UNIVERSITÀ CATTOLICA DEL SACRO CUORE
FACULTY OF BANKING AND FINANCE
MSC.IN STATISTICAL AND ACTUARIAL SCIENCES
DATA ANALYTICS FOR BUSINESS AND ECONOMICS

DYNAMIC PRICING UNDER COMPETITION IN AN E-COMMERCE SCENARIO: A DEMAND LEARNING AND PRICE OPTIMIZATION TECHNIQUE

Author:
Andrea Rafanelli
mat.4815168

Supervisor:
Marco L. Della Vedova

2019-2020

Dedicated to me,

to my parents,

to my love,

and to my friends.

ABSTRACT

The intent of this work is investigating how market sellers can increase their profits using automatic learning algorithms. Despite the benefits shown by automation in pricing, machine learning methods don't have great applicability in the real world.

The literature reported in this text is a proof of this fact: there are numerous studies and researches in which algorithms for the dynamic pricing are built. The construction of these, however, almost always takes place in simulated environments, not allowing a real generalization of these models.

Also in this thesis, using a method of reinforcement learning, a dynamic pricing tool has been built within a simulated market. The market simulates an online e-commerce scenario, with the presence of heterogeneity among consumers and competition between competitors.

Our work allows to build an algorithm that is able to vary prices by incorporating the information received on the market.

In this work it was not possible to make a direct comparison between our algorithm and those of the other competitors, due to lack of data about their revenues and sales. In fact, it was not possible to understand if our algorithm obtained more sales or more profits than the others. Nevertheless, it was found that the proposed algorithm is able to adapt to various

market situations, offering medium-high prices but often below those of other competitors.

The approach followed was to first estimate demand using historical market data through a regression model for learning demand. Second, to use the value iteration algorithm to find optimal pricing strategies.

Since this is an empirical work, at the end of the text some assumptions will be made and some limits of our model reported.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

RL	Reinforcement Learning
MDP	Markov Decision Process
GPI	Generalized Policy Improvement
DP	Dynamic Programming
MC	Monte-Carlo
TD	Temporal Difference
PI	Policy Iteration
WTP	Willingness-to-pay

CHAPTER 1

INTRODUCTION

1.1 Motivation

Price updates on the markets, especially online ones, are increasing year by year and are occurring more and more frequently.

Adapting prices automatically and quickly is called *dynamic pricing*.

Dynamic pricing has become an important tool for merchants to increase their profits and efficiently respond to market demands. In practice, this tool would allow companies to choose the right price at the right time.

As cited by Sairamesh and Kephart [29], markets are experiencing more and more pricing decisions made by algorithms. It is therefore necessary to develop these algorithms but also to understand the consequences of using them on the market.

However, the implementation of this methodology is often difficult.

Prices, in fact, depend on many variables both internal (for example the operating costs of the company) and external (for example consumer preferences).

In order to implement a correct pricing strategy, a large amount of data is therefore required. Companies must be able to process these data and to obtain useful information from them, at a minimum cost.

The difficulty for companies is twofold: on the one hand it requires the retrieval of data consistent with the foreseen needs, on the other hand it requires the ability, for firms, to process data and build algorithms as precise and fast as possible.

The hardest part is certainly the one that concerns the estimation of customers' purchasing attitudes on the market. To perform this estimation it is necessary that the algorithm is able to learn the consumer demand.

If the data are insufficient or incomplete, the algorithm may not learn enough information and be inefficient in choosing prices.

To build a good pricing algorithm, a good demand learning is essential.

Nowadays there are many types of algorithms and learning methods used by businesses to meet the need for a dynamic and automated pricing.

The use of reinforcement learning for dynamic pricing is a recently developing method that is not yet widespread.

In this thesis the use of such learning will be experimented.

The development of the thesis was designed to approach this new and current phenomenon, that of dynamic pricing, and to clarify the requisites needed by companies to implement it.

In addition, the reinforcement learning method has been explored, as a method that is developing and is demonstrating a good ability to adapt in very dynamic environments, such as the one explored here. Indeed, this method is able to learn from recent experiences and therefore to adapt its prices to complex and highly vulnerable market environments.

1.2 Goal and contributions of the thesis

The main objective of this thesis is to build an algorithm capable of predicting prices automatically by adapting to variations within the market in which it operates. The changes concern both customer behavior and

price changes by competitors.

