Ca' Foscari
University
of Venice

Master's Degree Programme

in Economics and Finance – Finance

Second Cycle (D.M. 270/2004)

Final Thesis

Q-Learning:

an intelligent technique for financial trading systems implementation

Supervisor:

Prof. Marco Corazza

Graduand:

Veronica Popa

843207

Academic Year

2017/2018

Contents

Introduction

The objective of this thesis is to develop and implement a *Financial Trading System*[1] (*FTS*) able to operate autonomously into the financial market. The developed *FTS* has to be able to manage, without supervision, an invested capital and to obtain positive performances.

The structure of the *FTS* is based on the implementation of an algorithm, the *Q-Learning algorithm*[2], belonging to the *Reinforcement Learning* techniques, that is a branch of the artificial intelligence. The considered algorithm is able to real time optimize its behavior on the basis of the feedbacks received by the environment in which it operates.

The theoretical framework on which the developed *FTS* is based, allowing for effective trading activities, is the *Adaptive Market Hypothesis (AMH)*, which can be viewed as an evolution of the *Efficient Market Hypothesis (EMH)*.

The *FTS* is applied on five real stock prices time series and on an artificial one. For each time series many configurations are tried with different parameters values characterizing each configuration. Moreover, many simulations are carried out for each configuration, in order to identify the most profitable, which is suitable for any financial asset. The structure of the thesis is developed in five chapters.

[1] A *Financial Trading System (FTS)* is defined as a system able to automatically make and submit orders on the financial market on the basis of predefined rules.

[2] The *Q-Learning algorithm* is an algorithm able to optimize, in real time, its behavior on the basis of the responses it receives from the surrounding environment.

The first chapter is a description of the theory which justifies financial trading activities, that is the *Adaptive Market Hypothesis (AMH)*, which derives from the *Efficient Market Hypothesis (EMH)*. In the same chapter is briefly described the objective of the thesis and an historical overview of the *Reinforcement Learning* is provided. The second chapter provides an accurate description of all the features of *Reinforcement Learning*, which will be useful to develop and implement the utilized *FTS*. In the third chapter, the specific elements and the values of the parameters, characterizing the *FTS*, are presented and justified. Furthermore, the six time series are described, reporting the main descriptive statistics and some graphical representations characterizing them. In the fourth chapter are presented all the results of all the configurations applied on each time series. The results are compared and explained through the use of some important statistics and graphs. The fifth chapter consists in the conclusions of the thesis.

Chapter 1

Market Hypothesis' evolution

This chapter is focused on the theory of financial market efficiency, *Efficient Market Hypothesis*, and on another one, *Adaptive Market Hypothesis*, that can be viewed as an evolution of the former. The latter theory finds its roots in evolutionary principles and it is introduced as it justifies the possibility of effective trading activities. In the second part of the chapter, the objective of the thesis is described, that is the development of a Financial Trading System through the implementation of the *Q-Learning algorithm*. The algorithm is considered as an agent that takes financial decisions and receives feedbacks from the environment in which it operates, that is the financial market. Finally, the last paragraph consists in an brief historical overview of the artificial intelligent branch here considered, that is the *Reinforcement Learning*. It is provided a brief description of the birth and evolution of the main features constituting the *Reinforcement Learning*, that is the *trial and error*, the *optimal control* problem and the *temporal-difference learning methods*.

The capital market's main purpose is the allocation of the capital stock, owned by the economy, among its partecipants. It can be viewed as an ecosystem in which different market agents[3] interact among them in order to seek profit opportunities. This interrelation among market agents and market resources define the *market ecology*. Indeed, ecology studies the interaction among organisms and the environment in which they act.
Here, I consider two theories relating the *market ecology*: one is the evolution of the another.

1.1 Efficient Market Hypothesis (*EMH*)

The more powerful and enduring theory regarding financial markets is the *Efficient Market Hypothesis (EMH)*. The illustration of this theory may be exemplified through an old joke, widely used among economists, that I will report from the paper of Andrew W. Lo:

> *An economist is strolling down the street with a companion. They come upon a $100 bill lying on the ground, and as the companion reaches down to pick it up, the economist says, "Don't bother, if it were a genuine $100 bill, someone would have already picked it up."*

This example of economic logic is useful to explain the *EMH*, which states that market prices, at any time, fully reflect all available information. Consequently, as new information comes into the market, such as the autenticity of the $100 bill on the ground, it is instantaneously exploited and incorporated in market prices, eliminating, in such a way, any profit opportunity.
So, in an efficient market, price changes are unpredictable as they incorporate all information. Assuming a discrete-time framework, an asset price variation in two subsequent time instants can be described as a stochastic process in which the variation is due to an unexpected new information:

[3]Market agents may be pension funds, hedge funds, financial institutions and retail investors.

$$P_{t+1} = \mathbb{E}(\tilde{P}_{t+1}|\Omega_t) + \tilde{\varepsilon}_{t+1}$$

where t and $t+1$ represent the two subsequent time instants, P_{t+1} is the price of an asset at time $t+1$, $\mathbb{E}(\cdot)$ is the expectation operator, $\tilde{P}_{t+1}$ is the random variable "financial asset price" at time $t+1$, Ω_t is the set of information available at time t, $\tilde{\varepsilon}_{t+1}$ is the random variable representing the financial asset prediction error at time $t+1$, with $\mathbb{E}(\tilde{\varepsilon}_{t+1}) = 0$.
The conditional expectation $\mathbb{E}(\tilde{P}_{t+1}|\Omega_t)$ is meant to imply that all the information available in t is fully utilized in determining the price at time t, allowing for an error term.
This efficient market model has two special characterization, that is the *Submartingale Model* and the *Random Walk Model* [11].

Submartingale Model is defined as follows:

$$\mathbb{E}(\tilde{P}_{t+1}|\Omega_t) \geq P_t.$$

The expression above states that the generating prices sequence follows a submartingale[4] with respect to the available information in each time instant, Ω_t. So, the submartingale expects the price in time instant $t+1$, that is projected on the basis of the available information Ω_t, to be equal to or greater than the current price.
An important implication, derived from the implicit assumption that the conditional returns on Ω_t are non-negative, suggests that any trading strategy can not be greater than a buying-and-holding policy.

Random Walk Model is based on the statement that the current price "fully reflects" all available information. This assumption implies that

[4]A submartingale is a stochastic process which gives the expected price of the next period, given the current informations, equal to or greater than the current price.

price changes, in subsequent time instants, are independent and identically distributed. So the model can be represented as follows:

$$\mathbb{E}(\tilde{P}_{t+1}|\Omega_t) = \mathbb{E}(\tilde{P}_{t+1}).$$

The expression states that the mean of the distribution of the stock price is independent of the available information Ω_t, consequently also the entire distribution of the generating stock prices process is.

So, the more efficient market is the one that generates the more random sequence of prices and, the most efficient market among all is the one in which price changes are random and unpredictable. This derives directly from the competition among market partecipants which attempt to profit even from the smallest information. In doing so, they incorporate the information into the market price eliminating, in such a way, any profit opportunity that justify their trades. If this happens instantaneously, no profit can be gathered as any new information has been already incorporated (as the $100 bill on the ground).

One of the central principle of modern financial economics concerns the necessity of taking into consideration the risk aversion of investors[5]. This yielded to a neoclassical version of the *EMH* in which price changes are unforcastable as they depend not only by the available information but also by the weight of investors' marginal utilities. So, the general belief can be summarized in the following three points [15]:

a. each investor forms expectations rationally[6], that is she/he makes optimal decisions, based on its own preferences, by trading off costs and benefits weighted by the statistically correct probabilities and marginal utilities;

[5] A risk averse investor is an investor who prefers to avoid loss rather than making a gain. Such investor prefers investments with a known and low degree of risk and, consequently, a low return rather than a higher return with a higher and uncertain risk.

[6] A rational investor is an investor who tries to maximize her/his own needs and that acts in her/his own interests. She/he makes rational expectations, that are expectations based on all available information, by trading off the costs and benefits among all possible action in order to choose the best one.

b. markets aggregate information in an efficient way;

c. equilibrium prices incorporate all available information.

1.2 The Three P's

The rational financial paradigm exposed above can be summarized by *The Three P's of Total Risk Management* [13], namely prices, probabilities and preferences. The interaction among prices, probabilities and preferences has its roots in the most fundamental idea of modern economics, that is the law of supply and demand. According to this principle, the price and the quantity of any traded commodity is determined by the crossing of the supply and demand curves. The demand curve represents the relationship between the price and the amount of a commodity that the consumer is able to buy at any given level of price. It is the aggregation of all individual consumers' demands, each originated from the optimization of individual's preferences, bounded by budget constraints dependent on prices, income, savings, borrowing charges. The supply curve is the relationship between the price and the quantity producers want to supply at any given level of price. It is the aggregation of outputs, whose production derives from the optimization of producers' production function, subjected to resources restrictions, that depend on prices, costs of materials, wages and financings. Consequently, the intersection between the two curves determines the equilibrium point, that is the price-quantity pair that satisfy both parts, while any other point satisfies only one. As both consumers and producers determine their consumption and production plans under uncertainty, as future income and business conditions are estimated, probabilities affect both sides of the market. Therefore, the interaction among the *three P's* form a general equilibrium, in which demand equals supply, in an uncertain environment, where rational agents act in order to optimize their own interests.

To determine an equilibrium, only two of the *three P's* are sufficient to determine also the third one. So, given preferences and probabilities of an equilibrium point, prices can be determined exactly. Alternatevely, an equilibrium in which prices

and probabilities are known, preferences can be inferred. Otherwise, if prices and preferences are given, probabilities can be extracted.
Summarizing, if prices contain all available information, the preferences and future plans of all markets' partecipants are known, the market environment is perpetually in equilibrium and stationary[7]. In practice, however, it is possible to note that the market is dynamic.

1.3 Sufficient Market Conditions for Efficiency

The *EMH* requires some conditions to make the market efficient, such as the lack of transaction costs for financial securities, the free availability of information to all market partecipants, same view of investors concerning the implications of present information on current prices and future prices' probability distribution of each security. Under these conditions, the market price fully reflects all available information.
However, these conditions are few realistic in the real financial markets as, for instance, transaction costs exist, free and available information for all investors is not feasible and market's partecipants could have not the same view about the implications of present and future probabilistic price evolution. Fortunately, these conditions are sufficient for an efficient market, however not necessary. Indeed, the market prices still reflect all available information even if there are transaction costs that tend to obstruct transactions flow. Furthermore, the market is efficient even if not all, but a suffient number of investors, have access to direct available information. Finally, market inefficiency can not derive from the disagreement among investors' expectations, unless there are investors' better or worst evaluations of available information that are already incorporated in market prices [11].

[7] A stationary market is a system in which the demanded and supplied quantities of goods and services remain constant over time and consequently also prices, incomes, population size and resources are kept constant.

1.4 Degrees of Efficiency

As information is one of the core elements in *EMH*, it can be classified in three different levels, based on the degree of information incorporated in the security prices.

In the first level, called *weak efficiency*, prices fully reflect only the information recorded in the past prices time series. Investors can obtain such information through a technical and quantitative analysis. The second level of market efficiency is called *semi-strong efficiency*. Information in this level derives from the *fundamental analysis*, that is prices reflect not just information of the *weak efficiency* level but also firms' business public information such as announcement of dividends, of the last quarter's earnings, new stock issue, annual reports and so on. The last level of efficiency is the *strong efficiency* that incorporates the two previous forms and, moreover, includes private informations obtained through a painstaking analysis of the business and the economy. This kind of information can be obtained by a privileged group of investors that have monopolistic access, as are insider and private informations, to any relevant information for the price formation.

The assertion behind the *weak efficiency* hypothesis is that no investor can gather profits higher than the market ones by studying past security prices. Under the assumption of the *semi-strong efficiency*, it is not possible to profit using public available information. On the basis of the *strong efficiency*, no investor is able to make systematic profits from monopolistic information.

Therefore, under the previous assumptions, especially under the *strong efficiency*, that includes the other two, the market is perfectly efficient. So, gather more information is useless and constitutes an expensive activity. Consequently, there is no reason to trade, as it is not possible to earn systematic profits, and the financial market would likely collapse.

1.5 Evolution to the Adaptive Market Hypothesis *(AMH)*

The *EMH*, however, that states that market agents are rational when making decisions, does not reflect the real financial market. In fact, common feeling suggests that humans, and therefore market agents, do not always behave rationally when making decision, especially under uncertainty. Indeed, in taking decisions, market agents' behavior is influenced by biases such as overconfidence, overreaction, risk aversion, herding, mental accounting, miscalibration of probabilities, hyperbolic discounting, regret. This behavioral distorsion, in decision making process, sometimes results in predictable and financially ruinous behavior, which is likely to bring to an inefficient market. Consequently, if the market is not efficient, this makes sense for investors to spend time to gather and trade informations, as security prices are not immediately and correctly updated with the new information. This gives rise to profit opportunities that represent a compensation for investors' effort in gather and trade information. In this new perspective, the previous efficient market levels, *weak efficiency*, *semi-strong efficiency* and *strong efficiency* can be viewed as the speed at which security prices adjust at new information.

The difference between the price adjustement processes in an efficient market and in a market influenced by behavioral biases, that is partly inefficient market, can be illustrated in the following figure:

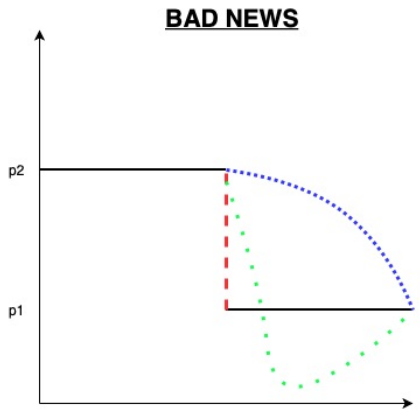


Figure 1.1: Price reaction when bad news arrive.

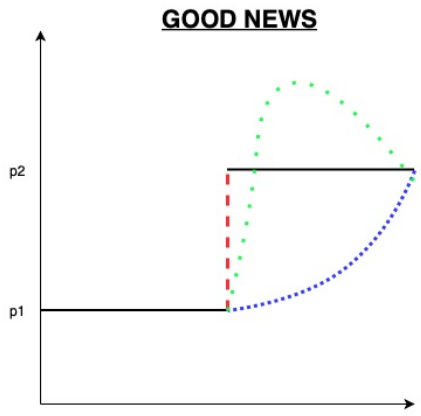


Figure 1.2: Price reaction when good news arrive.

As soon as bad or good news arrive, in an efficient market, price immediately handles it and instantly moves to a new level (red line). In this case there is no margin to act. Instead, in a market in which market partecipants are not rational, there might be a slow reaction to a new information (blu line) or an overreaction (green line). In this case the price adjusts itself to the new price level but, in the meanwhile, there is margin to operate.

To conciliate the *EMH* with its behavioral critics, a new theory arised, that is the *Adaptive Market Hypothesis (AMH)*. The *AMH* is based on a biological evolutionary perspective[8] view of the environment, in this case the financial markets, in which the economic agents interact and continously evolve [12]. Evolution is a consequence of changes of environmental conditions that determine a change in market agents' behavior. The dynamics of evolution make the markets efficient and determine the wax and wane of financial institutions, investment strategies and institutional and individual fortunes [15].

This approach can be traced to the recent development in the emerging discipline of *evolutionary psychology* that, in applying the evolutionary principles (competition, reproduction, mutation and natural selection) to social interactions, gives a compelling explanation of certain types of human behaviors, such as altruism, fairness, ethics, morality. Indeed recently, economist and biologists have begun to expand their study toward these connections under different perspectives, that is economic sociobiology, evolutionary game theory, evolutionary economics and economics as a complex system. Under this perspective is possible to pass from market efficiency to an evolutionary alternative, that is the *Adaptive Market Hypothesis*.

In this framework, in which different imperfections exist, the law of natural selection, or *survival of the richest* [16], determine the markets and institutions evolution. In fact, the presence of a high degree of competitiveness of global financial markets and the huge rewards that the "fittest" traders accumulate, imply that the natural selection determines the successful investor conventional profile. Market agents that are unable to adapt themselves to the environment mutation, are eliminated from the market after suffering a certain level of losses. Therefore, the number

[8] A biological evolutionary perspective is based on principles such as competition, reproduction, mutation and natural selection.

of competitors, their adaptability skills and the presence of profit opportunities, that characterize the market ecology, are the determinants of the market efficiency degree. Any counterproductive behavior will be reshaped and modified by the natural selection to better fit it to the current environment.

As the environment changes more or less continously, the market partecipants have to be able to adapt themselves to new situations by learning and developing more suitable behaviors. Such changes in behaviors are driven by changes in individual's preferences via the strength of natural selection. In fact, preferences are likely to change and to be shaped over time by factors internal to the individual, such as related to the individual's personality, and external to it, due to environmental conditions. So, it can be affirmed that behavior is not necessarily intrinsic in every agent and exogenous from the environment, but its evolution is subjected to the natural selection in the respective environment. Preferences and market forces are elements that interact among them, driven by competition, by natural selection and by the variety of individual and institutional behavior, in order to produce a dynamic environment.

Under the evolutionary perspective, individuals and institutions are viewed as organism that improve themselves through the natual selection process, by maximizing the survival of their "genetic" material. This postulate is opposite to that of the *EMH*, that considers individuals' rational expectation and expected utility maximization. As individuals are subjected to behavioral biases that decrese their degree of rationality, they are not capable to reach such kind of optimization. In fact, it is too costly and computationally expensive. Alternatively to optimization, humans can look for a degree of satisfaction, not necessarily optimal. This satisfactory point is reached not analitically, but through trial and error and, then, by a natural selection mechanism. Individuals base their choices on their past experience and make predictions on the potential optimal, receiving feedbacks from the environment. These techniques allow individuals to develop some heuristics in order to solve economic challenges. As long as the economic challenges remain constant in time, heuristics will adapt in order to achieve a near optimal solution. However, as the environment changes, the old heuristics will change in order to

best suit the new conditions. In such cases, it is possible to temporarily observe improper and suboptimal actions. However, rather than call such behaviors as irrational, is more proper label them as *maladaptive* [16]. Consequently, the *AMH* theory can be viewed as an evolution of the *EMH*, that is a combination of the *EMH* with evolutionary principles, which key characteristics are the following [16]:

a. Individuals act in their own interest;

b. Individuals commit mistakes;

c. Individuals learn and adapt their behaviors;

d. Competition leads adaptation and innovation;

e. Natural selection shapes *market ecology*;

f. Evolution defines the market dynamic.

In both theories individuals act for themselves. In efficient markets individuals do not make mistakes, and consequently they have no possibility to learn and adapt, as the environment is stationary and perpetually in equilibrium. However, in the *AMH* framework, individuals learn from errors and adapt themselves accordingly. Adaptation is driven by market forces, that constitute survival key factors. The interaction among the market agents is driven by natural selection and so, the environment is a consequence of this process. All the factors that drive evolution, such as egoism, adaptation, competition, natural selection and environmental states, are determinants of market dynamic.

The informations derived by the intersection among the environment conditions and the number and variety of *species* that constitute the *market ecology*, are fully reflected in market prices. The term *species* identifies different markets' groups, each with similar features. For example, one *specie* may be constituted by pension funds; another *specie* might be formed by retail investors; hedge funds are another *specie*; market makers a fourth one and so on. If these different *species* interact among them, competing for a limited quantity of resuorces, that market is probably higly efficient. Conversely, if the competition concerns a small number of species for an

abundant quantity of resources, the market is likely to be less efficient. The quantity of resources in a given environment is comparable to the profit opportunities in a given market. The competition depends by the amount of resources present. A high amount of resources is accompanied by a low degree of competition. Therefore, to an increase in competition, due to a diminuishing of resources supply or to an increase in population size, is associated the exhaustion of the former. This will bring to a population reduction and, in extreme cases, the extinction of certain species or permanent sources exhaustion, and so a decrease in competition. Then the cycle will start all over again.

Given that in the *AMH* framework the environment is dynamic and in continous evolution, reaching an equilibrium point, which is central to the *EMH*, here is not guaranteed. In many cases such equilibrium can not exist or the convergence is inexorably slow.

From the *AMH* framework can be derived some concrete and practical implications:

1. The first implication regards the relation between risk and reward. This relation tends to change over time as it is determined by the magnitude of the population, and its preferences in *market ecology*, and by institutional aspects, such as environmental regulation and tax law. A change in this factors will probably affect the risk-reward relationship. The logical consequence of this implication is that the equity risk premium[9] will also change over time as the risk preferences of all market partecipants are not constant but are shaped by the natural selection forces.

2. In the *AMH*, contrary to the *EMH*, gather information makes sense as security prices are not immediately updated with upcoming informations. This creates arbitrage[10] opportunities, otherwise the market would collapse. As such opportunities are exploited, they disappear. However, new opportunities appear continously because of the cycle of life of certain species and because of the change of institutions and business conditions. Therefore, in the *AMH*

[9]The equity risk premium is the reward demanded by investors for investing in equity or in a stock, so in a risky asset, rather than in a riskless asset.

[10]Arbitrage is the simultaneous purchase and sale of a security to lock in a profit from an imbalance in the price.

framework are observable complex market dynamics, such as trends, episodes of panic, obsessions, bubbles and crashes that give the reason for an active portfolio management.

3. The last implication regards investment strategies that can have good performances in certain environments and bad performances in others. The profitability of strategies created from the exploitation of arbitrage opportunities can have oscillating trends, so can be more or less profitable over time.

Therefore, under the perspective that the financial market is dynamic and constantly in evolution, so the population size tends to change, the investors' preferences change and environmental conditions change, in the *AMH* framework there makes sense an active portfolio management.

1.6 Aim of the Thesis

It is reasonable to assume that the *AMH* is valid and so it will be the reference theoretical framework in which the *Financial Trading System (FTS)*, that I will develop, will operate. To develop such a *FTS*, I will implement an algorithm[11], through a software created in MATLAB® environment, which might be described as an economic agent able to take any suitable financial decision in the financial market. To make this *FTS* profitable, is necessary to provide it with problem solving capabilities, so to render it "intelligent". Intelligence gives to an economic agent the ability of think, learn, understand and perceive informations, make judgements, make logical thoughts. Through this characteristics, intelligence makes an agent able to adapt herself/himself to the sorrounding environment, by adapting her/his behaviors to new situations. As I consider an area of artificial intelligence, that is the *Reinforcement Learning*, the agent that I consider is an artificial agent[12], subsequently called agent, that will act as a retail or as an institutional investor, so

[11] An algorithm is an unambigous procedure able to solve a class of operations within a limited amount of time.

