Università
Ca'Foscari
Venezia

Master's Degreee Programme –
Second Cycle (D.M. 270/2004)
in Economics and Finance

Final Thesis

Ca' Foscari
Dorsoduro 3246
30123 Venezia

A Simple Trading Strategy based on an Artificial Neural Network

Supervisor
Ch. Prof. Marco Corazza

Graduand
Enrico Brombin
Matriculation Number 842777

Academic Year
2016 / 2017

I would like to thank Professor Corazza, supervisor of this thesis, for the enthusiasm and support shown me during the writing period. Without you, this work would not have come to life.

A big thanks goes to my family who, with their support, both moral and economic, have allowed me to be here, contributing to my personal development. I would also like to thank my father, whose memory constantly strengthens and motivates me.

Finally, I thank my friends, in particular, Chiara, Cecilia, Andrea, Tommaso, Marco, Davide and Marco, who have always listened to me and supported me during my studies.

Table of Contents

Introduction

Uncertainty, or risk, is something that has always characterized the world's financial markets: stock market crashes, speculative bubbles, financial crises, unexpected recovery, irrational fears, the so-called black swan, and so on. Therefore, those who act in the markets have always been looking for a method to minimize the risk of losses while, possibly, maximizing the performance of their investments. Market operators have been studying artificial intelligence, and especially Artificial Neural Network (ANN), for minimizing financial risks, for more than twenty years.

In this thesis, we wanted, therefore, to assess whether there is the possibility that market operators effectively use ANNs to make investment decisions. In detail, we examined two mutual funds, such as market operators, we analyzed part of their composition and using, as net inputs, various Technical Analysis indicators based on the historical opening, closing, maximum and minimum prices of the funds' shares and, as target outputs, the historical returns of the shares, we built and trained an ANN. Subsequently, by presenting new inputs never seen before to the ANN, we tested its predictive capabilities and, using a simple trading strategy to simulate a market agent, we tested its performance and then compared them with the funds' ones.

The thesis is composed of three chapters: the first chapter describes the reasons that led us to use, as network inputs, indicators based on Technical Analysis and we discuss its validity also comparing it to the Fundamental's one; in the second chapter, instead, we wanted to focus on the theory of ANNs, we described the theories behind them, their nature and structure and, finally, we explained different "historical" models which form the basis of the most recent networks; in the third and last chapter, on the other hand, we focused on the Technical Analysis indicators we used, the shares object of our analysis, the structure and the architecture of the ANN we have constructed and, finally, we presented and critically explained the results obtained.

Chapter 1

The Theory behind the trading system

This paper describes the construction of a trading system based on an ANN. Therefore, we want to assess the significance of the theory which our trading system is based on, that is, Technical Analysis. Among the traders operating in financial markets, in fact, there are two main schools of thought, referred to Technical and Fundamental Analysis. Thus, in order to carry out this valuation, in this chapter both the theories will be described and compared.

1.1 Introduction

A pillar of financial economics is represented by the Efficient Market Hypothesis. The first theoretical developments are due to Samuelson and Mandelbrot in the 60s. An efficient market is a market in which prices reflect all the publicly available information. This is equivalent to saying that agents, in making their decisions, behave in a rational way, that is, they use all the information available at the moment in which the choice is made. If current prices reflect all available information, their variations can depend only on new events that are not known and, thus, unpredictable. For this reason, it is not possible to make accurate price's predictions, that means, there is not any market operator able to obtain particularly high profits from stocks trading.

The main implication of this theory is that financial markets are structurally stable, meaning that supply and demand are in equilibrium determining the price. For the market agents that accept this theory, the instruments used to analyze the markets and their evolution are referred to Fundamental analysis.

The framework described by the efficient market hypothesis on very rigorous assumptions, but not always a high analytical rigor corresponds to an equally high capacity to explain the reality. Considering only the last decades, we can count more than one financial crisis with global consequences. Too much to consider valid the theory of general stability of the market. When something happens with a certain regularity, at least it is suspected that it is not a random phenomenon, but rather something due to intrinsic mechanisms of the markets. Moreover, the assumption that agents behave in a rational and perfectly informed way is quite unrealistic. Human beings are prone to behave as a herd: this refers to the fact that when there are not sufficient elements to make rational decisions, we tend to conform to the behavior of other people.

In addition, according to Shiller, an important researcher of behavioral finance, operators make investment decisions based on past experiences. Thus, in order to recognize particular market situations, trends and anomalies, already happened in the past and properly evaluate them, operators developed different investigation procedures that are essentially referred to Technical Analysis.
In this chapter we thus describe and compare Technical Analysis and Fundamental one.

1.2 Technical Analysis

The study of Technical analysis started with Charles Dow, which developed the first American stock market index in 1884. He started monitoring the assets' prices through their charts, noticing that the prices reflected an information before it became publicly available. After several years, he developed a theory on the movement of stock prices that, among other things, affirms that prices move along recognizable paths. The set of articles that Dow wrote for the Wall Street Journal became known as the "Dow's Theory" and it constitutes the cornerstone which the actual Technical Analysis is based on. One of the most important concepts highlighted by Dow is that the price of an asset reflects the current balance between supply and demand, and that the distortions in the prices required by the two divisions of the market determine the positive and negative trends, recognizable through the charts. Then in the successive years and after World War II, other contributions to technical analysis came from other researchers (such as Bill Jiler, Alexander Wheelan and John Magee) concerning not only the charts used to represent trends, but also in different techniques to analyze price reactions. However, the most important step ahead made in technical analysis took place at the end of 1970s with the introduction of computers. This made it possible to develop many indicators helping to recognize trends through the study of price and volume movements. Technical Analysis includes a dual approach to the market: statistical and heuristic (heuristic refers to the fact that it is a technique which does not guarantee a perfect result, but it is sufficiently reliable for its purposes). It is the systematic study of the prices' behavior of assets in financial markets, mainly through graphical methods, with the aim of predicting their future trends. Despite a graphic may appear subject to purely objective valuation, using technical analysis there are significant references to stimulating aspects that, surprisingly, depend on who is studying the graph. In fact, as we can see from Figure 1.1, there could be different representations of the actual situation of an instrument.

Figure 1.1: Candlestick Chart with Trend lines of FTSE MIB index

Source: Personal Representation, based on data from www.prorealtime.com

In this case, looking at the last one hundred observations of the Italian market index FTSE MIB (30/06/2017 – 17/11/2017), a researcher could concludes that there is a positive trend (green line), another one could states that there is a negative trend (red line) and finally another one could asserts that there is not any trend (blue line). The most fascinating aspect of Technical Analysis consists precisely in the ability to understand the most interesting aspects of each instrument and to check their effectiveness in trading practices on financial markets. Research on Technical Analysis became increasingly important in parallel with the development of transactions on organized markets mainly due to globalization. The interest in the discipline is increased in the last decades primarily because of the difficulty of the classic theory to explain certain price movements, and even because of its relative simplicity of application.

According to a widespread opinion among financial operators, the price chart of an instrument contains most of the information useful to design its future evolution: this assumption is due to the fact that the price level at each timeframe is always an expression of the intersection between the supply and demand of that instrument.

The analysis of the price chart was the first expression of Technical Analysis and, since it still demonstrates a notable effectiveness, it can be considered the first step of each Technical Analyst approaching for the first time a new asset. As we previously said, Technical Analysis also uses statistical tools: this approach is essentially based on the signals provided by the interaction between different algorithmic and statistical indicators. Usually, statistical and chart analysis are used simultaneously, that means that they are conducted jointly, valuating and studying the chart and studying some statistical indicators in order to determine the presence of specific financial phenomena. Then, prices and volumes, as well as the quantities derived from basic computations and elaborations of these data, are collected in time series and studied mainly in graphic form. These charts are finally investigated with the aim of identifying recurrent configurations and particular sets of patterns in order to facilitate the recognition of the beginnings or ends of economic cycles and market trends. This type of analysis is therefore performed through the elaboration and the study of prices and volumes only, by graphs and mathematical models, essentially designed to measure the strength and weakness of the price movements, thus of the trends. The assumptions on which Technical Analysis is based can be summarized substantially in three features of the market:

1. The market incorporate all the information. This sentence means that the price of any instrument reflect all the information; thus any relevant variable, whether it is purely economic, monetary or political, when it comes out it is immediately reflected in the market price. Nobody cares about the determinant factors which led to that particular price: it is accepted as a perfect resume of all those information concerning the instrument and the linked listed business. Certainly, there are a wide variety of reasons for price increases and decreases, but, in order to understand and identify the direction of the market, since each of these factors is reflected in price movements and volumes, it is not necessary to know them.
2. History tends to repeat itself. This second assumption has a psychological nature, because it is based on the belief that, with identical stimuli given in different timeframe, individuals will respond with a similar and unchanged reaction and behavior[1].

[1] A. W. Lo et al, "*Foundations of Technical Analysis: Computational Algorithms, Statistical inference, and Empirical Implementation*", 2000, The Journal of Finance vol. 55 n. 4.

Thus, this means that, through the study of previous market turning points, it is possible to highlight some details useful to recognize important minimums and maximums in the price's time series. Human behavior, and hence the stock market trend, is extremely complex and thus it never repeats exactly in the same way; but the recurrence of similar of patterns is enough to allow the expert to identify the main critical points. This basically means that past price trends have some chance of repeating themselves in the future, and the price movements tend to reflect the psychological reactions of market participants.

3. An existing trend tends to remain in place. This means that a price movement characterized by a well-defined directionality is likely to follow the same path until a reversal signal occurs. One of the most important functions of the technical analysis is therefore to identify, from the very first stage of its development, the trend that characterizes the market in general and individual operations in particular. The identification of market trends is therefore the main objective of heuristic techniques, with the defined premise that the price trend of any instrument always tends to maintain substantially well-defined guidelines.

Some researchers, however, asserted that there is a certain inconsistency between what is contained in the first assumption and the other two conditions. Indeed, if the market were able to independently assess a financial asset taking into account all the information available at time t, then the likelihood that the price follows trends occurred in the past would be already reflected in the actual price, thus making recurring behaviors impossible. Essentially, they stated that if the price reflects all the information, it is not possible that a past behavior in the price could reoccur in the future, meaning that there cannot be any recognizable patterns in the price.

As already stated, the tools used by Technical Analysis are designed to assess and interpret the dynamics of the financial markets with the aim of predicting their future evolution. The expected evolution of the price of an instrument reflects two basic components. The first is certainly related to the effect that some macroeconomic variables (for instance, interest rates and inflation rate) have on the market price, while the second component concerns the correct interpretation of the behavior of active traders on financial markets. These agents are much more interested in the actual market price of an instrument than their intrinsic, fair value.

This operator's behavior has the effect of making the influence of the psychological component on the market price, in its various forms, far greater than the one exercised by the variables that should affect the intrinsic, objective value of an asset.

The Technical Analysis thus consists of a series of tools and methods, in constant evolution and expansion, which allow to study and interpret market dynamics based not only on the analysis of charts and statistical indicator but on a behavioral analysis too.

Thinking about this, some researchers stated that there seems to be an intrinsic weakness of Technical Analysis. If, in fact, all operators blindly follow the Technical Analysis, it may fall into the Keynesian paradox (beauty contest), that means that the market participants will not buy (or sell) the assets they think to be increasing in value (or decreasing) basing their decision on the available information, but they will buy (or sell) those assets they think others consider imminent to increase in value (or decrease)[2]. This way of thinking and acting, could inevitably lead to irrational behaviors in the markets, such as speculation[3]. However, Technical Analysis recognize that there is a Fundamental Analysis, meaning that in the market there are operators that act in the best way to be informed of all relevant information and events. Even the Technical Analyst will try to be informed of the most relevant information concerning the Company's stocks he is planning to trade. Thus, on the other hand, it is implicitly assumed that the Fundamental Analysis cannot ignore the Technical one. Indeed, if it is true that the individual behavior of an operator depends on the other one's behavior, even a Fundamental Analyst, in the preparation of his strategy, will not be able to avoid to try to identify the counterparty reaction, from the observation of the market data.

1.3 Fundamental Analysis

It is now necessary to define what is meant by Fundamental Analysis and on which assumptions it is based. The birth of Fundamental Analysis is usually dated back to 1934, when, just a few years after the Wall Street collapse, Benjamin Graham and David Dodd published their book "Security Analysis".

---

[2] S. Ward, M. Sherald, "*Successful Trading Using Artificial Intelligence*", 2006, Ward Systems Group Inc.

[3] Here, speculation is referred to situation in which an agent enters in the market just to benefit from a temporarily and very rapid increase (or decrease) of an asset's price, just because the entirety of the market operators is acting in the same way.

In this book, the authors proposed a clear definition of investment, which strongly differs from what was considered speculation. In fact, Graham considered a financial operation as an investment only if, after a precise analysis, it guaranteed security for the invested capital and promised an adequate return. Everything that did not have these characteristics was considered speculation.

From that moment, a large amount of researches has focused on specific economic variables as key determinants of assets' prices, leading to what today is known as Fundamental Analysis. Operating on the financial market through this approach means deciding to buy or sell an asset based on the eventual discrepancy between the current price and its true, fair value. It is therefore implicit to compute the theoretical value of an asset which will be used to compare, and thus analyze, the magnitude of the deviation between this value and the quotation expressed by the market. In fact, the Fundamental Analysis does not consist in the analysis of the exchange volumes or recurrent pattern in the price's time series, but rather in the analysis of the object traded, that is, the studies are not on the price but on the factors that determine that price. A listed good has its own actual value and Fundamental Analysts aim to determine it. Practically, the market tends to express this fair price as quickly as possible and, if analysts detect distortions, they will have to take consistent decisions, thus taking a long position if the asset is considered undervalued and a short position if the asset is considered overvalued.

The Fundamental Analysis process is divided mainly into two levels of generality: first, there is an economic analysis, a kind of structural analysis that examines the macroeconomic environment. Secondly, there is a particular analysis, which consists in a sectorial and in a business analysis, aimed to identify the company's economic and financial potential in order to make a correct estimation of the income flows associated with that particular asset.

By using complex quantitative models, analysts are able to interpret and describe the particular relationships existing between the economic variables that affect the price. While these macroeconomic models are necessary for governments in order to define their economic policies, they are less useful to the daily market operations of an agent. In fact, such models have some limitations and drawbacks:

- The significance of the data depends on a constant and accurate review, with consequent high costs.

- There is a huge amount of parameters, thus making them difficult to manage, and since they vary over time, the results of the models are heavily affected by small variations of these parameter.

- Usually a model is structured for a specific asset, or class of assets, and for a specific time horizon, thus making difficult its applicability to other markets or other periods.

- Data processing usually take long time, making difficult for the model to provide timely indications.

It is not possible, however, to take decisions based exclusively on the results provided by these type of models.
Anyways, they provide an initial knowledge of the possible effects of national and international political and economic events on certain linked macroeconomic variables which determine the asset valuation process.
As we said, the second analysis, the particular one, is aimed to understand the economic and financial potential of the company.
First, the company's market segment is considered, in order to provide an estimate of the company's level of competitiveness. In this step, the analysis is focused on the relationship between supply and demand, as well as the level of competition within the industry, the costs and the life cycle of the company's goods, the possible presence of seasonality in the production and the possibility to be affected by national or foreign regulations. The second step, finally, consists in the specific analysis of the company, thus considering the dynamic study of the balance sheet of several successive years. In fact, through historical comparisons and prospective evaluations and comparisons with the competitors, this analysis aims to investigate the abilities of the company to produce future earnings, to maintain a good balance between assets and liabilities and to maintain its capital over time.
Finally, for operational purposes, the Fundamental Analysis resort to different and very useful tools and indicators based on market data and data from the financial statements.

1.4 Technical versus Fundamental Analysis

In the previous paragraph, the concepts of Fundamental and Technical analysis have been defined. In this paragraph, instead, we propose a brief comparison between the two analysis in order to better understand them and to properly assess the bases on which the information is provided to the operator.

1.4.1 Assumptions

Technical Analysis assumes that the price is the ideal synthesis of all information, without importance whether they are known or not.

According to the Fundamental Analysis, instead, the market price is the result of many economic and financial factors that need to be identified and evaluated. From this underlying hypothesis, many conflicting arguments arise.

Since it is based on a broader set of publicly available information and it does not consider only the historical price's time series, the Fundamental Analysis would be preferred to Technical Analysis. Fundamental Analysis is thus considered, for all intents and purposes, consistent with the three levels of information efficiency of financial markets:

1. In the weak market efficiency, prices reflect all the information contained in the historical price's time series. It is not possible, then, to develop a trading strategy with an expected return higher than the market return, based only on past price information.

2. In the semi-strong market efficiency, prices reflect all the information contained in the historical price's time series and any other publicly available information. Here, again, it is not possible to develop a trading strategy with an expected return higher than the market return based information that everybody knows.

3. In the strong market efficiency, prices reflect the information contained in the historical price's time series, any other publicly available information and any private information[4].

[4] E. F. Fama, "*Efficient Capital Markets: A Review of Theory and Empirical Work*", 1970, The Journal of Finance vol. 25 n. 2, p. 383-417.

The opposite theory, then, states that if all the information that may be relevant are always reflected in the price, then the Technical Analysis implicitly contains the Fundamental Analysis. The price, thus, would be such a fast and efficient indicator able to anticipate all the fundamental information. Then, it would not be possible to identify market trends based on Fundamental Analysis' variables, as they would be known too late to the agent and they could not consider other important variables affecting the market price.

As we already asserted, however, this view is clearly incomplete because, if the market would be characterized by such a strong information efficiency, the theory does not explain how some typical patterns recognized by Technical Analysis can occur. Finally, there is a third theory that can be placed between the previously described ones because it re-evaluate the utility of both Technical and Fundamental Analysis. In fact, this theory consider that the value of an asset is affected by both short-term and long-term factors. The formers propagate rapidly causing sudden and short-term changes in the market price. The latters, identified by Fundamental Analysis, carry out their influence on prices very slowly, establishing their long-term trend[5]. Consequently, the information set is divided into two different classes. It is, thus, not certain that in its evaluations the Fundamental Analyst can identify the short-term factors, especially from a timing point of view. This incompleteness recognize, then, the value and importance of the technical approach to the short-term movements of market prices.

1.4.2 Purposes

Technical Analysis attempts to identify the reversal points in a price trend, in order to estimate the best moment to enter or to leave the market. Moreover, it aims to recognize also the recurrence of significant historical price movements. Fundamental Analysis, instead, seeks to identify the intrinsic, fair value of an asset which, then, will be constantly compared with the current market price in an attempt to capture those discrepancies suggesting to buy or to sell the asset.

Sometimes, however, the discrepancies between the market price of an asset and its fair value could be so larger and volatile to be explained by valuation errors.

"*One share of United States Steel, for example, was worth $261 in the early fall of 1929, but you could buy it for only $22 in June of 1932! By March 1937, it was selling for $126 and just 1 year later for $38.*

[5] E. Coliva, L. Galati, "*Analisi Tecnica Finanziaria*", 1993, UTET.

In May of 1946, it had climbed back up to $97, and 10 months later, in 1947, had dropped below $70. […] The book value of this share of U.S. Steel, according to the corporation's balance sheet, was about $204 in 1929 (end of the year); $187 in 1932; $151 in 1937; $117 in 1938, and $142 in 1946"[6].

In the light of this, what appear clear is that the aim of Fundamental Analysis is attenuated as, essentially, its ability to identify and evaluate each factor vanishes. This example, reported by Edwards, declares the superiority of the market price as a perfect resume of all the variables, even irrational.

1.4.3 Instruments

Any market analyst who rely on one of the two approach may recall episodes in which his conclusions were clearly mismatching the market trend. However, there is a significant difference between the two theories.

Technical Analysis, in fact, is a "self-correcting" process, that is the operators recognize and learn from their own mistakes, using them to refine the predictive capacity of their tools. The Technical Analysis systems, indeed, are constantly evolving and they are systematically and periodically tested.

On the other hand, the Fundamental Analyst acts in a much more dangerous way: assumes that, if an agent concludes that the fair value of an asset is $100 while its market price is $80 then, by definition, the agent will become even more bullish[7] when the market price, for instance, will drop to $70. Thus, this behavior could lead him to suffer huge losses, depending on the level of confidence he has for his valuation.

From this distinctive characteristic of Fundamental Analysis, we can deduce that this approach requires a high degree of specialization, that is, the Fundamental Analyst must have a very good sectorial knowledge of the asset treated, as well as a huge amount of information to process.

These show the limits of Fundamental Analysis, its lack of flexibility and adaptability. The same analysis, in fact, cannot be applied to any instrument.

---

[6] R. D. Edwards et al., "*Technical Analysis of Stock Trends*", 9th edition, 2007, CRC Press.

[7] Investors who believe that a stock price will increase over time are said to be bullish. Investors who believe that a stock price will decline, instead, are said to be bearish.

For example, the analysis of a stock consists mainly in the valuation of the company's balance sheet and of its economic and financial structure, while the analysis of a currency's exchange rate concerns the investigation of both the countries' balance of payments, the public debt and the inflation rate. The strength of the Technical Analysis is, thus, its extreme flexibility: the same tools can be used to analyze very different markets, such as the stock, bond, future, currency or commodity markets, with consequently lower costs compared to the fundamental analysis.