To accomplish this goal, an algorithm was built to compete within an online competition (Dynamic pricing competition, Haensel AMS)[1], in order to obtain data and evaluate its performance within a simulated market.

The simulated environment was formalized the environment as a Markov Decision Process. A Markov decision-making process allows to solve optimization problems -in this case we talk about price optimization- and can be solved with dynamic programming or linear programming techniques.

The method used in this work is a dynamic programming method called *value iteration*, which enables to find an optimal pricing policy given the variations within the environment.

Furthermore, in this thesis, particular importance has been given to the demand learning part. This learning was carried out using a regression model that made it possible to process and incorporate information on customers and competitors in the value iteration algorithm, but also to build transition and reward functions for the implementation of the algorithm itself.

Through this work, the following contributions were carried out:

- The approximation of consumer demand and competitors' strategies through historical market data using demand learning;
- The construction of a price optimization algorithm through the Dynamic programming method;
- The continuous recalculation of the variables through the retrieval of more and more historical data;
- Computational time improvements on the algorithm to allow participation in the competition;

[1] `https://dynamic-pricing-competition.com/`

1.3 Thesis outline

This thesis is divided into six chapters.

Chapter 2 reports the literature on the topic of dynamic pricing. In this chapter the origins and improvement of this technique is explained. Subsequently in section 2.3 some elaborations on the use of reinforcement learning methods for the development of dynamic pricing algorithms are illustrated.

Chapter 3 describes the methodology underlying reinforcement learning: what it is, how to formalize it. A Markov decision-making process is also described in section 3.2.

Section 3.5 explains the possible resolution methods of RL problems: Dynamic Programming, Monte-Carlo, and Temporal Difference.

Chapters 4 and 5 are those dealing with the experiment. In 4 the problem and the scenario in which the algorithm will have to compete is described; 5 shows the structure with which the experiment was built.

Finally, in chapter 6 the results of the simulations are described and in chapter 7 the assumptions, hypotheses and limits of the algorithm and of the market in which it competed are reported.

CHAPTER 2

DYNAMIC PRICING

In the Big Data era it is becoming increasingly usual in online marketplace that pricing decisions are made automatically. Automated pricing techniques are fundamental in revenue management[2], for their ability of changing prices as often as possible. Dynamic pricing is the study of deciding ideal prices of products or services in a setting where prices can frequently be adjusted, as A.V. Den Boer [41] wrote.

In this chapter, the advantages, limitations and applications of Dynamic Pricing will be presented, also introducing various literature on the subject.

[2]Assortment of methodologies and strategies that organizations use to manage demand for their services and products [38]

2.1 Dynamic pricing advantages

Dynamic pricing refers to a system that includes pricing strategies, rules, and approaches that collaborate with each other to achieve business goals.

In particular, it allows companies to decide the prices of a product or service based on market demand and offer, competitors' prices, customer behavior and their purchasing attitudes, and other external factors that can influence market decisions.

Basically, dynamic pricing is an approach built specifically for the online markets. In fact, given the multitude of information and data present in this type of market and the speed required in making business decisions, dynamic pricing is able to perform well to this context. It allows pricing makers to quickly change prices based on market and company needs.

W.Reinartz [35] proposed five conditions for the application of effective dynamic prices strategies. These conditions are:

1. Customers must be heterogeneous in their willingness to pay;
2. The market must be segmentable;
3. The arbitrage potential is limited;
4. The cost of segmentation and policy must not exceed the revenue increases due to personalization;
5. Violations of perceived fairness must not be committed;

The first condition, as Reinartz [35] writes, is that customers are willing to pay different prices for the same goods or services. This is the case of heterogeneous markets, where each customer has its own willingness to pay that is influenced by preferences, costs, experiences, etc.

In online markets, customers show greater price sensitivity than offline ones (Rangaswamy et al. [34]). Moreover, Reinartz [35] cites Porter [33], who argues that the Internet is likely to lead to more competitive markets with several pricing changes and lower profit margins.

Regarding the second condition, also in this case the online market improves the ability of a company to segment its market. This, happens *"by tracking individual customer purchases through the internet. Building up profiles over time allows firms to assess consumers' price sensitivity and then to start marketing campaigns either to individuals, or more likely, to segments that have varying degrees of willingness-to-pay"*.