[12] An artificial agent is an algorithm created to replicate specific features of human agent.

it must be provided with intelligence. An intelligent agent is an autonomous and computational entity able to act in an environment in order to reach its objectives. It optimizes its goals by using all available informations. In particular, it is able to adapt itself and its actions, in a flexible and rational way, to environmental changes, in order to overcome any obstacle. This is possible thanks to the capacity of an intelligent agent to learn, to solve problems, to plan and to make decisions.

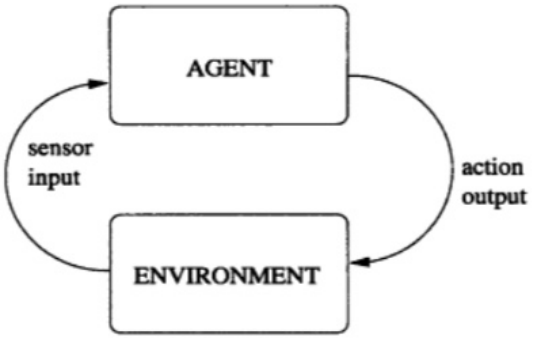


Figure 1.3: The agent discloses the environment
and produces action outputs in order to affect it.
([23])

In the figure above is represented the relationship existing between agent and environment. On the basis of the perception of the environment and on the available informations about it, an intelligent and rational agent takes some actions and form intentions, that will affect the environment, trying to maximize agent's objectives. However, to reach such objectives, the agent must have the following characteristics [23] [25]:

a. *autonomy*: the agent has absolute control upon its behavior and actions;

b. *reactivity*: it perceives the environment and promptly adapts itself to environmental changes;

c. *proactiveness*: it is able to takes goal-directed initiatives;

d. *social ability*: it can interact with other agents or humans.

So, an intelligent agent must take financial decisions, such as when to buy an asset, when to sell it or when to stay out of the market, in order to manage a portfolio in

a profitable way.

All theories of learning and intelligence are based on the underlying idea of learning by interacting. In fact, thanks to the interaction with the environment, it is possible to develop optimal behaviors.

1.7 Historical Threads of Reinforcement Learning

The three main threads of *Reinforcement Learning* were pursued independently before they came together into the modern *Reinforcement Learning*.

One thread, which has its roots into the psychology of animal learning, regards learning through *trial and error*. This learning process is at foundation of all theories of learning and intelligence of all living beings, that discover the unknown sorrounding environment by trying all the actions and movements available to them. The learning source consists in the feedback received by the environment, that could be a reward or a punishment. Consequently, the memory of what is good and what is bad play a central role in this process, as this will bring to the selection of actions with a better performance.

The second thread regards the question of *optimal control* and its solution through the use of value functions and dynamic programming. *Optimal control*, which term was coined in the late 1950s, concerns the problem of designing a controller to reduce at minimum a measure of dynamic system's behavior in time. To solve this kind of problem, several approaches were developed, some of which use the concept of dynamical system's states[13] and of value function[14], also called *optimal return function*, in order to define a functional equation that can be solved by a class of methods called *dynamic programming*. *Dynamic programming* was considered the only suitable method for solving general stochastic optimal control problems[15].

The third thread is related to *temporal-difference learning methods*, based on the

[13] A dynamical system is a system that evolves over time and, consequently, each state also changes.

[14] A value function is a function that evaluates what is good for an agent, in terms of expected return, in the long run.

[15] A stochastic problem concerns processes which are driven by randomness and whose generating model, that is unknown, can be approximated.

difference between temporally subsequent value estimates. Also the origin of this thread derives from animal learning psychology.

In the 1980s these three elements were fully brought together and this was essential for the birth of the modern field of *Reinforcement Learning*. The 1989 is the year in which the *Q-Learning* was developed, as a consequence of the intersection between the *temporal-difference* and *optimal control* threads. *Q-Learning* is the method used in this thesis for the development of the algorithm for financial trading.

Chapter 2

Reinforcement Learning

In this chapter, I will provide a computational approach, that is the *Reinforcement Learning*, able to solve dynamic optimal stochastic control problems. The idea behind this method is to learn by interacting directly with the environment, suitably memorizing past experiences. As a model of the environment is unknown and, in order to adapt the method to the specific problem to solve, qualifiable as stochastic control problem, the *Temporal Difference Learning* methods are used. In particular the *Q-Learning algorithm*, belonging to the branch of the *temporal difference* methods, is considered and is described in detail in order to be implemented.

Reinforcement Learning consists in learning how to behave in given situations by interact directly with the surrounding environment. This concept is at the basis of all theories of learning and intelligence and it is the principal source of knowledge of the envirnonment. The simplest example might be the infant that discovers the world around her/him by trying all the actions that she/he can do. The infant, that is the *learner*, have not a teacher, but she/he has to learn and memorize the consequences of her/his actions in order to perform better in the subsequent actions.

So, the *learner* has to descover the most suitable actions which bring to the best consequences by trying them. In this way, the *learner* uses its experience to improve its performance over time. The *learner*, also called *agent*, has no prior knowledge about the environment, so she/he is driven by the evaluative feedback, that is the reward. The reward signal, that can be positive or negative, is the premium or the loss, consequence of a chosen action. As the *learner* does not know which actions to take, she/he must discover which ones maximize the reward by selecting them. The actions selected affect the immediate reward but also affect the environment in future situations and, through that, the future rewards. So, maximizing the immediate reward is not the only aim of the *learner*, but also subsequent rewards are. So, the goal of the agent is to perform actions that bring to the maximization of the long run reward. These two factors, *trial and error* and *delayed reward*, are the most important features of *reinforcement learning*.
As the *agent* has no knowledge about the environment, she/he has to grasp the state of the environment and consequently, on the basis of her/his past experiences, select the actions that are able to affect her/his own state. In doing so, the agent must have an objective or objectives concerning the state of the environment. In her/his decision process must be included the following three aspects:

1. sensation;
2. action;
3. objective.

Unfortunately, there are no examples of optimal behavior that can represent all the situations in which the agent can act. This fact brings to the one of the challenges of the *Reinforcement Learning*, that is the trade-off between *exploration* and *exploitation*[16]. The *agent*, in order to get the best results, must have a preference for actions that she/he already tried in the past and found to be winning. But, to select such winning actions, the *agent* has to prove actions never taken before. The only feedback that she/he receives is the reward. The *agent* in this way explores

[16]Exploration consists in the act of exploring the environment while exploitation is the act of selecting that action, among the explored ones, that gives the higher result.

the environment by trying different kind of actions, (*exploration*), in order to select the best one that gives the better result, (*exploitation*). As the environment is generally not stationary and continously changing, the *agent* has to supervise it and appropriately react and update her/his strategy. This is the *exploration-exploitation* dilemma, and none of them can be pursued without falling at the task [20]. A balance is needed, so the agent can learn by exploring and can perform well by exploiting what is already known.

2.1 Markov Decision Process

The problem of *Reinforcement Learning* can be formalized using the *Markov Decision Process*[17] (MDP) framework, which determines how the *agent* interacts with the environment throughout three signals: a state signal, an action signal and a scalar reward signal. The environment is everything the agent can not control and can not change arbitrarly. Here is not assumed that the *agent* does not know anything of the environment. Even if the *agent* has a complete knowledge of the environment, it still faces a challenging task. For example, we might know how a Rubik's cube works, but we still be incapable to solve it. The boundary between the *agent* and the environment represents the limit of the absolute control of the *agent* and not of its knowledge.

The *agent* interacts with the environment in order to achieve a goal. It selects the actions and the environment gives feedback to these actions, in the form of rewards. In particular, the environment is defined as a descrete time steps $t = 0, 1, 2, 3, ...$ In each time step the environment is represented by a *state*, which is a unique representation of all what is relevant for the problem that is modelled. In each time state t, the *agent* gets a representation of the environment's *state*, $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of available states. On the basis of s_t, the *agent* selects an action $a_t \in \mathcal{A}$, where $\mathcal{A}(s_t)$ represents all actions available in the state s_t. Selecting an action a_t in

[17] A *Markov Decision Process* is a formalization of sequential decision making process. In such a model, the environment is configured as a set of states in which the selection of an action, that depends only on the current environmental state, has to maximize immediate and future rewards.

a state s_t, the system makes a *transition* from s_t to a new state s_{t+1}. The transition probability, in the MDP, depends only on the current state, and does not depend on previous actions and states, that is:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ...) = P(s_{t+1}|s_t, a_t)$$

In the current state s_t, it is incorporated all the necessary informations in order to take an optimal decision, so it does not matter which states or actions preceded it. In the subsequent state s_{t+1}, the *agent* receives a reward $r_{t+1} \in \mathcal{R}$, as a consequence of the selected action a_t. The reward represents the evaluation of the one-step decision-making performance and, as said previously, the goal is to maximize the long term performance, which is assessed by the total accumulated rewards.
The interaction between the agent and the environment can be represented in the figure below:

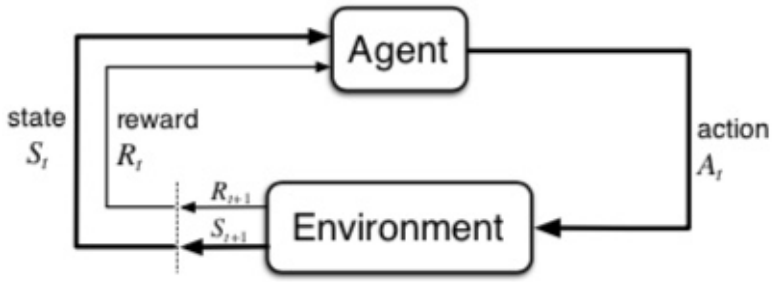


Figure 2.1: Interaction between agent and environment
in Markov Decision Process
([20])

We can observe in the figure that the *agent* chooses an action in a state and the perception it receives from the environment consists in the environment's state after the action plus a scalar reward signal associated to that step.
The behavior of the *agent* is carried on by its *policy*, π_t, that is a function $\pi_t(s, a)$ that outputs an action $a_t = a$ for each state $s_t = s$. As the goal of the *agent* is to maximize the reward in the long run, it has to implement its *policy* in each step. *Reinforcement Learning* methods point out the way in which the *agent* has to improve its policy, on the basis of its experience. The assumption underlying the

Reinforcement Learning is that a model of MDP is not known, that is, is not known the transition and the reward models. Here come in play the role of the interaction of the algorithm with the environment that induces the *exploration-exploitation* trade-off, in order to optimize its behavior, being directed by the subsequent rewards.

2.2 Reinforcement Learning elements

The *Reinforcement Learning* system is constituted by four subelements: a *policy*, a *reward function*, a *value function* and a *model* for the environment, that is optional [20].

A *policy* determines the way in which the agent behaves at a givent time instant. More specifically, the *policy* is a mapping of the set of actions to be taken when the agent will find itself in a set of environment states. The *policy* is sufficient to determine the behavior of the agent in reaching its objective.

A *reward function* determines the immediate goal of the agent. The *reward* for being in a state or taking an action in a state is represented by a single number. It is the primary indicator for changing the *policy* as it specifies what are good and bad events for the agent. For example, in a biological system, the *reward* corresponds to the experiences of pleasure or pain. Consequently, it can be viewed as the direction towards which the system, that is the MDP, must be addressed. Thus, it indicates where the agent is directed, but not how to reach it. This means that the computation of the *reward function* is external to the agent, so it can not be arbitrarily changed as it is outside the agent's control, thus, it is part of the environment. However, the objective of the agent is the maximization of the *reward* in the long run, which can be measured by the *value function*.

The *value function* measures the value of a state as a whole amount of *rewards* that can be obtained over the future by the agent, starting from the actual state in which it is. So, it measures how good is for the agent being in a state, or taking action in a state, in the long run. Since *rewards* are a measure of the immediate and intrinsic desirability of a state, the *values* measure the long-run desirability

of states, taking into consideration the probable future states and the associated *rewards*. For instance, a state might lead a low *reward* in the short-run, but a high *reward* in the long-run. Also the reverse is true. The agent has to prefer always the former respect to the latter.

So, as *rewards* derive directly from the environment, they are interpreted in a primary sense, whereas *values*, as predictions of *rewards*, are interpreted in a secondary sense. In particular, without *rewards* is not possible to estimate *values* but, to estimate *values* is necessary to realize *rewards*. As the primary objective of the agent is the maximization of the *value*, decisions are made on the basis of *values* estimation. The agent has to seek actions that bring states with the highest *value* not the highest reward, as this actions bring the greater amount of *rewards* in the long-run. While the *rewards* derive directly from the environment, *values* have to be estimated and reestimated by the agent in each step, at each time instant.

The *model* of the environment can forecast the behavior of the environment, that is the *model* can conduct the agent toward the next state and the next *reward* based on the actual state. *Models* are useful for designing future actions to take future states. However, in *Reinforcement Learning* algorithms a model for the environment is unknown and, so, I consider only the model free methods.

2.3 Goals, Rewards and Optimality criteria

In *Reinforcement Learning* the goal of the agent is the maximization of the cumulative reward it gather in the long run, and not the immediate reward $r_{t+1} \in \mathcal{R}$, expressed as a single number. The sequence of rewards the agent cumulates after time step t can be denoted $r_{t+1}, r_{t+2}, r_{t+3}, ...$ The sum of these sequence of rewards determines the *return* R_t, whose expected value maximization constitutes the objective of the agent. R_t is a function of the actual and future rewards and can be represented as follows:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T$$

where T is the final time step. When T is known, the formula above determines the model of optimality for a *finite horizon* case. However, in the *infinite horizon* model, where T is unknown and might be infinite, the *discounting* concept must be introduced. Based on this approach, the agent attempts to select actions so that the sum of the discounted rewards it receives over time is maximized. In particular, the agent selects the action a_t, in the time step t, in order to maximize the discounted return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

where the parameter $\gamma \in [0, 1]$ is the *discount rate*. The *discount rate* determines the present value of future rewards. Rewards gathered later in time are discounted more than rewards obtained earlier, as they are actualized. Indeed, a reward received k time steps in the future is worth γ^{k-1} times. The *discount rate* guarantees that the sum of the rewards in the *infinite horizont* model is finite. When $\gamma = 0$, the agent is said to be myopic as it is concerned about maximizing immediate rewards and not the future ones. When $\gamma = 1$, the agent is more farseeing as it gives heavy weights to future rewards.

2.4 Value Functions

Reinforcement Learning algorithms imply the estimation of *value functions* that evaluate "how good", in terms of expected return, it is for the agent to be in a state, or to select a given action in a given state. The rewards the agent expects to receive varies on the basis of the action selected, that is on the basis of particular policies. So, technically, the *value* of a state s under a policy π, denoted $V^\pi(s)$, is the expected return from s following π. Formally, $V^\pi(s)$, so called *state-value function for policy* π, can be defined as:

$$V^\pi(s) = \mathbb{E}_\pi\{R_t | s_t = s\} = \mathbb{E}_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}$$

where $\mathbb{E}_\pi\{\cdot\}$ is the expected value under policy π.

Analogously, the value of choosing an action a, in a given state s, following a policy π, is the expected return from s, selecting the action a, under the policy π. The function $Q^\pi(s,a)$ is called *action-value function for policy* π:

$$Q^\pi(s,a) = \mathbb{E}_\pi\{R_t|s_t = s, a_t = a\} = \mathbb{E}_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\right\}.$$

It is possible to estimate both *value functions* from experience. Indeed, the agent should maintain one of the two *value functions* as parameterized function and adjust the parameters of the chosen function, on the basis of observed rewards, in order to match better the subsequent states.

Value functions used in *Reinforcement Learning* satisfy a fundamental property that is the satisfaction of certain recursive relationships. The expression of the *state-value function*, $V^\pi(s)$, for any policy π and any state s, can be defined in terms of the so-called *Bellman Equation*[18], in which the following consistency condition holds between the value of s and the value of each possible subsequent state:

$$\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi\{R_t|s_t = s\} \\
&= \mathbb{E}_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\right\} \\
&= \mathbb{E}_\pi\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t = s\right\} \\
&= \mathbb{E}_\pi\left\{r_{t+1} + \gamma V^\pi(s_{t+1})|s_t = s\right\}
\end{aligned}$$

The *state-value function* $V^\pi(s)$ is the unique solution to the *Bellman Equation*. The *Bellman Equation* averages all the possible next states, weighting each by its

[18]The *Bellman Equation* is a recursive formula that expresses the relationship between the value of a state and all successive states' values.

possibility of happening. It establishes that the value of the actual state must equal the discounted value of the expected subsequent states, plus the corresponding reward.

A similar result is available for the *action-value function*, $Q^{\pi}(s,a)$, which is a unique solution for the *Bellman Equation*. Indeed, for any policy π, for any state s and for any action a, there is a consistency condition that is valid between any pair value (s,a) and the value of each subsequent pair:

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}_{\pi}\{R_t | s_t = s, a_t = a\} \\ &= \mathbb{E}_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\} \\ &= \mathbb{E}_{\pi}\left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a \right\} \\ &= \mathbb{E}_{\pi}\left\{ r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) | s_t = s, a_t = a \right\}. \end{aligned}$$

2.5 Optimal Value Functions

Achieving a *Reinforcement Learning* objective, for any given MDP, means finding the best policy that receives the highest reward in the long run by exploring the environment on the basis of own experience. A policy π is stated to be better than or equivalent to a policy π' if its value function is higher or equal to that of π' for each state. In particular, $\pi \geq \pi'$ if and only if $V^{\pi}(s) \geq V^{\pi'}(s)$ for all $s \in \mathcal{S}$ or $Q^{\pi}(s,a) \geq Q^{\pi'}(s,a)$ for all $s \in \mathcal{S}$ and all $a \in \mathcal{A}$. There is always an *optimal policy*, π^*, that is better than or equal to all other policies. The *optimal policy* defines the *optimal state-value function*, V^*, that can be written as:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad for\ all\ s \in \mathcal{S}$$

or the *optimal action-value function*, Q^*, defined as:

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \quad for\ all\ s \in \mathcal{S}\ and\ all\ a \in \mathcal{A}.$$

As V^* and Q^* are *value function* for a policy, they satisfy the *Bellman equation* and, as they are the *optimal state-value function* and *optimal action-value function*, they satisfy the *Bellman optimality equation*[19]. It specifies that the value of a state in which an optimal policy is followed, have to be equal to the expected return obtained by selecting the best action in that state.
The *Bellman optimality equation* for $V^*(s)$ can be represented as follows:

$$\begin{aligned}
V^*(s) &= \max_{a\in\mathcal{A}} Q^{\pi^*}(s,a) \\
&= \max_a \mathbb{E}_{\pi^*}\{R_t | s_t = s, a_t = a\} \\
&= \max_a \mathbb{E}_{\pi^*}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \\
&= \max_a \mathbb{E}_{\pi^*}\left\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a\right\} \\
&= \max_a \mathbb{E}_{\pi^*}\left\{r_{t+1} + \gamma V^{\pi^*}(s_{t+1}) | s_t = s, a_t = a\right\}.
\end{aligned}$$

Analogously, the *Bellman optimality equation* for $Q^*(s,a)$ is:

$$Q^*(s,a) = \mathbb{E}\Big[r(s_t, a_t, s_{t+1}) + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\Big].$$

The *Bellman optimality equation* gives a system of equations, one for each state. Solving the system made by n equations in n unknowns value functions, for each n states, means finding a unique solution, $V^*(s)$, independent of the followed policy. Then, to find the optimal policy associated to $V^*(s)$, the action that gives the higher

[19]The *Bellman optimality equation* is that equation in which the *value function* satisfies the *Bellman equation* for the optimal value function.

value in each state, among all the available actions in that state, is selected to move the agent in the next state. In this case the policy is called to be *greedy*, that defines a policy which selects actions looking only at short-term consequences. But, using $V^*(s)$ in evaluating the one-step consequences, the *greedy* policy is optimal also in the long run, as $V^*(s)$ takes into consideration the rewards related to future actions. Consequently, by means of $V^*(s)$ the optimal expected long-term return is a value that is locally and immediately available for each state. So, looking for an action that gives an optimal policy in the next state yields to the long-term optimal actions.

2.6 Policy evaluation, Policy improvement and Value iteration

The *Reinforcement Learning* approach in achieving its objectives consists in estimate value functions in order to determine good policies. The first step is the *policy evaluation*, that consists in computing a state-value function, V^π, for an arbitrary policy π, through an iterative method, that is through successive approximations. This method starts from an initial approximation value of the state-value function, $V_0^\pi(s)$, that is chosen arbitrarily. Each next state-value function is obtained computing the Bellman equation for V^π, using it as an update rule, which updates the current value function V_k^π into V_{k+1}^π:

$$V_{k+1}^\pi(s) = \mathbb{E}_\pi\{r_{t+1} + \gamma V_k^\pi(s_{t+1}) | s_t = s\} \quad for\ all\ s \in \mathcal{S}.$$

Each iteration of this method computes each successive approximation by applying the same operation to each s, that is by updating the old value of s with a new value. This algorithm, called *iterative policy evaluation*, assures the convergence of the sequence $\{V_k^\pi\}$ to V^* as $k \to \infty$ if $\gamma < 1$. However, in practice it is not possible to evaluate the convergence in the limit so it must be stopped before, when the quantity $|V_{k+1}^\pi(s) - V_k^\pi(s)|$ is suitably small.

Once the value function of a policy is known, the next step consists in improving the policy. This process, called *policy improvement*, can be done by computing the value of all actions through the action-value function, $Q^{\pi}(s,a)$, and check if it is higher than $V^{\pi}(s)$ for some action $a \in \mathcal{A}$. If it is greater, then, selecting a in s and then follow π determines a better policy. So, *policy improvement* produces a strictly better policy except when the original policy is already optimal. It can be extended to all actions in all states. The *policy improvement* may determine an increse of the global return in the long-run, choosing the best action in each state, as follow:

$$a = \begin{cases} \pi^{'}(s) \ with \ probability \ 1 \ - \varepsilon \\ \\ a \in \mathcal{A}(s) \ with \ probability \ \varepsilon \end{cases}$$

where $\varepsilon \in (0,1)$ and $\pi'(s)$ is the candidate policy which maximizes $Q^{\pi}(s,a)$. That is, $\pi'(s)$ is the *greedy* policy and can be expressed as:

$$\begin{aligned} \pi^{'}(s) &= \arg\max_{a} Q^{\pi}(s,a) \\ &= \arg\max_{a} \mathbb{E}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, a_t = a\} \end{aligned}$$

where $\arg\max_a(\cdot)$ indicates the best action that maximizes its argument. This selection approach, the $\varepsilon-$*greedy policy*, is applicable both to deterministic policy and to stochastic policy. The action is selected between two policies. In particular, the *greedy* policy, that is deterministic, is selected with probability $1 - \varepsilon$ and the non-*greedy* policy is selected with probability ε. Selecting an action from the *greedy* policy means exploiting the current knowledge of the action values. Instead, selecting a random action from those available in the non-*greedy* policy means exploring the actions available in a state in order to select the one that is better than the *greedy* action. The first policy, called *exploitation* policy, maximizes the expected return in the short-run. The *exploration* policy, that is the last one, may increase the reward in the long-run.