1.5 Conclusions

As previously stated, the cornerstone of Technical Analysis is the reading and interpretation of charts that describe the market's behavior. Depending on the correct reading of the chart, it could be perceived what have been the effects of the past behavior of the traders operating in the market on the supply and demand and, thus, on the asset price. Since these behaviors will tend to repeat in a similar way in the future, probable scenarios of future market evolution can be estimated.

Even if an analyst classifies itself as strictly technical, however, it is good to remember that having an overview of both Technical and Fundamental Analysis is in general advisable in order to operate profitably on financial markets. Technical Analysis has in fact the advantage to be flexible and adaptable to any type of market; it focuses exclusively on those markets characterized by clearly recognizable trends and it moves from these to other markets if the directionality is not clear. The Fundamental Analyst, instead, tends to specialize himself on a particular market; indeed, he can change the asset under analysis only at the expense of highly time-consuming new analysis. Nevertheless, since Technical Analysis should be used with a short-term horizon and the Fundamental one should be used with a long-term horizon, when operators have to decide when to enter in the market, the two theories must be considered complementary and not mutually exclusive. In this context, two moments are distinguished: selection and timing. Fundamental Analysis is, in fact, used to determine the quality of the assets being the objects of the investment, while Technical Analysis, on the other hand, provides signals indicating the best moment to enter in the market.

In order to implement a rational investment policy in every market, it is therefore necessary to resort to both fundamental and technical analyses: the former to assess the asset in a long-term perspective and the latter to act in the most profitable moment.

However, since we are interested to apply a short-term investment strategy, in the training phase of the ANN, only Technical Analysis' indicators will be considered.
A complete and detailed description of the technical indicators and the methods used to train the network will be given in the third chapter.

Chapter 2
Artificial Neural Networks

2.1 Introduction

An Artificial Neural Network (ANN) is a quantitative model based on the so-called artificial intelligence, inspired by the structure and functioning of the human brain. Such a model uses a computational connectivity approach: it is made up of a group of interconnected unit exchanging information between each other in order to produce an output.

An ANN receives external signals through a layer of input nodes, each of which is connected to a number of internal nodes, organized in multiple layers. Each node, then, processes the received signals and transmits the result to subsequent nodes until the output node.

In most cases, an ANN is an adaptive system that changes its structure based on external or internal information flowing through the network during a learning phase.

The learning phase, carried out through different techniques that will be described later, consists in the adjustment of the weighted connections between different nodes, in response to external stimuli.

It is useful to premise that this process occurs following the calculation of the error committed by the network, consisting in the difference between a desired output and the actual output provided by the network. Once the weights have been updated, the error is computed again until the values of the weights converge to an optimal solution, ideally when the error is zero.

In practical terms, an ANN is not programmed to execute a certain activity, but it is trained to execute it, providing it with a series of examples of the reality it should model.

They can be used to recognize and simulate complex relationships between inputs and outputs that other approaches cannot represent or even recognize. For these reasons, ANNs attracted the interest of many researchers and technicians in several disciplines such as Psychology, Medicine, Finance, Engineering, Informatics and so on; in fact, applications can vary from speech or image recognition to pattern recognition in time series analysis.

2.2 Biological Neural Network

Since, as we said, ANNs are inspired by the structure of the human brain we will briefly describe how a natural nervous system works.

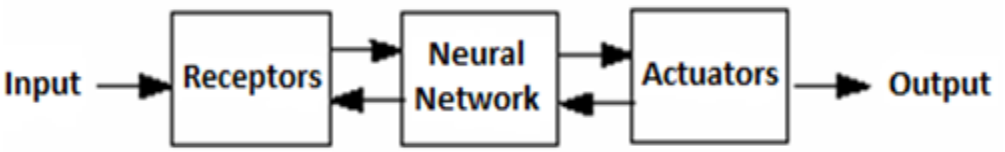
The human nervous system can be described as a 3-stage system:

1. Receptors, which convert external stimuli into electrical impulses that are sent to the neural network;

2. Neural network, that constantly receives information, understands and makes appropriate decisions;

3. Actuators, which convert electrical impulses generated by the neural network into responses to the external system.

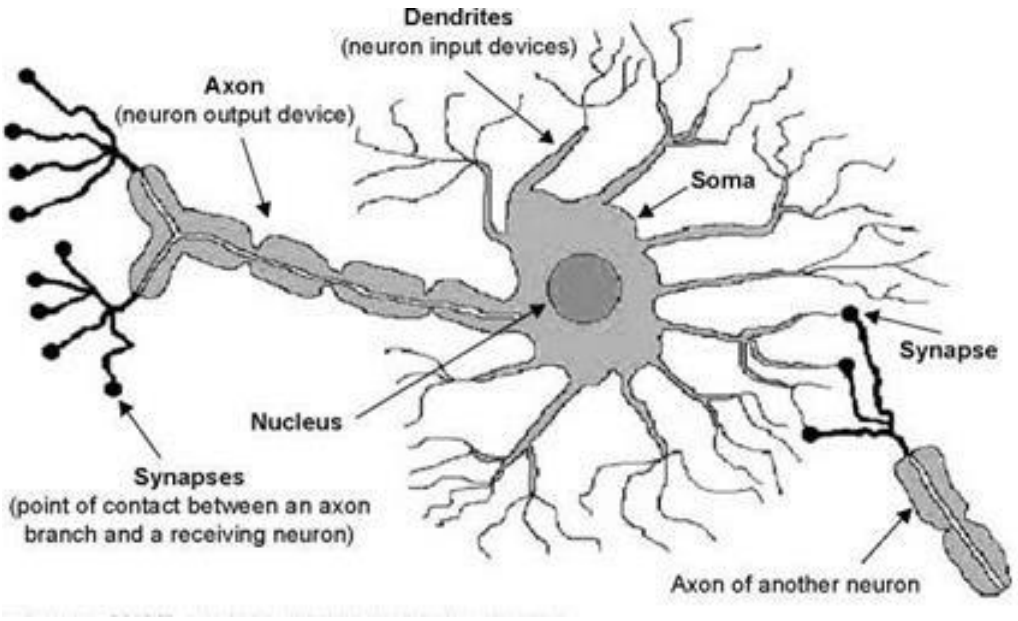
Figure 2.1: Block diagram of the nervous system



Source: S. R. Bulò, "Appunti di Reti Neurali"

The human brain in made up of a number that varies between 10^{10} and 10^{12} neurons interconnected between each other; the number of these connections is estimated to be 10^4 for each neuron. The neuron is essentially composed by the cell body or "soma" that is the computational unit, the axon that is the actuator or the sending-signals unit (and that end with the synapse) and multiple dendrites which are the receptor units. In Figure 2.2 there is an illustration of a biological neuron. The soma, which takes care of the nourishment necessary to the functioning of the cell, receives signals, through the dendrites, from thousands of contact points with other neurons, called synapses. When the received signals are strong enough, that is, they are higher than a certain threshold, the neuron is activated, it emits a signal through the axon and thanks to the synapse it is transmitted to another neuron.

Figure 2.2: Structure of a Biological Neuron



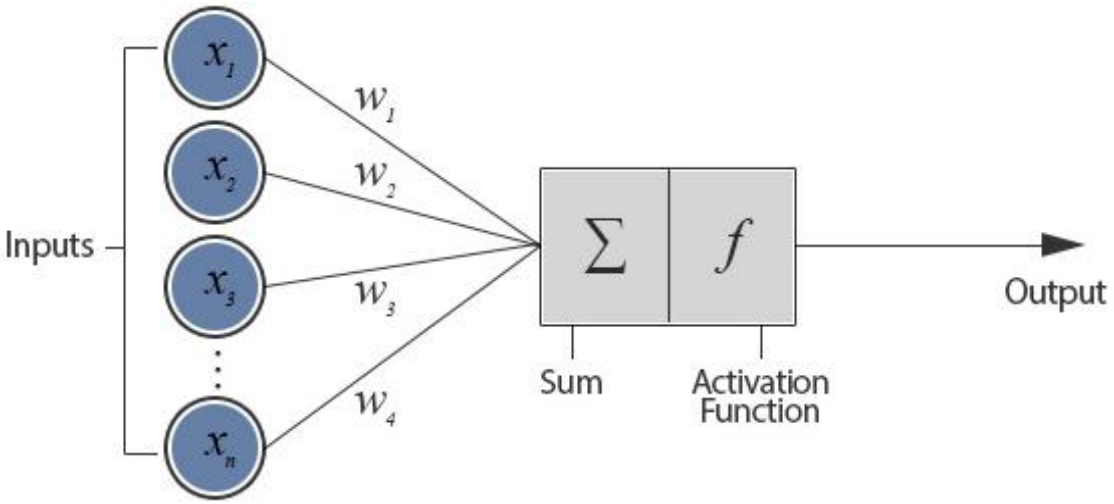
Source: www.energyprofessionalsymposium.com

2.3 Artificial Neural Network

An artificial neuron is nothing more than a computational model inspired by the mechanics of the biological neuron in the human brain, certainly the most efficient product of the evolution. As previously stated, in the human brain a dense network of interconnected neurons ensures its functionality, varying their configuration based on different signals that can come from the external environment. Since this capacity can be fully considered in terms of learning, researchers have been intrigued of this peculiarity and started to reproduce the mechanisms of the biological network.

As we can argue from Figure 2.3, the central body receives inputs from the external context or other units, they are multiplied by some given weights (meant as the strength of the signals) and then the network sums the weighted signals. In order to activate the output neuron, and thus to produce the final output of the network, it is necessary to compare the sum of the weighted inputs with a certain threshold previously determined. This comparison is carried out by a so-called activation function, which will be described later. Then, if the sum of the weighted inputs is equal or higher than the threshold, the activation function, as the name suggests, activates the output neuron. ANNs therefore combine artificial neurons to process information.

Figure 2.3: Structure of a Simple ANN



Source: www.theprojectspot.com

The troublesome part of this process is the determination of the right weights to assign at each input. Since we cannot know a priori the correct weights, when we are faced with hundreds or thousands of neurons we usually set randomly the initial weights and then we use algorithms that help to adjust these measures in order to obtain the desired output. This is what is meant when we say that the network learns.

2.4 Brief resume of the literature

The beginning of the studies on neural networks can be dated back to the first attempts to translate the principles of biological processing into mathematical models. The most ancient theories of brain and mental processes were conceived by Plato (427-347 B.C.) and Aristotle (384-322 B.C.). But the first practical achievement and important contribution is attributable to the neuroscientist W.S. McCullock and to the mathematician W. Pitts which in their work in 1943, "A logical calculus of the ideas immanent in nervous activity", elaborated what is considered the first neural network model. Basically, their network, belonging to the category called linear threshold combiner[8], consists of a set of threshold binary units.

[8] A linear combiner is an ANN that, as the name suggests, gives as output a linear combination of its inputs. A linear threshold combiner, thus, is an ANN that gives as output a linear combination of its inputs only if it is higher than a threshold.

Threshold binary units can be defined as units characterized by the possibility to implement multiple binary data as inputs, while giving only one binary data as output. A sufficient number of these units, interconnected between each other as a network allows us to compute simple Boolean function[9].

For the successive important historical event for the development of ANNs we have to thank the psychologist Donald O. Hebb that in 1949, in his book "The Organization of Behavior: A Neuropsychological Theory", he proposed the first studies on associative learning based on the mechanism of neural plasticity.

Basically, he studied the connection between the biological function of the brain as an organ and the higher function of the mind. The Hebb's Rule, as is commonly called, defines that: *"When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased"*[10].

In other words, this means that if two units are activated at the same time, the existing connections between them strengthen and then, if one of the two units is activated at a later time, it will send a stronger impulse to the other unit that will tends to act. This mechanism formalizes the idea of association[11]. The "Hebbian Learning", typically considered as a rule of unsupervised learning, and its successive variants have been the basis for the important advances in the years to come. Then, we had to wait other nine years for the successive important step in the ANNs' history. In 1958 the psychologist Frank Rosenblatt introduced the first neural network scheme, known as Perceptron, that consists in a recognition and classification algorithm aimed to provide an interpretation of the general organization of biological systems. The probabilistic model of Rosenblatt is therefore focused on a mathematical analysis of function, such as information storage, and on their influence on pattern recognition: this represents a crucial advance to the binary model of McCulloch and Pitts since, in Rosenblatt's network, the synaptic weights are variable and hence the Perceptron is able to learn.

---

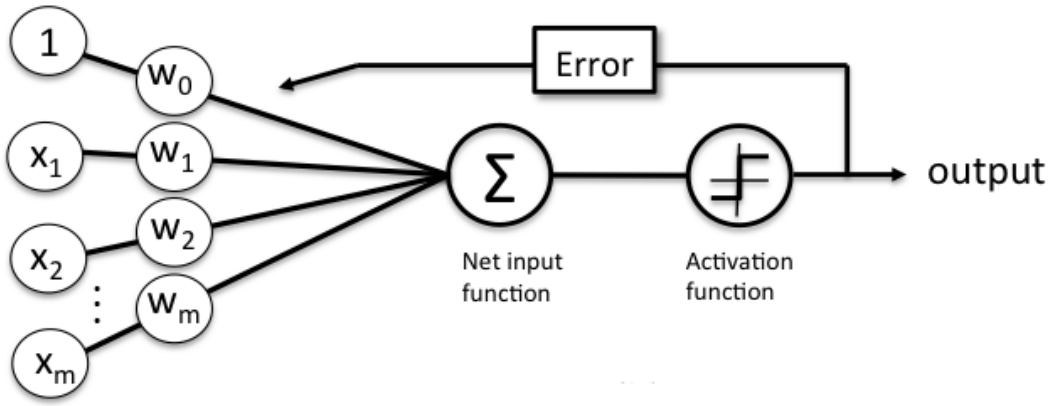
[9] A Boolean function is a function of k variables of the type: $f(x_0, x_1, \dots, x_k)$: $B^k \rightarrow B$, composed of the x_i Boolean variables that can assume values within the Boolean space $B = \{0,1\}$.

[10] D. O. Hebb, "*The Organization of Behavior: A Neuropsychological Theory*", 1949, Wiley and Sons, New York.

[11] In Psychology, association refers to a mental connection between concepts, events or mental states that usually stems from specific experiences.

Thus, this category of network is the basis of supervised learning, which consists of a teacher, outside the network, that provides the correct output the network should give at each particular input; then, based on the difference between its output and the output given by the teacher, the network changes the input weights in order to reduce those differences. We can see an example of this process illustrated in Figure 2.4.

Figure 2.4: Rosenblatt's Perceptron



Source: www.sebastianraschka.com

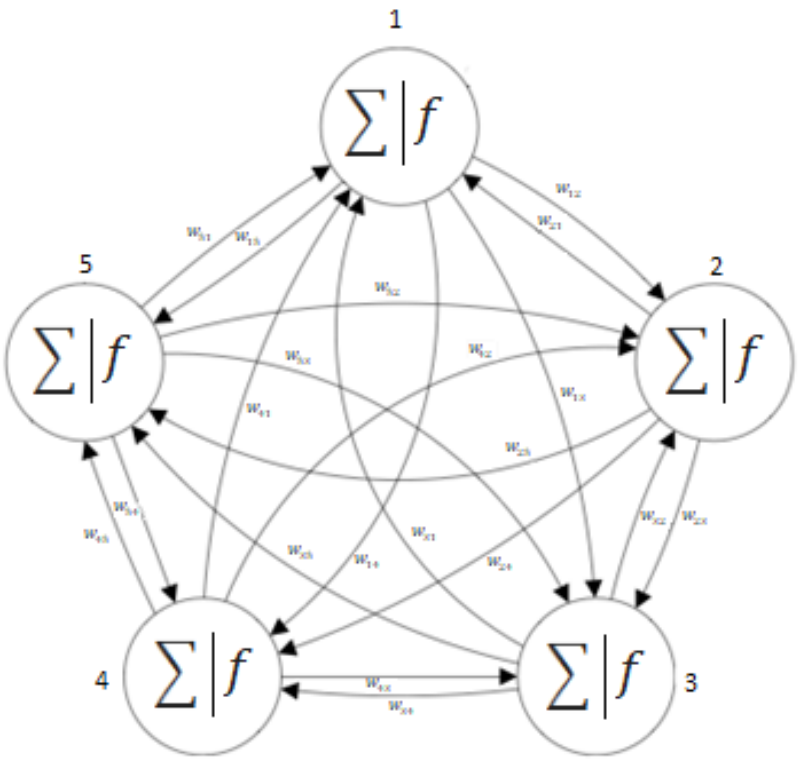
Even if the work presented by Frank Rosenblatt generated a strong interest in the scientific community and caused several successive studies in the field, in 1969, Marvin Minsky and Seymour Papert, published a book entitled "Perceptrons": an introduction to computational geometry" in which they highlighted the limitations of the model developed by Rosenblatt proving that it was not able to solve the class of classification problems called non-linearly separable problems such as the XOR function[12]. Because of this serious limitation the studies of artificial intelligence experienced a period of substantial indifference during which the enthusiasm and euphoria for the subject disappeared. Substantially, the biggest problem shown up caused the awareness that ANNs were still far from being able to match the artificial realization of a pattern with its natural structure. Despite few researchers continued to study artificial intelligence, only in 1982, thanks to John J. Hopfield, the development of ANNs re-gained its deserved importance and since that year other models and theories have been developed but they will be discussed later as belonging to the branch of Recurrent Neural Network.

[12] XOR function or "Exclusive or" or "Exclusive disjunction" is a logical operation that outputs true only when inputs differ (one is true and the other is false).

2.5 Network Architecture

ANNs can be classified on the basis of their architecture, that means according to the way in which their nodes are organized. Usually neurons are organized in layers, which are nothing more than “vectors” of units each of which is interconnected with the units of the previous and the successive layer. For example, we can identify ANNs with one, two, three or more layers and even without any layer, that is with fully connected units. Basically, in the ANNs with three or more layers the external ones perform the function of input and output layer respectively, just as in a biological brain where there are cells that are directly connected to the sensory organs (thus they receive signals) and cells that are in charge of the movement and the transmission of voice messages, that is, output signals. In this type of network the connections of a neuron can be only with neurons of a successive layer and there cannot be connections between neurons within the same layer nor with neuron of layers that are not immediately close. In a non-layered network, instead, also called fully connected, each neuron is connected with all other neurons. A representation of this type of network is given in Figure 2.5.

Figure 2.5: Fully Connected Neural Network



Source: www.neurdon.com

All the various connections between the neurons can be represented by a square matrix X with dimension equal to the number of nodes in the network, where the generic element w_{ij} expresses the weight of the connection between the neuron i and the neuron j.
Here below we will quickly describe the Single-Layer Perceptron and the Multi-Layer Perceptron.

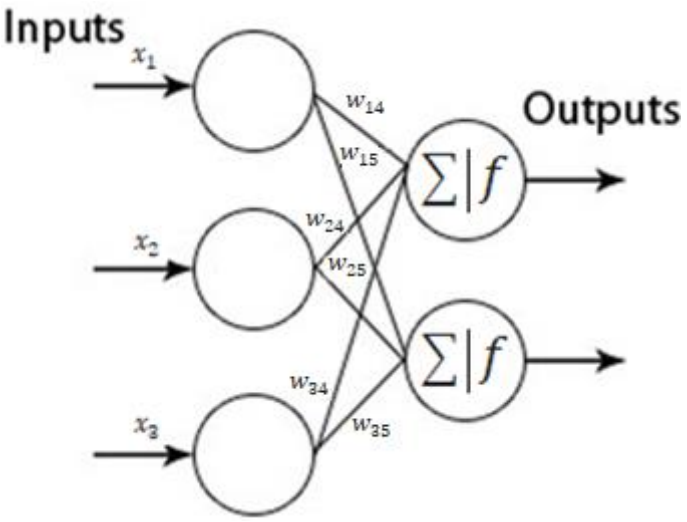
2.5.1 Single-Layer Perceptron

The Single-Layer Perceptron, belonging to the class of feed-forward networks, is the first and simplest example of ANN. In feed-forward networks there are no connections between units in the same layer nor between units of "non-close" layers (since they can have multiple intermediate layers) and thus information travel only in one direction from the input layer to the output layer, so there are not any cycles or repetition. The Single-Layer Perceptron model is characterized by only one layer of output nodes and several inputs that, through a series of weights, simply pass the signals to the output. The sum of the product between the input and the weight is computed by each node and if the value is greater than a certain threshold the neuron is activated and, typically, takes value equal to plus one. Otherwise, if the value is lower than the threshold the neuron is deactivated and, typically, takes value equal to minus one. Since there is not any computation in the input layer, this is considered as a single-layer network. The Ronsenblatt's Perceptron has been such an innovation from the McCulloch and Pitts' model because of the fact that the perceptron was able to learn changing the weights of the inputs and the threshold for the activation of the neurons. In this model, the learning process simply consists in start assigning random values of the weights and threshold and providing certain known examples, which must be learned by properly varying the weights and the threshold values. More precisely, for each example to learn, the output Y is calculated. For example, if Y is a binary output, that is, it can assume only two different values (zero and one or plus one and minus one), and it results to be zero when it should be 1, then the various weights are increased or left unchanged, depending on the respective input value, in order to increase the power of the activation. If, instead, we obtain $Y = 1$ and the desired result is 0, the weights are decreased. This process is iterated until the examples to learn finish or the correct result is obtained. At this point the weights are frozen and we can proceed to the validation and test phase, where new data are introduced as input and the ANN recognizes or classifies them based on the stored ones.