The third condition is that consumers' ability to arbitrage is limited. Reinartz [35] writes: *"A person who bought a product at a lower price should not be able to resell it for a profit to customers who have a higher willingness-topay. If this condition does not exist, firms have no incentive to discriminate on price"*.

The fourth criterion concerns the implementation cost, of dynamic pricing strategies, which must be reasonable.

Finally, the last condition is that which concerns the fairness perceived by the customer during the purchase phase: *"perceived fairness is when the buyer feels that both parties in a transaction have gained"*.

Having said that, since there is currently a large presence of online markets, the possible great applicability of dynamic pricing is understandable.

Dynamic pricing is carried out thanks to the use of algorithms that exploit data such as: customer purchase preferences, customer demographic information, historical sales data, historical data on competitors' prices etc.

These algorithms provide important benefits for companies, as Z. Y. Brown and A. MacKay [7] write, i) the use of algorithms reduces the cost of updating prices, facilitating more frequent prices. In fact, prices can be changed once a day, every hour, every ten minutes or even several times a minute, it depends on the technology investments made by the company; ii) the algorithms produce fewer errors than a human agent and allow to make more accurate pricing decisions by efficiently combining information from multiple sources; iii) the algorithms guarantee the pursuit of a strategy.

The price is decided according to strategies established ex-ante by the firms.

Dynamic prices have many advantages. Initially, profit increases, as R. Phillips said [22], *"Provides automated demand estimates and price adjustments that help managers increase revenues, get customer information and respond to changing requests"* , it is also able to capture a wider audience by obtaining more loyal customers (price customization).

The dynamic pricing algorithms can be developed in many different ways, it depends on the needs of the firms, on the type of industry in which they are applied, on the data available and on many other factors. The benefits, however, are tangible. A research by Forrester[3] states that dynamic pricing is capable of increasing profits by 25%.

In fact, for example, access to real-time data allows to check the changing trends on the market and establish appropriate prices, also increasing the control by sellers. From what has been said, there are therefore numerous factors that lead a company to rethink its price strategy and to adopt a dynamic pricing technique.

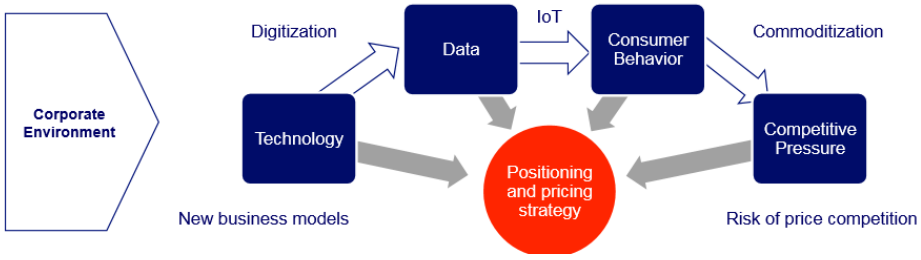

Figure 2.1: Impact of a corporate environment on pricing models

source: Krämer A. and Kalka R., How Digital Disruption Changes Pricing Strategies and Price Models [19]

[3] https://go.forrester.com/

2.2 Origin and development of dynamic pricing

Airlines were the first industry where dynamic pricing was applied. The price of airline tickets depends on many factors: on what day of the week the flight takes place, at what time of day, how long the journey is, how many seats are present etc.

Before that, in 1972, Littlewood [20] wrote an article in which he proposed the so-called *Littlewood rule*. This rule stated that discounted fare tickets should only be accepted if their revenue value exceeded the anticipated revenue of future full fare tickets.

Until 1980, clerks were in charge of changing prices several times a day according to the company's needs.

Dynamic pricing was born around 1980, when American Airlines lost a lot of money due to the invasion of People Express Airlines.

People Express was a low-cost airline offering customers discounted flights. Robert Crandall, president of American Airlines, decided to overcome this problem by starting to adjust prices dynamically based on the type of ticket, customer preferences, seat place on the plane etc. He realized that computers could change prices automatically using the information available on the market.

The proposed program assigned reservations to different classes of passengers, in this way the airline could divide passengers into business, personal and leisure travelers, customizing the ticket prices according to the type of travel required. Smith et al. [31] wrote about American Airlines that their goal was to *"sell the right seats to the right customer at the right prices"*.

Given the success, since 1980, many airlines started investing in software for automatic price updating. Other industries, from the pharmaceutical to the hotel, up to e-commerce have followed this trend.