This method of finding an optimal policy through the sequence of *policy evaluation* and *policy improvement* is called *policy iteration*. However, in the iterative computation of *policy evaluation* the convergence to V^* occurs only in the limit and it is computationally expensive. A solution is provided by the *value iteration Bellman* algorithm, through which it is possible to break off the evaluation after one iteration. This algorithm combines a truncated version of *policy evaluation* with the *policy improvement* without the need to wait for full convergence. Indeed, it is possible to stop the evaluation after the first step and, based on this evaluation, proceed with the policy improvement. This iterative process can be represented by the following expression:

$$V^{\pi}_{k+1}(s) = \max_{a} \mathbb{E}\{r_{t+1} + \gamma V^{\pi}_{k}(s_{t+1}|s_t = s, a_t = a)\}$$

where $V^{\pi}_{k+1}(s)$ represents the updated estimate of the state-value function under the new policy, determined by the improvement strategy in the step $k+1$, taking into consideration the estimation of the policy in step k. This expression is identical to the policy evaluation expression with the difference that it requires the maximum to be taken among the actions. Like *policy evaluation*, also *value iteration* stops, in practice, when the state-value function changes by only a small amount.

2.7 Temporal-Difference Learning

Temporal Difference (TD) learning algorithms update estimates of values on the basis of other learned estimates, without the need to wait for a final outcome. This method of estimation, based on experience, is called *bootstrapping*, and it does not require an environmental model.

TD methods solve the prediction control problems[20] using the past experience as the result of a followed policy π. For each nonterminal state s_t, they update

[20]The prediction control problem consists in the estimation of a value function V^{π} for a given policy π (prediction) and in finding an optimal policy (control) for the estimated function.

their estimate V of V^* using the reward r_{t+1} occured in time $t+1$. The simplest method, known as *TD(0)*, which consider only one step ahead when adjusting value estimates, has the following update expression:

$$V_{k+1}(s_t) = V_k(s_t) + \alpha[r_{t+1} + \gamma V_k(s_{t+1}) - V_k(s_t)]$$

where $\alpha \in (0, 1]$ is the *learning rate* and measures the degree of the update. The expression in the brackets, that is the difference between $r_{t+1} + \gamma V_k(s_{t+1})$, the *target*, and $V_k(s_t)$, the old estimate, is an *error* in the estimate:

$$\delta_k = r_{t+1} + \gamma V_k(s_{t+1}) - V_k(s_t)$$

Whenever there is a change in successive predictions, that is a change in the *error*, a learning source is produced, that is an amount able to adjust and improve the value of the current policy. *TD* methods try to minimize the difference between subsequent predictions and, in this way, the value of the current policy is adjusted and improved in each time step. For this reason *TD* methods are said to be implemented in an online incremental fashion.

The main advantage of the *TD* methods is that they do not require a model of the environment neither of the transitions probability distribution for two subsequent steps. In fact, they learn from each transition independently of the action selected. This makes the *TD* methods an appropriate instrument to solve stochastic control dynamic problems, such as financial ones.

To adapt *TD* methods to financial data, that are often non-stationary time series, that is, have no constant mean and no constant variance over time, it would be more appropriate to give more weight to recent rewards, so to what is learned latest, than to rewards received in steps far in the time. A widespread method is to use a constant *learning rate*. Replacing $R_{t+1} = r_{t+1} + \gamma V_k(s_{t+1})$, it is possible to rewrite the previous expression of $V_{k+1}(s_t)$ as follows:

$$V_{k+1}(s_t) = V_k(s_t) + \alpha[R_{t+1} - V_k(s_t)].$$

This result in $V_{k+1}(s_t)$ is a weighted average of past rewards and the initial estimate $V_k(s_0)$:

$$\begin{aligned} V_{k+1}(s_t) &= V_k(s_t) + \alpha[R_{t+1} - V_k(s_t)] \\ &= \alpha R_{t+1} + (1-\alpha)V_k(s_t) \\ &= \alpha R_{t+1} + \alpha(1-\alpha)R_t + (1-\alpha)^2 V_k(s_{t-1}) \\ &= \alpha R_{t+1} + \alpha(1-\alpha)R_t + \alpha(1-\alpha)^2 R_{t-1} + ...+ \\ &\quad \alpha(1-\alpha)^{T-1}R_1 + (1-\alpha)^T V_k(s_0) \\ &= (1-\alpha)^T V_k(s_0) + \sum_{t=1}^{T} \alpha(1-\alpha)^{T-t} R_t. \end{aligned}$$

This is a weighted average sum in which the sum of the weights is $(1-\alpha)^T + \sum_{t=1}^{T} \alpha(1-\alpha)^{T-t} = 1$. Given that $(1-\alpha) < 1$, the weight given to the rewards decreases exponentially as the number of intermediate rewards increases.
In stationary problems there are two conditions to be met in order to assure the convergence, with probabiliy 1, of the estimate of the state-value function $V_k(s_t)$ to the true value V^*:

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty.$$

The first condition ensures that the steps are enough numerous to overcome any initial conditions or random fluctuations. The second condition is required to ensure that the steps become small enough in order to converge to the optimal state-value function V^*, or to the optimal action-value function Q^*. The two conditions are met when the *learning rate*, α, decreases over time. Indeed, if α decreases after each time step, the estimation of $V_k(s_t)$ becomes more precise until the convergence

to V^*. However, in a non-stationary environment is preferable a constant *learning rate*. In this case, the second condition $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$ is not satisfied, meaning that the estimates might never completely converge to V^* but might continue to change on the basis of the last received rewards.
Therefore, for any fixed policy, π, the algorithm converges in the mean to the true value V^* if the *learning rate* is constant and sufficiently small.

2.8 Q-Learning Algorithm

Among the *TD* methods, the *Q-Learning (QL) algorithm* is one of the most used method in solving prediction control problems. This method estimates an action value function $Q(s, a)$ in a state s, by repeteadly executing all available actions a, following a policy π. By trying all actions in a state, the algorithm learns which is the best in terms of rewards and, by using the one-step lookahead it updates the previous action-value function. In such a way both the policy and the value function are improved.
QL is an *online off-policy control algorithm* which has the following update-rule structure:

$$Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q_k(s_{t+1}, a) - Q_k(s_t, a_t)].$$

The agent pass from state s_t to state s_{t+1} by taking action a_t and obtaines a reward r_{t+1} in such a way that $Q_k(s_t, a_t)$ is maximized at each state. It is *online* as it updates its estimates after each visit of each time step, with no need to wait until the terminal one. It is *off-policy* as the improvement process passes through two policies, that is a policy that is used to estimate value functions while using another policy which is more exploratory. More precisely, the algorithm learns regardless the performed actions as it updates on the basis of the action that maximizes the value function on the next step, that is on the basis of the greedy action, albeit in

the actual state-action pair is selected a greedy action or an exploratory one. It is a *control algorithm* as it selects actions in order to reach its objective, that is the highest estimate of the action-value function. Indeed, the *QL algorithm* converges to the optimal action-value function, Q^*, regardless the exploration policy followed, under the conditions that each state-action pair is executed an infinite number of times, and the learning rate α decreases appropriately. For this reason the *QL algorithm* is said to be *exploration-insensitive*.

There are different exploration strategies that can be used by the *QL algorithm*. The most widely used is the ϵ-greedy one, in which the greedy action is selected with probability 1-ϵ and another random action is selected with probability ϵ from an uniform distribution. One drawback of this strategy consists in the fact that, as exploration actions are selected uniform randomly, it is likely to select an action that is not the best action. An improved and more complex strategy uses the Boltzmann distribution function:

$$\frac{e^{\frac{Q_t(s_t,a)}{\tau}}}{\sum_{i=1}^{n} e^{\frac{Q_t(s_t,a_i)}{\tau}}}$$

where τ is a positive parameter called *temperature*. If τ is set to be very high, $\tau \to \infty$, the actions are selected randomly, so they are equiprobable. If τ is small, with $\tau \to 0$, this strategy approaches to the greedy method.

2.9 Value Function Approximation and Optimization

In the financial problem that I will consider, that is the application of a *FTS* on real time series, stock returns are continous and stochastic, so an exact representation of the action-value function cannot be provided. Indeed, approximation is needed. An effective approximation is based on two prerequisites. The first one consists in chosing an approximation function with enough elements so that it can approximate

closely the function that have to be approximated. The second one requires an efficient algorithm to calibrate the parameters of the approximation function. However, the two mentioned objectives are often in conflict. In fact, a large number of variables characterizing the approximation function requires a large number of parameters to be tuned or can happen that the dependence among the parameters is non linear. These elements increase the computational complexity.
Using parametric approximator, it is possible to represent the action-value function $Q(s_t, a_t)$ by a parameterized functional form with a fixed parameter vector $\boldsymbol{\theta}_t \in \Theta$.

$$\boldsymbol{\theta}_t = [\theta_0 \quad \theta_1 \quad \theta_2 \quad \theta_3 \quad \dots \quad \theta_N]'.$$

The approximator of the action-value function, $Q(s_t, a_t; \boldsymbol{\theta}_t)$, totally depends by the parameter vector that might vary step by step. As the objective is to find the optimal parameter vector $\boldsymbol{\theta}^*$ that gives the best approximation of the action-value function, that is which minimizes the "gap" between the unknow action value function Q^* and its estimate $Q(s_t, a_t; \boldsymbol{\theta}_t)$, the minimization of the *mean-squared error (MSE)* aproach is widespread used:

$$\min_{\boldsymbol{\theta}_t} MSE(\boldsymbol{\theta}_t) = \sum_{s \in \mathcal{S}, a \in \mathcal{A}(s)} [Q^*(s, a) - Q(s_t, a_t; \boldsymbol{\theta}_t)]^2.$$

An optimal objective would consist in finding the *global optimum*, in correspondence of which, for the vector $\boldsymbol{\theta}^*$, the $MSE(\boldsymbol{\theta}^*) \leq MSE(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$. However, this result can be reached if the approximation function is linear, otherwise it may converge to a *local optimum* [20], that is to a vector $\boldsymbol{\theta^*}$ for which $MSE(\boldsymbol{\theta^*}) \leq MSE(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$ in the proximity of $\boldsymbol{\theta^*}$.
As this optimization problem concerns the minimization of the *MSE*, the concepts of *global* and *local minimizer* will be respectively given:

> *Local minimizer*: A vector $\boldsymbol{\theta}^*$ is a *local minimizer* of a function f if the following inequality $f(\boldsymbol{\theta}^*) \leq f(\boldsymbol{\theta})$ is valid for all $\boldsymbol{\theta}$ near $\boldsymbol{\theta}^*$.

Global minimizer: A vector $\boldsymbol{\theta}^*$ is called *global minimizer* of a function f if $f(\boldsymbol{\theta}^*) \leq f(\boldsymbol{\theta})$ is valid for all $\boldsymbol{\theta}$.

To guarantee that a *local minimizer* is also a *global minimizer*, the action-value function must satisfy the convexity property. Before proceeding, some definitions are appropriate [17]:

Definition: A set $C \subset E^n$ is said to be *convex* if $\forall \boldsymbol{x_1}, \boldsymbol{x_2} \in C$ and every real number $\alpha \in (0, 1)$, the point $\alpha \boldsymbol{x_1} + (1 - \alpha)\boldsymbol{x_2} \in C$, where $\boldsymbol{x}$ is an n-dimensional vector in the n-dimensional Euclidean space E^n.

From a geometrical point of view, a set is defined as *convex* if, given two points belonging to the set, every point on the line segment connecting the two points is also part of the set.

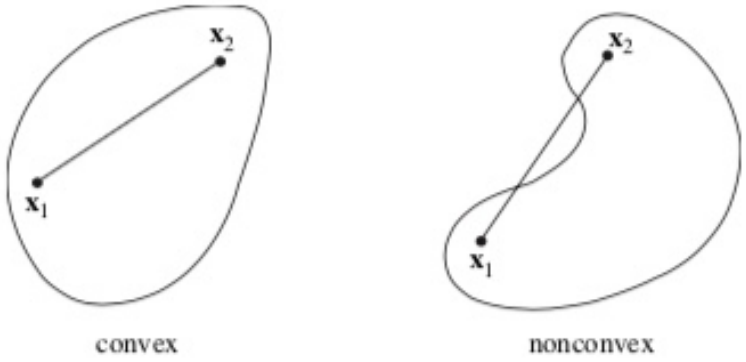


Figure 2.2: Convexity
([17])

Proposition: *Convex* sets in E^n satisfy the following relations:

a. If $C \subset E^n$ is a *convex* set and β is a real number, then the set

$$\beta C = \{\boldsymbol{x} : \boldsymbol{x} = \beta \boldsymbol{c}, \boldsymbol{c} \in C\}$$

is *convex*.

b. If C and D are two *convex* sets, then the union

$$C \cup D = \{\boldsymbol{x} : \boldsymbol{x} = \boldsymbol{c} + \boldsymbol{d}, \boldsymbol{c} \in C, \boldsymbol{d} \in D\}$$

is *convex.*

c. The intersection of any collection of *convex* sets is *convex.*

For a *convex function* with more than one variable, the following definition holds [17]:

Definition: A function f defined on a *convex* set Ω is said to be *convex* if $\forall \boldsymbol{x_1}, \boldsymbol{x_2} \in \Omega$ and every $\alpha \in [0, 1]$, the following expression is valid:

$$f(\alpha \boldsymbol{x_1} + (1 - \alpha)\boldsymbol{x_2}) \leq \alpha f(\boldsymbol{x_1}) + (1 - \alpha) f(\boldsymbol{x_2}).$$

If $\forall \alpha \in (0, 1)$ and $\boldsymbol{x_1} \neq \boldsymbol{x_2}$, the following is valid:

$$f(\alpha \boldsymbol{x_1} + (1 - \alpha)\boldsymbol{x_2}) < \alpha f(\boldsymbol{x_1}) + (1 - \alpha) f(\boldsymbol{x_2}).$$

f is said to be *strictly convex.*

In order to provide a graphical exemplification, a *convex function* (of one variable) is the one in which the line connecting two points on its graph lies above the curve representing the function.

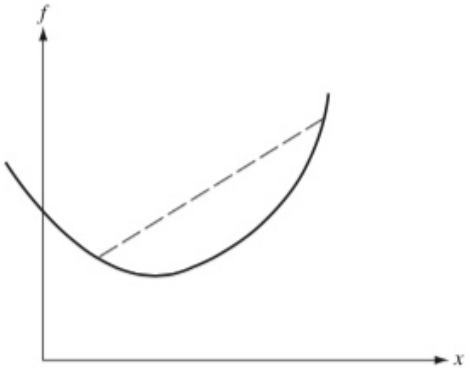


Figure 2.3: Convex function ([17])

The existence of a solution for the minimization problem here considered is guaranteed by the *Weierstrass theorem*, which enunciates that if f is continous and the set of parameters Θ is compact[21], a solution exists [17]. The minimization of the *MSE* concerns the minimization of a quadratic and unconstrained function, whose parameters can assume any value in the feasible set Θ, as $\Theta = E^n$, that is Θ is not compact. To reduce the MSE step by step until the minimum is reached, a minimizing sequence can be used.

To find the minimizer $\boldsymbol{\theta}^*$, a method that determines the direction of $\boldsymbol{\theta}$ can be used. It is based on a parameter column vector, $\boldsymbol{d_t} \in E^n$, which determines the descent direction, at each iteration, of the *MSE* from the parameter vector $\boldsymbol{\theta}$ toward $\boldsymbol{\theta^*}$. The consequent sequence vector $\boldsymbol{\theta_1, \theta_2, \ldots, \theta_T}$ is produced based on the following algorithm:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \alpha \boldsymbol{d_t}$$

where $\alpha \in (0, 1]$ is the *step-size parameter*, or *learning rate*, that determines the size of the movement. Consequently, step by step, the parameter vector is nearer to the minimizer $\boldsymbol{\theta}^*$.

2.10 Gradient Descent Method

In minimization problems of function approximation processes is widespreadly used the *gradient descent* method. This method, well suitable to *Reinforcement Learning*, is applied to the approximator of the action-value function in order to find the minimizer parameter vector.

In the *gradient descent* method the parameter column vector is a vector $\boldsymbol{\theta}_t$ and the approximate function $Q(s_t, a_t; \boldsymbol{\theta}_t)$ is differentiable for all $s \in \mathcal{S}$ and all $a \in \mathcal{A}$. Step by step the interaction between the agent and the environment produces examples

[21] A set is *compact* if it is both closed and bounded, that is if it is closed and it is contained within some sphere of finite radius [17].

of state-action pairs. The strategy used by the *gradient descent* method consists in minimizing the *MSE* between the unknown and estimated value function, so adjusting the parameter vector by a small amount, after each step, in the direction that reduces the error:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \frac{1}{2}\alpha\nabla_{\boldsymbol{\theta}_t}[Q^*(s,a) - Q(s_t,a_t,\boldsymbol{\theta}_t)]^2 \\ &= \boldsymbol{\theta}_t - \alpha[Q^*(s,a) - Q(s_t,a_t,\boldsymbol{\theta}_t)]\nabla_{\boldsymbol{\theta}_t}[Q^*(s,a) - Q(s_t,a_t,\boldsymbol{\theta}_t)]\end{aligned}$$

where $\nabla_{\boldsymbol{\theta}_t} f(\boldsymbol{\theta})$ is the vector of partial derivatives of $f(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, that is the gradient of the function $f(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, and where the direction is descending:

$$\begin{aligned}\boldsymbol{d}_t &= -\frac{1}{2}\nabla_{\boldsymbol{\theta}_t}[Q^*(s,a) - Q(s_t,a_t,\boldsymbol{\theta}_t)]^2 \\ &= -[Q^*(s,a) - Q(s_t,a_t,\boldsymbol{\theta}_t)]\nabla_{\boldsymbol{\theta}_t}[Q^*(s,a) - Q(s_t,a_t,\boldsymbol{\theta}_t)].\end{aligned}$$

As $Q^*(s,a)$ is the optimal but unknown value of the state-action function, the previous expression is rewritten as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha[Q^*(s,a) - Q(s_t,a_t,\boldsymbol{\theta}_t)]\nabla_{\boldsymbol{\theta}_t} Q(s_t,a_t,\boldsymbol{\theta}_t).$$

The *gradient descent* method proceeds in the direction of the negative gradient of the parametrized function which is to minimize. The negative gradient, step by step, is addressed towards the direction in which the *MSE* is minimized, so until $\nabla_{\boldsymbol{\theta}_t} Q(s_t,a_t,\boldsymbol{\theta}_t)$ approaches zero. However, the *descent gradient* takes only small steps towards the negative gradient as finding an action-value function with zero error in each state-action pair, that is a difference between the unknown action value function and its estimate near zero, is not feasible. In fact, what is needed is an approximation that balances the errors in different states. Finally, the

convergence of the *gradient descent* method to the local optimum is guaranteed, in a stationary environment, if the previously stated standard stochastic conditions, $\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$, are satisfied. In a non-stationary environment the convergence is reached in the mean.
As said previously, the $Q^*(s, a)$ is not available, so it can be replaced by an approximation to it:

$$Q^*(s, a) \cong q_t = r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \boldsymbol{\theta_t}).$$

If q_t is an unbiased estimate, that is if $E[q_t] = Q^*(s, a)$ for each t, therefore the convergence of $\boldsymbol{\theta_t}$ to a local minimum is guaranteed, under the usual stochastic approximation conditions for decreasing learning rate. The substitution gives the following parameter updating expression:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha[q_t - Q(s_t, a_t, \boldsymbol{\theta}_t)]\nabla_{\boldsymbol{\theta}_t} Q(s_t, a_t, \boldsymbol{\theta}_t) \\ &= \boldsymbol{\theta}_t + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \boldsymbol{\theta_t}) - Q(s_t, a_t, \boldsymbol{\theta}_t)]\nabla_{\boldsymbol{\theta}_t} Q(s_t, a_t, \boldsymbol{\theta}_t).\end{aligned}$$

So, as the parameter vector $\boldsymbol{\theta}_t$ is updated step by step, a move toward the optimal value is done.

2.11 Convergence Issues

As stated previously, the optimization problem which concerns the search of a parameter vector $\boldsymbol{\theta}_t$ that minimizes the difference between the unknown and the approximated action value function, cannot guarantee convergence of the parameter vector to a global minimum, in particular when the function to minimize is not convex, because of the presence of local minimum. In fact, the gradient descent method follows downwards the shape of the function, looking for a minimum, and so it is attracted by any type of minimum, either local or global. This constitutes

one of the weaknesses of the gradient descent method when dealing with non-convex functions with local minimum. In fact, once it reaches any stationary point[22], even a local maximum, it stops in that point. In theory it is possible, for a parameter vector $\boldsymbol{\theta}$, to have multiple limit points[23] if there exists a set of multiple local minima. The question now is to check if each limit point is also a stationary point. To assure that the vector $\boldsymbol{d_t}$ and the gradient $\nabla_{\boldsymbol{\theta_t}} f(\boldsymbol{\theta_t})$ will not become asymptotically orthogonal[24] in proximity of a nonstationary point, as two vectors can be orthogonal only in a stationary point, two conditions must be imposed on the feasible direction $\boldsymbol{d_t}$:

$$c_1 \parallel \nabla_{\boldsymbol{\theta_t}} f(\boldsymbol{\theta_t}) \parallel^2 \leq \ -\nabla_{\boldsymbol{\theta_t}} f(\boldsymbol{\theta_t})' \boldsymbol{d_t} \quad \text{and} \quad \parallel \boldsymbol{d_t} \parallel \leq c_2 \parallel \nabla_{\boldsymbol{\theta_t}} f(\boldsymbol{\theta_t}) \parallel$$

where c_1 and c_2 are positive scalars. The proof of the convergence for a constant learning rate is given by the following proposition [5]:

> Let $\boldsymbol{\theta_t}$ be a sequence generated by a gradient method $\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \alpha \boldsymbol{d_t}$, where $\boldsymbol{d_t}$ satisfy the just mentioned conditions. Assume that for some constant $L > 0$, one has
>
> $$\parallel \nabla_{\boldsymbol{\theta_t}} f(\boldsymbol{\theta_t}) - \nabla^*_{\boldsymbol{\theta}} f(\boldsymbol{\theta_t}^*) \parallel \leq L \parallel f(\boldsymbol{\theta_t}) - f(\boldsymbol{\theta_t}^*) \parallel, \quad \forall \boldsymbol{d_t}, \boldsymbol{d_t}^* \in \Omega$$
>
> and
>
> $$0 < \alpha < \frac{2c_1}{Lc_2^2}$$

Then either $f(\boldsymbol{\theta_t}) \to -\infty$ or else $f(\boldsymbol{\theta_t})$ converges to a finite value and $\lim_{t\to\infty} \nabla_{\boldsymbol{\theta_t}} f(\boldsymbol{\theta_t}) = 0$. Furthermore, every limit point of $\boldsymbol{\theta_t}$ is a stationary point of f[25].

