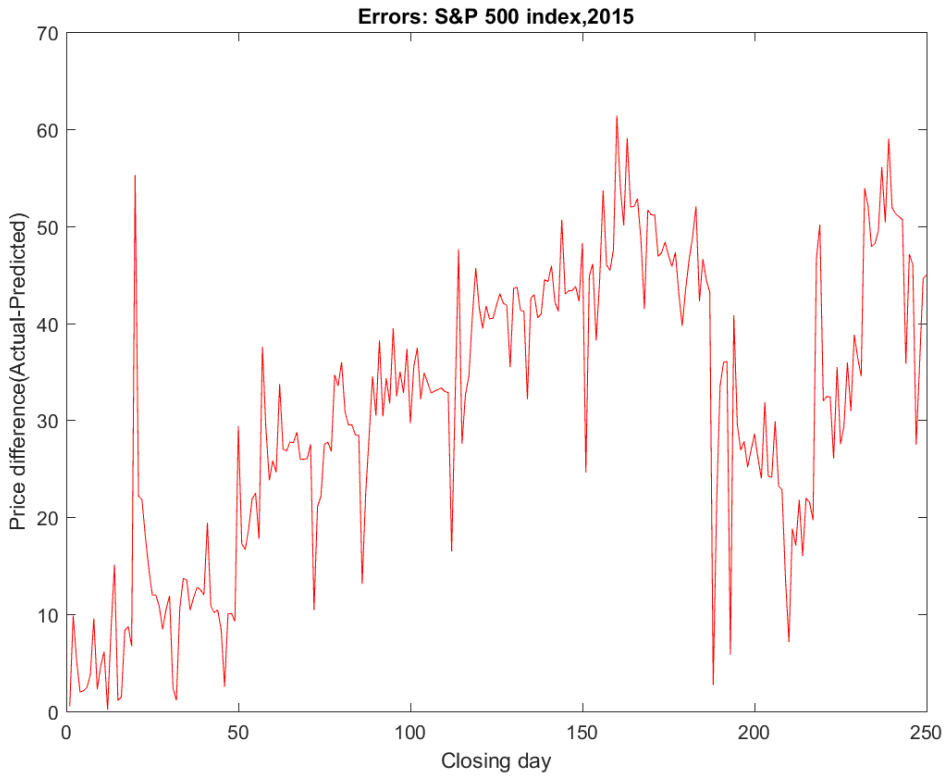

Fig. 70 - Difference between actual and predicted prices ($)