There has been a lot of progress in the area of dynamic pricing. As a matter of fact, an increase in academic discipline and literature can be seen (see K.T. Talluri and G.V. Ryzin,[38], or R. Phillips and O. Ozer,[23]).

According to M. Bernstein [3], price is one of the most important methods that a company has to communicate the value of its product or service and it is also one of the most important factors that determine profits.

Obviously, as mentioned before, dynamic prices have many applications: airlines, pharmaceutical sector, e-commerce and so on.

In this thesis, the main objective is the application of dynamic prices in an e-commerce scenario.

2.3 Dynamic Pricing in e-commerce

E-commerce has seen a rapid increase in the past few decades. With the debut of the internet, around the 90s, many companies started investing in new technologies which allowed the collection of many data on a daily basis.

These data contained a multitude of consumer information, such as demographic information, preferences, purchasing attitudes, etc. Companies collect data and use data analysis services to improve their decision making and optimize available resources.

The use of new technologies allowed the development of e-commerce. The term *"e-commerce"* was born in 2000 and really means the ability to buy any object or experience online. E-commerce has revolutionized people's purchasing habits and allowed retailers to reach new customers, expanding beyond the physical store (see Amazon or Ebay).

The spread of e-commerce enables new technologies and new software to evolve. The result is the use of pricing algorithms that are capable of replicating human action efficiently. Without these programs, sellers should

have pricing teams able to monitor price trends on the market and calculate new ones based on market variations and competitors' strategies.

The price becomes a fundamental aspect in this type of market, being, in many cases, what most interests consumers.

Many retailers invest in the development of dynamic pricing methods, others prefer to outsource to third parties (Capterra, Minderest, Omnia etc.).

It can be said that e-commerce is a perfect scenario for the use of dynamic pricing tools, precisely because in this market price decisions are made about 60 million times a day[4].

The dynamic pricing process can be inferred from a general revenue management process. Most dynamic pricing algorithms are based on three basic points:

1. The treatment of historical data;
2. Processing a demand function or learning it;
3. The use of mathematical processes for price optimization;

The first point, in particular, seems to be the key to the success of a good pricing algorithm. Companies need to have precise and transparent data immediately in order to better define a pricing strategy.

The e-commerce scenarios differ from each others. According to a classification made by M.Cheng and Z.L.Chen [10], these scenarios can have:

1. Three different structures of time-horizon: i) Continuous time-horizon, where price changes occur at any point in time, ii) Discrete time-horizon, where prices are changed periodically, iii) Customer based time-horizon, in which price changes arise at the point in time of customer arrivals;
2. Two different ways of setting prices: i) Continuous allowable prices, where firms set any possible price, ii) Discrete allowable prices, in

[4] https://www.omniaretail.com/blog/the-history-of-dynamic-pricing

which firms set prices within a range, following the rationale that certain prices are preferred by costumers;

3. Two different pricing schemes: i) Pre-announced pricing, where firms announce all the prices at the beginning of the sales period, ii) Contingent pricing, in which firms announce their prices over time;

There are, also, other characteristics in the e-commerce market, such as the replenishment or the non-replenishment of the inventory, the presence or not of strategic consumers, the dependency or the independency in time. All of these features affect in different ways pricing strategies and pricing decisions during time.

In online marketplace, what is common, nowadays, is the presence of competition. Price competition is an important aspect, that has become a relevant area of economics research since the 1800's last years (c.f. Bertrand [4] or Edgeworth [12]).

Current investigations on dynamic pricing models in competing environments have been carried out. This sort of literature has expanded rapidly as of late.

In the past, models for dynamic pricing under competition have hardly considered because of their complexity. In fact, the most critical issue when dealing with these kinds of models is the high number of possible market situations (the curse of dimensionality).

During recent years, with the high development of computation, it has become more accessible for researchers to face up to these sorts of problems. For example, V. Martinéz de Albéniz and K. Talluri [21] studied price competition on differentiated products in an oligopoly setting where each seller has a fixed number of products for sale. E. Adida and G. Perakis [1] proposed a competitive pricing model under the problem of inventory control.

B.D. Chung et al. [11] considered a problem of demand learning and dynamic pricing in both, monopolistic and oligopoly market. They used a

Markov Chain Monte Carlo method to estimate model parameters for demand learning and predicted new prices according to the proposed model.