[22] A stationary point is defined as a vector $\boldsymbol{\theta^*}$ satisfying the condition $\nabla_{\boldsymbol{\theta^*}} f(\boldsymbol{\theta^*}) = 0$ [5]

[23] A limit point, whithin some open sphere, is that point for which exists, at some small arbitrarly distance from it, other points different from the limit point.

[24] Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal when their scalar product is zero, that is $\boldsymbol{x'y} = 0$

[25] The stationary point of a function f is a point where the function's derivative or partial derivative is zero.

2.12 Linear Function Approximation

To find the approximator of the action-value function, $Q(s_t, a_t; \boldsymbol{\theta}_t)$, for which the parameter vector $\boldsymbol{\theta_t}$ converges to the optimal one, $\boldsymbol{\theta^*}$, is suitable to use a linear function approximation:

$$Q(s_t, a_t, \boldsymbol{\theta_t}) = \boldsymbol{\theta_t}\boldsymbol{\phi_{s_t}} = \sum_{n=0}^{N} \theta_t(n)\phi_{s_t}(n)$$

where the vector $\boldsymbol{\phi_{s_t}}$ is the so called *squashing function*, which is a suitable transformation of the states, and might be non-linear.
One property of the *squashing function* is the increasing monotonicity, that is given two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ that satisfy the following inequality:

$$\boldsymbol{a} \leq \boldsymbol{b}$$

then the following inequality is true for any increasing transformation of the function f:

$$f(\boldsymbol{a}) \leq f(\boldsymbol{b})$$

This property is useful in order to preserve the order of the transformed values, as in each state, it is selected that action that maximizes the value of the action-value function. In this way is assured that the approximator of the action value function increases in order to approach to the optimal unknow action-value function.

Chapter 3

Q-Learning at work

In this chapter, I describe the specific characteristics of the implemented *FTS*. In particular, I specify all the elements, such as the state descriptors variables, the actions that the *FTS* can choose, the reward functions and the squashing function used. Transaction costs and their application are also illustrated, to make the *FTS* more realistic.

The time series utilized are described and the main descriptive statistics are presented, with some graphical representation, in order to give an idea in which environment the *FTS* will operate.

Combining the different parameters values, different configurations are obtained and tested. Each configuration will be tested simultaneously K times. As the results are always different in each iteration, due to the random initialization of the parameters vector $\boldsymbol{\theta}$ and to the value of the exploratory actions. Finally, in order to obtain one operational trading signal to be adoperated in the real financial market, in each time step it is selected among the majority of the actions performed.

3.1 State representation: s

The environment, in each time step, is represented by a state that illustrates the most relevant informations for the agent. To evaluate the performances of the developed *FTS*, basic and simple environmental informations are used. Indeed, each state, in the implemented algorithm, is described by the actual logarithmic return and the $N-1$ past ones of each asset. So, at time t, the state of the environment can be represented by the following vector:

$$\boldsymbol{s_t} = [e_{t-N+1} \quad e_{t-N+2} \quad \dots \quad e_t],$$

where $e_\tau = ln(p_\tau / p_{\tau-1})$, in which p_τ is the stock price at time τ. In order to get a *FTS* that reacts promptly to new informations, the tests are carried out on the current return, $N = 1$, and on the current return plus the past five ones, $N = 5$, which represents a stock market week.

3.2 Action values: a

In each time step t, the *QL* algorithm tries all available actions, in order to learn and select which is the best in terms of the obtained reward, r_{t+1}. In my model, the actions that the algorithm can carry out, to reach the continous states $\boldsymbol{s_{t+1}}$, are of finite number. The possible actions are:

$$a_t = \begin{cases} -1 \\ 0 \\ 1 \end{cases}$$

where -1 means *"sell or stay-short-in-the-market-signal"*, 0 means *"stay-out-from-the-market-signal"*, implying the closing of any previous open position, and 1 means *"buy or stay-long-in-the-market-signal"* [4].

As consequence of the selected action a_t, the agent receives, in the next time step $t+1$, a reward r_{t+1}. The approach choosen by the algorithm, in selecting the actions, is the $\varepsilon-$*greedy policy*. This approach passes through two policies, that is a *greedy* policy, in which the agent exploits its knowledge of the environment, selected with probibility $1-\varepsilon$, and a non-*greedy* policy, selected with probability ε, in which the agent explores the environment by choosing the actions randomly. The selection of the action through this improvement criterion can be represented as follows:

$$a = \begin{cases} \arg\max_a Q(s_t, a_t, \boldsymbol{\theta_t}) & \textit{with probability } 1-\varepsilon \\ a_t & \textit{with probability } \varepsilon \end{cases}$$

where $\varepsilon \in \{2.5\%, 5\%\}$ and $a_t \sim A_d\{-1, 0, 1\}$. The parameter ε determines the rate of exploration actions, selected from a uniform distribution. Higher is the value, more exploration actions the agent will select. Due to the randomness selection, the agent may choose actions with bad performances, but also may select actions that can achieve good results. To find such winning actions that improve the performances, the agent has to try them. In this way, it explores the environment and then it can exploits the acquired knowledge in order to obtain positive results. The choice between exploration and exploitation is known as the *exploration-exploitation* dilemma.
Actions, in the action-value function computation, are represented as vectors and not as single values -1, 0, 1, that is:

- action -1, meaning *sell or stay-short-in-the-market*, is represented as the vector

$$\boldsymbol{a_{-1}} = [1 \quad 0 \quad 0]';$$

- action 0, meaning *stay-out-from-the-market*, is represented as the vector

$$\boldsymbol{a_0} = [0 \quad 1 \quad 0]';$$

- action 1, meaning *buy or stay-long-in-the-market*, is represented as the vector

$$\boldsymbol{a_1} = [0 \quad 0 \quad 1]'.$$

Such action vectors are used in order to differentiate among the different actions, even if the action value is always the same and, consequently, the sum of all elements in each vector is always 1. This choice is justified by the fact that, the linear regression used to estimate the action-value function for different possible actions, will always penalize some actions respect to others, if different action values will be used. In this case, as the parameters θ and state descriptors s are always the same for the three possible actions, the value of the action-value function will be determined by the value of each action. In fact, if the agent selects a greedy action, so the action which brings the highest value of the estimated action-value function, the action with value 1 (*buy*) will always be favored respect to the action with value -1 (*sell*), as the former value will be higher than the latter one. The action with value 0 will always be neutral. So, in each time step, the value of the three estimated action-value functions is conditioned by the value of the possible actions. In fact, the choice will depend by the value of the product of each action value and the same parameter θ. So:

- the action-value function related to the *sell or stay-short-in-the-market* would be:

$$Q(s_t, a_{-1}; \boldsymbol{\theta}_t) = \boldsymbol{\theta}_t \boldsymbol{\phi}_{\boldsymbol{s_t}} = \theta(0) + \sum_{n=1}^{N-1} \theta_t(n)\phi_{s_t}(n) + \theta(N)\phi(-1);$$

- the action-value function related to the *stay-out-from-the-market* would be:

$$Q(s_t, a_0; \boldsymbol{\theta}_t) = \boldsymbol{\theta}_t \boldsymbol{\phi}_{\boldsymbol{s_t}} = \theta(0) + \sum_{n=1}^{N-1} \theta_t(n)\phi_{s_t}(n) + \theta(N)\phi(0);$$

- the action-value function related to the *buy or stay-long-in-the-market* would be:

$$Q(s_t, a_1; \boldsymbol{\theta}_t) = \boldsymbol{\theta}_t \boldsymbol{\phi}_{\boldsymbol{s_t}} = \theta(0) + \sum_{n=1}^{N-1} \theta_t(n)\phi_{s_t}(n) + \theta(N)\phi(1).$$

The only value that changes in the three estimated action-value functions is the value of the selected actions and, as the squashing function must be monotonic increasing then, in a positive linear regression function, the actions will be ranked as follows:

$$a_{-1} < a_0 < a_1.$$

However, associating to each action a different vector, which sum is always 1, the selection among the actions will depend by the parameter value θ. Now, the action-value functions can be represented as follows:

- the action-value function related to the *sell or stay-short-in-the-market*:

$$Q(s_t, \boldsymbol{a_{-1}}; \boldsymbol{\theta}_t) = \boldsymbol{\theta}_t \boldsymbol{\phi}_{\boldsymbol{s_t, a_{-1}}} = \theta(0) + \sum_{n=1}^{N-1} \theta_t(n)\phi_{s_t}(n) + \theta(N)\phi_t \boldsymbol{a}_{-1};$$

- the action-value function related to the *stay-out-from-the-market*:

$$Q(s_t, \boldsymbol{a_0}; \boldsymbol{\theta}_t) = \boldsymbol{\theta}_t \boldsymbol{\phi}_{\boldsymbol{s_t, a_0}} = \theta(0) + \sum_{n=1}^{N-1} \theta_t(n)\phi_{s_t}(n) + \theta(N)\phi_t \boldsymbol{a}_0;$$

- the action-value function related to the *buy or stay-long-in-the-market*:

$$Q(s_t, \boldsymbol{a_1}; \boldsymbol{\theta}_t) = \boldsymbol{\theta}_t \boldsymbol{\phi}_{\boldsymbol{s_t, a_1}} = \theta(0) + \sum_{n=1}^{N-1} \theta_t(n)\phi_{s_t}(n) + \theta(N)\phi_t \boldsymbol{a}_1.$$

Now, each action vector has its own parameter θ, which assumes differen values. So, the choice depends by the value of one variable, that is the parameter θ multiplied by the value of the action vector, which is always 1.

3.3 Reward function: $r(s_t, a_t, s_{t+1})$

The ultimate aim of each investor is to obtain the higher return, or the higher economic utility or the higher risk-adjusted return. The reward represents, for the agent, the premium or the punishment for a chosen action.
One of the reward functions used in this thesis is the *Sharpe ratio*, that is a risk-adjusted return measure. The *Sharpe ratio*, that is the most used reward function in different *Reinforcement Learning* papers, will be used as a benchmark for the other reward functions.

- The *Sharpe ratio* is defined as:

$$SR_t = \frac{\mathbb{E}_L(e_t)}{\sqrt{\mathbb{V}ar_L(e_t)}} \in \mathbb{R}$$

  where SR_t is the *Sharpe ratio* at time t, $\mathbb{E}_L(\cdot)$ and $\mathbb{V}ar_L(\cdot)$ are, respectively, the sample mean and the sample variance of the returns e_t, obtained by the trading system, computed over the last L stock market days. So, the *Sharpe ratio* gives a measure of the return of an asset, adjusted by its risk. The higher the risk, the higher the return, as riskier assets give greater gains, but also greater losses.

The *Sharpe ratio* is based on the assumption that returns are normally distributed and this can bring to misleading results as, generally, returns are not simmetrically distributed, that is are skewed[26][1]. Furthermore, one drawback in using standard deviation as a measure of risk, consists in the fact that, positive and negative fluctuations are treated in the same way and with the same weight. As an investor looks at gains and losses assymetrically, she/he is more concerned about downside risk, that she/he seeks to avoid. Downside fluctuations are more reflective of investor's intuition of risk than the concept of variance do.

[26]Skewness is a measure of assymetry with respect to the Normal distribution. A positive skewness means that the right tail of the distribution is longer and so the data are concentrated in the left side of the distribution. Contrary, swekness is negative when the left tail of the distribution is longer.

One performance measure that is downside risk-adjusted, so that looks at the left tail of the distribution, is the *Sortino ratio*.

- The *Sortino ratio* is defined as follows:

$$Sortino_t = \frac{\mathbb{E}_L(e_t)}{SSD_L(e_t)} \in \mathbb{R}$$

  where $Sortino_t$ is the *Sortino ratio* at time t, $\mathbb{E}_L(\cdot)$ and $SSD_L(\cdot)$ are, respectively, the sample mean and the sample semi-standard deviation of the returns e_t, obtained by the trading system and computed over the last L stock market days. This ratio gives a measure of return of an asset, adjusted by its downside volatility, that is the negative volatility, so takes into account negative fluctuations.

Another, and more precise, downside risk-adjusted return measure that overcomes the limit of the normal distribution of data is the *Expected Shortfall ratio*.

- The *Expected Shortfall ratio* is represented as follows:

$$ESR_\epsilon = \frac{\mathbb{E}_L(e_t)}{ES_{\epsilon,L}(e_t)} \in \mathbb{R}$$

  where ESR_ϵ represents the ratio at time t between the sample mean of the returns e_t, obtained by the trading system, computed over the last L stock market days, and the *Expected Shortfall* ($ES_{\epsilon,L}$) at a confidence level $\epsilon \in [0, 1]$, computed over the same L market days of the returns e_t. The ES_ϵ formula is:

$$ES_\epsilon(e_t) = \frac{1}{\epsilon} \int_0^\epsilon VaR_p(e_t)\, dp.$$

The ES_ϵ is a coherent[27] measure of risk and can be defined as the expected value of losses below the *Value at Risk (VaR_p)*. Consequently, the computation of the ES_ϵ requires, first, the computation of the VaR_p.
The *Value at Risk* formula can be represented as follows:

$$VaR_p(e_t) = q_p(-e_t)$$

where $q_p(e_t)$ is the p-quantile of e_t. The VaR_p is defined as the maximum expected loss over a given time horizon and a given confidence level $p \in [0, 1]$ [9]. To compute the VaR_p, I used the *historical simulation*[28] method, which does not require the assumption of normal distribution of returns. The *historical simulation* method ranks the returns from the lowest to the highest. Using a confidence level p equal to 5%, the method compute the $95th$ percentile, over the past L stock market days, in correspondence of which the expectation of the worst daily loss will not exceed the worst 5% of returns. Based on the values of VaR_p founded, the ES_ϵ is computed. The latter is an average loss instead of a worst-case loss, like the former. The choice of the *Expected Shortfall* as a measure of risk, instead of the *Value at Risk*, is justified by the fact that this last one is not a coherent risk measure, as it is not subadditive.

The three considered reward functions are computed over the last $L = 11$ and $L = 22$ stock market days, this last corresponding to a stock month.
The reward signal that the *Q-Learning* based financial trading system receives in each time step is the only feedback that the environment gives. It represents the goodness of the selected action and it indicates the direction where the system is addressed.

[27] A measure of risk ρ is coherent if it is *monotonous*, *positevely homogeneous*, *translation invariant* and *subadditive*. The risk measure ρ, defined on a set V, is *monotonous* if, given two random variables $X, Y \in V, X \leq Y \Rightarrow \rho(X) \geq \rho(Y)$. It is *positively homogeneous* if, for $X \in V, h > 0, hX \in V \Rightarrow \rho(hX) = h\rho(X)$. It is *translation invariant* when, for $X \in V, a \in \mathbb{R}, X+a \in V \Rightarrow \rho(X+a) = \rho(X)-a$. It is *subbadditive* if, for $X, Y \in V, X+Y \in V \Rightarrow \rho(X+Y) \leq \rho(X) + \rho(Y)$.

[28] The *historical simulation* is a non-parametric model as, to be defined, it does not require the parameters as the mean and the standard deviation. Indeed, this method directly sample the returns in order to identify the value which corresponds to the required confidence level.

3.4 Transaction costs: δ

Everytime the agent takes an action that changes its postition on a stock, a transaction cost is applied. The transaction costs are applied both in the opening and in the closing of each position, for this reason they are half divided. The transaction cost, δ, can be defined as follows:

$$\delta = \frac{transaction\ cost}{2}.$$

On the tests carried out, the applied transaction cost, δ, is expressed in terms of percentage and it is equal to 12‰. The choice of this value is based on the main transaction costs applied by several Italian brokerage firms.
Transaction costs are applied when a new long or a new short position on a stock is opened or when an old one is closed. The net-of-transaction-costs return at time $t+1$ as consequence of the action taken by the FTS at time t can be represented as follows:

$$r_{t+1}^{net} = a_t e_{t+1} - \delta \mid a_{t+1} - a_t \mid$$

where $a_t e_{t+1} = r_{t+1}^{gross}$. The transaction costs, in such a representation, affect the net logarithmic return only if two subsequent actions are different. The application and the derivation of this formula can be shown below:

- The gross monetary gain, G_{t+1}^{gross}, is obtained from the multiplication between the gross percentage return, r_{t+1}^{gross}, and the gross invested capital, E_t^{gross}:

$$G_{t+1}^{gross} = E_t^{gross} r_{t+1}^{gross}$$

- The monetary transaction costs are obtained from the application of the

percentage transaction costs, δ, on the gross invested capital:

$$\Delta = E_t^{gross} \delta \mid a_{t+1} - a_t \mid$$

- In order to obtain the net monetary gain, G_{t+1}^{net}, the monetary transaction costs must be subtracted from the gross monetary gain:

$$\begin{aligned} G_{t+1}^{net} &= G_{t+1}^{gross} - \Delta \\ &= E_t^{gross} r_{t+1}^{gross} - E_t^{gross} \delta \mid a_{t+1} - a_t \mid \\ &= E_t^{gross} (r_{t+1}^{gross} - \delta \mid a_{t+1} - a_t \mid) \end{aligned}$$

- Now, it is easy to obtain the evolution of the net capital, that is the equity line:

$$\begin{aligned} E_{t+1}^{net} &= E_t^{gross} + G_{t+1}^{net} \\ &= E_t^{gross} + E_t^{gross} (r_{t+1}^{gross} - \delta \mid a_{t+1} - a_t \mid) \\ &= E_t^{gross} (1 + r_{t+1}^{gross} - \delta \mid a_{t+1} - a_t \mid) \\ &= E_t^{gross} (1 + r_{t+1}^{net}). \end{aligned}$$

Every time an action is taken to open a new position, the all amount of previous capital is invested and, when a previous open position is closed, the capital is disinvested.

3.5 Squashing function: ϕ

The squashing function plays the role of transformator of the state descriptors, in order to estimate the action-value function. The purpose of the transformation of the states consists in enhance the sensitivity of the process which select the greedy action.

The squashing function used here, for the trasformation of the states, is the *logistic function*, which has the following form:

$$\phi(x) = \frac{a}{1 + be^{-cx}} - d$$

where $a = 2$, $b = 1$, $c = 10^{15}$ and $d = -1$. The function is a S-shaped function which varies in the range $[-1, 1]$, as the daily stock returns, generally, vary in this interval and rarely assume the extreme values. Daily stock return usually approach values near zero. Consequently, I set up the *squashing function* in the range $[-1, 1]$, with zero as central value. In this way, almost all returns will be captured in an efficient way.

The sensitivity of the function can be increased by changing the value of the parameter c.

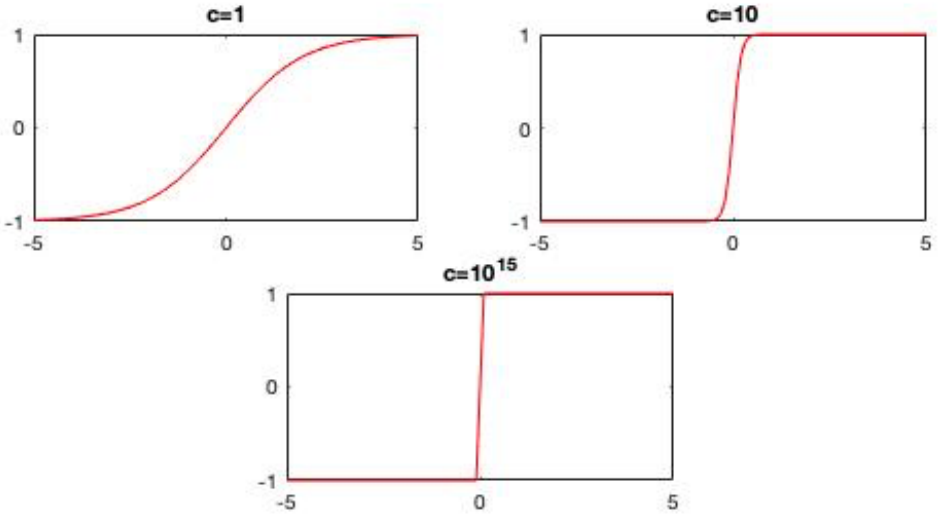


Figure 3.1: Comparison among different logistic functions with different values of c.

In the figure above are represented three different logistic functions with different values of c. As c increases, the form of the function becomes more steepest. This configuration translates in a more sensitive *FTS* as, close values will be distant after the transformation, allowing the process to differentiate between two subsequent state descriptors. The S-shaped form of the *logistic function* allows the algorithm

to perceive intermediate data with more accuracy, as they are more frequent. For this reason, intermediate values require more distinction among them, as they are closer to each other, allowing the agent to capture and learn the difference among these intermediate values. The high chosen value of c renders the algorithm more sensitive to two subsequent state descriptors.
As the states are continous, the process of transformation through the *squashing function* is comparable to a clustering process. Indeed, in this way the state values are organized and approximated.