It can be shown that, giving enough examples, this learning process always converges to an optimal solution, represented by the minimum error (ideally zero) between the desired output and the network output[13].

In Figure 2.6 we can see an illustration of this model. We can also notice that in this particular picture the output layer is composed by only two nodes but this is not the rule, that is, the number of nodes in each layer simply depends on the problem we have to solve and it does not depend on the structure of the network (single or multi-layer). For example, as showed in Figure 2.4, in the first single-layer network proposed by Rosenblatt, the output layer was composed by only one node.

Figure 2.6: Single-Layer Perceptron



Source: www.neuroph.sourceforge.net

2.5.2 Multi-Layer Perceptron

Multi-Layer networks represent a generalization of the basic principles of the perceptron. These are feed-forward networks with other layers, between the input and the output layer, which do not interact directly with the external environment.

[13] C. Macchiavello, "*Introduzione alle Reti Neurali*", 1992, Dipartimento di Fisica A. Volta, Università di Pavia.

For this reason, they are called hidden layers and they represent the solution to the computational limits of the perceptron. ANNs of this type allow to solve classification problems of non-linearly separable sets[14].

Minsky and Papert, in 1987, highlighted that a multi-layer network with only one hidden layer is always able to solve non-linearly separable problems, such as the previously mentioned XOR function. It can even be demonstrated that any function can be computed from a three-layer neural network, that is, with an output layer and two hidden layers (Kolmogorov's theorem).

As for the single-layer model, the connections in the multi-layer network are unidirectional, starting from the input units and then passing to those of a lower layer, that is closer to the output layer.

As we already said, the number of input and output units is determined by the problem that has to be solved, while unfortunately there are no any criteria to establish the optimum number of hidden layers and their units. However, in general, no more than one or two hidden layers are usually used and the number of their units is often lower than those of the input and output layers. Connections between the neurons of a layered network are represented by so many matrices as the number of couple of nearby layers. Each matrix, then, contains the weights of the connections between each couple of neurons of two adjacent layers.

As we have already emphasized, the learning process plays a crucial role for the proper functioning of the network: the update of the connection weights, in fact, is carried out in order to minimize the error between the computed output value and the desired output. This process is relatively simple for the single-layer perceptron, while, in the multi-layer one, since the desired output for the hidden units is not known, there was not a valid criterion to modify their connection weights.

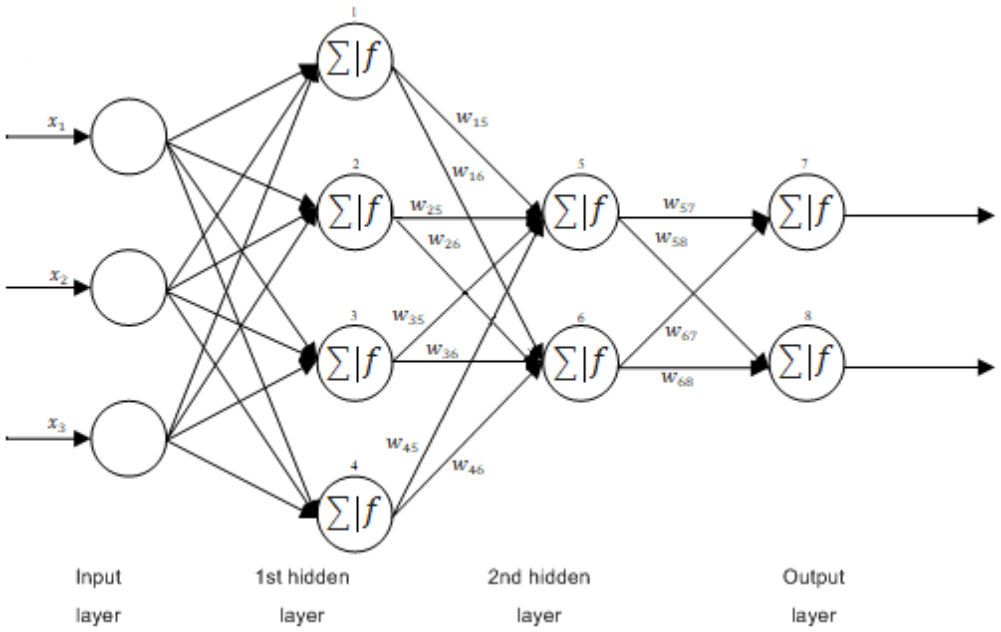
This problem, however, has been solved with the technique of Error Back-Propagation, developed by Rumelhart, Hinton and Williams in 1986, which will be explained later.

In Figure 2.7 there is an illustration of a first multi-layer feed-forward network containing a hidden layer with more units than the input and output layer and a second hidden layer with the same number of units of the output layer (for simplicity, the first set of connection weights, from the input to the first hidden layer, has been omitted).

[14] A training set is said to be linearly separable if there exists a hyperplane able to separate, in the input vector space, those inputs that require a positive output from those that require a negative one.

As we said, the neurons of both the hidden layers do not communicate directly with the external environment.

Figure 2.7: Multi-Layer Perceptron



Source: www.commons.wikimedia.org

2.5.3 Activation Function

As we have seen so far, in an ANN, the neurons are represented as units that can easily process the received signals from other neurons and the synapses are represented as connections between the units.

The electrical signal traveling along the axon is represented by a number, usually between zero and one, the power of the signal is represented by the connection weight and its effect on the signal is simply represented by the product between the signal itself and the respective weight before it reaches the successive unit. For example, if x is a signal that is sent from a neuron to another one, and w is the weight of the connection between the two neurons, the "weighted" signal that arrives at the second neuron will be equal to $w * x$.

Each unit perform two very simple operations on the incoming signals: they compute the activation potential and, if it is the case, they become active.

The computation for each neuron's activation potential is easily done by summing the "weighted" signals coming from all other connected neurons.

So for example if x_1, x_2, x_3 are the signals coming from the other neurons with connection weights equal to w_1, w_2, w_3 then, the activation potential a is obtained as $a = w_1 * x_1 + w_2 * x_2 + w_3 * x_3$. This means that, based on the activation potential received, each unit sends some signals to all the units they are connected with.

The calculation of these signals, which are sent to the units of the successive layer, is made by using a mathematical function called transfer functions or activation function.

More formally, the computation of the output value can be summarized with this equation:

$$a_j^i = \gamma\left(\sum_k \left(w_{jk}^i * a_k^{i-1}\right)\right) \tag{2.1}$$

where a_j^i is the activation of the j^{th} neuron in the i^{th} layer, γ is the activation function, w_{jk}^i is the weight of the connection between the k^{th} neuron in the $(i-1)^{th}$ layer and the j^{th} neuron in the i^{th} layer. Since there are several types of activation function that can be used, here below we will describe the most used ones.

Threshold Binary Function

As the name suggests, the threshold binary function, used in the first model developed by McCulloch and Pitts, is particularly useful in solving binary problems, that is, in situations in which the desired output can be zero or one. When the sum of the weighted signals, that is the activation potential, is greater than a certain threshold, the output takes value one and the unit is activated, otherwise when the activation potential is lower than the threshold the output takes value zero and the neuron is not activated.

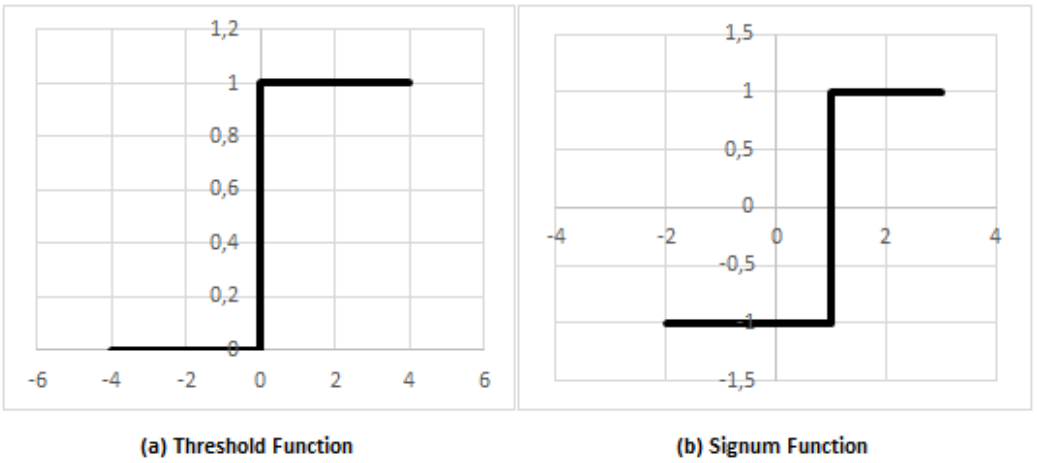
Thus, the equation of the threshold binary function can be written in this form:

$$\gamma(x) = \begin{cases} 1 & if\ x \geq \theta \\ 0 & if\ x < \theta \end{cases} \tag{2.2}$$

where $\gamma(x)$ represents the output value, x represents the activation potential and θ represents the threshold value. A variant of the threshold binary function is the Signum Function, in which the output is symmetric with respect to zero, that is, it can takes the value of plus one, if the activation potential is higher than the threshold, and value minus one, if the activation potential is lower than the threshold.

In Figure 2.8 (a), the threshold binary function is represented; in this particular case the threshold is equal to zero, in the *X* axis there is the activation potential, and in the *Y* axis there are the corresponding possible activation values of the unit. In Figure 2.8 (b), instead, there is an illustration of the signum function.
These activation functions reproduce an aspect of the biological neurons, characterized from the fact that the neuron become active only if the activation potential exceed a certain level.

Figure 2.8: Threshold and Signum Functions



Source: Personal representation

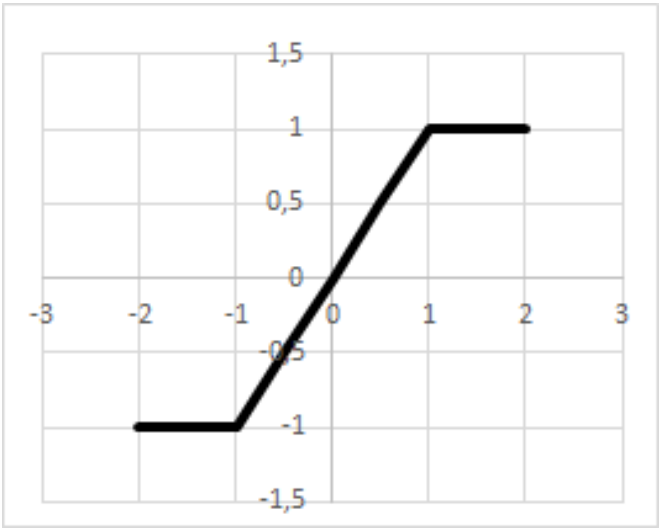
The advantages to use these functions are that they are fast and easy to use and they allow to take decisions without any doubt. But there are some drawbacks too: for example they are not invertible and they are never approximately linear, thus they cannot smoothly imitate functions.

Ramp Function

Ramp function, also called Piecewise Linear function, is similar to the signum function. In fact it can takes values equal to plus one or minus one, when the neuron's activation potential is greater or lower than the threshold respectively but with the substantial difference that the transition from plus one to minus one is not instantaneous.

In the following figure there is a representation of this function. In this particular case the threshold is equal to one.

Figure 2.9: Ramp Function



Source: Personal representation

The equation of the Ramp function thus can be written in this way.

$$\gamma(x) = \begin{cases} 1 & if \quad x \geq \theta \\ x & if \quad -\theta < x < \theta \\ -1 & if \quad x < -\theta \end{cases} \tag{2.3}$$

where $\gamma(x)$ is the output value, x is the activation potential and θ is a fixed threshold.
Even if this function is not invertible, there are still some advantages in using it: in fact, in the middle range, it is approximately linear and thus it can imitate linear functions and it is relatively easy to implement.

Linear Function

The linear function, typically used for the output layer, is drawn as a straight-line in which the output is directly proportional to the input and thus, in this way, the output can takes values among a range of activations (from $-\infty$ to $+\infty$), so no more binary activation.

Assume that we want to create a multilayer network: if we use only linear activation functions for all the network's units, the final output would just be a linear transformation of the initial inputs, which would not be sufficient to approximate the relationships between input and output (the real world is not linear)[15].

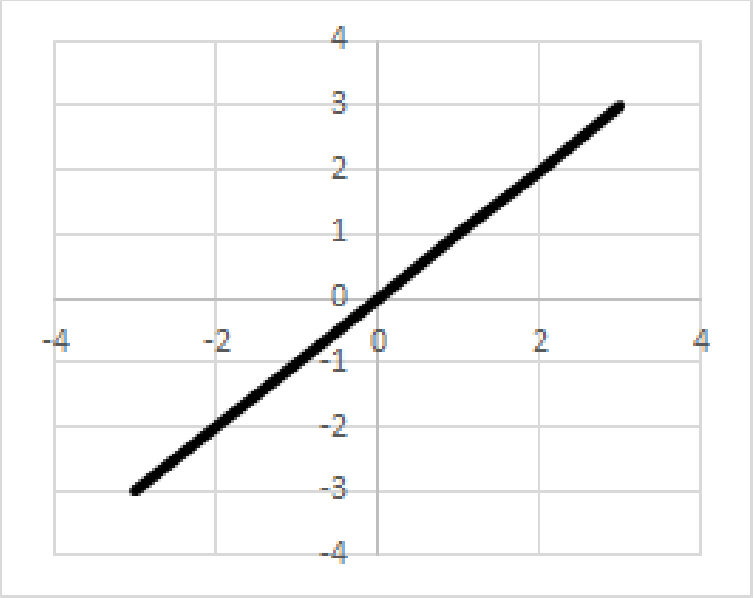
A neural network can have output neurons activated by a linear function, but only if the neurons of the other layers are activated by non-linear functions.

Thus, the equation of the linear function can be written in this way.

$$\gamma(x) = \alpha x \tag{2.4}$$

where $\gamma(x)$ is the output value, α is a constant. In Figure 2.10 there is a representation of the linear activation function; In that figure, α is the slope of the line and x is, as usual, the activation potential.

Figure 2.10: Linear Function



Source: Personal representation

[15] Independently on the number of layers of our network, this would just be as a single-layer network.

This is a very simple activation function: in classification problems, for example, the simultaneous activation of more than one output neuron means that the network is not able to correctly classify a given input. With a linear activation function, this problem disappears: if more than one neuron is activated, in fact, only the one with the highest activation value will be considered.

Problems with this function, however, arise in the learning phase of the network. In fact, as we will see in a later paragraph, using the technique of the gradient descent, the derivative of this function is a constant.

This implies that there will not be any relationship between the gradient and the activation potential, that is, if there will be an error in the output of the network, the changes of the input weights, through the back propagation, will be constant and not depending on the difference between the network's output and the desired output[16].

Sigmoid Function

The Sigmoid function, also called Logistic function, is a non-linear, continuous and derivable function, which has a non-negative derivative.

For these reasons it is used not only in the field of artificial intelligence but also in a wide range of disciplines: for example it is used in statistics as a cumulative distribution function. However, it is the most used activation function in ANNs for classification problems. Its equation can then be written in this way.

$$\gamma(x) = \frac{1}{1 + e^{-\alpha x}} \tag{2.5}$$

where α is a parameter indicating a kind of slope of the function. This function approximates the threshold function if $\alpha \to \pm\infty$.

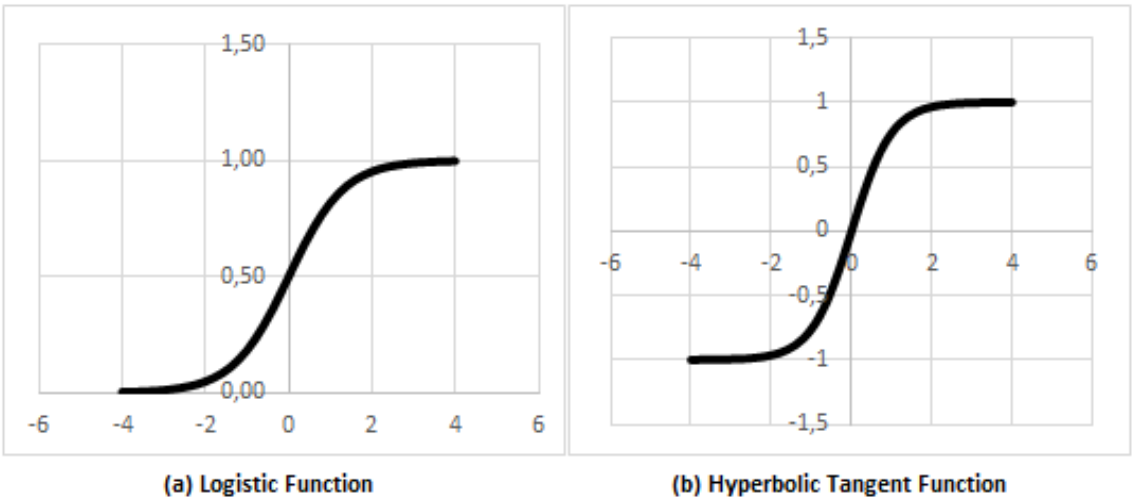
Looking at the Equation (2.5), we can figure out that, unlike the linear function, the Logistic function is bounded between zero and one, compared to the range from plus to minus infinity of the linear function. However, if the problem require a different output, we can use another sigmoid function, called Hyperbolic Tangent function, that, as we can deduce from equation (2.6), it is bounded between minus one and plus one.

[16] A. Sharma, "*Understanding Activation Function in Neural Networks*", 2017, The Theory of Everything, www.medium.com.

$$\gamma(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.6}$$

This range of output values is surely more clear looking at Figure 2.11, where there is a comparison between the Logistic function (a) and the Hyperbolic Tangent one (b).

Figure 2.11: Logistic versus Hyperbolic Tangent Function



Source: Personal representation

As we can notice from both the figures above, the sigmoid functions are approximately linear in their middle range. But this is true only for the middle region: in fact, large positive or negative changes in the x values provoke smaller changes in the y values, that means that these functions have a tendency to produce results at either the ends of the curve. Even if it seems a good property, at least for classification problems, this means that the gradient at the ends of the curve is very small, thus the network is no more able to learn further examples or the learning process will require more and more time. We will describe this problem more in details in the later paragraph.

2.6 Neural Network Learning

ANNs, since they are inspired by biological neural networks, should replicate even the learning process of the brain.

In the human brain the learning process involves the neural plasticity, a term that refers to the ability of the brain to change, to modify the connection between neurons and this process is carried on thanks to our experiences.

In a similar way, ANNs learn through experience: through a learning algorithm, in fact, they are able to properly configuring their weights, that is, the structure of the connections between neurons, in order to achieve effective results for the tasks they have been created for. However, there are different types of learning process, also called learning paradigms, and learning algorithms. With learning paradigm is meant the methodology used to train the network of neurons. We can divide the learning paradigms into two different categories: supervised and unsupervised learning. Simplifying as much as possible, it can be said that the biggest difference between the two learning paradigms lies in the fact that in the former the network is able to learn thanks to the feedbacks it receives from the external operator, while in the latter the network is able to learn independently, without any external suggestions. Once this choice has been made, the next step consists in deciding the algorithm that should be used in order to properly varying the connection weights, necessary to find the optimal solution, ideally represented by the difference between the actual network output and the desired one equal to zero. Both the supervised and the unsupervised learning will be described in the next paragraphs and, then, the Gradient descent will be presented as the most used learning algorithm, which is used by the technique of Error Back-Propagation.

2.6.1 Supervised Learning and Delta Rule

In order to use the technique of Supervised learning, first of all it is necessary to define the initial vector of input data and the vector of output data containing the desired output.

The second step consists in defining an activation function γ that has to be able to associate each input with its correct output. In fact, all the supervised learning techniques are based on the assumption that, by providing to the network an adequate number of examples, known as training set, an a-priori defined learning algorithm, usually the Error Back-propagation, will allow the network to approximate the existing relationships between inputs and outputs, learned during the training phase.

Then, for each pair of input and output data, the model computes the errors, which are represented as differences between the outputs actually obtained and the desired ones. Thus, the errors computed by the network can be summarized with the equation (2.7).

$$\varepsilon_j = \rho_j - a_j \tag{2.7}$$

where ε_j is the error, ρ_j is the desired output, known a priori, and a_j is the output resulted from the network for the j^{th} output unit.