R. Schlosser and R. Keven [36] built a dynamic pricing model under some assumptions: finite horizon, inventory level and customer behaviour as well as competitors' strategies were unknown.

2.4 Demand learning: exploration vs exploitation

Prices fluctuate thanks to pricing algorithms. These, exploit information on customers, on competitors and, in general, on the market situation for setting prices.

When firms deal with prices, they have to define strategies according to the inventory level, the minimum revenue, the costs, as well as the clients. To develop such pricing policies,firms have to forecast the demand in an accurate way.

One of the assumptions in many studies and research is that the demand curve is known a priori by decision makers. In reality, they do not have a full knowledge of market information and demand curve. As R. Rana and F.S. Oliveira [26] wrote, there are two sources of randomness in demand: i) customer arrival rate, ii) customer reservation price.

What companies observe is the realized demand over time, but not the demand function itself. Since it is important for these to understand customer behavior and formalize a demand curve in order to maximize their revenues, approaches for learning demand were born over time.

Many dynamic pricing and demand learning problems are accomplished with MAB (Multi-Armed-Bandit). MABs are often used to make pricing decisions under uncertainty (see A.V. Den Boer [41] or Paladino et al. [39]). By the way, in one of the first studies formulated for a dynamic pricing problem with an unknown demand curve, a MAB was used (Roth-

schild [28]).

Other approaches to deal with the lack of demand curve information, are those Bayesian. The Bayesian framework allows priors to be established ex-ante on the distribution of demand (see Carvalho and Puterman [8] or Harrison et al. [15]).

The problem of using a Bayesian approach is that pricing decisions are made on the basis of the prior distribution, which often comes from the so-called family-conjugated.

The use of prior allocation and Bayesian approaches in highly dynamic environments leads to large restrictions on models.

In recent years, data-driven approaches to learning demand have been developed (see Ito and Fujimaki [17] or Schlosser and Boisser [36]). These approaches allow, through different demand learning techniques, to estimate demand in real-life online markets.

For what has been said, dynamic pricing is not only about optimization of prices but, also, about learning the demand function. The above-mentioned tasks cannot be separated, but have to be considered simultaneously.

In practice, companies use statistical learning tools, on historical sales data, to estimate demand functions -that mostly depend on the customers' behaviour, on the size of competition, on the presence of differentiation in the market and so on-. The use of optimization techniques, instead, regard the choice of the right price, according to the obtained demand learning results. Consequently, Dynamic pricing methods belong to the class of sequential decision problem under uncertainty.

An area called reinforcement learning (RL) explores algorithms to make good sequential decisions in an uncertain environment (R.S. Sutton and A.G. Barto [37]). The most recent literature provides evidence that reinforcement learning is suitable for finding optimal, or near-optimal, pricing policies in simulated environments.

However, as their performance is measured in a simulated environment, it is difficult to evaluate the use of these algorithm in real markets.

The biggest problem is not only how quickly the algorithm reaches optimal prices but also at what cost. To learn the market demand a part of exploration is necessary, especially in the case of RL algorithms. During an exploration phase, random prices are usually chosen which can lead to a loss of profits.

Despite the possible limitations of applying such algorithms to real markets, there is a lot of literature on the subject.

Renzhi L. et al. [27] formulated a dynamic pricing model as an MDP and used a Q-learning algorithm to decide the retailer electricity price in an online learning process.

F. Trovò, S. Paladino et al. [39] designed a multi-armed bandit algorithm to learn optimal prices in an e-commerce scenario.

W. Cheng, H. Liu and D. Xu [9] explored a Q-learning algorithm mechanism to adjust prices if perishable products under demand uncertainty in a competitive market. They also showed up how optimal prices are influenced by customer preferences, retailer prices and customer demands.

C.V. Raju, Y. Narahari, K. Ravikumar [25] used a Reinforcement learning technique to identify optimal prices in an online retailer store.

CHAPTER 3

REINFORCEMENT LEARNING METHODS

This chapter summarizes and illustrates the methods and techniques behind Reinforcement Learning, which will help to better understand the thesis topic. In chapter 3.1 the basic concepts and foundations of the traditional reinforcement learning method are presented. Chapter 3.2 illustrates the formalization of this method as a Markov Decision Process. Finally in 3.3 and 3.4, optimization problem will be discussed. DP, MC, TD algorithms will be presented.

The preparation of this chapter was inspired by the discussion conducted by R.S. Sutton and A.G. Barto [37].