3.6 Time series

The algorithm is applied on five daily closing stock prices time series from the FTSE MIB Index and on an artificial one. The real time series are downloaded from Yahoo Finance site. The time series used are Assicurazioni Generali S.p.A., Buzzi Unicem S.p.A., Mediobanca S.p.A., Saipem S.p.A. and Telecom Italia S.p.A.. Each time series is selected from different economic sectors, such as insurance industry, construction industry, banking industry, petroleum industry and telecommunications industry, respectively.
The artificial time series is a GARCH process, that is a random walk price series with autoregressive trend process. The model used to generate such a time series is [4]:

$$p_t = exp\left\{ \frac{z_t}{maxz_t - minz_t} \right\},$$

where $z_t = z_{t-1} + \beta_{t-1} + 3a_t$, $\beta_t = 0.9\beta_{t-1} + \beta$, with $a_t \sim \mathcal{N}(0,1)$ and $b_t \sim \mathcal{N}(0,1)$. This process shares the features characterizing real financial price series, like a non constant volatility and trends on short time frames. The period taken into account goes from November 30, 1993 to November 30, 2018, that is about 6370 days, corresponding to 25 stock market years.
The main descriptive statistics of the daily returns of each time series are summarized

in the table below:

	Mean	Median	Max	Min	St. Dev.	Skewness	Kurtosis
Assicurazioni Generali S.p.A.	1.4851e-05	0.00000	0.1231	-0.1835	0.0188	-0.06990	9.8181
Buzzi Unicem S.p.A.	2.3004e-04	0.00000	0.1404	-0.1527	0.0213	0.00240	5.6617
Mediobanca S.p.A.	6.5264e-05	-0.00031	0.1533	-0.2385	0.0218	-0.07090	7.8887
Saipem S.p.A.	1.1278e-04	0.00000	0.1700	-0.4199	0.0253	-1.54210	27.3261
Telecom Italia S.p.A.	3.9668e-06	0.00000	0.6065	-0.5992	0.0393	0.21630	144.1168
Artificial time series	4.9014e-05	0.00005	0.0129	-0.0130	0.0038	0.01310	2.9707

Table 3.1: Main descriptive statistics of daily stock logarithmic returns.

The mean of returns of all time series is around zero. The Skewness, that measures the degree of asymmetry, range from -1.54210 to 0.21630. This index should be compared to the one of a Normal distribution, in which is equal to 0. The time series with the highest Skewness is the Saipem S.p.A., because of the presence of outliers in the left tail. The Kurtosis indicates how much the data are concentrated in the center and in the tails of the distribution, rather than in the shoulders. The comparison should be made with the Kurtosis of a Normal distribution, in which it is equal to 3. In the table is observable that, apart the Artficial time series, the other time series have a high Kurtosis, meaning that the data are distributed in the tails.

A graphical representation of such distributional features is given by the figures below. Each histogram is plotted against a Normal distribution. While in the second plot is represented the QQ-plot, that is a representation of the quantiles of the distribution against the quantiles of the Normal distribution. From the graph is observable that all real time series have heavy tails, compared to a Normal distribution, as they follow a convex-concave pattern and all real time series have outliers. Heavy tails are confirmed by the positive excess Kurtosis of each real time series. The returns of the Artificial time series, on the other hand, have a distribution near the Normal one.

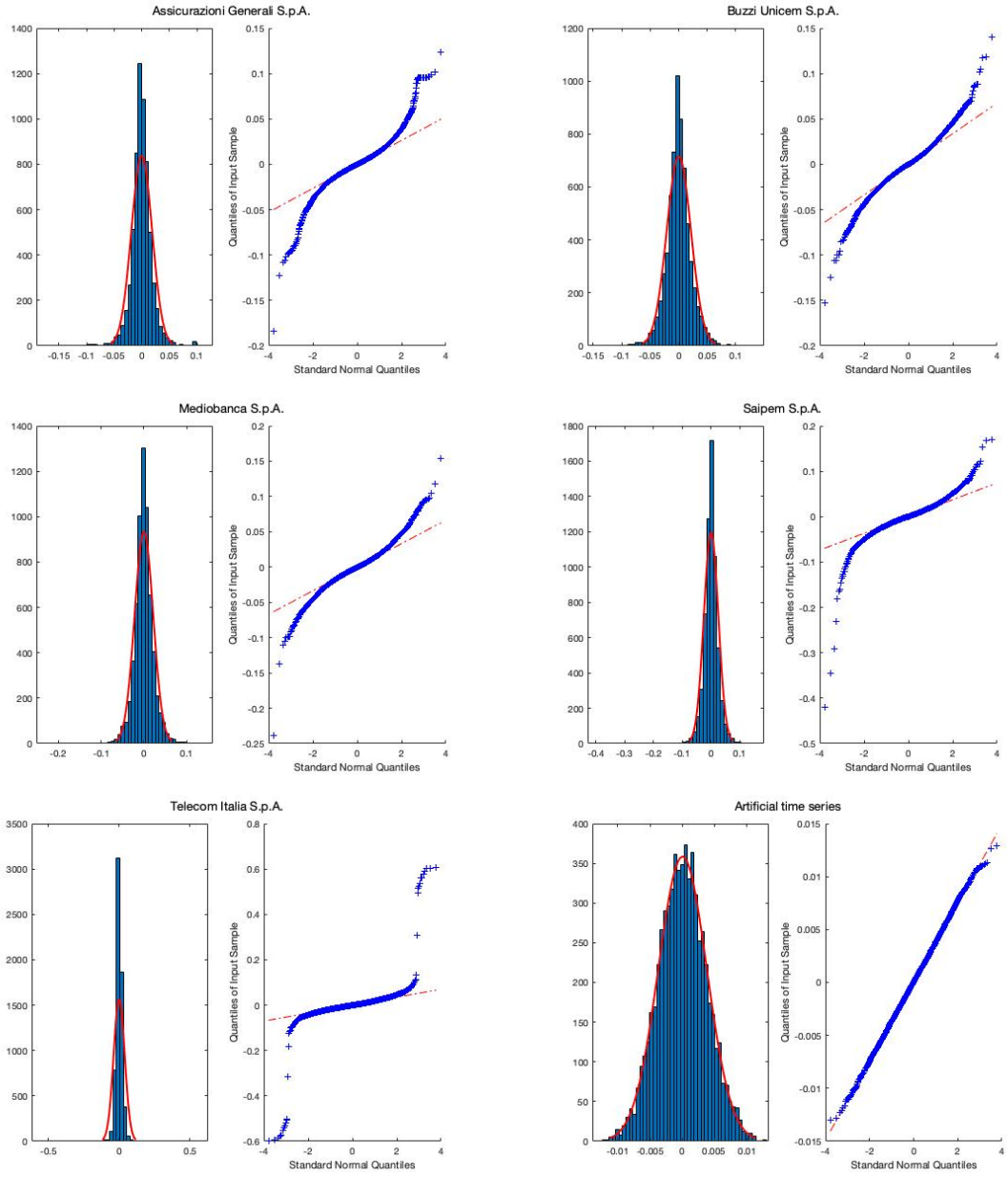
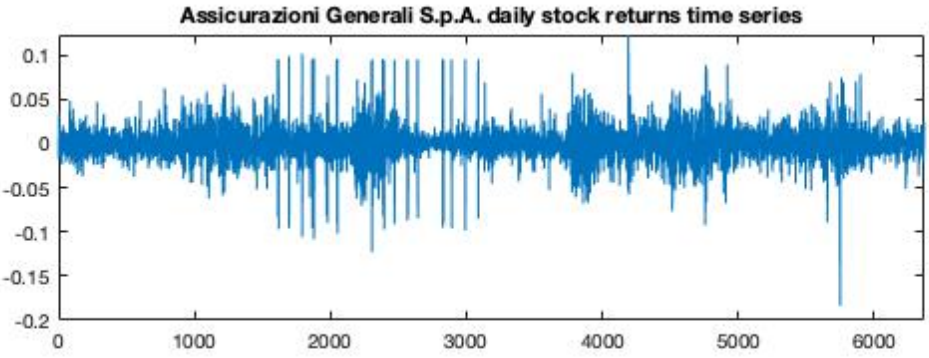


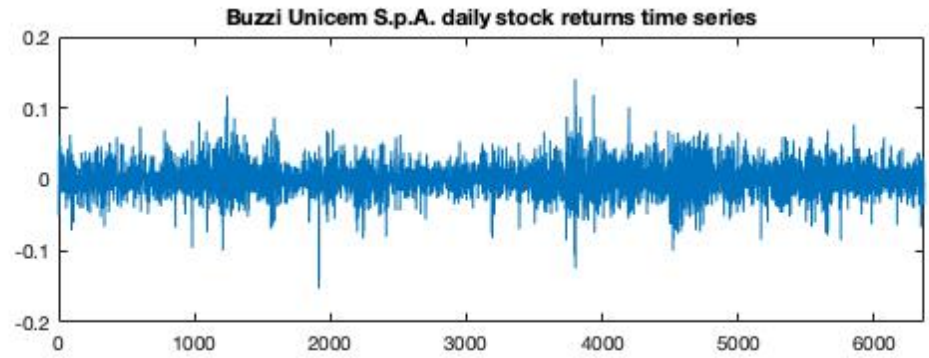
Figure 3.2: Histogram and Q-Q plot of each daily stock logarithmic returns.

Given that all the time series have different descriptive statistical values, the *FTS* will test different environments with, consequently, different performances.
In the figures below, the daily stock prices and the respective daily stock returns of each time series are represented. The returns of real time series present high variability, while in the GARCH process the volatility is more constant over time around a constant mean. Constant variability and constant mean define a stationary process so, it is feasible to expect that the Artificial time series will be easier to learn.

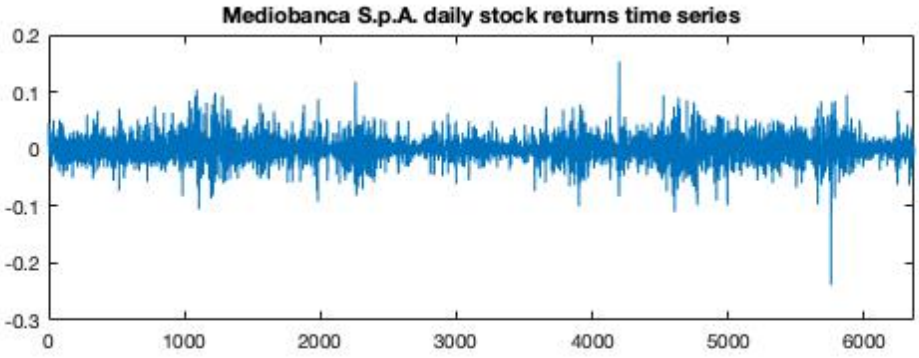

Assicurazioni Generali S.p.A. daily stock prices time series
50
40
30
20
10
0
0
1000
2000
3000
4000
5000
6000


Assicurazioni Generali S.p.A. daily stock returns time series
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
1000
2000
3000
4000
5000
6000

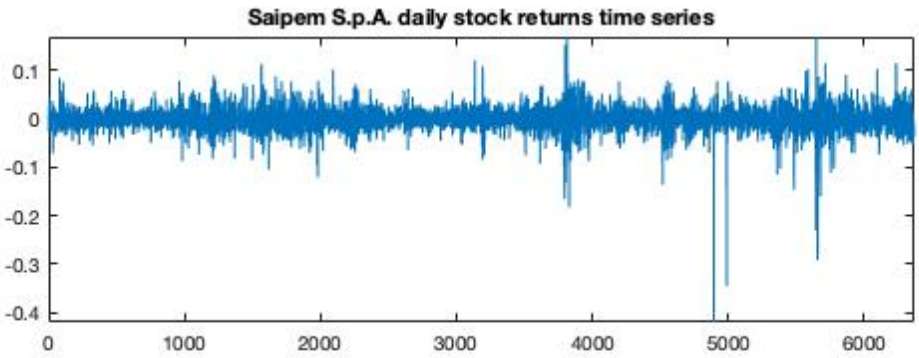

Buzzi Unicem S.p.A. daily stock prices time series
30
25
20
15
10
5
0
0
1000
2000
3000
4000
5000
6000


Buzzi Unicem S.p.A. daily stock returns time series
0.2
0.1
0
-0.1
-0.2
0
1000
2000
3000
4000
5000
6000


Mediobanca S.p.A. daily stock prices time series
20
15
10
5
0
0
1000
2000
3000
4000
5000
6000


Mediobanca S.p.A. daily stock returns time series
0.2
0.1
0
-0.1
-0.2
-0.3
0
1000
2000
3000
4000
5000
6000


Saipem S.p.A. daily stock prices time series
60
50
40
30
20
10
0
0
1000
2000
3000
4000
5000
6000


Saipem S.p.A. daily stock returns time series
0.1
0
-0.1
-0.2
-0.3
-0.4
0
1000
2000
3000
4000
5000
6000

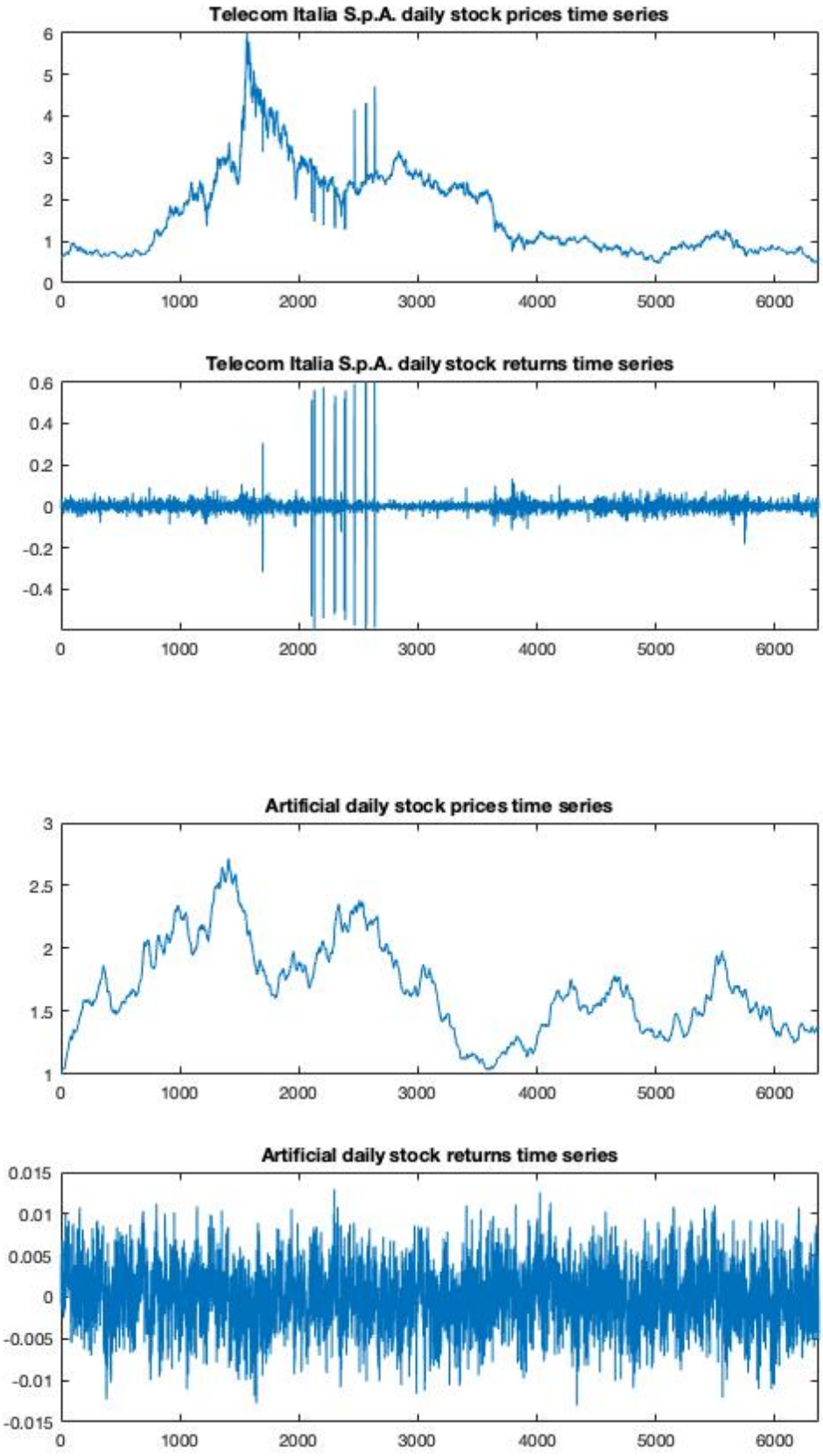


Figure 3.3: Daily prices and daily returns of all time series.

3.7 Simulations

The implementation of the *Q-Learning* algorithm, for finding the optimal policy π^*, can be summarized by the following steps [20]:

1. Set the discount factor γ and the learning rate parameter α;
2. Initialize randomly the parameter vector $\boldsymbol{\theta_0}$ and the starting state $\boldsymbol{s}_0$;
3. Repeat for each time step:
   - Estimate $Q_t(s_t, a_t, \boldsymbol{\theta}_t)$ for each available action a_t of state $\boldsymbol{s}_t$
   - Choose an action a_t from $\boldsymbol{s}_t$ following the ε-greedy policy
   - Take action a_t, observe the subsequent state $\boldsymbol{s}_{t+1}$ and the obtained reward r_{t+1}
   - Estimate and select the $\max_a Q(s_{t+1}, a_t, \boldsymbol{\theta_t})$
   - Update the parameter $\boldsymbol{\theta}_t$:
     $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \boldsymbol{\theta_t}) - Q(s_t, a_t, \boldsymbol{\theta}_t)]\nabla_{\boldsymbol{\theta}_t} Q(s_t, a_t, \boldsymbol{\theta}_t)$
   - $\boldsymbol{s_t} = \boldsymbol{s_{t+1}}$

   until the stopping criterion is satisfied.

In the tests carried out, I set $\gamma = 97.5\%$ and $\alpha = 40\%$, as are the values generally used in the prominent literature. The discount rate is smaller than 1 in order to assure that the infinite reward sum sequence has a finite value, and it approaches 1 in order to make the agent more farsighted. The learning rate α is set constant and enough high in order to allow the agent the possibility to learn a non-stationary environment.

The parameter vector $\boldsymbol{\theta}$, at the beginning of each trading period $t = 0$, is randomly initialized on the basis of a uniform probability distribution. For this reason, repeating many times the same configuration, the final results change each time. For this reason, I carried out simultaneously 500 simulations for each different configuration, in order to obtain a good approximation of the performances of the *FTS*.

To render the *FTS* realistic, transaction costs are considered. For each configuration,

the gross and the net results are presented, to highlight the impact of transaction costs.

Summarizing, for the development of the *FTS*, I use two different values of N (1 and 5), three different reward functions (*SR*, *Sortino* and *ESR*), two values of L (11 and 22) and two different values of ε ($2,5\%$ and 5%), for a total of twenty-four different configurations.

3.8 Operative signals

Each of the 500 simulations carried out, of each configuration, brings different results, due to the stochastic characteristics of the process taken into account. So, in each time step, for each time series, there are 500 possible actions to take. To render operative the algorithm, in order to obtain only one trading signal, in each time step, to perform in the real financial market, the action signal is determined on the basis of the majority of the action signals performed in each time step [19]. So, the action that the agent has to take in each time step, is an average of all the actions of all simulations performed in that time step. Formally:

$$\bar{a}_t = \frac{\sum_{k=1}^{K} a_{t,k}}{K}$$

where $\bar{a}_t$ represents the average action value at time t, $a_{t,k}$ is the action at time t of the k-th simulation of K total number of simulations. In particular, to translate into an operative action signal the average action value, $\bar{a}_t$, the following three intervals are used:

$$a_t = \begin{cases} -1 \ \textit{sell or stay-short-in-the-market-signal if } \bar{a}_t \in \Big[-1, -0.29\Big] \\ 0 \ \ \textit{stay-out-from-the-market-signal if } \bar{a}_t \in \Big(-0.29, 0.29\Big) \\ 1 \ \ \textit{buy or stay-long-in-the-market-signal if } \bar{a}_t \in \Big[0.29, 1\Big]. \end{cases}$$

Consequently, all the 500 equity lines performances are summarized into an operative equity line, that follows the trend in the mean. In this operative phase, transaction costs are applied when there is a change in stock positions, that is if a new position is taken or a previous one is closed.

Chapter 4

Application and outcomes

In this chapter I illustrate the results obtained from the application of the implemented *FTS* on several configurations. It is carried out on five real daily stock price time series and on an artificial one.
The performances are measured through four statistics, that is the gross and the net yearly logarithmic return, the percentage of times the capital invested is equal to or greater than the starting capital and the annual average number of transactions.
For each stock time series are illiustrated the results obtained by the application of several *FTS* configurations, which are stored in tables. For the best performance, of each stock price time series, two figures are provided, which illustrate the evolution and the behavior of the *FTS*.
The results, then, are clustered in a table on the basis of each single configuration parameters. This gives a general idea of the global performances of the financial trading system.

4.1 Statistics

The performances of the implemented *FTS* are measured through the average yearly logarithmic return and through the evolution of the invested equity, in order to check if the investment has bring to a positive return.
The twenty-four configurations are applied to each daily stock prices time series, that are Assicurazioni Generali S.p.A., Buzzi Unicem S.p.A., Mediobanca S.p.A., Saipem S.p.A., Telecom Italia S.p.A. and the artificial time series, configured as a GARCH process. The analysed period goes from November 30, 1993 to November 30, 2018.
I will present the results through several statistics, such as the gross and the net average yearly logarithmic return obtained during the trading period, the percentage of times in which the net equity line is greater than or equal to the starting capital, and the average number of transactions per stock market year taken by the system. This statistics are calculated on the results produced by the operative actions signals. The operative action signals, as stated in the previous chapter, consist on the average of all action signals of all 500 simulations in each time step, so:

$$\bar{a}_t = \frac{\sum_{k=1}^{500} a_{t,k}}{500}.$$

Once the average action value is computed, it is translated into an operative action signal through three defined intervals:

$$a_t = \begin{cases} -1 \ \textit{sell or stay-short-in-the-market-signal if } \bar{a}_t \in \Big[-1, -0.29\Big] \\ 0 \ \ \textit{stay-out-from-the-market-signal if } \bar{a}_t \in \Big(-0.29, 0.29\Big) \\ 1 \ \ \textit{buy or stay-long-in-the-market-signal if } \bar{a}_t \in \Big[0.29, 1\Big]. \end{cases}$$

At the end of the trading period, it is possible to define the final gross and net capital invested and, based on this results, it is possible to compute the average daily logarithmic return and then, the average yearly logarithmic return, both gross

and net:

- $Average\ daily\ return = (\frac{Final\ Capital}{Initial\ Capital})^{\frac{1}{T}} - 1$
- $Average\ yearly\ return = \bar{g} = (1 + Average\ daily\ return)^{252} - 1$

where T indicates the total trading days of the investment period, that are 6370, and 252 corresponds to the days of a trading year.
The percentage of time in which the net average capital is equal to or greater than the initial capital is computed as follows:

$$\% = \frac{\Sigma}{T} * 100$$

where Σ represents the number of times in which the equity line is equal or is above the initial invested capital. This last statistic gives a qualitative information about the capital invested as it considers the evolution of the capital during the trading period [4]. On the contrary, the average yearly return considers only the initial and the final capital. Based on this two considerations, a *FTS* that has generated a positive equity line during the trading period but, at the end it collapsed, giving a final capital lower of the initial invested capital, can be considered a satisfactory *FTS*.
The last statistic consider the average number of transactions per year and it is computed as follows:

$$\# = \frac{\Pi}{T/252}$$

where Π represents the number of performed actions during the trading period. This statistic is useful to check if the system takes too many actions, so opens or closes positions too many often, as too many transactions imply high transaction costs and consequently a lower final net capital.