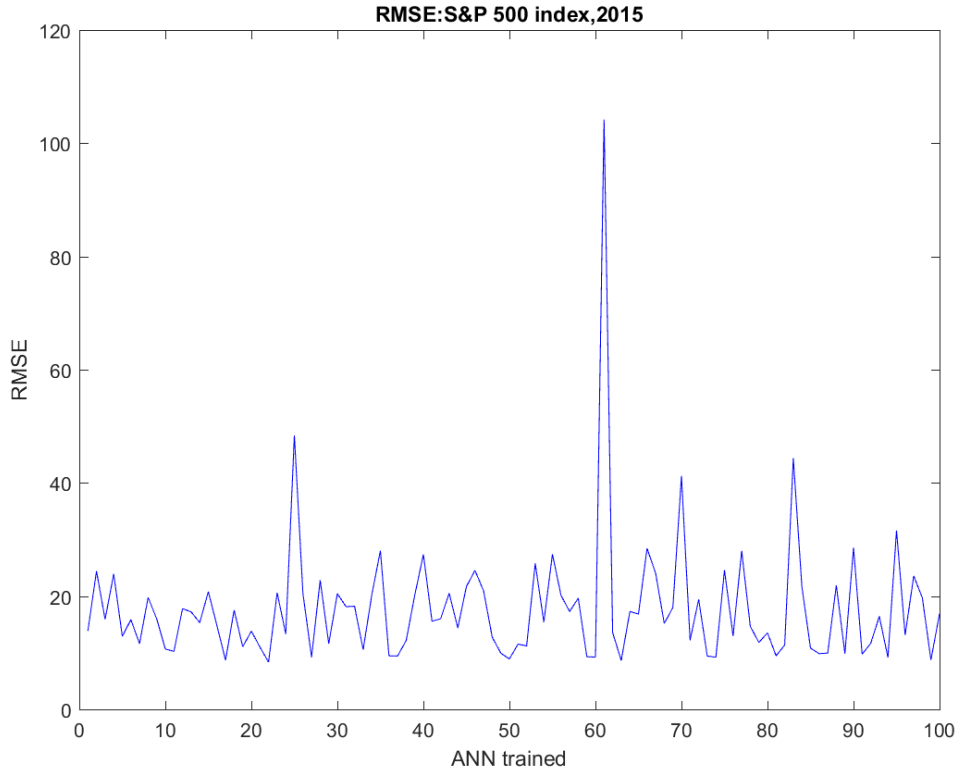

Fig. 71 - RMSE of the 100 ANNs trained

Fig. 69 shows the prediction curve vs the curve of actual prices. The predicted price, compared to the actual price, has a difference of at most 63$. The average RMSE is equal to 17.86 with a standard deviation of 11.15, meaning that on average the errors are relatively low. It reaches 100 but only in one peak. Moreover, the average accuracy is equal to 0.985 with a standard deviation of 0.013, meaning that on average the predicted prices differ from the actual ones by less than 2%. Finally, considering all the previous results and that the accuracy varies within a 7.4% range, it is possible to conclude that the price predicted by this test is reliable.

In the last step, the distribution of the training dataset is divided as follows:

- 48% of the training dataset for training;
- 48% of the training dataset for validation;
- 4% of the training dataset for testing.

Tables 32-33 and figures 72-74 show the results of this test.