3.1 Reinforcement Learning elements

Reinforcement Learning is a learning structure in which an agent interacts with the environment in a "trial and error" way.

In contrast of other machine learning methods, the agent explores the environment to realize the greatest amount of future rewards, usually looking for a goal, represented numerically by a reward. The main elements in Reinforcement learning methods are:

- The *"policy"*, π, is a mapping from states, S_t, of the environment to actions, A_t be taken in those states: $\pi(s) : S \rightarrow A$.

The state, S_t, is the current situation in which the agent is located.

A policy is effectively a probability distribution over actions given states. It defines and regulates the behavior of an agent; if the agent follows π at time t, $\pi(s)$ is the probability that $A_t = a$ if $S_t = s$:

$$\pi(a|s) = Pr[A_t = a|St = s] \tag{3.1}$$

- The *"reward"*, R_t, *is a stochastic function of the action and the state* [32]. It describes the payoff of taking a specific action given a state.

In each time step t, the environment sends to the agent a number, $R_t \in R$. The agent's goal is to maximize the whole reward it receives over the long run.

A negative reward leads to undesirable behavior, as the opposite of positive reward. In this sense, the reward is related to immediate desire, meaning that it communicates to the agent, in an instant sensation, what it would wish to realize.

- The *"value function"*, $V_\pi(s)$, is the total amount of reward within the future, ranging from a specific state. It indicates, what is good within the

long run (long-term desirability), taking into consideration the states that are likely to follow (the agent's path) and therefore the reward in each of these states.

The purpose of estimating value functions is to urge more rewards. It is important, therefore, to determine a way for efficiently estimating these values, since the values are estimated and re-evaluated for the whole lifetime of the agent.

- The model simulates the behavior of the environment and allows to formulate hypotheses on how the environment will behave.

"Models are used for planning, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced", says R.S. Sutton and A.G. Barto [37].

There are two different types of main models: *model-based* and *model-free*. The two models have been explained so far (sec.3.4).

3.2 Markov Decision Process

Formally, a Markov decision making (MDP) is employed to explain the interaction between the agent and its environment in terms of states, actions and rewards. Many reinforcement learning problems are often formalized as MDP.

To introduce a MPD, suppose a stochastic process[5] $X = \{X_t : t \in T\}$ on (Ω, F, P) with a time space T and a state space S. X_t is a random variable taking values in $S \vee t \in T$. Also a filtration, $\mathfrak{F} = \{F_t : t \in T\}$ has to be formalize.

The filtration is a collection of σ-algebras with the properties that $X(t)$ is measurable with respect to $\mathfrak{F}$. In practice, $\mathfrak{F}$ is a filtration and the process X is adapted to $\mathfrak{F}$.

[5]Formulas and theory taken from: www.randomservices.org/random/markov/General.html

With these elements, some properties and definitions can be described.

i) The random process X is a Markov process if:

$$P(X_{s+t} \in A|F_s) = P(X_{s+t} \in A|X_s) \tag{3.2}$$

Thus X has the *loss of memory* property. This property says that *"the future is independent of the past, given the present"*, Andrei Markov (1856-1922).

This means that what happens next in the environment depends only on the current state:

$$P[X_{t+1}|X_t] \doteq P[X_{t+1}|X_1, X_2..X_p] \tag{3.3}$$

In this way the current state can be a sufficient statistic that provides us with an equivalent long-term characterization as in the case of having the whole story.

ii) The random process X is a Markov process if:

$$E[f(X_{s+t})|F_s] = E[f(X_{s+t})|X_s] \tag{3.4}$$

Thus X is a Markov process relative to $\mathfrak{F}$. If X satisfies the Markov property relative to the filtration, then it satisfies the Markov property relative to any other filtration.

For a Markov state S_t and a successor state S_{t+1}, the state transition probability function is defined by:

$$P^a_{ss'} \doteq Pr[S_{t+1} = s'|S_t = s, A_t = a] = \sum_r p(s', r|s, a) \tag{3.5}$$

It is a probability distribution that, given preceding state and action, describes the probability of transiting from one state to another.

The matrix of all the state transition probabilities is called state transition matrix:

$$\begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,n} \end{pmatrix}$$

The Markov process is a random process without memory, a sequence of random states having the Markov property.