4.2 Results

The results are presented in the tables below, for each time series, with all the 500 simulations for each of the twenty-four configurations. Also the final capital, both gross and net, are reported. Transaction costs are applied on the operative phase. The columns labeled $g[\%]gross$ and $g[\%]net$ show, respectively, the gross and net average yearly logarithmic return. The column labeled % shows the percentage of times in which the net equity line is higher or equal to the initial invested capital. The column labeled # shows the average number of transactions in a trading year. Two figures, of the best performance in terms of the higher yearly net logarithmic return, are shown for each stock. The first figure shows all the equity lines of all the simulations and the gross and net operative equity line. The bold black line represents the gross operative equity line, while the bold green line represents the net one. The second figure is divided into four panels. The first panel shows the daily stock prices time series; the second panel reports the operative actions signals taken during all the trading period; the third panel shows the reward function; the fourth panel represents both the gross (blue line) and the net (green line) equity capital that is obtained by investing at time t_0 a starting capital of $C = 100$ euro until the terminal state T.

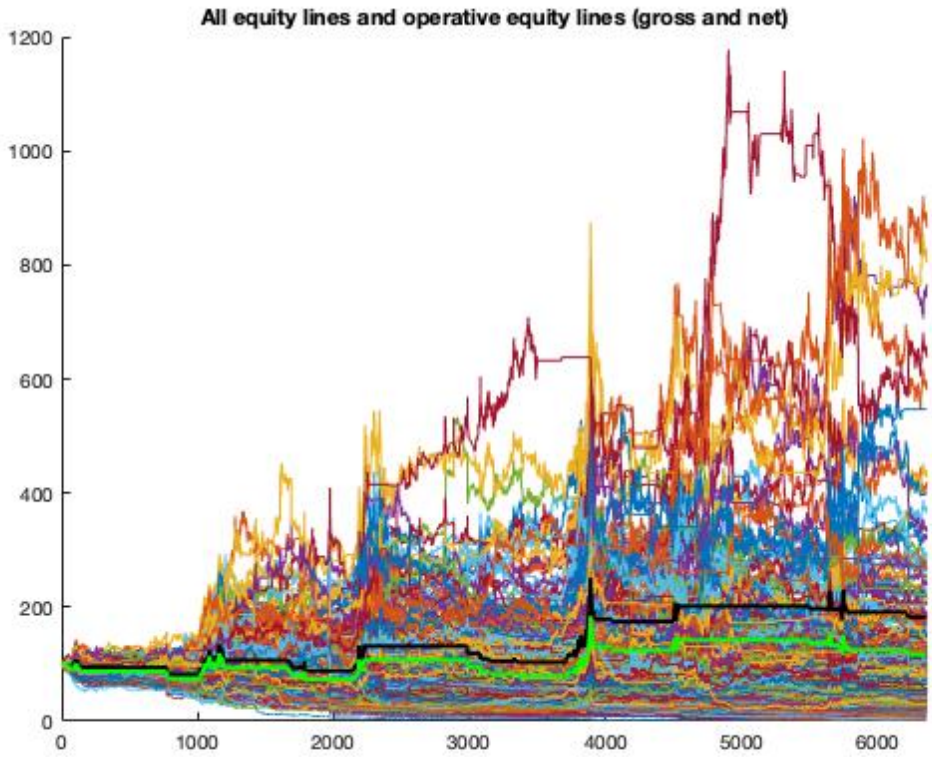
The first results reported are from the application of the *FTS* to the Assicurazioni Generali S.p.A. time series.

Ass.Generali			**N=1**					
Reward function	**L**	**ε**	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	1.40	-3.48	67.62	8.15	142.24	40.84
SR	11	5%	-3.95	-11.35	2.09	13.22	36.10	4.77
SR	22	2.5%	-1.87	-6.06	15.48	7.20	62.04	20.59
SR	22	5%	-1.11	-7.98	1.41	11.87	75.34	12.24
Sortino	11	2.5%	-1.67	-7.08	14.12	9.34	63.32	15.63
Sortino	11	5%	-0.72	-7.70	9.74	2.03	83.36	13.20
Sortino	22	2.5%	-2.49	-8.34	17.16	10.21	52.85	11.08
Sortino	22	5%	-1.65	-9.82	7.18	14.33	65.64	7.35
ESR	11	2.5%	-0.51	-3.66	-6.06	5.30	87.78	38.95
ESR	11	5%	-4.60	-11.70	1.49	12.74	30.44	4.31
ESR	22	2.5%	0.81	-3.11	53.17	6.57	122.78	44.98
ESR	22	5%	-2.72	-8.60	1.24	10.29	49.81	10.32

Ass.Generali			**N=5**					
Reward function	**L**	**ε**	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	0.37	-4.09	0.27	7.52	109.66	34.80
SR	11	5%	1.84	-2.83	0.38	7.76	158.33	48.44
SR	22	2.5%	2.06	-1.54	1.15	5.94	167.22	67.63
SR	22	5%	0.66	-3.81	0.79	7.52	118.04	37.47
Sortino	11	2.5%	1.94	-1.51	0.96	5.70	162.60	68.17
Sortino	11	5%	-1.40	-5.86	0.75	7.68	70.11	21.76
Sortino	22	2.5%	2.13	-1.23	1.62	5.54	170.35	73.10
Sortino	22	5%	-0.46	-4.47	0.77	6.81	88.96	31.52
ESR	11	2.5%	1.05	-1.39	1.12	4.04	130.33	70.30
ESR	11	5%	-1.49	-6.10	0.75	7.92	68.44	20.43
ESR	22	2.5%	2.41	0.61	54.02	2.93	182.54	116.62
ESR	22	5%	2.31	-0.78	21.29	5.07	177.88	2.11

Table 4.1: Assicurazioni Generali S.p.A. daily stock prices time series.

The best configuration, in terms of the higher net yearly logarithmic return obtained, is the one with $N = 5$, $L = 22$, $\varepsilon = 2.5\%$ and the *Expected Shortfall ratio* as reward function. The final capital, both gross (182.54) and net (116.62), is higher than the starting capital and, for 54.02% of times, the net equity line is equal or above the starting capital. Both logarithmic returns are positive: the gross is equal to 2.41%, while the net is 0.61%.



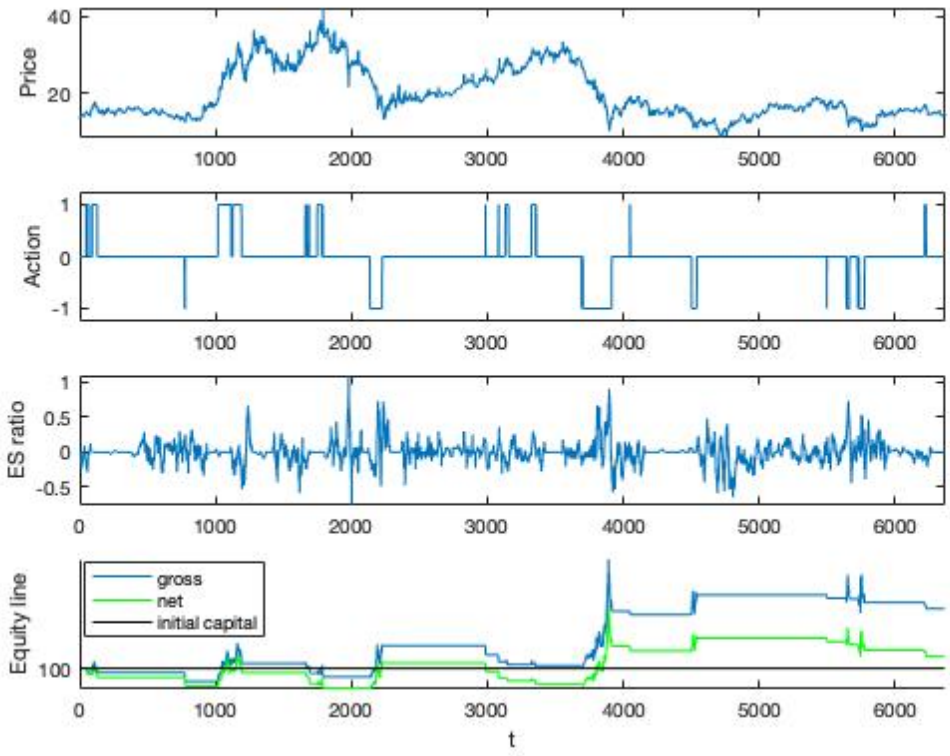


Figure 4.1: Assicurazioni Generali S.p.A.
$N = 5$, $L = 22$, $\varepsilon = 2.5\%$, *ESR*.

The graphics of this configuration are reported above. In the first figure are represented all the 500 equity lines and the gross (black line) and net (green line) operative equity lines. The equity lines of Assicurazioni Generali S.p.A. have not a main trend. For the fist 3700 observations the two operative equity lines, gross and net, have many drawdowns but with small magnitude. Then, they increase and maintain an approximately constant trend until the end of the trading period. In the second panel of the second figure the actions taken by the system are reported. The system does not take many actions. In fact, the number of transactions per year is of 2.93. It is interesting to note how the agent tries to learn the environment. In fact, it buys the stock when the price starts to increase and sells it when the price starts to decrease. The low number of transactions taken during the trading period can be attributed to the combination of two factors: a high learning rate and a low exploration rate. The high learning rate is necessary as the environment

is not stationary and the time series have several structural breaks[29]. Due to the structural breaks, the environmental features change and the *FTS* has to adapt itself to this changes. So it must unlearn what it already learned and must start to learn the new environmental characteristics on the basis of few informations. Therefore, a high learning rate brings to high updates of the action-value functions the system is trying to maximize. The exploration rate is low, so the system does not need to explore too much the environment, because of the high learning rate. The same configuration applied to the other two reward functions performs well only in terms of gross yearly returns, as the net logarithmic returns are negative. Among the three reward functions, the *Expected Shortfall ratio* is the one with the best performance, followed by *Sortino ratio* and the last, the *Sharpe ratio*. This is a predictable result, as the *Expected Shortfall* is a more precise measure of risk, compared to the other two.
There are others configurations with a positive gross logarithmic return but, due to the transaction costs, the net one is always negative. Another observation that can be pointed out, is related to the number N of returns used in the state descriptors. For each reward function, the higher performances are obtained when N is higher. So the system prefers more informations in order to learn and, consequently, takes actions.

The following table reports the results concerning the second time series, that is Buzzi Unicem S.p.A.. The best configuration, among all, is the one with $N = 5$, $L = 11$, $\varepsilon = 2.5\%$ and the *Expected Shortfall ratio* as reward function. Both final capitals are above the initial capital, 143.73 the gross one and 124.29 the net one. The gross and net yearly logarithmic return are 1.45% and 0.86%, respectively. For 95.80% of times, the net equity line is greater or equal to the starting capital.

[29] A structural break is an unaxpected change in a time series trend, typical of a non-stationary environment.

Buzzi Unicem			N=1					
Reward function	**L**	**ε**	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	-4.98	-10.81	7.01	10.45	27.49	5.55
SR	11	5%	-7.24	-16.12	7.65	5.28	14.95	1.18
SR	22	2.5%	-5.10	-10.74	9.16	10.13	26.66	5.66
SR	22	5%	-4.44	-13.57	8.68	16.58	31.70	2.51
Sortino	11	2.5%	-6.16	-13.16	6.74	12.78	20.05	2.83
Sortino	11	5%	-5.53	-13.49	8.31	14.52	23.75	2.57
Sortino	22	2.5%	-7.43	-13.70	4.18	11.59	14.22	2.41
Sortino	22	5%	-7.05	-15.90	7.35	16.50	15.76	1.26
ESR	11	2.5%	-5.71	-10.75	19.33	9.06	22.62	5.65
ESR	11	5%	-7.07	-14.79	4.60	14.29	15.67	1.75
ESR	22	2.5%	-7.42	-12.94	4.16	10.13	14.24	3.01
ESR	22	5%	-5.14	-12.55	3.74	13.42	26.34	3.37
Buzzi Unicem			**N=5**					
Reward function	**L**	**ε**	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	1.40	-0.28	13.67	2.77	142.00	93.06
SR	11	5%	1.18	-1.07	5.28	3.72	134.48	76.28
SR	22	2.5%	0.88	-1.45	12.29	3.88	124.84	69.13
SR	22	5%	0.72	-1.81	4.29	4.20	119.73	63.13
Sortino	11	2.5%	0.81	-0.68	10.29	2.46	122.50	84.27
Sortino	11	5%	0.69	-0.84	10.50	2.53	118.82	80.78
Sortino	22	2.5%	1.41	-0.36	52.22	2.93	142.48	91.24
Sortino	22	5%	0.17	-1.49	16.17	2.77	104.47	68.50
ESR	11	2.5%	1.45	0.86	95.80	0.95	143.73	124.29
ESR	11	5%	0.52	-1.76	9.63	3.80	114.09	63.02
ESR	22	2.5%	1.54	-0.34	46.24	3.09	146.92	91.86
ESR	22	5%	0.28	-1.28	10.09	2.61	107.44	72.19

Table 4.2: Buzzi Unicem S.p.A. daily stock prices time series.

A graphical representation of this configuration applied to the time series of Buzzi Unicem S.p.A. is reported below. The operative equity lines are almost straight, confirmed by both figures. There is a quite constant trend until observation 3000, then the equity lines increase in correspondence of a relative peak in the price time series, maintaining the trend for about three years. When the price reached another relative peak, in observation 3800, the equity lines increase again and then keep a constant trend until the end of the trading time. In fact, the system takes a small number of actions, with an average number of transactions in a trading year of 0.95. The reason can be attributed to the multiples structural breaks characterizing this time series and, as in the previous time series, to the combination of a high learning rate with a low explorative rate. Observing the table, the gross yearly logarithmic returns are all positive for $N = 5$ and all negative for $N = 1$. The net logarithmic return pass from -16.12% in the worst case, for $N = 1$, to -1.81% in the worst case, for $N = 5$. Even for this time series, the algorithm performs better with a state descriptor with more informations.

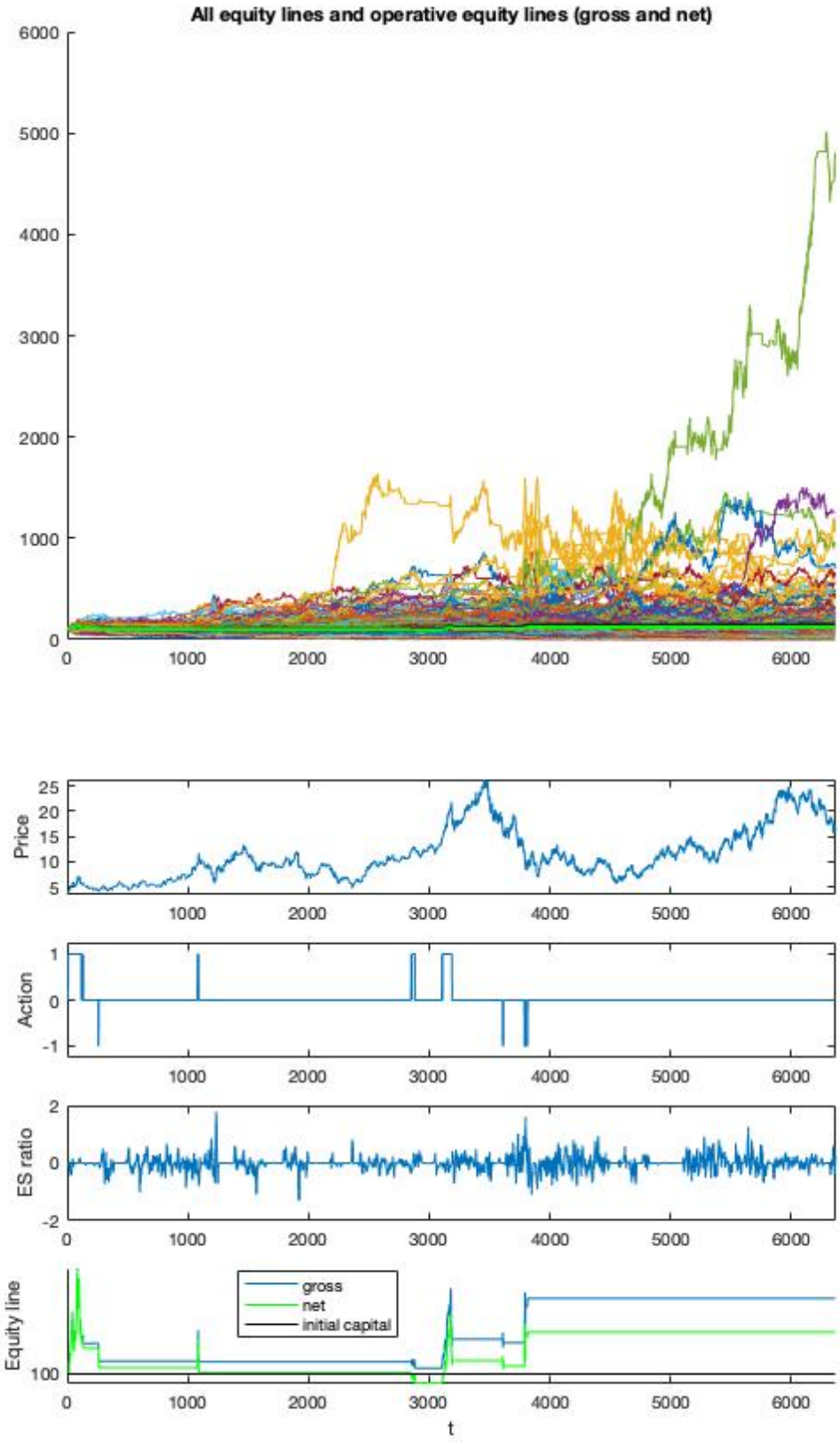


Figure 4.2: Buzzi Unicem S.p.A.
$N = 5$, $L = 11$, $\varepsilon = 2.5\%$, *ESR*.

The outputs of the time series Mediobanca S.p.A. are not optimal. In fact, all the net yearly logarithmic returns are negative. The best peformance, in terms of the lower negative net yearly logarithmic return, is given by the configuration with $N = 5$, $L = 22$, $\varepsilon = 5\%$, ESR as reward function. The gross yearly logarithmic return is 1.67% while the net one is -0.20%. The gross final capital is 151.98 and the net one is 94.96. This time series is hard to be learned as it presents many structural breaks that change the environmental characteristics. So the FTS has to unlearn what it has already learned and start the process again.

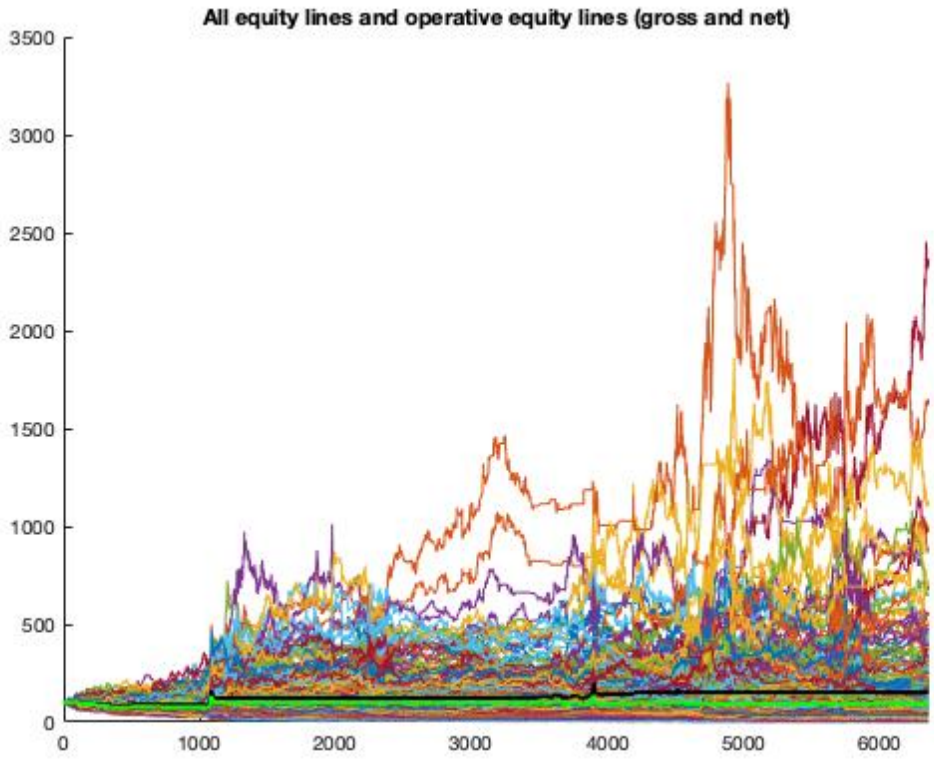
Mediobanca			**N=1**					
Reward function	**L**	**ε**	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	-5.52	-11.15	2.40	10.13	23.84	5.04
SR	11	5%	-2.50	-9.40	5.29	12.11	52.77	8.24
SR	22	2.5%	-3.81	-9.45	4.49	9.97	37.48	8.13
SR	22	5%	-4.91	-12.44	5.14	13.61	28.00	3.48
Sortino	11	2.5%	1.51	-4.36	9.59	9.81	146.12	32.43
Sortino	11	5%	-0.14	-8.44	3.94	14.33	96.62	10.77
Sortino	22	2.5%	-2.19	-8.19	2.51	10.45	57.19	11.55
Sortino	22	5%	-5.60	-13.80	1.08	15.04	23.32	2.33
ESR	11	2.5%	-1.29	-6.05	3.64	8.15	72	20.68
ESR	11	5%	-2.66	-10.63	2.40	14.09	50.63	5.85
ESR	22	2.5%	-2.03	-6.53	60.14	7.76	59.60	18.16
ESR	22	5%	-1.68	-9.49	3.91	13.62	8.04	65.23
Mediobanca			**N=5**					
Reward function	**L**	**ε**	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	0.52	-1.20	8.83	2.85	113.96	73.75
SR	11	5%	-0.34	-2.46	0.79	3.56	91.75	53.25
SR	22	2.5%	1.20	-0.39	4.48	2.61	135.07	90.69
SR	22	5%	-0.64	-2.99	1.13	3.96	84.97	46.43
Sortino	11	2.5%	-0.80	-2.68	1.67	3.17	81.58	50.30
Sortino	11	5%	-1.80	-3.98	1.41	3.72	63.26	35.88
Sortino	22	2.5%	-1.17	-2.67	1.18	2.53	74.35	50.53
Sortino	22	5%	-2.37	-4.90	1.15	4.36	54.58	28.08
ESR	11	2.5%	-1.29	-2.88	1.12	2.69	47.82	72.01
ESR	11	5%	-1.05	-3.44	0.66	4.04	41.31	76.55
ESR	22	2.5%	0.69	-1.02	47.56	2.85	119.09	77.15
ESR	22	5%	1.67	-0.20	3.69	3.09	151.98	94.96

Table 4.3: Mediobanca S.p.A. daily stock prices time series.