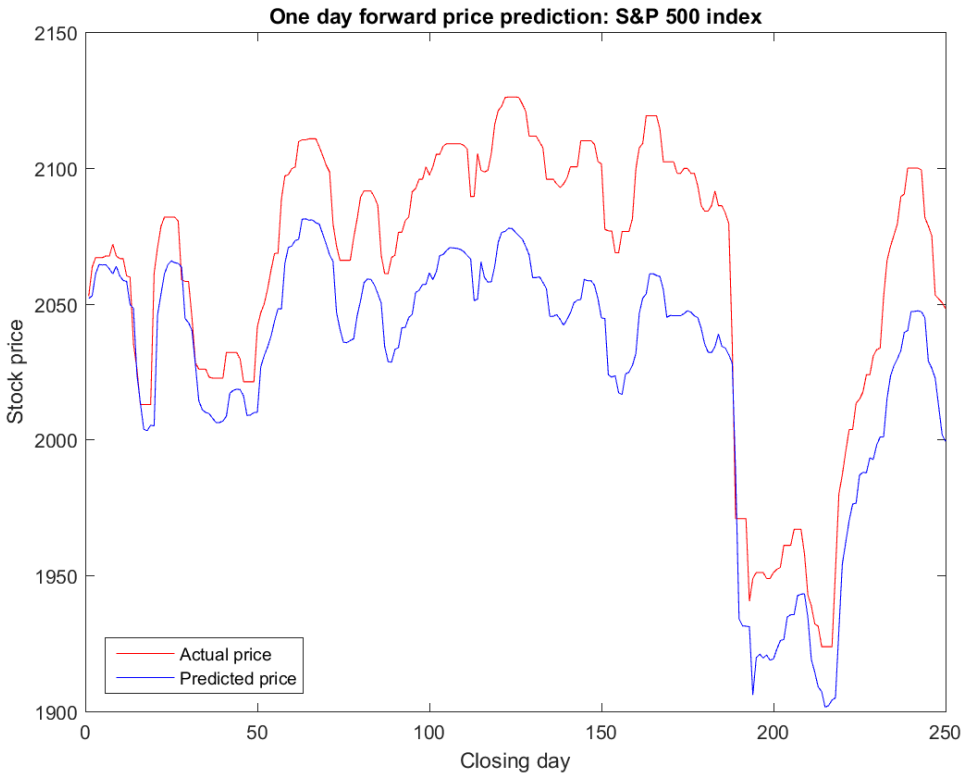

Fig. 72 - One-day ahead price prediction of S&P 500 index vs. actual data

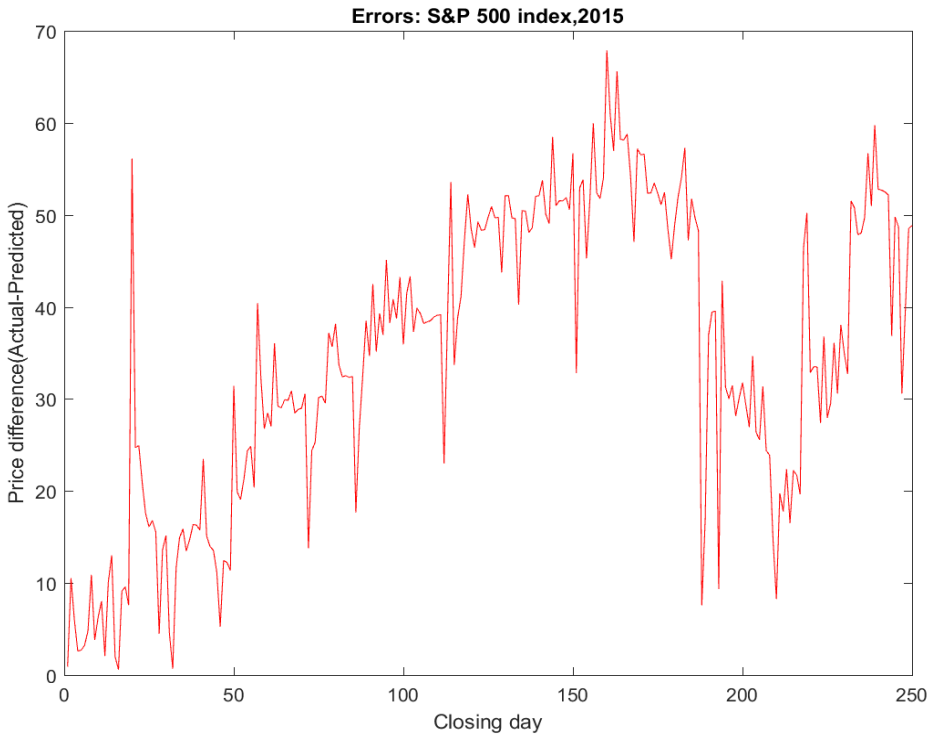

Fig. 73 - Difference between actual and predicted prices ($)