When the learning process is supervised, the most used rule to adjust the connection weights between neurons is the Delta Rule, also known as Widrow-Hoff rule. The Delta rule defines that the variation of the generic connection weight w_j is given by the equation (2.8).

$$\Delta w_j = \eta \varepsilon_j x_j \tag{2.8}$$

where ε_j is the error computed for the j^{th} unit, x_j is the input vector provided to the j^{th} neuron and η is a real number between zero and one and it represents the learning rate, that is, determines the "learning speed" of the neuron.

It is worthy to notice that the Delta rule modifies, proportionally to the error, only those connection weights that contributed to the error itself, that is the connection weights between the j^{th} neuron and those that sent a signal different from zero. Conversely, for those that sent a signal equal to zero, the connection weight is not modified since it is not possible to know whether they contributed or not to the error.

Then, the new value of the generic connection weight w_j is given by the formula (2.9).

$$w_j^t = w_j^{t-1} + \Delta w_j \tag{2.9}$$

where t and $t-1$ are only aimed to indicate the value of the connection weight at time t and $t-1$ respectively.

If the training is successful, the network learns to recognize unknown relationships that links new input variables to new output variables, and is therefore able to make predictions even when the output is not known a priori.

In the second phase, in fact, we have to evaluate the generalization capacity of the network and this is done by testing the model, providing it examples not used during the learning phase. Of course, the generalization capacity of an ANN depends heavily on the set of examples chosen for the learning phase. Such examples, in fact, must accurately represent the reality that the network has to learn.

2.6.2 Unsupervised Learning

Unsupervised learning is a learning technique that consists in providing to the ANN only a set of inputs. The network then, searching for unknown relationships in the data, analyzes, classifies and organizes the data based on their common features in order to learn to make reasoning and predictions on subsequent inputs. In other words this algorithm attempt to group input data identifying appropriate clusters representative of the data, typically using topological or probabilistic methods.

A typical example of this algorithm is the search engines. These algorithms, given one or more keywords, are able to create a list of the websites that they consider relevant for the research.

This type of learning is thus based on the comparison of the data and the evaluation of similarities and differences between them. Specifically, during the learning process, the system, using parameters derived exclusively from the input data, is able to adjust the connection weights in order to perform a self-rating classification of the output results.

The ANN therefore seeks to organize itself automatically, often resorting to the competition that arises between different units. In fact, a particular type of unsupervised learning is the Competitive learning in which the output units, indeed, are said to compete with each other so that only the winning neuron will be able to improve its ability to recognize a particular characteristic of the input data.

More precisely, during the learning phase, the output unit that provides the maximum activation to a given input is declared winner, and thus its connection is strengthened, while all the other connections remain unchanged.

The process underlying this type of ANN is thus based on the stimuli exercised during the learning phase and on the connection weights, and therefore not on the management of a supervisor.

2.6.3 Gradient Descent Algorithm

As we have already said, the learning process consists in changing properly the connection weights between units of adjacent network's layers in order to minimize the difference between the actual network's output and the desired output.

Below we will give a brief description of what is the process that allows us to minimize the error. Many learning algorithms have an iterative procedure to minimize the error function, thus consisting in the determination of the connection weights after many successive steps.

Usually, the error function to minimize is represented by the mean squared error. Consider an ANN with n inputs and p outputs, then assume that ρ_j and a_j are the desired output and the actual network output respectively of the j^{th} neuron. Then, the mean squared error, E_k on the k^{th} example, can be defined as:

$$E_k = \frac{1}{2}\sum_{j=1}^{p}\left(\rho_{kj} - a_{kj}\right)^2 \tag{2.10}$$

A very efficient learning technique is based on the so-called gradient descent algorithm, developed in 1847 by A. L. Cauchy but introduced for this purpose only in 1986 by D. E. Rumelhart.

In mathematics, the gradient is a function of multiple real variables, thus defined in a space with two, three or more dimensions. Consider a generic function $f(x, y)$ defined on a set $A \subseteq \mathbb{R}^2$ and let (x_0, y_0) be a generic point in the plane, belonging to A. If, in (x_0, y_0) both the partial derivative with respect to x and that with respect to y exist, $f'_x(x_0, y_0)$ and $f'_y(x_0, y_0)$, it is then possible to construct a vector in which its components are the partial derivatives, as shown in Formula 2.11.

$$\nabla f(x_0, y_0) = \left(f'_x(x_0, y_0), f'_y(x_0, y_0)\right). \tag{2.11}$$

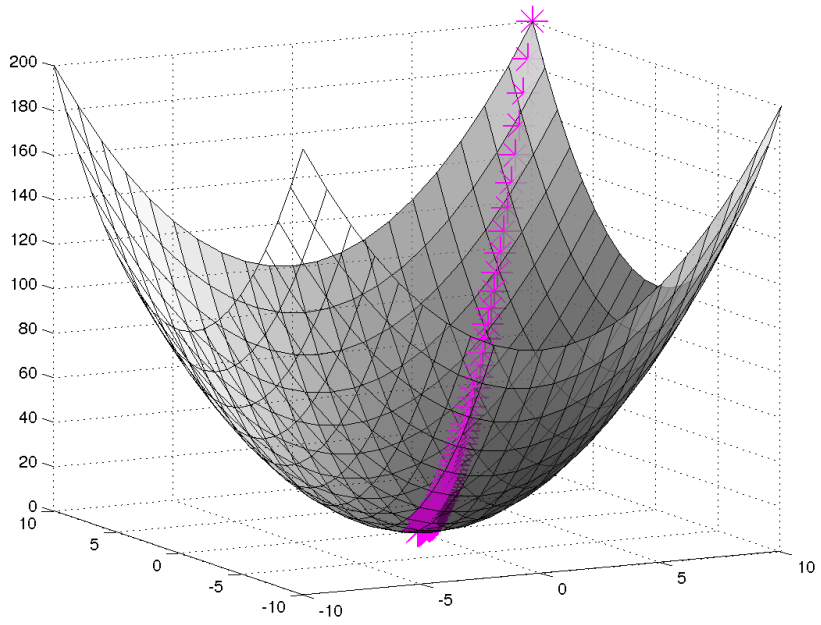
The vector is called the gradient of the function f evaluated in (x_0, y_0), or equivalently, $\nabla f(x_0, y_0)$. Similarly, the definition of gradient can be extended to a function $f : A \subseteq \mathbb{R}^n \to \mathbb{R}$. It will be sufficient to consider the vector in which its components are the n partial derivatives of the first order evaluated in the point $x_0 \in A$.

Gradient descent is a local optimization technique, meaning that, given a multidimensional mathematical function, the gradient descent allows to find a local minimum of this function. However, it can also be used to find the absolute minimum: in fact, once a local minimum has been found, it will be sufficient to repeat many times the iterative process, starting from a new point in the plane, chosen randomly.

In other words, this algorithm allows the network to learn, minimizing the error function E, modifying the connection weights in the opposite direction of the gradient, searching the absolute minimum.

The process can be graphically represented thanks to Figure 2.11: starting from the highest point, the gradient descent algorithm, after several steps (represented by pink asterisk) is able to find the minimum (the lower point in the figure).

Figure 2.11: Gradient Descent Algorithm



Source: www.mathworks.com

It is essential to underline that the process is divided into two phases. At every step of the iterative process, in fact, first, the derivatives of the error function are computed with respect to the weights and secondly, these derivatives are used to update the weights through an update formula, that we will see in the next paragraph, which is dedicated to the Error Back Propagation algorithm.

However, it is important to premise that this iterative process consists in a recursive formula through which, starting from the most external nodes (output nodes) it is able to back propagate the error, thus inversely to the direction of the forward-propagation, hence the name of back-propagation algorithm.

2.6.4 Error Back-Propagation Algorithm

The learning phase of Single-Layer network does not present any problems; in fact, the Delta rule can be easily used to update the connection weights.

Problems arise when we deal with Multi-Layer networks: applying the Delta rule, indeed, we would update only the connection weights of output neurons, but not those related to the hidden neurons. In fact, since the external agent provides the initial pairs of input-output data, the desired final output is known, but there is no way of knowing the desired output of the hidden neurons. Only in 1986, Rumelhart Hinton and Williams, as the solution to such a problem, proposed what in the successive years would have become one of the most known and effective methods for training ANNs, that is the so called Error Back-Propagation algorithm.

Considered as a kind of generalization of the Delta rule, this algorithm proceeds by two steps: first, it computes the error made by all the output neurons and, secondly, the error made by a hidden neuron is computed summing the errors computed in the first step considering, as weight, the connection weights between the hidden neuron and the output neurons. Hence the name error back-propagation. Then, the same procedure is repeated for all the neurons of the hidden layer.

In other words, the algorithm assumes that in the first phase, the signals travel from the input to the output layer. During this phase, which is called forward propagation, the connection weights are all fixed. In the second phase, called backward propagation, the ANN responses are compared to the desired outputs by obtaining the errors. These are propagated in the opposite direction with respect to the first phase, that is from the output to the input layer, and finally the connection weights are modified to minimize the differences between the current outputs and the desired outputs.

Since the algorithm requires the neurons activation function to be non-linear and derivable, usually a sigmoid function is adopted, which produces output values between zero and one, or hyperbolic tangent that produces output values between minus one and plus one.

Generally speaking, consider Equation (2.10), then, defining by s the weighted sum of the input signals, and by w_{ij} the connection weight between the neurons, we can express the derivative of the error as in equation (2.12):

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial s_j}\frac{\partial s_j}{\partial w_{ij}}. \tag{2.12}$$

Since ε_j in equation (2.7) can be expressed also as

$$\varepsilon_j = -\frac{\partial E}{\partial a_j}, \tag{2.13}$$

in this case, as we have no more linear neuron, we can express it as

$$\varepsilon_j = -\frac{\partial E}{\partial s_j} \tag{2.14}$$

and rewriting it as

$$\varepsilon_j = -\frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial s_j}. \tag{2.15}$$

Thus, since we know equation (2.10), we can write

$$\frac{\partial E}{\partial a_j} = -(\rho_j - a_j). \tag{2.16}$$

Then, for the activation function γ, the output is

$$a_j = \gamma(s_j) \tag{2.17}$$

then

$$\frac{\partial a_j}{\partial s_j} = \gamma'(s_j), \tag{2.18}$$

from which we obtain

$$\varepsilon_j = (\rho_j - a_j)\gamma'(s_j). \tag{2.19}$$

Defining with x the input, as usual, we know that

$$s_j = \sum_i x_i w_{ij}. \tag{2.20}$$

Then it is clear that

$$\frac{\partial s_j}{\partial w_{ij}} = x_i \tag{2.21}$$

thus, from the initial Equation (2.12) we obtain

$$\frac{\partial E}{\partial w_{ij}} = -(\rho_j - a_j)\gamma'(s_j)x_i = -\varepsilon_j x_i. \tag{2.22}$$

Finally, applying the learning rate, we obtain the formula for the generalized Delta rule:

$$\Delta w_{ij} = \eta \varepsilon_j x_i \tag{2.23}$$

which is equal to Equation (2.8). As we already said, we cannot know the error of a hidden neuron since we do not know its desired output. However, we can adjust it proportionally to its contribution to the errors made by its connected output neurons. The contribution of each of these errors depends, of course, on both the size of the error generated by the relative output neuron and the connection weight between the hidden and the output neuron. In other words, an output neuron with a big error greatly contributes to the error of any hidden neuron to which it is connected with a high weight. Thus for a hidden neuron, we can say that the error is given by:

$$\varepsilon_j = \gamma'(s_j) \sum_u \varepsilon_u w_{ju} \tag{2.24}$$

where u is the index of the neurons of the layer from which the errors are sent back.

At this point, it is useful to figure out another problem concerning the learning phase of the ANN, that is the particular choice of the rate at which the ANN should be able to modify the connection weights with respect to the significance of the error committed, said the learning rate.

The descent to the minimum of the function, operated by the generalized delta rule, can be approximated to the gradient descent algorithm if and only if at every step of the iterative learning process the learning rate η is so small that it does not modify too much the connection weights[17]. Conversely, if the η value is too small, the descent process can be extremely slow. But too high values of the learning rate can cause strong oscillations of the descent. These oscillations are related to the presence of deep valleys, with almost no slope, at the bottom of the surface of the error function. In these areas, the connection weights, at each update, continues to point to the minimum of the error function but at the same time the high learning rate could makes them deviate to the opposite direction. A good compromise is a modification of the generalized delta rule by adding a term, called momentum, which is multiplied by the direction of the previous change:

$$\Delta w_{ji}(t+1) = \eta \varepsilon_j x_i + \mu \Delta w_{ji}(t) \tag{2.25}$$

where t and $t+1$ are the numbers of the update and μ is a constant, the momentum, that determines the effect of the last variation of the weight w_{ji} on the new weight's direction[18]. The addition of the momentum allows to decrease the oscillations. Summing up consecutive weight corrections, in fact, cancel out opposite variations while sums concordant variations direct to the minimum. Learning becomes even faster when both the momentum and the learning rate have higher values.

In order to determine an appropriate value for the learning rate, we should use information on the size of the input vector. In order to determine the right value for the momentum, instead, we would need to know the shape of the surface of the error function.

---

[17] M. Nielsen, "*How the Backpropagation Algorithm works*", 2017, www.neuralnetworks-anddeeplearning.com.

[18] M. Stella, "*Le Reti Neurali: Isomorfismo tra i modelli di Hopfield e di Ising*", www.dmf.unisalento.it, 2009, Facoltà di Scienze Matematiche, Fisiche e Naturali, Università degli studi del Salento.

Even if we know that it should be a convex function, we do not know its exact shape (almost for the possible presence of local minima), thus one method to determine the value of the momentum that best fit our data and model is through a trial and error process.

2.7 Recurrent Neural Network

Recurrent Neural Networks are a particular type of ANN, which operate as sequential systems, that is, they associate the ANN outputs to the inputs presented, recognizing relationships that evolve over time. In other words, the recurrent ANN is able to take into consideration what the neuron activations were in previous time steps.

Practically, an ANN of this type computes the output at time t and, among the others, it will be used as input at time $t + 1$.

These ANNs are used, for example, when we want to forecast a future data based on some current and past data. In fact, if the time window we want to consider is relatively large, we cannot use a classic feed-forward network because it could take into consideration only the current data, or just data of one or few time steps behind, when, instead, the possible dependences could be between input values far apart in time from each other.

The output of a recurrent ANN, thus, does not depend only on the structural features of the network itself, as it occurs in the classic feed-forward ANN, but it is also a function of previous activations (also of other units and not only of the output units).

In this section we will present some of the most famous recurrent ANN architectures developed and used through the years.

2.7.1 Hopfield Network

As we said at the beginning of the chapter, after the Rosenblatt's Perceptron, the studies of artificial intelligence experienced a period of substantial indifference, during which the enthusiasm and euphoria for the subject disappeared.

However, in 1982, thanks to John J. Hopfield, the development of ANNs made another important step. In its famous article "Neural networks and physical systems with emergent collective computational abilities" he argued and expressed the concept that the associative system of an ANN should be based on the functioning of the human brain, thus allowing neurons to communicate between each other and not just with the successive layer of neurons.

In fact this is a totally connected ANN, as the one showed in Figure 2.5, in which the connections are symmetrical ($w_{ji} = w_{ij}$) and the neurons cannot be connected with themselves ($w_{ii} = 0$). Furthermore, each unit has a binary output, thus it can takes values plus one or minus one, or zero and one, and there is not distinction between input and output units: the ANN receives many inputs as the number of its units and, after processing them, presents the outputs through the units themselves.

The learning process of the Hopfield ANN is based on the Hebb's rule that, as already stated at the beginning of the chapter, defines that if two units are activated at the same time, the connection between them strengthens. This means that if one of these units is activated in a successive time step, the signal that it sends to the other units tends to activate this second unit.

Now, defining with a_i the activation potential of the neuron i at time t, where w_{ij} is the connection weight between neurons i and j and where θ_i is the threshold of each unit, the equations that govern the dynamics of the Hopfield network are:

$$a_i(t+1) = \gamma_i\big(x_i(t)\big) \tag{2.26}$$

where x_i is the input to neuron i at time t, and

$$x_i(t) = \sum_{j=1}^{n} w_{ji} a_j - \theta_i \tag{2.27}$$

where γ_i is the activation function, a hyperbolic tangent or a logistic function, since the units have a binary output. This rule is applied asynchronously, that is one unit at each step, according to a defined order or randomly. Then the process is repeated iteratively until the ANN relaxes in an equilibrium state, which corresponds to the final output.

Hopfield, finally, stated that its network is associated to a function, called Energy, described by Equation 2.28.

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i w_{ij} a_j + \sum_{i=1}^{n} \theta_i a_i. \tag{2.28}$$

In fact, he demonstrated that the "energy" decrease during the entire process, until it reach its minimum[19]. Hopfield also stated that its network, given its properties, is characterized by associative memory, that is the ability to recognize items (a particular pattern or an object) comparing them with previously learned examples, even if they are corrupted or are characterized by some biases. This is why this type of network is typically used in pattern or image recognition problems. However, the Hopfield network presents some drawbacks: it has been demonstrated that the ANN is able to learn and store only $k \simeq 0.14n$ pattern or configuration, where n is the number of neurons.

Thus, making an example to better understand the problem, if we want to store the Italian alphabet letters, that is $k = 21$, our network should have at least $n = \frac{21}{0.14} \simeq 150$ neurons and $\frac{n(n-1)}{2} \simeq 11000$ different connections.

Moreover, the Hopfield network, similarly to the Multi-layer network, is characterized by the presence of relative minimums in the energy function, thus making burdensome the iterative process.

2.7.2 Elman Network

A significant and well-known contribution in the field has been given by Jeffrey L. Elman that in 1990, in its research "Finding Structure in time"[20], proposed a new type of recurrent ANN. The Elman network can be thought as a variation of the Multi-layer network we have seen in paragraph 2.5.2: precisely it is composed by three layers (input, hidden and output) with the addition of another layer called context.

Except for this context layer, the network works as a classic feed-forward network. In fact at each instant, the inputs are propagated, through the hidden units, to the output units and then a learning algorithm, usually the Back-propagation, is used to train the network.

This context layer has the unique objective of providing the ANN with a kind of memory. The context neurons, indeed, receive the inputs from the hidden neurons thanks to specifically dedicated connection. Since these connections have weights equal to one, their outputs is simply a copy of the outputs from the hidden layer; these outputs are then re-introduced as input into the hidden layer, in the successive time step.

---

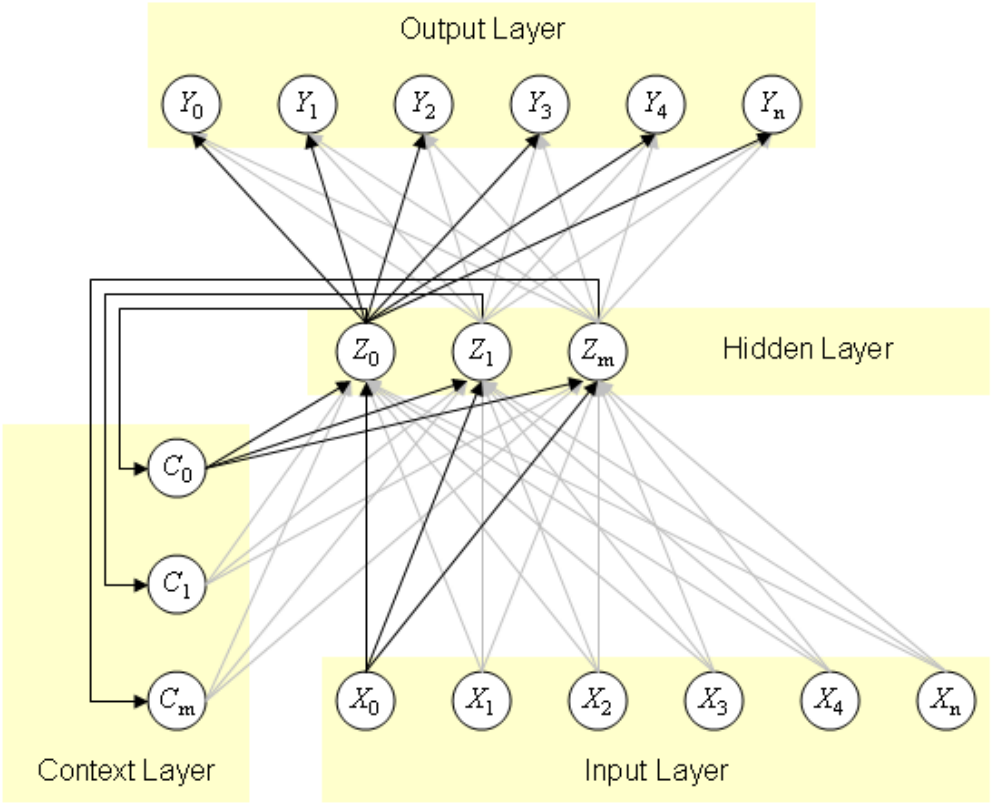
[19] A. Fernàndez and S. Gòmez, "*Portfolio Selection using Neural Networks*", 2005, Computers & Operations Research n.34, p. 1177-1191.

[20] J. L. Elman, "*Finding Structure in Time*", 1990, Cognitive Science n.14, p. 179-211.