Observing the panel reporting the operative actions taken by the system, it is clear how the system acted. After it took a long position, when the price started to decrese, the system took many short and neutral positions, as it was hesitant on the position to be taken. This burned the initial capital from the beginning. Then the system took long positions only when the price started to increase and reached

the major peaks. The subsequent short positions, when the price began to decrease, brought the gross equity line above the starting capital, but not the net one, due to the transaction costs. In fact, the net equity line is above the starting capital for 3.69% of trading time. However, for most of the time, the *FTS* took neutral positions because of many drawdowns of the price that made the system to forget what it already learned. For this reason the system preferred to stay out from the market for most of the time. In fact, the annual average number of transactions is 3.09.

Also for this time series, the best configuration is given by the one that uses the *Expected Shortfall ratio* as reward function. Moreover, the configurations with $N = 5$ gives better results than the configurations with $N = 1$.



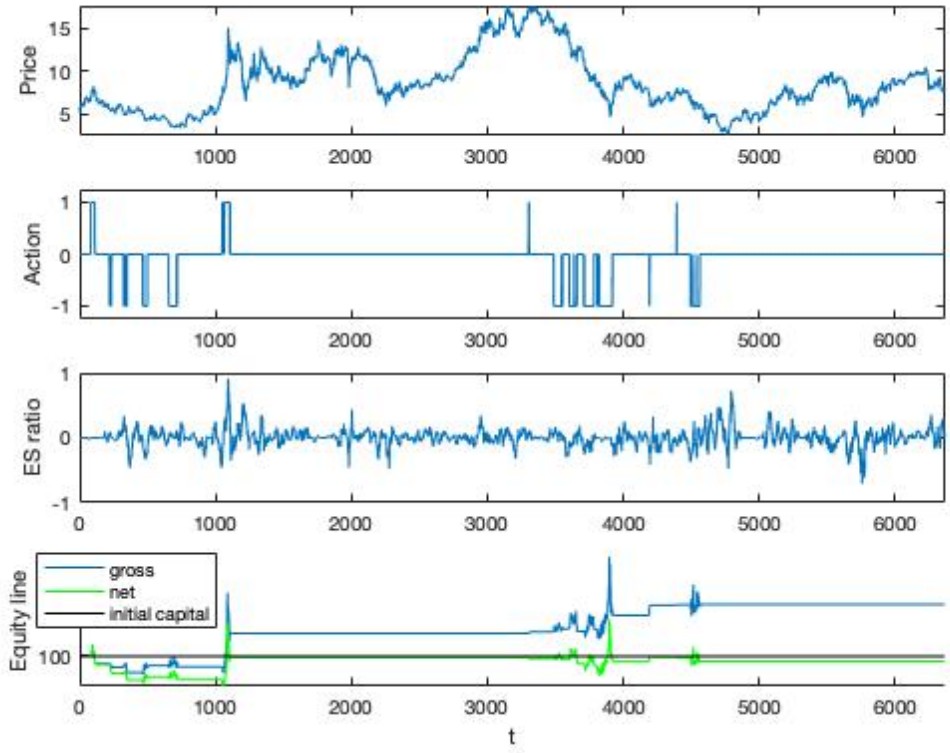


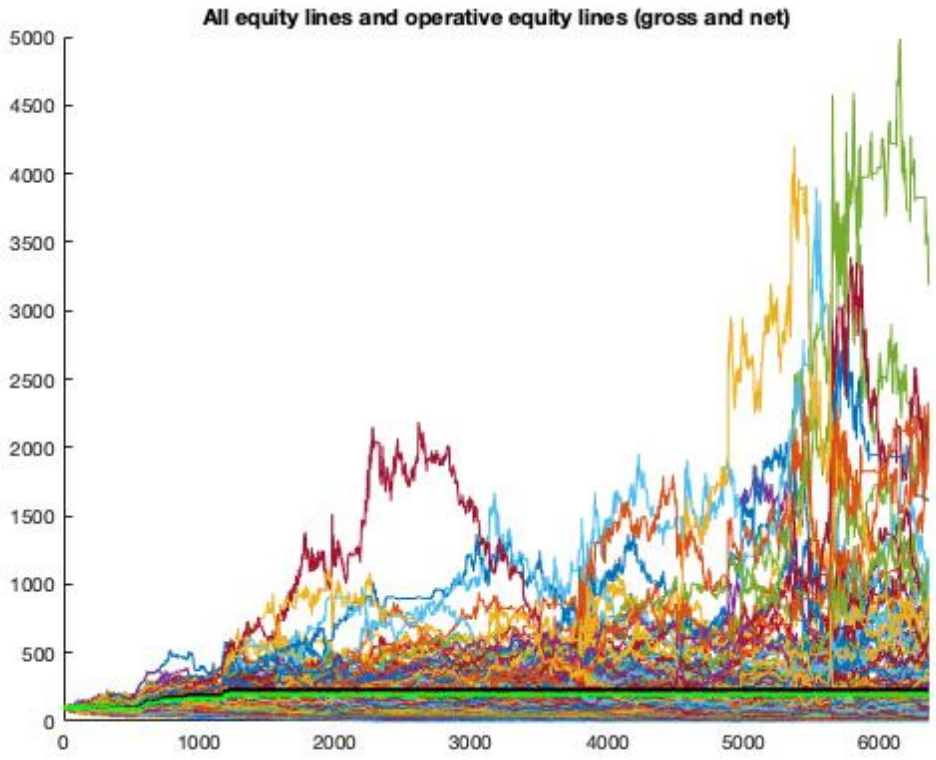
Figure 4.3: Mediobanca S.p.A.
$N = 5$, $L = 22$, $\varepsilon = 5\%$, *ESR*.

The best performance, for the Saipem S.p.A. time series, is given by the configuration with $N = 5$, $L = 11$, $\varepsilon = 5\%$ and *Sortino ratio* as reward function. This configuration gives a gross yearly return of 3.33% and a net one of 2.69%. The gross and net final capital is of 228.78 and 195.50, respectively. This time series performed quite well in all configurations, at least for what concerns the gross yearly logarithmic return. The net one is always positive for configurations with $N = 5$.

Saipem			N=1					
Reward function	**L**	**ε**	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	3.99	-0.56	42.16	7.36	268.53	86.87
SR	11	5%	4.18	-3.62	13.27	12.82	281.12	39.39
SR	22	2.5%	2.99	-3.19	15.14	10.21	270.71	44.10
SR	22	5%	3.46	-3.92	8.87	12.19	236.07	36.38
Sortino	11	2.5%	7.79	0.88	48.54	10.92	611.03	114.81
Sortino	11	5%	4.74	-4.82	14.98	15.75	321.99	28.67
Sortino	22	2.5%	5.75	-0.41	44.63	9.89	411.01	90.07
Sortino	22	5%	7.29	-2.49	4.02	15.75	591.38	52.90
ESR	11	2.5%	-1.19	-7.87	0.46	11.56	73.90	2.59
ESR	11	5%	0.28	-8.08	0.39	14.33	107.28	11.90
ESR	22	2.5%	1.25	-5.24	3.82	10.92	136.73	25.65
ESR	22	5%	5.58	-2.97	5.46	13.93	394.46	46.62

Saipem			N=5					
Reward function	**L**	**ε**	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	2.09	1.27	90.92	1.35	168.73	137.50
SR	11	5%	1.63	0.80	86.50	1.35	150.24	122.23
SR	22	2.5%	1.81	1.18	91.72	1.09	157.48	134.58
SR	22	5%	1.27	0.11	86.57	1.90	137.49	102.84
Sortino	11	2.5%	1.91	1.19	96.02	1.19	161.39	134.67
Sortino	11	5%	3.33	2.69	96.17	1.03	228.78	195.50
Sortino	22	2.5%	2.52	2.03	99.95	0.79	187.31	165.99
Sortino	22	5%	1.88	1.15	95.82	1.19	159.90	133.36
ESR	11	2.5%	0.93	0.21	87.37	1.19	126.43	105.41
ESR	11	5%	1.54	0.57	88.83	1.58	147.26	115.56
ESR	22	2.5%	1.64	0.63	94.75	1.66	150.99	17.26
ESR	22	5%	1.37	0.40	93.04	1.58	140.87	110.52

Table 4.4: Saipem S.p.A. daily stock prices time series.



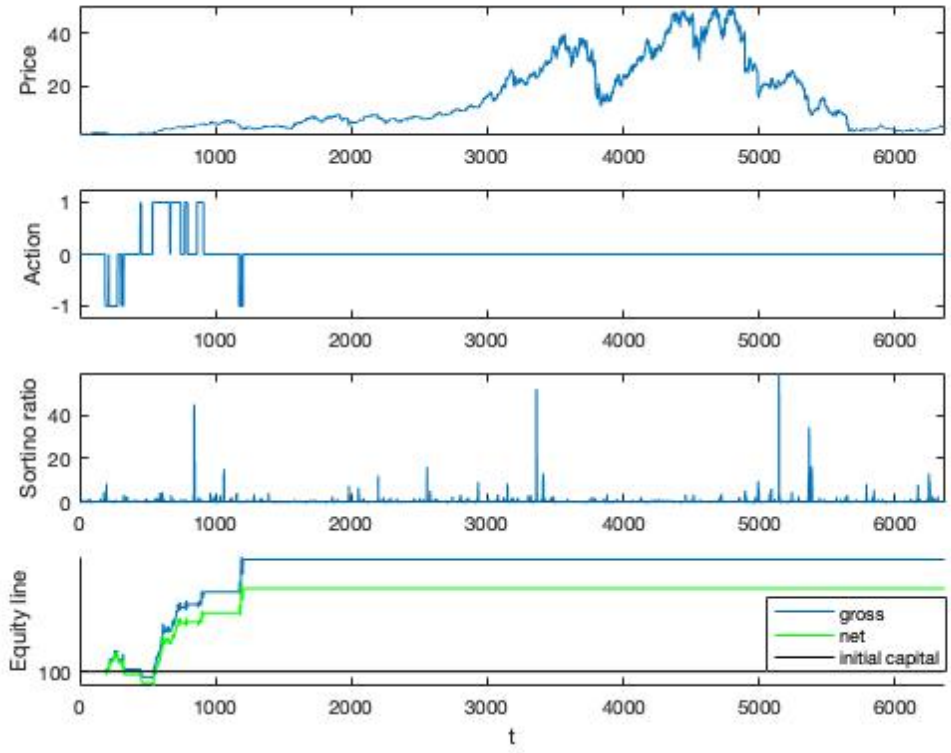
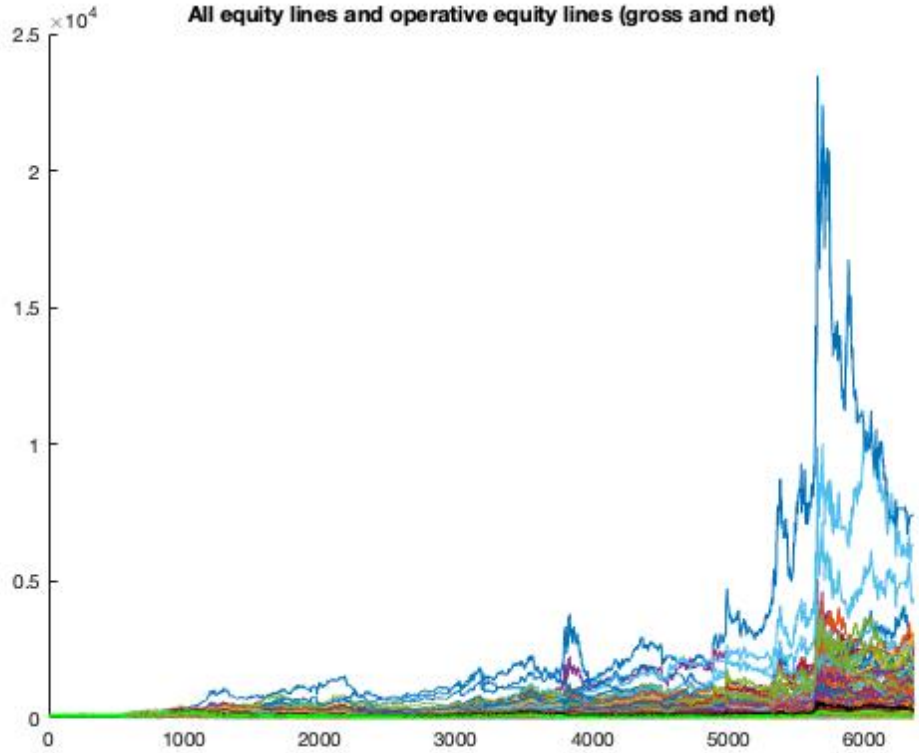


Figure 4.4: Saipem S.p.A.
$N = 5$, $L = 11$, $\varepsilon = 5\%$, *Sortino*.

In the two figures above the behavior of the time series under the best configuration is illustrated. The two operative equity lines are quite straight. Indeed, after about observation 1100, the agent exits the market. The average annual number of transactions is of 1.03 and are all performed before. The explanation of this behavior can be attributed, in part to the structural breaks the time series present and the consequently difficulty to learn a changing environment; in part to the relatively high learning rate. In fact, until observation 1100, the price trend is quite constant. When the price reached a local mimima in observation 1100, the system sold the asset and decided to exit the market. However, due to this choice, the system missed other peaks that the price time series reched in the subsequent observations. The choice to sell just in that moment is justified by the fact that, the price trend was quite constant in the previous observations and then started to decrease. Once a local minima was reached the price remained constant at the minimum level and the system decided to exit. Due to the sell, the equity lines

returned above the initial capital and, until the end of the trading period, keep constant. Indeed, the percentage of time in which the net equity line is equal or above the starting capital is of 96.17%.

To give an idea of the effects the learning rate has on the *FTS* behavior, below are reported the figures of the same optimal configuration, that is $N = 5$, $L = 11$, $\varepsilon = 5\%$ and *Sortino ratio*, but with a lower learning rate, $\alpha = 10\%$. The system, now, takes more actions than in the previous case, trying to learn the environment, with an annual average of 7.44. The outputs of this configuration are worse compared with the previous configuration. The gross yearly logarithmic return is still positive, 4.42%, but the net one is negative, -0.19%. The operative equity lines do not follow a main trend but start to increase on the last years of the trading period, when the system take short positions, as the price starts to decrease. The final capitals are 298.06, the gross one, and 95.30, the net one.



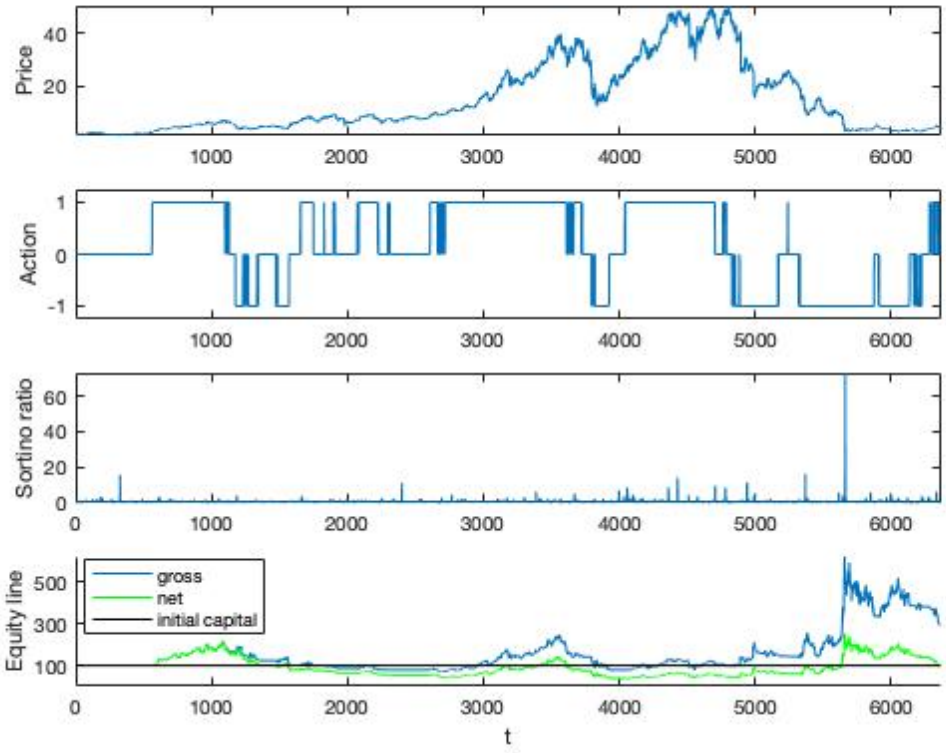


Figure 4.5: Saipem S.p.A.
$N = 5$, $L = 11$, $\varepsilon = 5\%$, *Sortino*, $\alpha = 10\%$.

The table below reports the results of the Telecom Italia S.p.A. asset. The configuration that brings the best outputs have $N = 5$, $L = 11$, $\varepsilon = 2.5\%$ and *ESR*. The $g[\%]gross = 4.69$ and the $g[\%]net = 3.40$. The final capitals are 318.09, the gross one, and 232.39, the net one. For most of the time, 99.78%, the net equity line is above or equal to the starting capital. This is shown in the two figures below. The system took only 2.06 actions per year. In fact, for most of the trading period it stayed out from the market. In particular, it bought the asset when the price started to increase and reached its main peaks. Then the system sold the asset when the price started to decrease and, as the trend of the time series after observation 4000 is quite constant, the *FTS* preferred to stay neutral. Also in this case, the choice to exit the market when the trend remains constant, with no variability, can be traced back to the high learning rate.

Observing the table below, the difference among the values with $N = 1$ and that with $N = 5$ is substantial. The net yearly logarithmic returns pass form neg-

ative to positive and also the final capital, both gross and net, increases considerably.

Telecom			**N=1**					
Reward function	**L**	ε	**g[%] gross**	**g[%] net**	**%**	#	**F.C gross**	**F.C net**
SR	11	2.5%	-3.97	-9.06	18.28	8.98	35.96	9.08
SR	11	5%	-9.81	-16.33	21.25	12.35	7.36	1.1
SR	22	2.5%	-7.27	-11.75	22.14	8.11	14.83	4.24
SR	22	5%	-9.33	-15.46	22.75	11.52	8.43	1.43
Sortino	11	2.5%	-13.23	-18.52	21.88	10.33	2.77	0.56
Sortino	11	5%	-10.75	-17.34	20.21	12.54	5.64	0.81
Sortino	22	2.5%	-10.34	-15.88	34.85	10.49	6.34	1.27
Sortino	22	5%	-8.39	-15.03	27.70	2.31	10.93	1.63
ESR	11	2.5%	0.10	-3.76	70.24	6.49	102.50	37.94
ESR	11	5%	-5.81	-12.06	10.58	11.24	22.03	3.89
ESR	22	2.5%	-1.94	-5.08	77.17	5.34	60.95	26.78
ESR	22	5%	-9.52	-15.51	14.93	11.16	7.98	1.41
Telecom			**N=5**					
Reward function	**L**	ε	**g[%] gross**	**g[%] net**	**%**	#	**F.C gross**	**F.C net**
SR	11	2.5%	4.90	2.87	95.79	3.25	335.08	204.21
SR	11	5%	3.73	1.42	95.07	3.72	252.16	142.93
SR	22	2.5%	4.70	2.77	99.83	3.09	319.31	199.21
SR	22	5%	3.94	2.46	99.83	2.36	265.52	184.92
Sortino	11	2.5%	3.51	1.51	96.15	3.25	239.03	145.90
Sortino	11	5%	3.69	1.39	95.34	3.72	249.99	141.73
Sortino	22	2.5%	3.49	2.35	95.25	1.82	237.75	179.94
Sortino	22	5%	3.03	1.13	97.64	3.09	212.33	132.75
ESR	11	2.5%	4.69	3.40	99.78	2.06	318.09	232.39
ESR	11	5%	1.69	-0.86	37.96	4.20	152.67	80.46
ESR	22	2.5%	3.54	2.51	99.69	1.66	240.95	186.99
ESR	22	5%	3.45	1.30	99.80	3.48	235.34	138.54

Table 4.5: Telecom Italia S.p.A. daily stock prices time series.