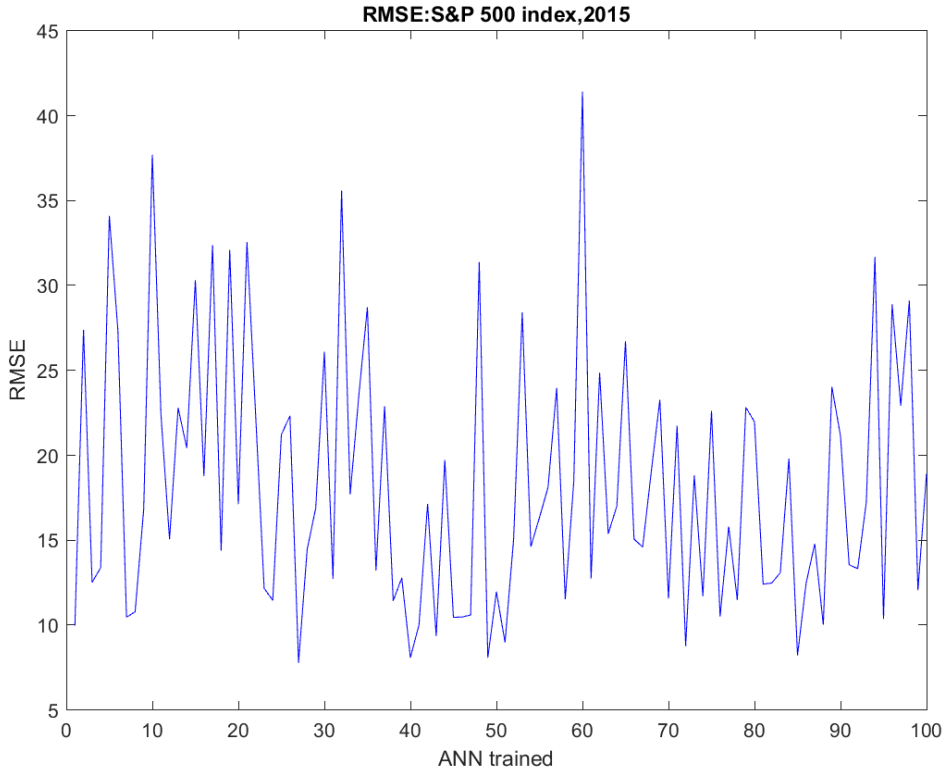

Fig. 74 - RMSE of the 100 ANNs trained

Fig. 72 shows a fairly aligned prediction curve. The predicted price, compared to the actual price, has a difference of at most 68$. Even though the errors increases over time, the curve of the actual price is similar to the one of predicted price, at least with respect to the prices rising and decreasing. Furthermore, the average RMSE is equal to 18.25, meaning that on average the errors are relatively low. The RMSE has a standard deviation equal to 7.67. Moreover, the average accuracy is equal to 0.983 with a standard deviation of 0.011, meaning that on average the predicted prices differ from the actual ones by less than 2%. Finally, considering all the previous results and that the accuracy varies within a 4.2% range, it is possible to conclude that the price predicted by this test is reliable.

Training set %	Validation set %	Testing set %	Accuracy	Standard deviation
34	33	33	0.984	0.014
90	5	5	0.982	0.013
5	90	5	0.976	0.031
5	5	90	0.980	0.043
4	48	48	0.972	0.041
48	4	48	0.985	0.013
48	48	4	0.983	0.011

Table 34 - Accuracy for each configuration

Training set %	Validation set %	Testing set %	RMSE	Standard deviation
34	33	33	18.36	8.86
90	5	5	19.63	8.47
5	90	5	28.83	16.61
5	5	90	30.22	14.82
4	48	48	31.52	17.02
48	4	48	17.86	11.55
48	48	4	18.25	7.67

Table 35 - RMSE for each configuration

Tables 34-35 show a summary of the results of the third test. Looking at these results, it is possible to see that the accuracy is above 98% and RMSE is relatively low when the training dataset is sufficiently large (i.e. 34%, 90%, 48%). Furthermore, when the training data has this configuration the accuracy and RMSE have a lower standard deviation than in the other cases. Thus, it is clear that there has to be enough data for the training set or the ANN will not perform at best.

The testing dataset is not used for training the ANN, but only to check its performance. Therefore, it has no effect on the performance of the ANN.

Finally, the relation between the training and validation data is investigated. When the validation dataset is bigger than the training dataset, the performance of the ANN in terms both RMSE and accuracy is poor. In the opposite case, the ANN has a better performance. When training and validation datasets have equal distribution, both good and poor performances can be seen. Considering the previous results, there seem to be no relation at all between training and validation sets. It is possible to conclude that only the distribution of the training set affects the performance of the ANN.

CHAPTER 6 CONCLUSIONS

"Technology periodically steals a leaf from nature's book."
Steven Harp and Tariq Samad

In this thesis the application of ANNs to the task of stock index prediction was divided into two main parts: selecting the relevant input variables and predicting the price of the S&P 500 index.
In the first part an ANN-based variable selection model developed by Edward Gately was described and tested with actual data. First of all, the final inputs found by Gatley were used to train an ANN and its performance was measured. Even though these input variables were selected in relation to the financial markets condition of 20 years ago, the results were reasonable.