Markov decision process has evolved to understand the issue of creating decision sequences in conditions of uncertainty, in which each decision can depend on previous decisions and their results.

In a MDP, we have:

- **S**, finite set of Markov states $s \in S$;
- **A**, finite set of actions $a \in A$;
- **P**, the state-transition probability (3.2), a function that characterizes dynamics of finite MDP;
- **R**, the reward function that describes the reward value that the agent receives from the environment after an action. It depends on the state and the action:

$$R(S_t, A_t) \doteq \mathbf{E}[R_{t+1} | S_t = s, A_t = a] = \sum_r r \sum_s^{'} p(s', r | s, a) \tag{3.6}$$

- $\gamma \in [0, 1]$, a discount factor;

Thus, an MDP can be described as a tuple (**S, A, P, R,** γ).

The key elements to formalize the sequential decision-making process are returns, value functions and Bellman equations.

3.2.1 Agent-Environment interaction

The agent interacts continuously with the environment over a sequence of discrete-time steps.

The agent selects the actions and therefore the environment responds to the actions by changing and evolving into new situations. The environment also offers the agent rewards that the agent tries to maximize over time by choosing actions.

To formalize, given a set of discrete-time steps t=0,1,2...T, the agent receives a representation of the environment's state, $S_t \in S$ and, given the state, selects an action $A_t \in A$.

One step later the agent receives a reward $R_{t+1} \in R$ and find himself in a new state S_{t+1}.

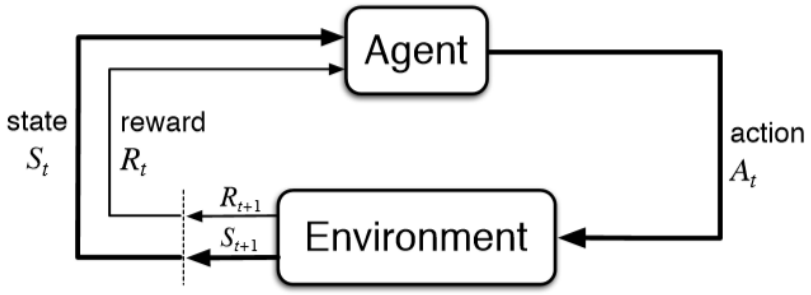

Figure 3.1: The agent–environment interaction in a Markov decision process
source: Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, pag.49

The probability that particular values of R_t and S_t occur at time t, given specific values of preceding state and action is described in (3.5).

3.2.2 Goal and reward

At each time t, the agent receives a reward $R_t \in R$. The agent's goal is to maximize the full amount of the reward he receives.

Markov Reward Process indicates the amount of reward accumulated through a particular sequence. Here we do not mention immediate reward but long term cumulative returns.

G_t is the total rewards discounted by the temporal passage t and it is called *"expected return"*. The goal is to maximize this return:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + .. = \sum_{k=0}^{+\infty} \gamma^k R_{t+k+1} \tag{3.7}$$

where k is the number of times a reward is received, γ is the discount rate that informs the agent how much cares about rewards in the distant future relative to those in the immediate future. If γ = 0, the agent maximizes the immediate reward, if γ = 1, equation(3.7) is undiscounted and therefore the agent takes care of all future rewards. The discount factor is advantageous because it allows to avoid infinite cycles within the Markov process, ensuring the convergence of the algorithm.

Moreover, it is also used because the model is not perfect: γ defines a kind of finite-horizon in which the agent-that is not sure what it will happen in the future- may consider immediate reward rather than waiting for getting a bigger reward in the future.

3.2.3 Policy and value function

As mentioned in paragraph 3.1, the value function estimates “how good” is for the agent to be in a given state and do a specific action.

Formally, it is defined as the state-value function for the policy π.

The state-value function of a MDP is the expected return starting from state S_t and then following the policy π:

$$v_\pi(s) \doteq E_\pi[G_t|S_t = s], \forall s \in S \tag{3.8}$$

Similarly, the state-action value function for the policy π, is the expected return the agent receives, starting from state S_t, taking action A_t and following the policy π:

$$q_\pi(s,a) \doteq E_\pi[G_t|S_t = s, A_t = a], \forall s \in S, \forall a \in A \tag{3.9}$$

Both, $v_\pi(s)$ and $q_\pi(s,a)$ can be estimated from experience.

3.3 Bellman expectation equations

State-value function (or state-action value function) satisfies the *"recursive relationship"*. This property allows to compute the corresponding value at the current state-taking transition probabilities into account.