Elman describe it better as:

"*Both the input units and context units activate the hidden units; the hidden units then feed forward to activate the output units. The hidden units also feed back to activate the context units. This constitutes the forward activation. Depending upon the task, there may or may not be a learning phase in this time cycle. If so, the output is compared with a teacher input, and back propagation of error is used to adjust connection strengths incrementally. Recurrent connections are fixed at 1.0 and are not subject to adjustment. At the next time step,* $t + 1$*, the above sequence is repeated. This time the context units contain values which are exactly the hidden unit values at time* t*. These context units thus provide the network with memory*". Thus, with this ANN, the hidden layer processes both the initial inputs and its previous activation, contrary to a hidden layer in a classic feed forward network that in fact only develops internal representations of the input patterns to produce the correct output. A graphical representation of the Elman Network is given in Figure 2.12.

Figure 2.12: The Elman Network



Source: www.mnemstudio.org

As we can see from the figure, there are one-for-one connections between hidden and context unit, meaning that there are an equal number of hidden and context units.
Moreover, each hidden unit is connected to only one context unit, but the connections context to hidden are fully distributed, such that each context unit is connected to all the hidden units.
Except for the modifications concerning the context layer, the equations for the computation of the outputs from inputs are similar to those of the common feed-forward ANNs. Even the learning algorithm in the training phase, as we said is equal to that used in feed-forward ANNs.

2.7.3 Back-propagation through time and the vanishing gradient problem

Recurrent Neural Networks are very difficult to train, given their specific architectures. This is because we must consider the time component that characterizes these ANNs.
Usually, the learning algorithms used for recurrent ANNs are based on the computation of the gradient of a cost function related to the connection weights.
This should sound familiar, as it is the same technique on which the learning phase of classic feed-forward networks is based. In fact, in order to train recurrent ANNs, it is used an algorithm, called back-propagation through time, that is nothing more than a generalization of the back propagation algorithm for classic feed-forward networks, but with the substantial difference that in the backward propagation phase also the time is taken into consideration[21].
Time, in this case, is simply expressed by a set of well-defined and ordered computations, linking an epoch to another. Thus, adding the time, as another variable, only extends the range of functions for which the computation of the derivatives for the gradient descent is needed.
When we use activation functions such as the logistic or the hyperbolic tangent one, there is the risk to run into the vanishing gradient problem. Basically, when the values of the activation potential approach the horizontal asymptotes of the activation function, the gradient tends to zero. For the back-propagation phase, therefore, the derivative calculation produces results very close to zero, as well as the gradient.

[21] Y. Bengio, P. Simard and P. Frasconi, "*Learning Long-Term Dependencies with Gradient Descent is Difficult*", 1994, Transactions on Neural Networks, vol. 5, n. 2 p. 157-166.

Consequently, the next connection weights update is very small: continuous infinitesimal weights updates make them tending to a fixed value, from which they can no longer move away. If this happen, thus, the network is no more able to learn. This phenomenon becomes increasingly relevant as the depth of the network increases, that is, as the number of hidden layers increases.

One solution to overcome the problem could be to reduce the learning rate, thus changing the weights of very small quantity at each iteration.

However, in this way, the gradient descent requires many small steps to converge to a solution: this is a problem if we use an old and slow computer that takes a lot of time for each step, but if we have a computer with a fast GPU, which performs many steps in rapid succession this becomes a minor problem.

Alternatively, in order to overcome the problem, some researchers proposed different ANN architectures that are not exposed to vanishing or exploding gradient.

Thus, in the successive paragraph The LSTM and the GRU models will be described.

2.7.4 Long Short Term Memory and Gated Recurrent Unit

Long Short-Term Memory (LSTM) is a particular architecture of Recurrent Neural Network. It has been originally proposed, in 1997, by Hochreiter and Schmidhuber in their research "Long Short-Term Memory"[22].

This ANN has been recently rediscovered, even by the colossus of informatics such as Google, because it is not exposed to the problem of vanishing or exploding gradient, and because in practice it offers excellent results and performance.

LSTM can be considered as a classic Multi-layer network, but with the substantial difference that one of the intermediate (hidden) layers hides a sort of second ANN.

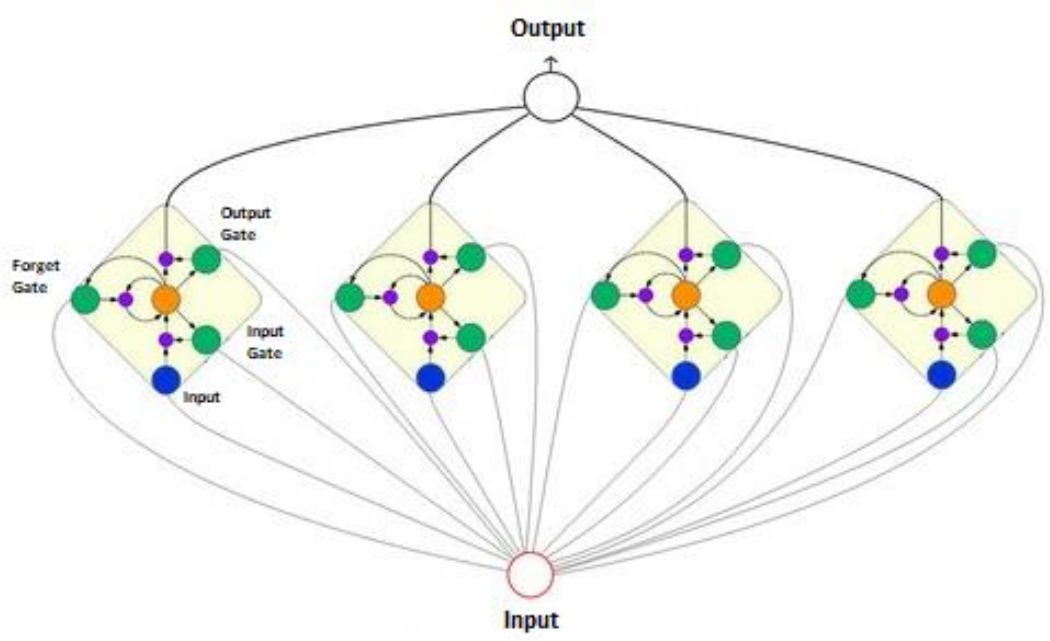
Within this second ANN the layers interact in a very special way. It includes, in fact, storage units that allow the storage of long-term dependencies in the data and in the output of previous iterations. Moreover, each unit is composed of three gates, which can be considered as "classic" artificial neurons, as in multi-layer ANN, and a cell, which is substantially the memory of the ANN.

Precisely, there are an Input gate, that, if it is the case, add values to be stored in the memory cell, a Forget gate, that screen and delete the values stored in the memory cell if these are no more considered useful and an Output gate, that determine the right output to be showed.

[22] S. Hochreiter, J. Schmidhuber, "*Long Short-Term Memory*", 1997, vol. 9, n. 8 p. 1735-1780.

Generally, these gate are driven by a sigmoidal activation function ranging from zero to one, where zero indicated total inhibition (for example, if the result coming from the input gate is zero, the cell will maintain stored the previous state, while if it is one the current state will be combined with the new input) while one indicated total activation.
Thus, the basic idea of LSTM is that it is possible to remember a value for an arbitrary period of time. The gates, then, have the task of deciding when a value is significant enough to be remembered, that is, if a value is sufficiently greater than a certain threshold to be stored in the memory, how long it must be stored, and when it should be presented as output.
In order to better understand the model, it could be useful to look at Figure 2.13 in which there is a graphical representation of the Long Short-Term Memory model. In this case, the input and the output layer have only one unit, even if this is not the rule.

Figure 2.13: Long Short-Term Memory



Source: www.dukascopy.com

There is another version of the LSTM model, called Gated Recurrent Unit, that has been proposed for the first time in 2014, thus it is the most recent model presented.
The core idea of the Gated Recurrent Unit model is very similar to that of the LSTM model.

However, the former has only two gates, a reset and an update one: the reset gate determines how to link the new input with the previously stored state and the update gate defines whether, and in which percentage, to maintain previous state or to discard it[23].

If, hypothetically, we fix the weights of the reset gate at one and the weights of the update gate at zero we would get a recurrent ANN in its simplest form. In many applications, both ANNs offer similar performance. Thus, in this case, the choice of the parameters and the total size of the ANN become the most important decisions for effectively implementing a model using one of the two architectures.

GRU architecture has fewer parameters and then is able to learn slightly faster or require less data to generalize than the LSTM model. On the other hand, if we have enough data, the greater expressive power of LSTM can lead to better results.

[23] D. Britz, "*Recurrent Neural Network Tutorial, Part 4 - Implementing a GRU/LSTM RNN with Python and Theano*", 2015, www.wildml.com.

Chapter 3

Research and Analysis

3.1 Neural Network Design and Methodology

In this section, we will describe and follow step by step the common procedure to properly design an ANN. Precisely this procedure, introduced by I. Kaastra and M. Boyd in 1996[24], recognizes eight steps in designing a neural network forecasting model:

Step 1: Variable selection → definition of the inputs to be presented to the ANN;

Step 2: Data collection → collection of the variables previously defined;

Step 3: Data preprocessing → analysis and transformation of input and output data;

Step 4: Train, test and validation sets → divide the time series in three different sets;

Step 5: Neural network paradigms → definition of the network architecture;
- number of hidden layers
- number of hidden neurons
- number of output neurons
- transfer function

Step 6: Evaluation criteria → definition of the error function to be minimized;

Step 7: Neural network training → definition of the training process;
- number of training iterations
- learning rate and momentum

Step 8: Implementation → ANN deployment.

In an analogous way, we will describe the design of our network, getting into details and resuming when necessary.

[24] I. Kaastra, M. Boyd, "*Designing a neural network for forecasting financial and economic time series*", 1996, Neurocomputing n.10, p. 215-236.

3.1.1 Variable Selection

As previously stated in the first chapter, since Technical Analysis demonstrated its validity in recognizing patterns in an asset price's time series, and therefore in predicting its future trends, our ANN will be trained presenting to it, as inputs, several technical indicators used in practice by analysts.

Specifically, both trend-reversal and trend-following indicators will be presented to the ANN[25]. These indicators, based on the open, closing, minimum and maximum values of the price of each asset, as well as on their rates of change, aim to signal respectively a possible trend reversal or the presence of a trend, bearish or bullish, so that the trader that is using them can appropriately take advantage of them. In addition to the Technical Analysis indicators, we will present to the ANN also three external inputs, meaning that they are not directly related to the stocks under analysis. These inputs are the price of oil Brent, the Volatility Index and the 3-months interest rate Euribor.

Brent, together with the WTI, is one of the major benchmarks for the price of crude oil, that is oil as extracted from the oilfields, thus before the refinement process. Precisely, the Brent, which is a type of crude oil mined in the North Sea, is used as a reference for defining prices in Europe, Africa and the Middle East. The Vix Index is a measure of the implied volatility of the S&P 500 computed through a kind of weighted average of the volatility of its options. It represents volatility market expectations over the next 30 days.

Finally, the Euribor (Euro Interbank Offered Rate) is the average interest rate at which the financial transactions in Euro occur between the major European banks.

Then, here below, we will give a description, of each indicator we will use to train the network.

Open, Closing, Minimum, Maximum Price, Exchange Volume and Log-Return

The opening price represents the price at which the market started to trade the asset at the beginning of the day; the closing price, instead, represents the last price at which the market traded the asset before closing.

The minimum price and the maximum price represent, respectively, the minimum and the maximum price at which the market traded the asset during the day.

[25] A trend-following indicator aims to identify a trend in order to open a position that will benefit from the continuation of the trend. A trend reversal indicator, instead, aims to identify a change in the market direction in order to open a position coherent with the reverse of the current trend.

Then, the Volume indicates the amount of shares of an asset that has been traded during the day and, finally, the log-return is the percentage change in the asset price between two consecutive trading days.

Exponential Moving Average (EMA)

The most used indicators in Technical Analysis are the moving averages. Moving averages are simply computed summing the closing price of n periods and dividing the sum by the n periods considered. Then, maintaining fixed the number of periods, anytime there is a new observation and it is added to the previous sum, the oldest observation is subtracted and the resulting amount is again divided by n.

The main purpose of moving averages is to identify the real market trend smoothing possible short-term biases in the price trend. There are several types of moving averages, such as the simple or the weighted ones.

However, we will consider only the Exponential Moving Average. It is more complex than the other types but it is more reactive with respect to the price's variations. The Exponential Moving Average, gives more importance to the most recent data and less to the earliest ones.

For its computation, it is necessary to define a parameter called decay factor, usually between zero and one, which allows to give a higher weight to the latest values and a lower weight to the older ones. The equation of the Exponential Moving Average is described by the Formula 3.1, where EMA_t and EMA_{t-1} represent respectively the current Exponential Moving Average and the previous one, C_t represents the last closing price and ϕ represents the decay factor, usually computed as $\phi = 2/(n+1)$ where n is the number of time periods considered:

$$EMA_t = (C_t - EMA_{t-1}) * \phi + EMA_{t-1} \tag{3.1}$$

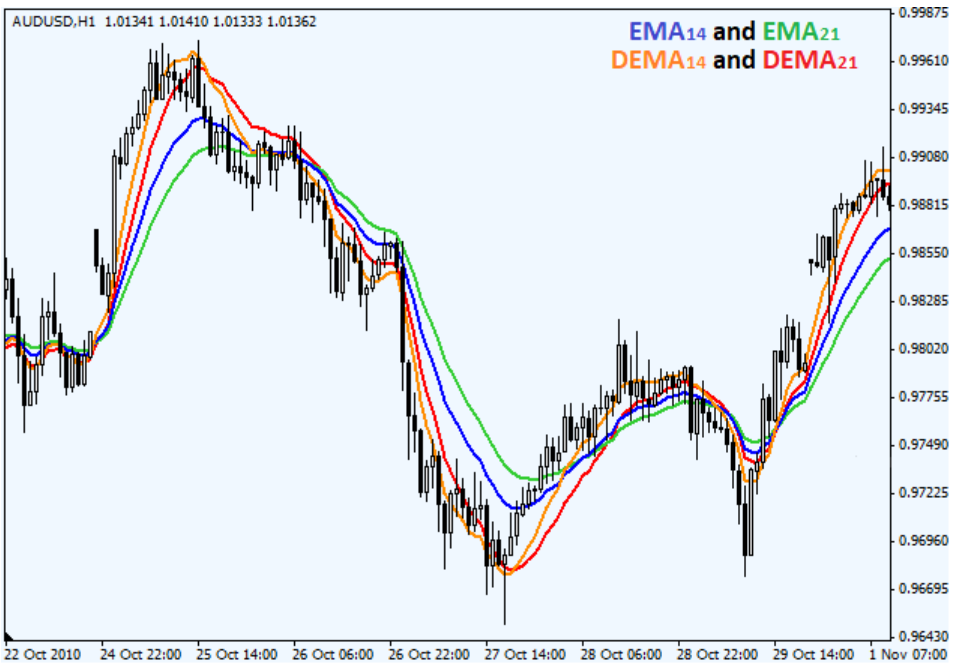
The computation of the Exponential Moving Average assumes that the first value is equal to the simple moving average, and then the other values are computed using the Formula 3.1. Usually, technicians use two EMA considering different periods, generally fourteen and twenty-one respectively, looking for bullish or bearish signals: the former occurs when the short-term EMA crosses above the long-term one and the latter occur when the short-term EMA crosses below the long-term one.

Double Exponential Moving Average (DEMA)

DEMA is a variant of the classic exponential moving average. Since the EMA is subjected to a certain degree of retard in signaling a trend, Patrick Mulloy, in its article "Smoothing Data with Faster Moving Averages" published in 1994, introduced the Double Exponential Moving Average. Often the name of this indicator generates misunderstanding. This technique is not simply based on the moving average of a moving average, but it is a combination of the two.

Its aim is to better identify trends reducing the delay that characterizes the so-common EMA: with an equal number of periods considered, in fact, DEMA is far more reactive than the classic EMA. Usually, as with the classic exponential moving average, in order to recognize bullish or bearish trends, technicians use two DEMA considering different periods and the buy and sell signals are generated when the short-term DEMA crosses above the long-term one and when the short-term DEMA crosses below the long-term one, respectively. This can be clearly noticed in Figure 3.1, where there are two DEMA and two EMA, both considering fourteen and twenty-one periods, respectively, applied to the exchange rate between the Australian Dollar and the U.S. Dollar between October 22nd and November 1st, 2010.

Figure 3.1: Comparison of EMA and DEMA



Source: www.forexwiki.it

The equation to compute the DEMA is in fact described by Formula 3.3, where EMA_t is an exponential moving average of the closing prices and the second term is the EMA of the so-computed EMA.

$$DEMA_t = 2 * EMA_t - EMA(EMA_t) \quad (3.3)$$

Empirical studies have shown how the analysis of the combination of two DEMA can provide less false signals than a technique based on the combination of two single exponential moving averages.

Moving Average Convergence Divergence (MACD)

The MACD is substantially the difference between two exponential moving averages considering different periods, precisely twelve and twenty-six periods respectively.

Anyways, there are two methods to use the MACD. One consists in compare it to the zero line, resulting in bullish signals if the MACD crosses from below to above zero and in bearish signals if the reverse happens. The second, widely used, method consists in compare the MACD to a linked indicator (graphically represented as a line, and thus called Signal Line) that is nothing more than an exponential moving average, generally considering nine periods, of the MACD. In this case a bullish signal is generated if the MACD crosses from below to above its exponential moving average, while a bearish signal is generated if the reverse happens. The Moving Average Convergence Divergence is simply computed through the following formula, where the 12 and 26 indicates the number of periods considered and the EMA is computed with Formula 3.1.

$$MACD = EMA_{12} - EMA_{26}. \quad (3.2)$$

Also the Signal Line is computed with Formula 3.1, but in place of the closing price there is, obviously, the MACD.

Accumulation Distribution Index (ADI)

The Accumulation Distribution Index, a kind of weighted volume indicator, is based on the widespread belief that exchange volumes, in a certain manner, foresee the price trend.

First of all, we have to consider another indicator: the Close Location Value (CLV), which focuses on the price trend of a given time frame, studying the relationship between open and closing price and the price range of the period considered, thus providing an accurate information. The value of the CLV is easily obtained through Formula 3.4.

$$CLV = \frac{[(C_t - L_t) - (H_t - C_t)]}{(H_t - L_t)} \tag{3.4}$$

where C_t is the closing price of the time frame considered, usually a day, L_t is the minimum value reached by the price during a day and H_t is the maximum value reached by the price during the day. Then, the Accumulation Distribution Index is simply computed as:

$$ADI = CLV * V_t \tag{3.5}$$

where V_t is the value of the exchange volumes.

This indicator, as many other ones, is used looking at the divergences between the indicator's trend and the prices' one, as well as to confirm the strength of a trend, smoothing, as more as possible, unreliable price movements providing false signals.

What analysts have seen, for example, is that at the end of a correction in a bullish market, it is very likely that the Accumulation Distribution Index will start to increase again, reaching very quickly new maximums, higher than before the correction, before prices behave in the same way, thus predicting the continuation of the bullish trend with satisfactory accuracy.

Disparity Index

The Disparity Index represents, in percentage terms, the position of the last closing price of an asset with respect to a moving average, usually the exponential one, of the price. It can take positive or negative values: a positive value indicates that the price is increasing quickly, while a negative value indicates that the price is decreasing rapidly; a value of zero, finally, means that the current price is exactly consistent with its moving average.

A bullish trend signal is generated when such an index crosses from below to above zero, indicating the increasing pressure operated by the buyers. Conversely, a bearish trend signal is generated when it crosses from above to below zero, indicating a growing pressure from sellers.

The Disparity Index is computed through the following Formula 3.6.

$$Disparity\ Index = \frac{C_t - EMA_{13}}{EMA_{13}} * 100 \tag{3.6}$$

where C_t is the last closing price and EMA_{13} is an exponential moving average considering the last thirteen closing price. The number of periods considered by this indicator, in fact, is thirteen, but it possible to consider different periods depending on the strategy to be pursued. Its developer, Steve Nison, states that the Disparity Index can also indicate whether an activity is in a condition of overbought, in the case of a positive value, or oversold, in the case of a negative value. Therefore the presence of a bullish trend is suggested when the asset is overbought and the presence of a bullish trend is, instead suggested when the asset is oversold.

Stochastic Oscillator

Stochastic oscillator is one of the most used technical indicators in the field of Technical Analysis. Created by George Lane in 1970, this indicator relates the closing prices of an asset to its range over a time period. The stochastic oscillator identifies, like other indicators, overbought and oversold areas, which provide to the trader buy and sell signals on a particular financial instrument. The computation of this oscillator is based on a kind of ratio between the minimum and maximum values of the price over a given period of time. This computation results in the parameter $\%K$, known as the fast line, from which, then, is computed the parameter $\%D$, known as the slow line. The former is called fast because it is more sensitive to price variations, while with the latter buy and sell signals are generated. The two parameters are computed using Formula 3.7 and 3.8.

$$\%K = \frac{C_t - L_n}{H_n - L_n} * 100 \tag{3.7}$$

$$\%D = EMA_3(\%K) \tag{3.8}$$

where C_t is the last closing price and L_n and H_n represent, respectively, the lowest and the highest values reached by the price during the periods considered n.