Secondly, every step of Gatley's variable selection model was replicated and the results were particularly promising. After all the tests, I concluded that this model is able to provide input variables to predict the price of the S&P 500 index.

In the last test of the variable selection model the input variables with a low contribution to the network were deleted and an ANN was trained using these input variables. This ANN had a better performance that the Benchmark network, that was trained with the input variables provided by Gatley's variable selection model. Thus, ANN with a better generalization than the Benchmark network was found.

In the second part of this thesis study the application of ANNs to the prediction of S&P 500 index price was described. In this part, the influence of the training algorithm, the hidden layer size and the distribution of the training data to the accuracy of the ANN was investigated.

The results obtained with different training algorithms, hidden layer sizes and training dataset distribution led to the following conclusions. First of all, the ANN was trained with three different training algorithms. The Levenberg-Marquardt algorithm was found to be the most suitable algorithm to predict the price of the S&P 500 index.

Furthermore, the hidden layer size should be chosen in relation to the complexity of the problem. Indeed, it is required to retrain the ANN with different hidden layer sizes in order to find an ANN that has the best performance. The latter was reached with an hidden layer size set at 10 neurons.

Finally, the results obtained using different configurations of the training data lead to the conclusion that only the training set influences the performance of the ANN. Besides the distribution of the available training data, it is important to have enough training data otherwise the ANN will not be able generalize the problem.

The ANN in almost all its configurations performed well. The one-day ahead prediction of the S&P 500 index price was considered as reliable. Therefore, it can be used within one or two other technical indicators, such as the Bollinger Bands and the RSI, as an indicator for buying or selling the S&P 500 index.

Considering the results of the whole price prediction model, it is possible to conclude that the ANNs are an effective tool for financial time series forecasting and can be used on the real world data.

In a possible future work one could use a different architecture for the ANN, for example a recurrent neural network. This type of ANN permits cycles and loops in the network and it would thus allow for more complex learning.

Finally, one could increase the dimensionality of the training dataset, that is the number of inputs variables, and compare the results with the ones of lower dimensionality datasets.

REFERENCES

Kar, Abhishek (2010). Stock prediction using ANNs, Dept. of Computer Science and Engineering, IIT, Kanpur.

Stark, A. (2010). A study of business decisions under Uncertainty: The Probability of Improbable. Dissertation.Com.

Engelbrecht, A. P. (2007). Computational Intelligence: An Introduction, 2nd edition, John Wiley & Sons.

Brabazon A. & O'Neill M., (2006). Biologically inspired algorithms for financial trading. Springer.

Ferrari, Antonella (1996). Aspetti applicativi delle reti neurali artificiali. Franco Angeli.

Refenes, Apostolos-Paul N. (1995). Neural Networks in the Capital Markets. Wiley.

Choen, B. (1997). The edge of chaos. Financial Booms, Bubbles, Crashes and Chaos. Wiley.

Vanstone B. & Finnie G. (2008). An empirical methodology for developing stockmarket trading systems using ANNs, Expert Systems with Applications, vol. 36, 6668-6680.

Malkiel, Burton G. (2003). The Efficient Market Hypothesis and Its Critics, Journal of Economic Perspective, vol. 17, 59-82.

Jones, Charles P. (1998). Investments: Analysis and Management 6th edition. John Wiley & Sons.

Baily D. & Thompson D.M. (1990). Developing neural networks applications, AI Expert, Vol. 5, 38-47.

Rumelhart, D. E. et al. (1986). Parallel distributed processing: explorations in the microstructure of cognition, vol. 1, 318-362.

Enke D. et al. (2005). The use of data mining and neural networks or forecasting stock markets returns, Expert Systems with applications, vol. 29, 927-940.

Floreano, D. (1996). Manuale sulle reti neurali. Il Mulino.

Mercadal, D. (1990). Dictionary of Artificial Intelligence. Van Nostrand Reinhold Computer.