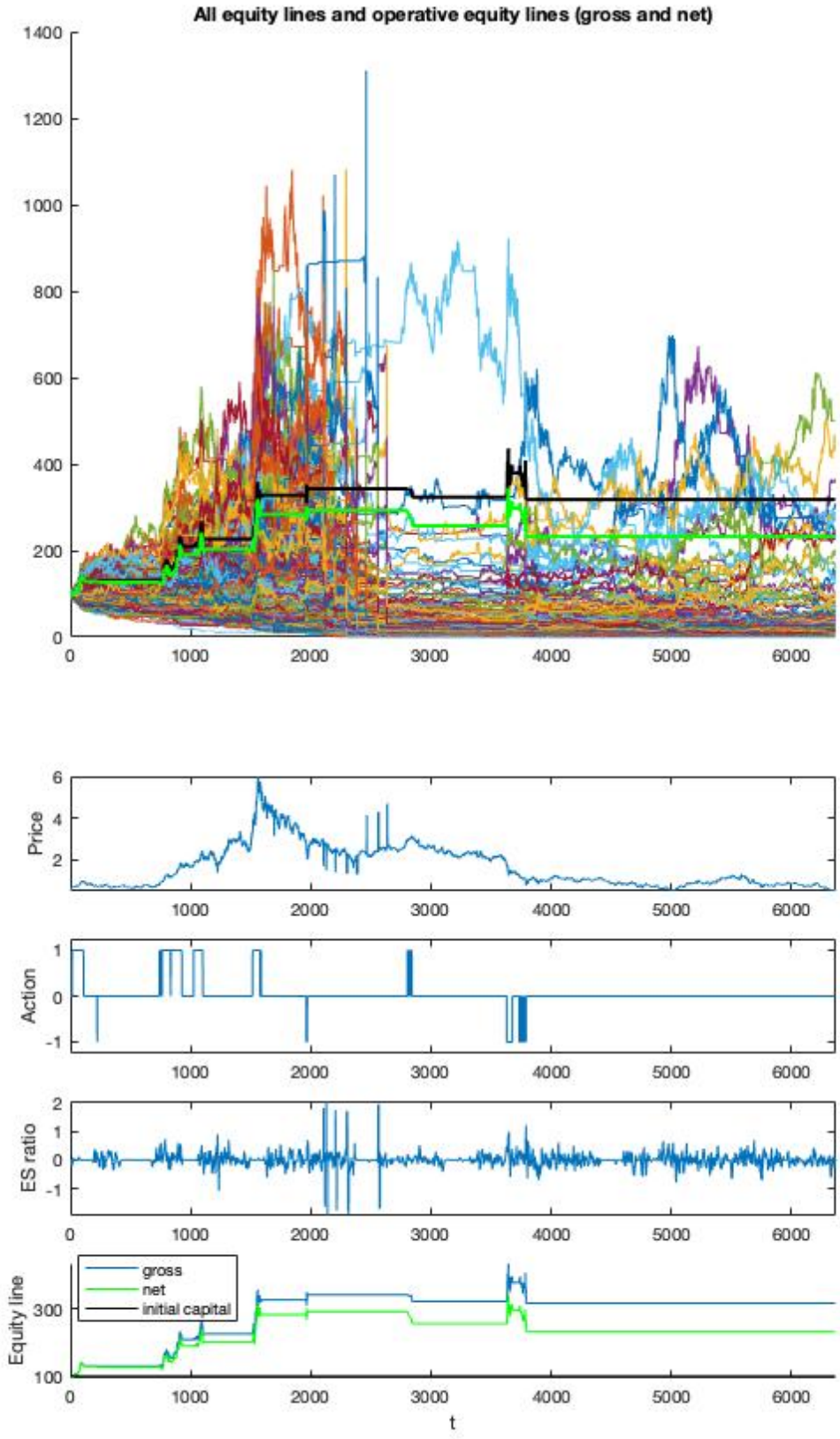


Figure 4.6: Telecom Italia S.p.A.
$N = 5$, $L = 11$, $\varepsilon = 2.5\%$, *ESR*.

The Artificial time series is the series with the best performances. All returns are positive, both gross and net. The best configuration is the one with $N = 5$, $L = 11$, $\varepsilon = 5\%$ and *ES ratio* as reward function. The gross yearly logarithmic return is equal to 14.31% and the net one is equal to 8.52%. The operative equity lines have a constant increasing trend. In fact, the net equity line is above the starting capital for 99.98% of trading time, with an annual average number of transactions of 8.63. Such a constant positive trend is due to the fact that the environment in which the *FTS* operates is known, as this time series is generated by a GARCH process. Moreover, this time series has the lowest standard deviation, 0.038, meaning that is easier to learn such an environment as it does not have high variability. The final capitals are high compared to the ones of the real time series. The gross one is 2931.61, while the net one is 788.03.

Artificial series			**N=1**					
Reward function	**L**	ε	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	13.17	6.98	99.97	9.34	2278.46	549.66
SR	11	5%	15.16	8.08	99.97	10.53	3539.25	712.58
SR	22	2.5%	11.15	6.22	99.97	7.52	1445.23	460.04
SR	22	5%	12.12	6.13	99.97	9.10	1798.94	450.10
Sortino	11	2.5%	10.25	3.82	99.97	9.97	1176.30	257.81
Sortino	11	5%	11.66	3.66	99.98	12.35	1623.10	247.99
Sortino	22	2.5%	9.48	3.73	99.97	8.94	985.47	252.53
Sortino	22	5%	9.59	0.92	93.44	13.62	1011.17	125.89
ESR	11	2.5%	11.89	6.73	99.97	7.84	1707.91	518.14
ESR	11	5%	14.97	7.70	99.98	10.84	3397.93	652.03
ESR	22	2.5%	9.75	4.59	99.97	7.99	1049.04	310.90
ESR	22	5%	11.48	5.03	99.97	9.89	1557.16	345.39
Artificial series			**N=5**					
Reward function	**L**	ε	**g[%] gross**	**g[%] net**	**%**	**#**	**F.C gross**	**F.C net**
SR	11	2.5%	12.73	8.15	99.98	6.89	2061.87	722.72
SR	11	5%	14.47	7.89	99.98	9.82	3034.56	680.93
SR	22	2.5%	10.25	6.07	99.98	6.41	1175.89	443.20
SR	22	5%	12.08	6.76	99.95	8.08	1782.97	521.94
Sortino	11	2.5%	8.96	4.43	99.75	7.05	873.41	299.05
Sortino	11	5%	9.47	3.13	99.87	9.90	981.42	217.85
Sortino	22	2.5%	8.08	3.84	99.80	6.65	711.98	258.95
Sortino	22	5%	7.55	1.33	99.73	9.90	629.20	139.72
ESR	11	2.5%	12.24	7.52	99.98	7.13	1845.86	624.00
ESR	11	5%	14.31	8.52	99.98	8.63	2931.61	788.03
ESR	22	2.5%	9.38	4.53	99.98	7.52	961.00	306.44
ESR	22	5%	11.74	6.03	99.98	8.71	1648.82	438.30

Table 4.6: Artificial daily stock prices time series.

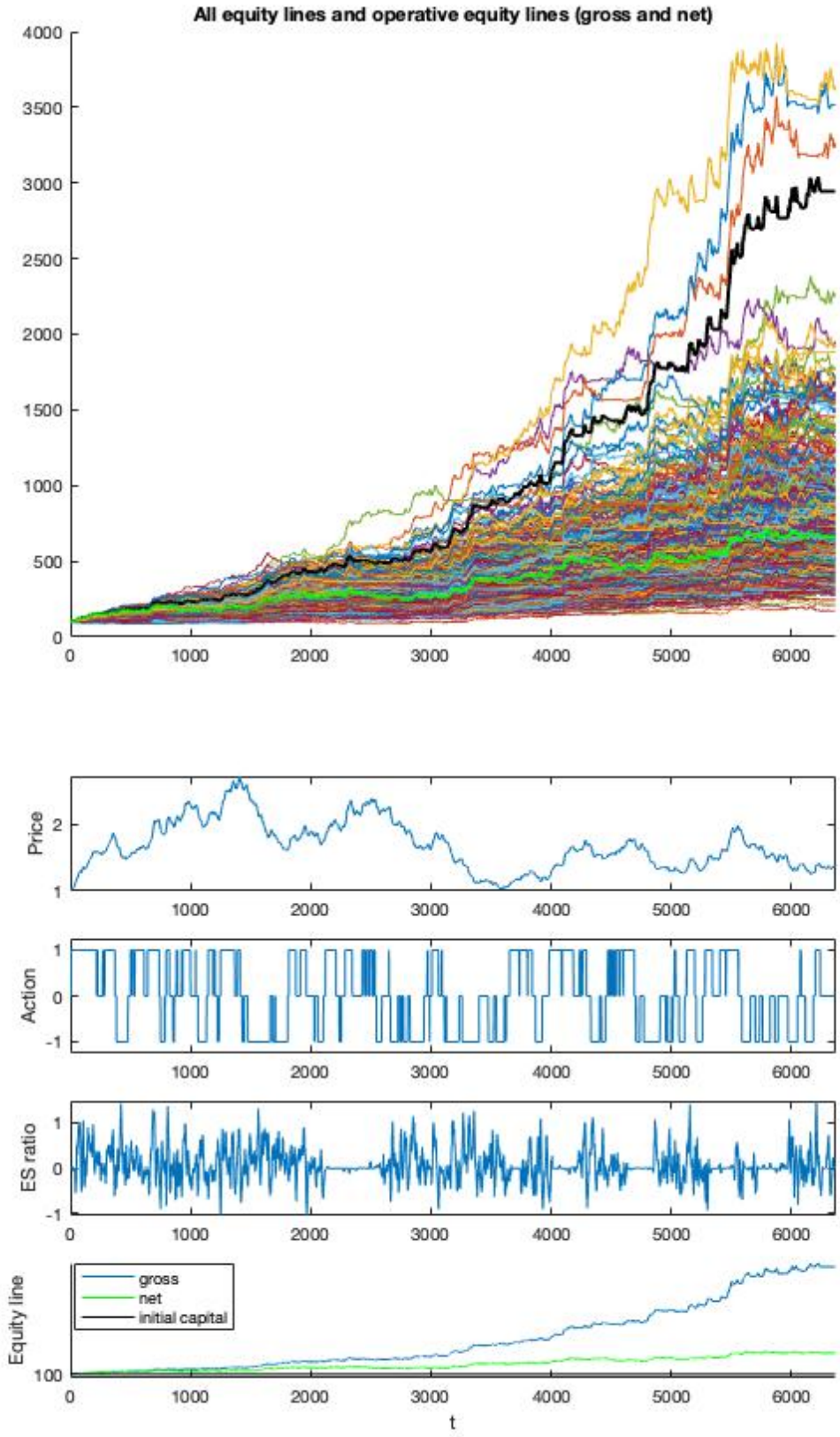


Figure 4.7: Artificial time series
$N = 5$, $L = 11$, $\varepsilon = 5\%$, *ESR*.

4.3 Remarks

In order to globally evaluate the performances of the system, the results are agglomerated on the basis of each parameter configuration. In the table below are showed the percentage of configurations with a positive net average yearly logarithmic return and the percentage of configurations in which the net equity line is equal or above the starting capital for at least half of the trading period. A distinction is made on the basis of the application of such statistics over all the stock prices time series or over only the real ones.

	All series		**Real series**	
	g[%]net>0	**%>50**	**g[%]net>0**	**%>50**
All configurations	34.72%	38.19%	21.67%	25.83%
N=1	18.06%	23.61%	1.67%	8.33%
N=5	51.39%	52.78%	41.67%	43.34%
L=11	34.72%	36.11%	21.67%	23.33%
L=22	34.72%	40.28%	21.67%	28.33%
$\varepsilon = 2.5\%$	37.50%	44.44%	25.00%	33.34%
$\varepsilon = 5\%$	31.94%	31.94%	18.34%	18.33%
SR	33.34%	35.42%	22.50%	22.50%
Sortino	35.42%	35.42%	22.50%	22.50%
ESR	35.42%	43.75%	32.50%	32.50%

Table 4.7: Summary statistics.

The difference between the results obtained from all series and those from only the real series is, on average, of 10% points. Only 21.67% of configurations, applied only on real time series, have a positive average yearly logarithmic return. The percentage is higher, 34.72%, if also the artificial time series is considered. An interesting result is the difference among the statistics obtained from the configurations with $N = 1$ and $N = 5$. For all time series, the percentage of configurations with a positive net average yearly logarithmic return passes from 18.06% to 51.39%, and for only the real time series the difference is even more evident. Indeed, this percentage moves from 1.67% to 41.67%. In fact, in correspondence of $N = 5$, the percentage of configurations in which the invested capital is equal or over the starting capital, is higher. It is of 52.78% for all series and 43.34% for only real series.

For what concerns the number L of trading days on which the reward functions are computed, there is no difference between the two values. Another observation can be done observing the explorative rate ε. According to the observations made in the previous part, a low explorative rate is combined with a high learning rate. In fact, the configurations with $\varepsilon = 2.5\%$ give better performances than those with $\varepsilon = 5\%$. For what concerns the reward functions, the real series perform better utilizing the *Expected Shortfall ratio*. Indeed, 32.50% of configurations obtain a positive net yearly logarithmic return with the *ESR*, against a 22.50% for the other two reward functions. Four of the five real time series' best configurations use the *Expected Shortfall ratio* as reward function. This mean that, this reward function which takes into account the downside risk of an investment, is preferred. Making an average of the net yearly logarithmic returns of each type of reward function, emerges that the *Expected Shorfall ratio* gives higher results for Assicurazioni Generali S.p.A., Buzzi S.p.A., Mediobanca S.p.A. and Telecom Italia S.p.A. Only Saipem S.p.A. performes better, in the mean, utilizing the *Sortino ratio* and the Artificial time series gives better results utilizing the *Sharpe ratio*.

Given the great difference in performances when the number N of returns, utilized in the state descriptors, increases, I illustrate this variations. In the table 4.8 are reported the percentage increases of the net yearly logarithmic returns when N increases from $N = 1$ to $N = 5$. For all the real time series is observable a consistent improvement in the results. The time series that records the higher increment is Telecom Italia S.p.A., with an average increment of 122%. The Artificial time series does not report significant improvements as, in this case, the *FTS* is able to learn the environment even with a simpler state descriptor, as it is produced by a known process, that is the GARCH one.

			Δ[%] of g[%] net					
Reward function	**L**	**ε**	**Ass.Generali**	**Buzzi**	**Mediobanca**	**Telecom**	**Saipem**	**Artificial**
SR	11	2.5%	-18%	97%	89%	132%	327%	17%
SR	11	5%	75%	93%	74%	109%	122%	-2%
SR	22	2.5%	75%	86%	96%	124%	137%	-2%
SR	22	5%	52%	87%	76%	116%	103%	10%
Sortino	11	2.5%	79%	95%	39%	108%	35%	16%
Sortino	11	5%	24%	94%	53%	108%	156%	-14%
Sortino	22	2.5%	85%	97%	67%	115%	595%	3%
Sortino	22	5%	54%	91%	64%	108%	146%	45%
ES	11	2.5%	62%	108%	52%	190%	103%	12%
ES	11	5%	48%	88%	68%	93%	107%	11%
ES	22	2.5%	120%	97%	84%	149%	112%	-1%
ES	22	5%	91%	90%	98%	108%	113%	20%
Mean			62%	94%	72%	122%	65%	9%

Table 4.8: %variation of g[%]net with $N = 1$ and g[%]net with $N = 5$.

The table 4.9 illustrates the percentage increments of the net final capital between the configurations with $N = 1$ and those with $N = 5$. Also in this case Telecom Italia S.p.A. is the time series which reports the higher improvements, with an average increment of 9228%. The Artificial time series was not affected by the change of the number of returns utilized in the state descriptors.

			Δ[%] of F.C. net					
Reward function	**L**	**ε**	**Ass.Generali**	**Buzzi**	**Mediobanca**	**Telecom**	**Saipem**	**Artificial**
SR	11	2.5%	-15%	1577%	1363%	2149%	58%	31%
SR	11	5%	916%	6364%	546%	12894%	210%	-4%
SR	22	2.5%	228%	1121%	1015%	4598%	205%	-4%
SR	22	5%	206%	2415%	1234%	12831%	183%	16%
Sortino	11	2.5%	336%	2878%	55%	25954%	17%	16%
Sortino	11	5%	65%	3043%	233%	17398%	582%	-12%
Sortino	22	2.5%	560%	3686%	337%	14069%	84%	3%
Sortino	22	5%	329%	5337%	1105%	8044%	152%	11%
ES	11	2.5%	80%	2100%	248%	513%	737%	20%
ES	11	5%	374%	3501%	1209%	1968%	871%	21%
ES	22	2.5%	159%	2952%	325%	598%	357%	-1%
ES	22	5%	696%	2042%	46%	9726%	137%	27%
Mean			328%	3085%	643%	9228%	300%	10%

Table 4.9: %variation of F.C. net with $N = 1$ and F.C. net with $N = 5$.

Conclusions

In this thesis, the development and the implementation of financial trading systems, based on several configurations of the *Q-Learning algorithm*, is illustrated. The *Q-Learning*, an algorithm which belongs to the *Reinforcement Learning* methods, is able to real-time interact with a dynamic environment in order to exploit its knowledge and achieve profitable results. The *Adaptive Market Hypothesis* is the theoretical framework in which the algorithm operates. Such theory justifies the possibility to make profitable trading, as security prices do not immediately and correctly fully reflect new informations coming into the market. The achievement of some positive results is a proof that the *Adaptive Market Hypothesis* theory is valid.

The *FTS* is applied on five real daily stock market price time series and on an artificial one, with different parameters values which give different configurations. After the analysis and the comparison of the outputs from the different configurations, some aspects stand out. The first concerns the quantity of informations contained in the state descriptors. Even if the system uses simple state descriptors, emerged that more informations help the agent to take more profitable and efficient actions, which bring to higher results. Indeed, state descriptors which consider the actual return plus the past four ones give higher results than the ones wich consider only the actual return.

A general aim may be finding a configuration that makes the *FTS* works profitable in each environment. A first result, in this sense, could regard the reward function. This is an another considered aspect. Indeed, most of the real stock price time series perform better with the *Expected Shortfall ratio*. Respect to the *Sharpe ratio*, the *Sortino ratio* and the *Expected Shortfall ratio* are more realistic performance measures, as they takes into account a downside measure of risk. However, this last one can be considered a more precise measure of risk as gives higher results.

An equilibrium between the learning rate and the exploration rate should be found. In the obtained outputs, a high learning rate requires a low exploration rate. The high learning rate is justified by the non stationarity of the environment, in which the system operates, and by the presence of many structural breaks. Given a high learning rate, the system has no need to explore too much the surrounding environment.

Due to the use of operative signals, it is possible to employ such a *FTS* into the real financial market. In fact, through the 500 simulations, which require 500 initial capitals and likewise transaction costs, the system has 500 actions to take. Obviously, such an approach is not feasible, as it is too expensive. Making an average of all the actions in each time step, it is possible to obain operative action signals, that is sell, out or buy. This method makes the implemented *FTS* practical and, for some configurations, profitable.

Even if the results obtained by my *FTS* are not optimal, it can be considered as a starting point for further improvements.

Acknowledgements

I would like to express my gratitude to Professor Marco Corazza for the support, advices, patience and precision he provided me throughout the time I wrote the thesis.

A big thank you to my parents, that gave me the opportunity to continue my studies and supported me all the time.

Bibliography

[1] Adcock, C., Areal, N., Armada, M., Cortez, M. C., Oliveira, B. & Silva, F. (2010). *Does the use of downside risk-adjusted measures impact the performance of UK investment trust?*. SSRN Electronic Journal.

[2] Bartle, R.G., & Sherbert, D.R. (2011). *Introduction to real analysis*. John Wiley & Sons, Inc. [Fourth edition].

[3] Bertoluzzo, F., & Corazza, M. (2012). *Reinforcement Learning for automatic financial trading: Introduction and some applications*. The Working Paper Series-Departement of Economics, Cà Foscari, University of Venice. Vol. 33, pp. 1-13.

[4] Bertoluzzo, F., & Corazza, M. (2014). *Reinforcement Learning for automated financial trading: Basics and applications*. Recent Advances of Neural Networks Models and Applications, Heidelberg, Springer, vol. 26, pp. 197-214.

[5] Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

[6] Brealey, R. A., & Myers, S. C., & Allen, F. (2014). *Principles of Corporate Finance*. McGraw-Hill/Irwin [Eleventh edition].

[7] Buşoniu, L., Babuška, R., Schutter, B.D., & Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press imprint of Taylor & Francis Group.

[8] Corazza, M., & RedexRiskManagement. *Apprendimento per rinforzo e Forex Robot*. Tratto da YouTube: www.youtube.com/watch?v=A3IQb5_KrBU

[9] Edwards, D. (2014). *Risk Management in Trading: Techniques to Drive Profitability of Hedge Funds and Trading Desks*. John Wiley & Sons Inc., pp. 141-176.

[10] Estrada, J. (2006). *Downside Risk in Practice*. Journal of Applied Corporate Finance, Vol. 18, No. 1, pp. 117-125.

[11] Fama, E. F. (1969). *Efficient Capital Markets: A Review of Theory and Empirical Work*. The Journal of Finance, Vol 25, No. 2, pp. 383-417.

[12] Farmer, J. D., & Lo, A. W. (1999). *Frontiers of finance: Evolution and efficient markets*. Proc. Natl. Acad. Sci. USA, Vol. 96, pp. 9991-9992.

[13] Lo, W.A. (1999) *The Three P's of Total Risk Management*. Financial Analysts Journal.

[14] Lo, W.A. (2002) *Bubble, Rubble, Finance in Trouble?*. Journal of Psychology and Financial Markets.

[15] Lo, W. A. (2004, August). *The adaptive market hypothesis*. The Journal of Portfolio Management.

[16] Lo, W. A. (2005) *Reconciling Efficient Markets with Behavioral Finance: The Adaptive Markets Hypothesis*. The Journal of Investment Consulting, Vol. 7, No. 2, pp. 21-44.

[17] Luenberger, D. G., & Ye, Y. (2008). *Linear and Nonlinear Programming*. Springer Science+Business Media, LLC [Third edition].

[18] Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). *Performance functions and Reinforcement Learning for trading systems and portfolios*. Journal of Forecating, Vol. 17, pp. 441-470.

[19] Moody, J., & Saffel, M. (2001). *Learning to Trade via Direct Reinforcement*. IEEE Transactions on Neural Networks, Vol. 12, N. 4, pp. 875-889.

[20] Sutton, R.S., & Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. The MIT press.

[21] Tasche, D. (2002). *Expected shortfall and beyond*. Journal of Banking & Finance, Vol. 26, pp. 1519-1533.

[22] Watkins, C. J. (1989). *Learning from Delayed Rewards*. Ph.D. Thesis. King's College.

[23] Weiss, G. (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge.

[24] Wiering, M., & Otterlo, M. v. (2012). *Reinforcement Learning State-of-the-Art*. Springer Heidelberg New York Dordrecht London.

[25] Wooldridge, M., & Jennings, N. R. (1995). *Intelligent agents: theory and practice*. The Knowledge Engineering Review, Vol. 10:2, pp. 115-152.