Thus, $L_{14} = 28$, for example, means that the minimum price, over the last fourteen periods, is 28. The $\%D$, instead, is an Exponential moving average of the last three observation of the $\%K$.

As we said, using this oscillator, traders are able to detect overbought and oversold areas. Particularly, the oscillator varies between zero and one hundred and an overbought phase is identified when both the $\%K$ and the $\%D$ parameter fluctuate between eighty and one hundred, while an oversold phase is identified when the parameters moves between zero and twenty.

There are, however, three main variants of the stochastic oscillator:

- Fast Stochastic Oscillator: it is the original oscillator described so far, based on George Lane's original formula.

  The parameter $\%K$ is computed considering fourteen periods, while the parameter $\%D$ is the three-periods exponential moving average of $\%K$. This formula is quite criticized because the oscillator obtained is very reactive, making difficult its application.

- Slow Stochastic Oscillator: this version is directly derived from the Fast Oscillator. In fact, in order to slightly smooth its reactivity, the parameter $\%K$ of the Slow Oscillator is computed as a three-periods exponential moving average of the parameter $\%D$ of the Fast Oscillator. The parameter $\%D$ of the Slow Oscillator, then, is computed as a three-periods exponential moving average of the parameter $\%K$ of the Slow Oscillator. Using this version, however, the overbought area is identified between seventy and one hundred and the oversold area is identified between zero and thirty.

- Complete Stochastic Oscillator: this is the most used version, because it is based on the Slow version, but in this case the parameters are completely customizable in terms of number of periods[26].

[26] C. Mitchell, "*Ultimate Guide to the Stochastic Oscillator*", 2014, Trading Indicators, www.traderhq.com.

The greatest advantages in using this oscillator come when the market is in a so-called lateral phase, that is when there is not a clear trend, as this oscillator will indicates overbought and oversold areas with the consequently imminent reverse of the price trend. When there is a clear and well-defined trend, instead, since it could last from a couple of days to several weeks, the oscillator, as a result, could provide too many false signals.

Aroon Oscillator

The Aroon oscillator, developed in 1995 by Tushar Chande, is used as a measure of the strength of a trend. Aroon is a Sanskrit word that means "dawn's early light" or "the transition between day and night"[27]. The basic principle of this indicator is that the price of an asset should close near its maximum value, reached during the period under consideration, during bullish trend and near its minimum value, reached during the period under consideration, in bearish trend. It has been created in order to help the trader to find the beginning of a new market trends and to define its strength.

It is composed by two parameters, the Aroon Up and the Aroon Down.

The former represents, as a percentage, the number of days passed from the highest maximum of the period considered and the latter, instead, represents, as a percentage, the number of days passed from the lowest minimum of the period considered. The two parameter are then computed as in Formula 3.9 and 3.10.

$$Aroon\ Up = \frac{n - number\ of\ days\ passed\ from\ the\ max}{n} * 100 \qquad (3.9)$$

$$Aroon\ Down = \frac{n - number\ of\ days\ passed\ from\ the\ min}{n} * 100 \qquad (3.10)$$

where n is the number of periods considered that, usually, is fourteen.

These parameters vary between zero and one hundred. If the last closing price is the new maximum of the period, for example, the Aroon Up is equal to one hundred; if, instead, the maximum of the period was at the beginning of the series considered, the Aroon Up is equal to zero. An analogous reasoning is valid for the Aroon Down, with the minimum in place of the maximum.

[27] www.stockcharts.com, ChartSchool, Technical Indicators and Overlays.

An Aroon Up above seventy, or an Aroon Down below thirty, are evidence of a bullish trend, while the reverse, Aroon Down above seventy and Aroon Up below thirty, are evidence of a bearish trend. When both the parameters move parallel, without a clear direction, it can be considered as a consolidation of the trend. When, instead, the Aroon Up crosses from below to above the Aroon Down, it is a strong signal of a bullish trend and when the Aroon Down crosses from below to above the Aroon Up, it is a strong signal of a bearish trend. The Aroon Oscillator, finally, is the difference between the Aroon Up and the Aroon Down, thus making it fluctuating between minus and plus one hundred. Then the beginning of a new trend is signaled by the changes in the sign of the Aroon Oscillator. Particularly, a buy signal is generated when it changes from negative to positive and, analogously, a sell signal is generated when it changes from positive to negative.

Momentum and Rate of Change

The Momentum is an indicator that allows identifying and anticipating trend reversal. Its computation is very simple: in fact, it is nothing more than the difference between the last closing price and the closing price of n periods ago, as shown in Formula 3.11.

$$Momentum = C_t - C_{t-n} \tag{3.11}$$

Just as any type of vehicles decrease its speed before changing direction, in the same way, prices decrease their growth speed before a trend reversal. If we want the momentum expressed as a percentage, it is sufficient to divide it by the closing price of n periods ago and then multiply the result by one hundred: the so-computed result is called Rate of Change, shown in Formula 3.12.

$$RoC = \frac{C_t - C_{t-n}}{C_{t-n}} * 100 \tag{3.12}$$

where C_t is the actual closing price and n is the number of periods considered.

Bullish and bearish signals are generated when the indicator crosses from below to above zero, or a specific value determined in advance, and when the indicator crosses from above to below zero, or a specific value determined in advance, respectively.

Know Sure Thing Oscillator (KST)

KST Oscillator is a technical indicator used to determine a kind of momentum in price trends. It fluctuates above and below the zero line, providing trading signals based on the crossover with a signal line and on the divergences between oscillator's trend and price's trend. Martin J. Pring developed this indicator with the premise that price trends, at any time, are determined by the interactions of several economic cycles considering different time horizons. In fact, he noticed that, in some cases, when there is a trend reversal, the movements of the price are not always the same. He concluded that, when a short-term economic trend changes direction, probably it would not have a big impact on the price because there are other longer and more important trends overlapping it. On the other hand, if two or three medium and long-term trends change direction, then a substantial price movement is likely to happen. Thus, Pring elaborated the KST Oscillator, which is nothing more than a sum of the weighted averages of four price's rate of changes considering different periods. Substantially, he argued that, given the impossibility of following a large number of indicators, the solution lies in the construction of a single indicator that contains the characteristics of the cycles influencing the market, combining them in an appropriate manner. The computation is divided in three steps: first of all it is necessary to compute the four rates of change with Formula 3.12. In this case n, that is the number of periods considered, assumes the values of ten, fifteen, twenty and thirty for the four rates, respectively. Then, it is necessary to compute a simple moving average of each rate of change, considering ten periods for the first three rates and fifteen periods for the last rate. Finally, the KST can be computed through the Formula 3.13.

$$KST = SMA_{10}(RoC_{10}) + \left(2 * SMA_{10}(RoC_{15})\right) + \\ + \left(3 * SMA_{10}(RoC_{20})\right) + \left(4 * SMA_{15}(RoC_{30})\right). \tag{3.13}$$

All the numbers of periods considered for the construction of the parameters described so far are suggested by Pring, the developer of the oscillator, but nothing preclude an agent from changing them based on the asset or market considered. In order to generate buy and sell signals, then, another parameter, called signal line, is created: it is simply an exponential moving average of the KST oscillator, which consider nine periods. Purchase signals occur when the KST crosses over the zero line, or when it crosses over the signal line.

Sell signals, instead, occur when the KST crosses below the zero line, or when it crosses below the signal line. When the KST remains above zero during a bullish trend, this means that the trend is confirmed, while when it remains below zero during a bearish trend, it means that the trend is confirmed. Finally, it could even be compared to the price: indeed, if the price reaches a new maximum while the KST does not, it means that the price momentum is decreasing, signaling that the trend is going to changes from upward to downward, and vice versa when the price reaches a new minimum while KST does not. This can be clearer looking at Figure 3.2, which, considering daily data of Azimut's share price, shows these divergences between the price's trend and the oscillator's trend between February 26th and April 4th 2014 and between April 13th and May 18th 2015. The upper chart is the price's time series represented with candle bar, that is, showing the minimum, open, close and maximum price of the timeframe selected (daily in this case), while the lower chart shows the KST Oscillator as a blue line and the signal line as the pink one.

Figure 3.2: Divergences between price's trend and KST Oscillator's trend

Source: Personal representation, based on data from www.prorealtime.com

Coppock Curve Indicator

The Coppock Curve has been developed, in 1962, by Edwin Coppock. The indicator is computed as a weighted moving average of the sum of two different Rate of Change. The default periods are eleven and fourteen for the two Rate of Change and ten for the weighted moving average.

The main objective of this indicator is to highlight the accumulation periods that anticipate the bullish trends and the distribution periods that anticipate the bearish ones[28]. Formula 3.14 describes its computation.

$$Coppock\ Curve\ Indicator = WMA_{10}(RoC_{11} + RoC_{14}). \tag{3.14}$$

The weighted moving average considering ten periods requires some steps: first of all, multiply the coupled RoCs by a coefficient (from one to ten) assigning one to the first observation and ten to the last one, sum them in order to obtain a weighted sum, and finally divide the result by the sum of the coefficients, that is fifty-five. The signals are highlighted by the crosses of the zero line; when the indicator crosses it from below to above it, in fact, a bullish signal is generated, while when the indicator crosses from above to below zero, a bearish signal is generated.

As with other indicators, the divergences between the trend of the Coppock Indicator and the price trend highlight a possible inversion of the trend. Thus, for example, if the Coppock indicator is increasing while prices are still falling, this could anticipate a possible bullish reversal.

Williams Percentage Range (%R)

Williams Percentage Range is a simple but still effective technical analysis' tool, developed in 1973 by Larry Williams. It is very similar to the stochastic oscillator but, while the latter uses the difference between the closing price and the minimum price of a given period, for the construction of the Williams $\%R$ it is necessary to consider the difference between the closing price and the maximum price of a given period.

[28] The first stage of a bull market is referred to as the accumulation phase, which is the start of the upward trend: this phase typically comes at the end of a downward trend, thus once reached a low and attractive price, informed investors enter in the market. In the last stage the upward trend, instead, called distribution phase, the market is characterized by irrational exuberance, last buyers enter in the market thinking that the trend will continue but informed investors start to close their positions: this anticipate a bearish trend.

It should be more clear looking at the Formula 3.15.

$$Williams\ \%R = \frac{H_n - C_t}{H_n - L_n} * (-100) \tag{3.15}$$

where C_t is the last closing price and H_n and L_n represent, respectively, the highest and the lowest values reached by the price during the periods considered n. The indicator fluctuates between zero and minus one hundred, thus, practically, on an inverted scale compared to normal oscillators such as the stochastic one. It is used to recognize overbought areas, identified at values above minus twenty, and oversold areas, identified at values below minus eighty. This oscillator is particularly efficient in identifying some of the characteristic situation of the market, most of all, the phases of exhaustion of a trend. In fact, in a strong bullish market, if the prices are no more able to close near the maximums, it could be an indicator of the exhaustion of the trend, that is, a trend reversal is likely to occur, and thus giving the opportunity to open a bearish position. The same reasoning is valid for bearish trend.

The oscillator is often used to identify also the divergences with the price, that is, a bearish divergence occurs when prices make new maximums while the oscillator has declining maximums and, on the other hand, a bullish divergence occurs if prices reach new minimums while the oscillator has growing minimums. Since, however, the $\%R$ could provide false signals, sometimes traders add a moving average to the $\%R$ chart, in order to increase the possibility to avoid them.

Average Directional Index (ADX)

The Average Directional Index was elaborated by Welles Wilder in order to recognize and analyze the transition from a congestion, or price lateral movement, to a defined trend, and then to measure the strength of this trend. The ADX does not consider the direction of the trend, that is, it quantifies the strength of a trend without taking into account whether it is an increasing or a decreasing one.

In order to provide to the agent the information of the direction of the trend, however, the ADX is used with other two parameters, which, in addition, compose the ADX itself. These parameters are the Positive and the Negative Directional indexes (+DI and –DI): the former measures the strength of the positive trend and the latter measures the strength of the negative one.

The computation of the ADX requires some steps: first of all we have to compute the Up and Down movements, occurred between two periods, with Formula 3.16 and 3.17.

$$Up\ Movement = H_t - H_{t-1} \tag{3.16}$$

$$Down\ Movement = L_{t-1} - L_t \tag{3.17}$$

where H_t and L_t represents respectively the maximum and the minimum values reached by the price in the period $[t-1, t]$.

From these two values we can compute the Positive and the Negative Movements: if the Up Movement is higher than zero and higher than the Down Movement, then the Positive Movement (+DM) is equal to the Up Movement, otherwise it is zero. Similarly, if the Down Movement is higher than zero and higher than the Up Movement, then the Negative Movement (-DM) is equal to the Down Movement, otherwise it is zero. Then, in order to compute the Positive and Negative Directional Indexes it is necessary to calculate the True Range of the price. As shown in Formula 3.18, it is the maximum between the difference between the actual maximum and minimum prices, the absolute value of the difference between the actual maximum and the previous closing price and the absolute value of the difference between the actual minimum and the previous closing price.

$$True\ Range = max\{H_t - L_t; |H_t - C_{t-1}|; |L_t - C_{t-1}|\}. \tag{3.18}$$

Then the Positive and the Negative Directional Indexes are computed using Formula 3.19 and 3.20.

$$+DI = \frac{EMA_{14}(+DM)}{EMA_{14}(True\ Range)} * 100, \tag{3.19}$$

$$-DI = \frac{EMA_{14}(-DM)}{EMA_{14}(True\ Range)} * 100. \tag{3.20}$$

Finally, the Average Directional Index can be computed using Formula 3.21.

$$ADX = EMA_{14}\left(100 * \frac{|(+DI) - (-DI)|}{(+DI) + (-DI)}\right). \quad (3.21)$$

The developer of this indicator, Wilder, originally used a smoothed moving average, while, in the practice, actual technicians use an exponential moving average considering fourteen periods.

ADX by its definition can take values between zero and one hundred, but values above sixty are actually very rare. Usually when the value of ADX is growing above twenty, it is believed that the current trend is strengthening, while when it is falling below forty, it is believed that the trend is slowing down and, thus, that it is likely to reverse.

Relative Strength Index (RSI)

The Relative Strength Index has been created by Welles Wilder and presented in his book "New Concepts in Technical Trading System", in 1978.

Despite what its name may suggest, its purpose is not to measure the relative strength of an asset (compared to the market, for example), but to measure the speed of its price movement. Like many indicators seen so far, it oscillates between zero and one hundred and, according to the default setting, a value above seventy is considered a sign of overbought area, while the oversold one is identified with values below thirty. It is calculated using the ratio between the moving average of positive price variations and the moving average of negative price variations. Precisely, it is computed using Formula 3.22.

$$RSI = 100 - \frac{100}{1 + \frac{EMA_{14}(UM)}{EMA_{14}(DM)}}. \quad (3.22)$$

where the EMAs of UM and DM are the exponential moving averages of the upward and downward movements occurred during the last fourteen periods. A movement is defined as upward when the last closing price is higher than the previous one. Analogously, it is defined as downward when the last closing price is lower than the previous one. The exponential moving average considers, by default, fourteen periods but it can be increased or decreased to modify the sensitivity of the indicator.

True Strength Index (TSI)

The True Strength Index, elaborated by William Blau, is a kind of momentum indicator, used by traders to recognize overbought and oversold areas. The TSI is based on the combination of two exponential moving averages, usually considering thirteen periods, which in turn, are based on two exponential moving averages considering twenty-five periods. First, it is necessary to compute the price momentum, as shown in Formula 3.11, and a second price momentum considering absolute values (thus, this second momentum will have only positive variation).

Then, we have to compute a twenty-five periods exponential moving average of both the momentums.

The third steps consists in generate the so-called double smoothing, computing a shorter-term exponential moving average, usually of thirteen periods, of both the EMA considering twenty-five periods. Finally, the True Strength Index is computed as the ratio between the thirteen-periods EMA that considers price variations and the thirteen-periods EMA that considers absolute price variations.

Formula 3.23 describes the computation.

$$TSI = \frac{EMA_{13}\big(EMA_{25}(momentum)\big)}{EMA_{13}\big(EMA_{25}(|momentum|)\big)} * 100 \tag{3.23}$$

The numerator therefore gives the sign to the TSI, depending on whether the price is rising or falling. As for the Relative Strength Index, which assumes particular significance when its value is below thirty or above seventy, the TSI highlights oversold areas if it is below minus twenty-five, or overbought areas if it is above twenty-five. In fact, a series of large positive price changes implies high positive values for the TSI, signaling a strong upward momentum, while, on the other hand, strong price falls push the TSI to assume negative values.

Volatility Ratio

The Volatility Ratio is based on the True Range of an asset price. As already stated, while describing the Average Directional Index, the True Range is defined as the highest value among these three computation: (Current Maximum Price – Current Minimum Price), |Current Maximum Price – Previous Closing Price| and |Current Minimum Price – Previous Closing Price|, as described in Formula 3.18.

Once computed the True Range, it is necessary to proceed with the computation of the Average True Range, developed by Welles Wilder in the 1970's, which is nothing more than an exponential moving average of the True Ranges, usually considering fourteen periods. Then, the Volatility Ratio is computed as the current True Range divided by the Average True Range, as shown in Formula 3.24.

$$Volatility\ Ratio = \frac{TR_t}{ATR_{14}}. \tag{3.24}$$

This ratio is called Volatility Ratio because it provides a measure of the fluctuations of the actual price with respect to its usual range of the period considered. When it increases above 1.5, or decreases below 0.5, it signals a possible imminent trend reversal.

3.1.2 Data Collection

The core idea behind this paper is to assess whether or not a trading system based on an ANN is sustainable and profitable. In order to perform this valuation, then, we thought to compare the performances of two mutual funds with the performance obtained applying our trading system. Thus, given all the publicly available information[29], we selected nine stocks among the compositions of each fund, in order to replicate as accurately as possible their investments[30]. We focused on European mutual funds investing only in European stocks. Precisely, we selected Pioneer Funds – Italian Equity E Eur, Isin: LU0085424223 and Morgan Stanley Investment Funds – Eurozone Equity Alpha Fund A, Isin: LU0109967165. We thus selected the following stocks: Intesa San Paolo (8.12%), Eni (7.10%), Enel (6.64%), UniCredit (5.23%), Atlantia (4.30%), Assicurazioni Generali (4.22%), Tenaris (3.51%), Luxottica Group (3.47%) and STMicroelectronics (3.23%) which compose the 45.82% of the Pioneer Fund.

[29] We obtained all the information from www.morningstar.com.

[30] In order not to worsen, in computational terms, the training process of the network, we decided to consider the first 10 stocks in which both the Funds invest the most, but with the constraint of being listed for at least ten years: this constraint led us to select the 9 stocks described (as an example, Fiat Chrysler Automobiles' stock has been excluded from our analysis because it is listed since October 2014).

The stocks SAP (5.04%), Allianz (4.73%), Bayer (4.67%), ASML Holding (4.40%), BNP Paribas (4.33%), Siemens (4.33%), Unilever (4.30%), ING Groep (4.14%) and Airbus (3.82%), instead, compose the 39.76% of the Morgan Stanley Fund[31].

The prices' time series are from 04/01/2005 to 29/09/2017, thus resulting in 3226 observations for each stocks, both the compositions of the funds have been checked on 29/11/2017 and all the stocks have been downloaded from www.bloomberg.com.

Carrying out the valuation with two different funds will allows us to assess the performance of the ANN, its generalization capacity, that is the ability to learn the existent relationship between input and output and thus to recognize them when new inputs are presented to it, its forecasting ability, whether or not the portfolio selection affects significantly the ANN's performance and, finally, if it is plausible that fund managers use ANNs to make investment decisions.

3.1.3 Data Preprocessing, Training, validation and test sets

Data preprocessing is a well-known and used approach to transform the data in order to improve and to reduce the time required for the ANN's learning phase. This process consists in the transformation of the inputs and the target outputs to normalize as far as possible the distribution of the variables, to minimize the presence of noise and biases, highlight important relationships and detect possible trends. Usually the most common transformations are the first differentiation, the logarithmic transformation and rescaling the variables between the upper and lower bounds of the transfer function. In our model, starting from the assets' prices, we computed the logarithmic returns to be used as target outputs. Then, in order to minimize the presence of noise and biases, we deleted 27 observations from the inputs and the target outputs of Morgan Stanley Fund and 14 observations from the inputs and target outputs of the Pioneer Fund because considered outliers (we deleted all observations corresponding to returns outside the range between plus and minus fifteen percent). Finally, since we use the hyperbolic tangent as hidden layers' activation function, using the Matlab function 'mapminmax', thus using as references the maximum and the minimum values of each series, we rescaled all the variables between the lower and upper bound of the transfer function, that is, between minus and plus one.

[31] The percentages next to the stocks' name represent the percentage of the Funds' reserves invested in the stocks.