Michael A. E. (1994). Neural network time series forecasting of financial markets. Wiley.

Peters, E. E. (1991). Chaos and Order in the Capital Markets. Wiley.

Peters, E. E. (1994). Fractal Market Analysis: Applying Chaos Theory to Investment and Economics. Wiley.

Peters, E. E. (1999). Patterns in the Dark: Understanding Risk and Financial Crisis with Complexity Theory. Wiley.

Peters E. E. (2001). Complexity, Risk, and Financial Markets. Wiley.

Gately, E. (1996). Neural networks for financial forecasting. Wiley.

Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work, The Journal of Finance, Vol. 25, No.2, 383-417.

Fabbri G. & Orsini R. (1993). Reti neurali per le scienze economiche. Franco Muzzio Editore.

Luger, G. F. (2009). Artificial Intelligence. Pearson.

Zhang G. et al. (1998). Forecasting with ANNs: the state of art, international journal of forecasting, vol. 14, 35-62.

Baker G. L. & Gollub J. P. (1990). Chaotic Dynamics: An Introduction. Cambridge University Press.

Deboeck, G. J. (1994). Trading on the edge: neural, genetic and fuzzy systems for chaotic and financial markets. Wiley.

Yu H. & Wilamowski B. M. (2010). Levenberg-Marquardt Training, Intelligent Systems, Vol. 12, 2-12.

Zhang, H. (2010). Efficient market Hypothesis and Technical analysis.

Kaastra I. & Boyd M. (1996). Designing a neural network for forecasting financial and economic time series, Neurocomputing, vol. 10, 215-236.

Moody J.E. & Darken C. (1989). Learning with Localized Receptive Fields. In D. Touretzky, G. Hinton, and T. Sejnowski editors, Proceedings of the Connectionist Models Summer School, 133-143.

Katz, J.O. (1992). Developing neural networks forecasters for trading, Stocks and Commodities, Vol. 10, 160-168.

Lorie J. H. et al. (1985). The Stock Market: Theories and Evidence. Irwin Professional Publishing.

Klaussen K.L. & Uhrig J.W. (1992). Cash soybean prediction with neural networks, Conf. on Applied Commodity Analysis, Price Forecasting, and Market Risk Management Proc., 58-70.

Cunningham, L. A. (1994). Capital Market Theory, Mandatory Disclosure, and Price Discovery. Washington and Lee Law Review.

Patel, M. B. et al. (2014). Stock Price Prediction Using ANN, International Journal of Innovative Research in Science Engineering and Technology, vol.3, 13755-13762.

Bale, M. H. et al. (2015). Neural Netwrok Toolbox User's Guide.

Roe, M. J. (1996). Chaos and evolution in Law and Economics, Harvard L. Rev., 641-641

Pendharkar, P. C. (2002). A computational study on the performance of ANNs under changing structural design and data distribution, European Journal of Operational Research, vol 138, 155-177.

Glen, P. J. (2005). The Efficient Capital Market Hypothesis, Chaos Theory, and the Insider Filing Requirements of the Securities Exchange Act of 1934: The Predictive Power of Form 4 Filings, vol. 11, article 4, 85-98.

McNelis P. D. (2005). Neural Networks in finance: gaining predictive edge in the market. Academic Press.

Ball R. (1994). The Theory of Stock Market Efficiency: Accomplishments and Limitations, Managerial Finance, Vol. 30, 3-48.

Ball R. (2010). The Global Financial Crisis and the Efficient Market Hypothesis, 2010 General meeting.

Su et al. (2010). Reelin is required for class specific retinogeniculate targeting. Journal of Neuroscience, vol. 31, 578-586.

Masters, T. (1993). Practical Neural Network Recipes in C++. Morgan Kaufmann.

Hazen, T. L. (1991). The short-term/long-term dichotomy and investment theory, N.C. L. REV., 137-157.

Zhang Y. & Wu L. (2009). Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network, Expert Systems with Applications, vol. 36, 8849-8854.