Once performed these transformations, then, in order to assess the forecasting ability of the network, we left approximately the last 500 observations of the series, as never-seen inputs, to be successively presented to the ANN after the training phase. Then we proceeded to divide the remaining 2644 observations in three different sets, training, testing and validation sets, 70%, 15% and 15% respectively. The default method to divide the series is with the Matlab function 'dividerand' but since with this function the division is carried out randomly and thus it destroys the correlation between the data we chose the Matlab function 'divideblock'. This function allows the ANN to be trained exactly on the first 70% of the observation, tested on the successive 15% and validated on the last 15% of the observations[32]. The division of the series into three separate sets is common practice: the training set is used to adjust and define the connection weights between the neurons, the validation set is used to verify that the level of accuracy achieved in the training set is maintained with new data, not present in the training set and, finally, the test set is used to test and confirm the actual generalization capacity of the network.

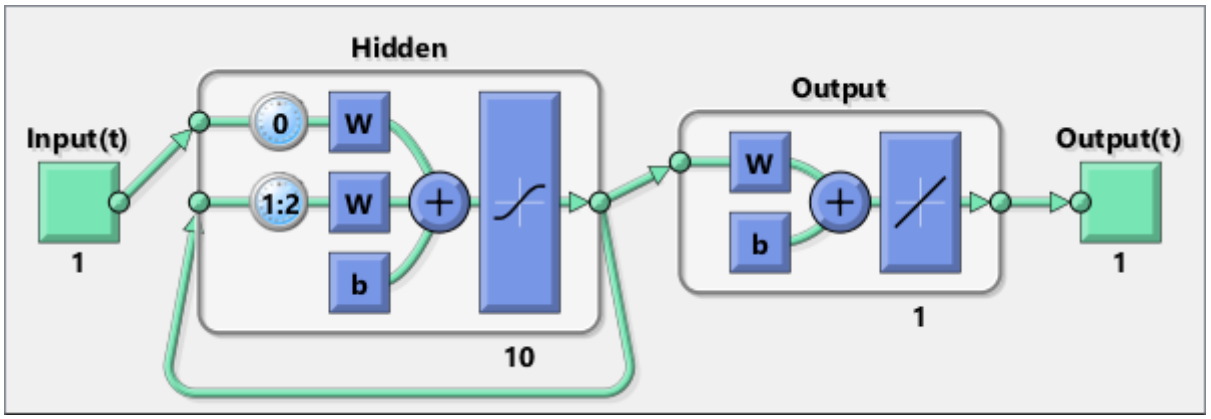
3.1.4 Network Design: Paradigms, Evaluation Criteria and Training

The ANN we aimed to design is of the type of Recurrent ANNs, which, as we already said in the previous chapter, are characterized by the presence of a time component, meaning that the ANN, once computed an output, is able to present it to itself as a new input in a successive iteration. Precisely the network we used is a Layer Recurrent Neural Network ('layrecnet' with MatLab notation) which substantially is a generalization of the original Elman ANN. In fact, the 'layrecnet' function allows to construct a network with more than one hidden layer and to set up different activation function for the output neurons (thus no more only linear activation function). Starting from the default configuration of this network, we ran different experiments and with a trial and error process we modified its structure, almost changing the number of hidden neurons: since we have a total number of 31 inputs for each stock, our ANN, for the first sets of experiments is composed of 279 input neurons, two hidden layers with a number of neurons that varies between 31 and 187 each and finally nine output neurons.

[32] Using the function 'dividerand' the training of the ANN is performed on the 70% of the observations, but they are chosen randomly, that is, the ANN could be trained starting from the 50th observation, passing to the 1450th one and finishing with the 3rd one. This is why 'dividerand' destroys the correlations between the data.

Since 31 inputs could result in too much noise in the learning phase, in some experiments, we also tried to run the ANN using a fewer number of inputs (equal to seven): this allowed us to set different numbers of hidden neurons, particularly, between 63 and 253.
The activation function of all the hidden neurons is the hyperbolic tangent and the output neurons' one is linear, thus resulting in an output bounded between plus and minus one.
In Figure 3.3 there is a graphic representation of the Layer Recurrent ANN with one input neuron, one hidden layer with ten hidden neurons and, finally, one output neuron.

Figure 3.3: Graphic representation of Layer Recurrent Neural Network



Source: www.mathworks.com

The performance of the ANN are measured by the mean squared error, that is the error function to be minimized, for which we defined a goal equal to 0.001 in order to avoid overfitting.
During several experiments, we tried different number of training iterations, varying between five hundred and three thousand. Finally, the training function we used is the back-propagation with momentum equal to 0.9 and adaptive learning rate starting from 0.01 ('traingdx' with MatLab notation).

3.1.5 Implementation

In the first implementation we trained the ANN providing it with the inputs and the target output in such a way that it should have learnt the relationships between the observation at time t of the inputs and the observation at time $t + 1$ of the target outputs.

However, in the training phase the results were not very satisfactory, that is, the network did not learn to associate the inputs to the correct output with a sufficient level of generalization and thus, in the prediction phase, where new inputs have been presented to the ANN for the first time, the generalization was even worst. Thus, in order to obtain better results in both the training and the prediction phase, we thought to train the ANN in such a way that it should have learnt the existent relationship between a certain number of previous observation of the inputs, that is, from $t-n$ to t, and the observation at time $t+1$ of the target output. Thus, we ran three sets of experiments with n equal to 4, 9 and 19 respectively and, for each set, we implemented the ANN with different numbers of neurons, precisely 31, 63, 125 and 187 for both the hidden layers[33] and one experiment with 187 and 63 neurons respectively for the first and the second hidden layer, totally resulting in fifteen experiments for both the datasets. Even if there is not a recognized method to successfully choose the right number of hidden neurons (it depends on the problem to be solved and on the network architecture), in 1956, the Russian mathematician, Andrej Nikolaevič Kolmogorov, in its theorem[34], stated that any continuous function can be computed using a Recurrent ANN with k input neurons, $2k+1$ hidden neurons and m output neurons and with totally connected layers. Thus, in our case, with 279 input neurons we should have defined 559 neurons for both the hidden layers. However, in order to define the number of hidden neurons, and to successfully run our network, it has been necessary to increase the virtual memory of the personal computer used for the implementation up to the maximum available, which is 300 gigabytes, so that Matlab could use the hard disk as RAM[35]. This is why we were not able to define $2k+1$ hidden neurons: even after this memory adjustment, it were not possible.

[33] The number of hidden neurons has not been chosen randomly, but it is based on the number of input neurons: in fact, if k is the number of input neurons defined, that is $k=279$, we obtain $\frac{k}{9}=31$, $\frac{2k}{9}+1=63$, $\frac{4k}{9}+1=125$ and $\frac{6k}{9}+1=187$, where 9 is the number of stocks being analyzed simultaneously.

[34] V. Kůrková, "*Kolmogorov's theorem and Multilayer Neural Network"*, 1992, Neural Networks, n. 5, p. 501-506.

[35] The computer used for the implementation is a Lenovo IdeaPad S510p, o.s. Windows 10 Home, CPU Intel® Core™ i7-4500U 1.80GHz 2.40GHz, HD 465Gb, RAM 4Gb, 64-bit operating system, x64-based processor.

Given the huge amount of input and hidden neurons, the number of epochs has been fixed at 500 (one epoch is comprehensive of the forward and the backward phase of all the training examples). The time required to complete each cycle, that is for the whole 500 epochs, ranged between 20 and 120 minutes, depending on the experiments: 20 minutes were required for the cycle with the ANN with 31 neurons in both the hidden layers and training the network linking 5 previous input observations (from $t-4$ to t) to the target output observation in $t+1$, while 120 minutes were required for the cycle with the ANN with 187 neurons in both the hidden layers and training the network linking 20 previous input observations (from $t-19$ to t) to the target output observation in $t+1$.

Then, as we already said, in order to try to reduce the presence of noise and biases in the inputs, we also ran some experiments with a lower number of inputs, precisely seven, thus resulting in a lighter network with 63 input neurons. As with the previous architecture, we ran three sets of experiments in which the network should had linked the observations from $t-n$ to t of the inputs with the observation at time $t+1$ of the target output, with n equal to 4, 9 and 19 respectively. In this case, however, given the substantial lightening of the ANN, we have been able to run each experiments with $4k+1$ (253) and $2k+1$ (127) neurons, respectively, in the first and in the second hidden layer, with 127 neurons for each hidden layer and with $2k+1$ and k (63) neurons, respectively, in the first and in the second hidden layer.

In this case, we fixed the number of epochs at 3000. The time required to complete each cycle, that is for the whole 3000 epochs, ranged between 100 and 310 minutes, depending on the experiments: 100 minutes were required for the cycle with the ANN with 127 and 63 neurons in the two hidden layers and training the network linking 5 previous input observations (from $t-4$ to t) to the target output observation in $t+1$, while 310 minutes were required for the cycle with the ANN with 253 and 127 neurons in the two hidden layers and training the network linking 20 previous input observations (from $t-19$ to t) to the target output observation in $t+1$.

3.2 Results

In this last section of the chapter we will present the results obtained from the network's prediction phase and the related performance deriving from the application of a simple trading strategy.

First of all, since we previously pre-processed the data, rescaling them between -1 and +1 for the training phase, once obtained the output of the prediction phase, it is necessary to post-process them reversing the procedure, in order to make our analysis. Then, it is necessary to define a measure of the goodness of the prediction: we do not want the network to predict as perfectly as possible the assets' return but, for our purposes, it is sufficient that the network predicts the exact sign of the returns.

Thus, taking a cue from a quotation, dated 1996, of "Atti del ventesimo convegno annuale A.M.A.S.E.S."[36] we decided to use two indicators, the Correct Sign (CS) and the Correct Trend (CT): defining with ρ_t and ρ_{t+1} the desired output at time t and $t+1$, respectively, with a_t and a_{t+1} the network output at time t and $t+1$, respectively, and with S the number of observations resulting from the prediction phase, the two indicator are computed as in Formula 3.25 and Formula 3.26.

$$CT = \left(\frac{100}{S-1}\sum_{t=1}^{S-1}\alpha_t\right)\% \in [0,100], \alpha_t = \begin{cases} 1 \; if & (\rho_{t+1}-\rho_t)(a_{t+1}-\rho_t) > 0 \\ 1 \; if & \rho_{t+1} = \rho_t = a_{t+1} \\ 0 & otherwise \end{cases}, \quad (3.25)$$

$$CS = \left(\frac{100}{S}\sum_{t=1}^{S}\beta_t\right)\% \in [0,100], \beta_t = \begin{cases} 1 \; if & \rho_t a_t > 0 \\ 1 \; if & \rho_t = a_t = 0 \\ 0 & otherwise \end{cases}. \quad (3.26)$$

For both the indicators the optimal values are identified in the range between 50 and 100%, that is, the values of CS and CT are considered satisfactory if the network is able to correctly predict the sign and the trend of, at least, the 50% of the returns. Generally speaking, however, we expect the CT to take higher values than the CS, since the latter is a measure of more accurate prediction.

Finally, we will describe the performance obtained from the application of a simple trading strategy.

[36] P. Belcaro, E. Canestrelli, M. Corazza, "*Modelli Previsivi Neurali: Un'applicazione al mercato finanziario Italiano*", 1996, Atti del ventesimo convegno annuale A.M.A.S.E.S. (Associazione per la Matematica Applicata alle Scienze Economiche e Sociali), Università degli Studi di Urbino.

We decided to measure the performance with logarithm returns, obtained using Formula 3.27.

$$r = LN(P_{t+1}) - LN(P_t) = LN\left(\frac{P_{t+1}}{P_t}\right), \tag{3.27}$$

where r is the return, LN is the natural logarithm and P_t and P_{t+1} are, respectively, the initial amount invested and the final amount obtained.

The strategy we applied, simply consists in buying an asset when its price is predicted to increase and sell it when its price is predicted to decrease.

In this case, the optimal results are identified with the values higher than the Fund Performance. We have to specify, however, that, in order to act as similarly as possible like a mutual fund, our strategy does not imply the practice of short selling[37], that is, we will sell an asset only if we already bought it.

We divided a virtual amount of one hundred thousand of euros among the assets, in order to conform to the proportion defined by both mutual funds: for example, the 8.12% invested in Intesa San Paolo by the Pioneer Fund, in our strategy it becomes (8.12/45.82)=17.7% (where 45.82 is the percentage of the total composition of the fund we invested in).

Here below, for each network architecture, we present both the indicators of the goodness of prediction and a comparison between the performances obtained from the application of our Network Based Strategy (NBS), the performances of a buy-and-hold (B&H) strategy and the performances of both the funds during the same period (22/09/2015 – 29/09/2017).

The name of the tables will contain the name of the fund considered, the number of inputs used in the network training phase and the number of inputs' observations the network has been trained to link to the output observation at time $t + 1$. In the tables, the various network architectures are identified with the number of hidden neurons (HN) defined for both the hidden layers ([31 31] HN). We begin describing the results obtained with the experiments in which we provided the networks with 279 input neurons.

[37] The short selling of shares is a financial transaction that consists in the sale of shares without owning them, hoping to buy them back later at a lower price, thus making a profit.

Table 3.1: NET_PIONEER_FUND_279_INPUTS_5D

PIONEER FUND

Fund Performance	6,81%		Inputs' obs.	Outputs' obs.	
B&H Performance	7,36%		$t-4 \rightarrow t$	$t+1$	
	[31 31] HN	[63 63] HN	[125 125] HN	[187 187] HN	[187 63] HN
CS	48,59%	49,15%	48,87%	49,23%	49,21%
CT	73,26%	68,68%	69,54%	67,07%	68,83%
NBS Performance	1,49%	-6,04%	3,28%	8,27%	9,73%

Even if the results of the CS indicator are not satisfactory, it is interesting to point out that the network actually made correct sign predictions for some stocks: for example, the percentage of correct sign predictions of Enel's and Eni's returns is higher than 50% in 4 architectures out of 5, or the percentage of correct sign predictions of Assicurazioni Generali's and Unicredt's returns is higher than 50% in 2 architectures out of 5. Moreover, as we can notice, applying the trading strategy with the last two architectures we obtained positive performances, higher than the Fund's ones.

Table 3.2: NET_PIONEER_FUND_279_INPUTS_10D

PIONEER FUND

Fund Performance	6,81%		Inputs' obs.	Outputs' obs.	
B&H Performance	7,36%		$t-9 \rightarrow t$	$t+1$	
	[31 31] HN	[63 63] HN	[125 125] HN	[187 187] HN	[187 63] HN
CS	49,90%	48,82%	49,39%	49,45%	48,16%
CT	70,32%	69,59%	65,05%	67,96%	63,16%
NBS Performance	5,63%	0,80%	1,68%	-2,75%	7,90%

Also in this case the network actually made correct sign predictions for some stocks: for example, the percentage of correct sign predictions of Atlantia's, Enel's and Unicredit's returns is higher than 50% in 3 architectures out of 5.

Table 3.3: NET_PIONEER_FUND_279_INPUTS_20D

PIONEER FUND

Fund Performance	6,81%		Inputs' obs.	Outputs' obs.	
B&H Performance	7,36%		$t-19 \rightarrow t$	$t+1$	
	[31 31] HN	[63 63] HN	[125 125] HN	[187 187] HN	[187 63] HN
CS	48,78%	49,59%	49,04%	47,63%	50,03%
CT	67,11%	72,86%	63,36%	66,31%	74,07%
NBS Performance	-1,02%	15,53%	8,78%	-3,74%	5,71%

As expected, looking at the results presented, the Correct Trend indicator assumes values higher than the 50% and higher than the Correct Sign indicator in all the experiments. The Correct Sign indicator, instead assumes a value higher than the 50% only in one experiment, that is, the network is able to correctly predict the 50,03% of the sign of the assets' returns only when we set 187 and 63 neurons in the first and second hidden layers, respectively, and training the network presenting the inputs and the target outputs in such a way that it should have learnt the relationship between twenty previous inputs observations (form time $t-19$ to t) and the output observation at time $t+1$. We have to specify, however, that in this experiment the number of epochs has been fixed at one thousand. However, also in this case, the network made correct sign predictions for some stocks: the percentage of correct sign predictions of Atlantia's and Eni's returns is higher than 50% in 4 architectures out of 5. Finally, it has to be noticed that, even if 14 network architectures out of 15 do not reach a good percentage of correct predictions, the performance obtained from the application of the simple trading strategy based on the networks' predictions, are higher than the fund performance, and even higher than the buy-and-hold strategy's performance, in 5 cases out of 15, and higher than zero in 11 cases out of 15, meaning that the network is able to predict the sign of the 50% most significant price changes, allowing us to take advantage of them.

Considering the CS indicator and the CT indicator, the best experiment can be identified in the network architecture with 187 and 63 neurons in the first and second hidden layers, respectively, and training the network to recognize the relationships between 20 previous input observations and the output observation at time $t+1$, while considering only the performance obtained with our trading strategy, the best experiment is identified, in the same set of experiments, with the network architecture with 63 neurons in both the hidden layers, with which, applying our strategy, we obtained a return equal to 15,53%, 872 basis point more than the Fund's one.

Table 3.4: NET_MORGAN_STANLEY_FUND_279_INPUTS_5D

MORGAN STANLEY FUND					
Fund Performance	19,03%		Inputs' obs.	Outputs' obs.	
B&H Performance	30,60%		$t-4 \rightarrow t$	$t+1$	
	[31 31] HN	[63 63] HN	[125 125] HN	[187 187] HN	[187 63] HN
CS	48,65%	49,86%	51,37%	48,98%	49,46%
CT	54,37%	59,35%	59,82%	58,65%	59,20%
NBS Performance	24,96%	15,23%	28,26%	13,19%	23,77%

Table 3.5: NET_MORGAN_STANLEY_FUND_279_INPUTS_10D

MORGAN STANLEY FUND					
Fund Performance	19,03%		Inputs' obs.	Outputs' obs.	
B&H Performance	30,60%		$t-9 \rightarrow t$	$t+1$	
	[31 31] HN	[63 63] HN	[125 125] HN	[187 187] HN	[187 63] HN
CS	51,70%	50,64%	50,93%	50,35%	47,61%
CT	55,95%	62,99%	55,76%	58,02%	56,55%
NBS Performance	15,02%	21,40%	30,08%	14,05%	9,63%

As shown in Table 3.4, even with that set of experiments ran on Morgan Stanley's stocks, the network actually made correct sign predictions for some stocks: indeed, the percentage of correct sign predictions of Asml Holding's and Unilever's returns is higher than 50% in 4 architectures out of 5 and the one of Bayer's, Bnp's and Sap's returns is higher than 50% in 3 architectures out of 5.

Even with the Morgan Stanley Fund, we can notice that the Correct Trend indicator always assumes higher values than the Correct Sign indicator. In this case, however, the Correct Sign indicator assumes values higher than the 50% in 7 experiments out of 15.

Finally, the performance obtained from the application of our strategy are higher than the fund performance in 6 experiments out of 15 and higher than zero in all the experiments.

Table 3.6: NET_MORGAN_STANLEY_FUND_279_INPUTS_20D

MORGAN STANLEY FUND					
Fund Performance	19,03%		Inputs' obs.	Outputs' obs.	
B&H Performance	30,06%		$t-19 \rightarrow t$	$t+1$	
	[31 31] HN	[63 63] HN	[125 125] HN	[187 187] HN	[187 63] HN
CS	49,21%	50,50%	47,58%	51,56%	48,37%
CT	54,99%	58,77%	59,15%	53,11%	54,02%
NBS Performance	8,99%	17,13%	9,60%	25,92%	2,66%

Here, the best experiments are identified in the same set of experiments, the one in which the network is trained linking 10 past input observations to the output observation at time $t+1$: considering the CS indicator the best experiment is identified in the architecture with 31 neurons in both the hidden layers, considering the CT indicator the best experiment is identified in the one with 63 neurons in both the hidden layers, while considering the performance of our strategy, the best experiment is identified in the one with 125 neurons in both the hidden layers, with which, applying our strategy, we obtained a return equal to 30,08%, 1105 basis points more than the Fund's one.

Now we pass to describe the results obtained with the experiments in which we provided the networks with 63 input neurons.

Table 3.7: NET_PIONEER_FUND_63_INPUTS_5D

PIONEER FUND			
Fund Performance	6,81%	Inputs' obs.	Outputs' obs.
B&H Performance	7,36%	$t-4 \rightarrow t$	$t+1$
	[127 63] HN	[127 127] HN	[253 127] HN
CS	49,38%	48,33%	48,57%
CT	73,39%	72,98%	73,77%
NBS Performance	4,96%	4,33%	7,83%

Using a lower number of input neurons and different number of neurons in the hidden layers, in this set of experiments, the CS does not assume values higher than 50%. However, even in this case, the network predicted the correct sign of the returns for certain stocks: the percentage of correct sign predictions of Enel's returns is higher than 50% in 2 architectures out of 3 and the one of Atlantia's, Luxottica Group's and Unicredit's returns is higher than 50% in 1 architectures out of 3. However, only in this case the performance obtained with our strategy seems to be more stable with respect to the set of experiments ran using more input neurons.

Table 3.8: NET_PIONEER_FUND_63_INPUTS_10D

PIONEER FUND			
Fund Performance	6,81%	Inputs' obs.	Outputs' obs.
B&H Performance	7,36%	$t-9 \rightarrow t$	$t+1$
	[127 63] HN	[127 127] HN	[253 127] HN
CS	50,57%	50,16%	48,39%
CT	73,56%	73,95%	71,08%
NBS Performance	26,64%	0,23%	3,03%

Table 3.9: NET_PIONEER_FUND_63_INPUTS_20D

PIONEER FUND			
Fund Performance	6,81%	Inputs' obs.	Outputs' obs.
B&H Performance	7,36%	$t-19 \rightarrow t$	$t+1$
	[127 63] HN	[127 127] HN	[253 127] HN
CS	47,79%	49,90%	49,50%
CT	74,80%	71,45%	71,25%
NBS Performance	-7,57%	2,72%	0,72%

Even in this last set of experiments on the Pioneer Fund's stocks, none of the Correct Sign indicators is higher than 50%.

However, the network predicted the correct sign of the returns for certain stocks: the percentage of correct sign predictions of Atlantia's returns is higher than 50% in all the 3 architectures and the one of Luxottica Group's is higher than 50% in 2 architectures out of 3. Among all the 3 sets of experiments, the performance obtained from the application of the simple trading strategy based on the networks' predictions, are higher than the fund performance, and even higher than the buy-and-hold strategy's performance, only in 2 cases out of 9 but higher than zero in 8 experiments out of 9, with a peak of 26,64% of return against the 6,81% of the Fund. Considering the CS indicator and the performance obtained with our trading strategy, the best experiment is identified in the network architecture with 127 and 63 neurons, respectively in the first and second hidden layers and training the network linking 10 past input observations to the output observation at time $t+1$, while considering the CT indicator the best experiment is identified in the network with the same number of hidden neurons and training it linking 20 past input observations to the output observation at time $t+1$.

Table 3.10: NET_MORGAN_STANLEY_FUND_63_INPUTS_5D

MORGAN STANLEY FUND			
Fund Performance	19,03%	Inputs' obs.	Outputs' obs.
B&H Performance	30,60%	$t-4 \rightarrow t$	$t+1$
	[127 63] HN	[127 127] HN	[253 127] HN
CS	48,32%	49,13%	50,03%
CT	72,42%	69,04%	67,63%
NBS Performance	14,73%	11,50%	16,78%

In this set of experiments, involving the Morgan Stanley Fund's stocks, the CS indicator is higher than 50% only in one experiment. However, the network predicted the correct sign of certain stocks' returns: the percentage of correct sign predictions of Airbus's, Bayer's and Sap's returns is higher than 50% in 2 architectures out of 3. In this set, even if we obtained performances higher than zero in all the experiments, none of them beat the Fund's performance.

Table 3.11: NET_MORGAN_STANLEY_FUND_63_INPUTS_10D

MORGAN STANLEY FUND			
Fund Performance	19,03%	Inputs' obs.	Outputs' obs.
B&H Performance	30,60%	$t-9 \rightarrow t$	$t+1$
	[127 63] HN	[127 127] HN	[253 127] HN
CS	49,56%	49,98%	50,00%
CT	65,78%	66,25%	64,18%
NBS Performance	11,29%	20,45%	23,24%

Analogously to the previous set of experiments, even in this case only in one experiment we obtained a CS indicator higher (or equal in this case) than 50%. On the other hand, the network predicted the correct sign of certain stocks' returns: the percentage of correct sign predictions of Siemens's returns is higher than 50% in all the 3 architectures and the one of Airbus's, Allianz's, Bayer's and Sap's returns is higher than 50% in 2 architectures out of 3. In addition, in this case, applying our strategy, in 2 experiments out of 3, we obtained performances higher than the Fund's one, with a peak return of 23,24%, 421 basis point more that the Fund's return.

Table 3.12: NET_MORGAN_STANLEY_FUND_63_INPUTS_20D

MORGAN STANLEY FUND			
Fund Performance	19,03%	Inputs' obs.	Outputs' obs.
B&H Performance	30,60%	$t-19 \rightarrow t$	$t+1$
	[127 63] HN	[127 127] HN	[253 127] HN
CS	51,38%	51,24%	52,03%
CT	57,12%	60,22%	67,44%
NBS Performance	22,44%	21,13%	19,88%

In this last set of experiments on the Morgan Stanley Fund's the CS indicator takes always values higher than 50%. Among all the 3 sets of experiments, the performance obtained from the application of the simple trading strategy based on the networks' predictions, are higher than the fund performance in 5 cases out of 9, (3 of which in the last set) but higher than zero in all the experiments. Considering the CS indicator, the best experiment is identified in the network architecture with 253 and 127 neurons, respectively in the first and second hidden layers and training the network linking 20 past input observations to the output observation at time $t+1$, considering the CT indicator the best experiment is identified in the network architecture with 127 and 63 neurons, respectively in the first and second hidden layers and training it linking 20 past input observations to the output observation at time $t+1$.

Finally, considering the performance obtained with our trading strategy, the best experiment is identified in the network architecture with 253 and 127 neurons, respectively in the first and second hidden layers and training the network linking 10 past input observations to the output observation at time $t + 1$.

With the same number of inputs used to train the networks and among the same set of experiments, since for the same network architectures we obtain different results for the two sets of stocks analyzed, we can conclude that the portfolio selection could affect the network predictions. For example in the set of examples in which we trained the network to recognize the relationships between 10 previous input observations and the output observation at time $t + 1$, we obtained good results with the network architectures with [63 63] hidden neurons and [125 125] hidden neurons in the experiments involving the Morgan Stanley Fund's stocks, while in the experiments on the other Fund's stocks we obtained good results only with the network architectures with [187 63] hidden neurons.

It is interesting to notice, however, that while in the set of experiments in which we trained the network to recognize the relationships between 5 previous input observations and the output observation at time $t + 1$ the differences are not so clear, when we pass to the other two set of experiments the results are almost opposed: for example in the set of experiments in which we used 63 input neurons and trained the network to recognize the relationships between 20 previous input observations and the output observation at time $t + 1$, none of the architectures provided satisfactory results for the Pioneer Fund's stocks, while in the experiments involving the Morgan Stanley Fund's stocks, all the architectures provided good results.

Conclusions

The network proved to make poor predictions of the signs' returns: this could depend on several network features as the number of inputs, the length of the time series used to train the network, the network parameters, its architecture or the type of network itself. Although we presented the results obtained from 48 experiments, we would like to point out that other experiments have been carried out, almost seventy in total, but have not been described because the results were completely in line with the described ones. Even if the CS indicator assumed satisfactory values, that is higher than 50%, generally speaking, it is not sustainable, and certainly not recommendable, to base a trading strategy on predictions with a poor rate of correctness such as 50% (it is comparable to make investment decisions based on a coin flip). Indeed, this is reflected on the performance obtained by applying a simple trading strategy: with only the 50% of correct predictions, in fact, we obtained better performances than the funds' ones in 18 experiments out of 48 presented, that is the 37.5% of the total. Furthermore, it is necessary to clarify that the shares object of our analysis form the 45.82% and the 39.76%, respectively, of the compositions of the Pioneer and Morgan Stanley funds, hence the performances obtained with our trading strategy cannot be considered absolutely comparable with the performance of the two funds, without even considering the fact that we did not consider all the fees that a mutual fund usually pays, that is, we presented gross performances.

However, we can conclude that the portfolio selection could affect the network outputs: since analyzing the Pioneer Fund's stocks, we obtained good results, (at least, performances higher than the Fund's ones), only in 7 experiments out of 24, while analyzing the Morgan Stanley Fund's stocks, we obtained good results in 11 experiments out of 24, we can deduce that, depending on the time series of inputs-desired output used to train the network, the results could be different.

An ANN is not perfect, just like a human being, but, if correctly and successfully trained, it should never makes mistakes in the evaluation of reliable and recurring data. The human being inevitably takes small risks every day, sometimes succeeding and sometimes not, while the ANN elaborates information and processes based on the recurrence of data in order to obtain a result. Some researchers are even trying to develop an ANN provided with imagination, that is, an ANN capable of reasoning and of making plans for the future, without human instructions[38].

[38] Y. Li et al., "*Agents that imagine and plan*", 20/07/2017, www.deepmind.com.

This is why we can affirm that it is very plausible that Fund managers take investment decisions based on ANN's predictions: using different types of network, training them with different and more sophisticated inputs and using advanced learning techniques we are almost sure that they are able to obtain predictions with a level of accuracy higher enough to construct a superior and highly developed trading strategy based on them.

Bibliography

Allen F. and Karjalainen R., (1999), "*Using genetic algorithms to find technical trading rules*", Journal of Financial Economics, vol. 51, p. 245-271, www.sciencedirect.com.

Andreeski C.J. et al., (2006), "*Elman NN and Time Series in Forecasting Models for Decision Making*", World Automation Congress, www.researchgate.net.

Arévalo A. et al., (2016), "*High-Frequency Trading Strategy Based on Deep Neural Networks*", Springer International Publishing Switzerland, www.researchgate.net.

Ashwood A.J., (2013), "*Portfolio Selection using Artificial Intelligence*", Queensland University of Technology, eprints.qut.edu.au.

Belcaro P. et al., (1996), "*Modelli Previsivi Neurali: Un'applicazione al Mercato Finanziario Italiano*", Atti del Ventesimo Convegno Annuale A.M.A.S.E.S., Università degli studi di Urbino.

Bengio Y., et al., (1994), "*Learning Long-Term Dependencies with Gradient Descent is Difficult*", IEEE Transaction on Neural Networks, vol. 5, n.2, p. 157-166, ai.dinfo.unifi.it.

Bengio Y., et al., (2012), "*On the difficulty of training recurrent neural networks*", arXiv, www.researchgate.net.

Bergmeir C. and Benìtez J.M., (2012), "*Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS*", Journal of Statistical Software, vol. 46, www.jstatsoft.org.

Bodén M., (2001), "*A guide to recurrent neural networks and backpropagation*", School of Information Science, Computer and Electrical Engineering, Halmstad University, www.researchgate.net.

Britz D., (2015), "*Recurrent Neural Network Tutorial – Implementing a GRU/LSTM RNN with Python and Theano*", Artificial Intelligence, Deep Learning and NPL, www.wildml.com.

Cammarata S., (1990), "*Reti neuronali: Una introduzione all' "altra" intelligenza artificiale*", Gruppo Editoriale Fabbri.

Cogo G., (2015), "*MultiLayer ANNs: predicting the S&P 500 index*", Università Ca' Foscari di Venezia, dspace.unive.it.

Coliva E. and Galati L., (1992), "*Analisi tecnica finanziaria*", UTET Università.

Dai Y. and Zhang Y., (2014), "*Machine Learning in Stock Price Trend Forecasting*", Stanford University, cs229.stanford.edu.

Dario U., (2014), "*Forecasting Energy Market: an Artificial Neural Network Approach*", Università Ca' Foscari di Venezia, dspace.unive.it.

Dash R. and Dash P.K., (2016), "*A hybrid stock trading framework integrating technical analysis with machine learning techniques*", The Journal of Finance and Data Science, vol. 2, p. 42-57, www.sciencedirect.com.

Dempster M.A.H. and Leemans V., (2006), "*An automated FX trading system using adaptive reinforcement learning*", Expert Systems with Applications, vol. 30, p. 543-552, www.jbs.cam.ac.uk.

Di Cartesio S., (2010), "*Analisi Tecnica o Analisi Fondamentale?*", www.intermarketandmore.com.

Dixon M. et al., (2016), "*Classification-based Financial Markets Prediction using Deep Neural Networks*", www.researchgate.net.

Donn S.F., (2010), "*Stock Market Timing Using Artificial Neural Networks and Genetic Algorithms*", Nquant.com.

Edwards R.D. et al., (2007), "*Technical Analysis of Stock Trends, Ninth Edition*", CRC Press, www.workofark.com

Elman J.L., (1990), "*Finding Structure in Time*", Cognitive Science, vol. 14, p. 179-211, psych.colorado.edu.

Fama E.F., (1970), "*Efficient Capital Markets: A Review of Theory and Empirical Work*", The Journal of Finance, vol. 25, p. 383-417, efinance.org.cn.

Fernández A. and Gómez S., (2007), "*Portfolio selection using neural networks*", Computers & Operations Research, vol. 34, p. 1177-1191, www.sciencedirect.com.

Gallo C., (2007), "*Reti Neurali Artificiali: Teorie e Applicazioni*", Dipartimento di Scienze Economiche, Matematiche e Statistiche, Università degli Studi di Foggia, www.ideas.repec.org.

Grau I. et al., (2013), "*Backpropagation through Time Algorithm for Training Recurrent Neural Networks using Variable Length Instances*", Computación y Sistemas, vol. 17, n. 1, p. 15-24, www.researchgate.net.

Grippo L. and Sciandrone M., (2003), "*Metodi di ottimizzazione per le reti neurali*", Università La Sapienza di Roma, www.dis.uniroma1.it.

Guo X., (2016), "*How can machine learning help stock investment*?", Stanford University, pdfs.semanticscholar.org.

Hebb D.O., (1949), "*The Organization of Behavior: A Neuropsychological Theory*", Wiley and Sons, s-f-walker.org.uk.

Hochreiter S. and Schmidhuber J., (1997), "*Long Short-Term Memory*", Neural Computation, vol. 9, p. 1735-1780, www.bioinf.jku.at.

Kaastra I. and Boyd M., (1996), "*Designing a neural network for forecasting financial and economic time series*", Neurocomputing, vol. 10, p. 215-236, www.sciencedirect.com.

Kara Y. et al., (2011), "*Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange*", Expert Systems with Applications, vol. 38, p. 5311-5319, www.sciencedirect.com.

Karpathy A., (2015), "*The Unreasonable Effectiveness of Recurrent Neural Networks*", Andrej Karpathy Blog, karpathy.github.io.

Kondratenko V.V. and Kuperin Y.A., (2002), "*Using Recurrent Neural Network to Forecasting of Forex*", Department of Physics, St. Petersburg State University, www.researchgate.net.

Krogh A. et al., (1991), "*Introduction to the theory of neural computation*", Addison-Wesley Publishing Company, citeseerx.ist.psu.edu.

Kuo R.J. et al., (2001), "*An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network*", Fuzzy Sets and Systems, vol. 118, p. 21-45, www.sciencedirect.com.

Kůrková V., (1992), "*Kolmogorov's Theorem and Multilayer Neural Networks*", Neural Networks, vol. 5, p. 501-506, www.sciencedirect.com.

Längkvist M. et al., (2014), "*A review of unsupervised feature learning and deep learning for time-series modeling*", Pattern Recognition Letters, vol. 42, p. 11-24, www.sciencedirect.com.

Lee A. et al., (2016), "*Determination of Optimal Initial Weights of an Artificial Neural Network by Using the Harmony Search Algorithm: Application to Breakwater Armor Stones*", Applied Science, vol. 164, n. 6, www.mdpi.com.

Li X. and Yu W., (2002), "*Dynamic system identification via recurrent multilayer perceptron*", Information Sciences, vol. 147, p. 45-63, www.sciencedirect.com.

Li Y. et al., (2017), "*Agents that imagine and plan*", www.deepmind.com.

Lipton Z.C. et al., (2015), "*A Critical Review of Recurrent Neural Networks for Sequence Learning*", University of California San Diego, www.zacklipton.com.

Lo A.W. et al., (2000), "*Foundations of Technical Analysis: Computational Algorithms, Statistical Inference and Empirical Implementation*", The Journal of Finance, vol. 55, n. 4, web.mit.edu.

Lu D.W., (2017), "*Agent Inspired Trading Using Recurrent Reinforcement Learning and LSTM Neural Networks*", www.researchgate.net.

Macchiavello C., (1992), "*Introduzione alle Reti Neurali*", Dipartimento di Fisica A. Volta, Università di Pavia, vision.unipv.it.

Manjula B. et al., (2012), "*Tracking the Trends of Financial Applications Using Genetic Algorithm*", International Journal of Computer Applications, vol. 48, n. 16, www.researchgate.net.

Milosevic N., (2016), "*Equity Forecast: Predicting long term stock price movements using machine learning*", School of Computer Science, University of Manchester, www.arxiv.org.

Mitchell C., (2014), "*Ultimate Guide to the Stochastic Oscillator*", Trading Indicators, www.traderhq.com.

Niaki S.T.A. and Hoseinzade S., (2013), "*Forecasting S&P 500 index using artificial neural networks and design of experiments*", Journal of Industrial Engineering International, vol. 9, n. 1, www.researchgate.net.

Nielsen M., (2017), "*How the Backpropagation Algorithm works*", www.neuralnetworks-anddeeplearning.com.

Olah C., (2015), "*Understanding LSTM Networks*", Colah's Blog, colah.github.io.

Parlos A.G. et al., (1994), "*Application of the Recurrent Multilayer Perceptron in Modeling Complex Process Dynamics*", IEEE Transactions on Neural Networks, vol. 5, n. 2, p. 255-266, alumnus.caltech.edu.

Parlos A.G., (2004), "*Algorithm for Training a Recurrent Multilayer Perceptron*", NASA Tech Briefs, www.ntrs.nasa.gov.

Pererveńko V., (2015), "*Third Generation Neural Networks: Deep Networks*", www.mql5.com

Raei R., (2007), "*Risky Portfolio Selection through Neural Networks*", Iranian Accounting & Auditing Review, n. 46, p. 70-83, journals.ut.ac.ir.

Rota Bulò S., "*Appunti di reti neurali*", Università Ca' Foscari di Venezia, www.dsi.unive.it.

Sak H. et al., (2014), "*Long short-term memory recurrent neural network architectures for large scale acoustic modeling*", Interspeech, www.researchgate.net.

Salakhutdinov R. and Hinton G., (2012), "*An Efficient Learning Procedure for Deep Boltzmann Machines*", Neural Computation, vol. 24, Massachusetts Institute of Technology, pdfs.semanticscholar.org.

Sartorello M., (2014), "*Prezzi predatori e reti neurali: un approccio sperimentale*", Università Ca' Foscari di Venezia, dspace.unive.it.

Sezer O.B. et al., (2017), "*An Artificial Neural Network-based Stock Trading System Using Technical Analysis and Big Data Framework*", www.researchgate.net.

Sharma A., (2017), "*Understanding Activation Function in Neural Networks*", The Theory of Everything, www.medium.com.

Shen S. et al., (2012), "*Stock Market Forecasting Using Machine Learning Algorithms*", Department of Electrical Engineering, Stanford University, cs229.stanford.edu.

Stella M., (2009), "*Le Reti Neurali: Isomorfismo tra i modelli di Hopfield e di Ising*", Facoltà di Scienze Matematiche, Fisiche e Naturali, Università degli studi del Salento, www.dmf.unisalento.it

Sutskever I. et al., (2015), "*Recurrent Neural Network Regularization*", International Conference on Learning Representations, www.arxiv.org.

Takeuchi L. and Lee Y.Y., (2013), "*Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks*", Stanford University, cs229.stanford.edu.

Tan C.N.W., (1997), "*An Artificial Neural Network Primer with Financial Application Examples in Financial Distress Predictions and Foreign Exchange Hybrid Trading System*", School of Information Technology, Bond University, www.datascienceassn.org.

Tiong Ching Ow L., (2012), "*Financial Time Series Prediction using Machine Learning Algorithms*", Sunway University, eprints.sunway.edu.my.

Tutschku K., (1995), "*Recurrent Multilayer Perceptrons for Identification and Control: The Road to Applications*", Institute of Computer Science, University of Würzburg, www3.informatik.uni-werzburg.de.

Unadkat S.B. et al., (2001), "*Recurrent Neural Networks, Design and Applications*", CRC Press, cdn.preterhuman.net.

Vanstone B. and Finnie G., (2009), "*Financial Trading Systems Using Artificial Neural Networks*", IGI Global, www.researchgate.net.

Voegt T., (2017), "*Artificial Neural Networks in Trading Systems*", University of Brighton, School of Computing, Engineering and Mathematics, www.researchgate.net.

Wang J. et al., (2016), "*Financial Time Series Predictions Using Elman Recurrent Random Neural Networks*", Computational Intelligence and Neuroscience, vol. 2016, www.hindawi.com.

Ward S. and Sherald M., (2006), "*Successful Trading Using Artificial Intelligence*", Ward System Group Inc, c.mql5.com.

Werbos P.J., (1990), "*Backpropagation Through Time: What It Does and How to Do It*", Proceedings of the IEEE, vol. 78, n. 10, p. 1550-1560, axon.cs.byu.edu.

Yu L. et al., (2005), "*Mining Stock Market Tendency Using GA-Based Support Vector Machines*", Springer-Verlag Berlin Heidelberg, pdfs.semanticscholar.org.

www.commons.wikimedia.org.

www.dukascopy.com.

www.energyprofessionalsymposium.com.

www.forexwiki.it.

www.mathworks.com.

www.mnemstudio.org.

www.morningstar.com.

www.neurdon.com.

www.neuroph.sourceforge.net.

www.prorealtime.com.

www.sebastianraschka.com.

www.stockcharts.com.

www.theprojectspot.com.