For any policy and any state, the next condition of consistency holds between the value of the current state and the value of its possible successor states:

$$v_\pi(s) \doteq E_\pi[\, G_t|S_t = s\,] = E_\pi[R_{t+1} + \gamma\,(G_{t+1})|S_t = s] \tag{3.10}$$

Remember from equation (3.7):

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + .. = R_{t+1} + \gamma\,(G_{t+1})$$

and from (3.6)

$$E_\pi[R_{t+1}|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)r$$

Then (3.8) becomes:

$$\begin{aligned}
&= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)(r + \gamma E_\pi[(G_{t+1})|S_{t+1} = s]) = \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)(r + \gamma v_\pi(s')) = \\
&= E_\pi[R_{t+1} + \gamma\, v_\pi(S_{t+1}|S_t = s)],\ \forall s \in S
\end{aligned}$$

The equation (3.10) is the Bellman equation for v_π, where the first term is the immediate reward and the last one is the discounted value of the successor state.

Considering all the actions that the agent could take later and considering all the states that the environment could present to the agent, for each of the states, there have been many successive ones. The agent could land in one of those states. It is important to know how good it is to be in a specific state, and to continue with the same policy. In practice: *how much reward will the agent receive for continuing from that moment on?*

We have to consider the average over possible events that might happen. Summing all those things together can explain how good is to take that particular action from that particular state S.

The recursive relationship can be represented with a backup diagram that shows the relationships belonging to the update operations (fig. 3.2).

From figure 3.2, the update operations transfer the value information back to a state from its successor.

Similarly, the action-value function can be rewritten using the Bellman

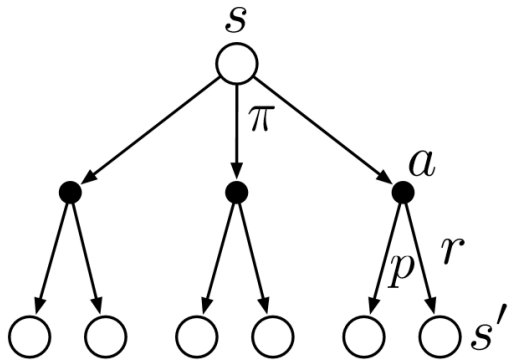

Figure 3.2: Backup diagram for v_π

source: Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, pag.59

equation:

$$
\begin{aligned}
q_\pi(s,a) \doteq & E_\pi[G_t|S_t=s,A_t=a] = \\
& = E_\pi[R_{t+1}+\gamma(G_{t+1}|S_t=s,A_t=a)] = \\
& = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)(r+\gamma E_\pi[(G_{t+1})|S_{t+1}=s]) = \\
& = \sum_{s',r} p(s,r|s,a)(r+\gamma \sum_a^{'} E_\pi[G_{t+1}|S_{t+1}=s',A_{t+1}=a'] = \\
& = \sum_{s',r} p(s,r|s,a)(r+\gamma \sum_a^{'} \pi(a'|s') q_\pi(s',a')) = \\
& = E_\pi[R_{t+1}+\gamma q_\pi(S_{t+1},A_{t+1})|S_t=s,A_t=a],\ \forall s \in S,\ \forall a \in A
\end{aligned}
\tag{3.11}
$$

If the agent is in state S and takes an action, it will get an immediate reward for that action, then we will examine where the agent ends up.

What is the value of the action within the state in which the agent is, under the action that it will select from that moment on? The sum of these things together tells us how good it was to request that action from that specific state.

Also for the state-action value is it possible to represent the recursive relationship with the backup diagram:

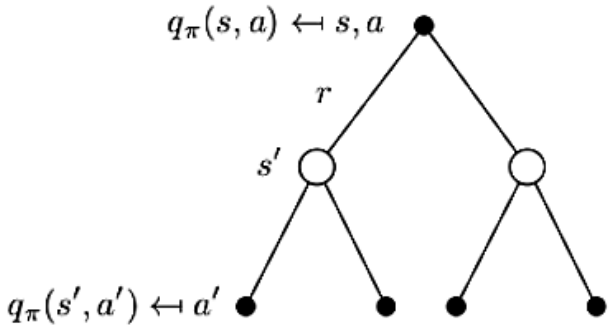

Figure 3.3: Backup diagram for q_π
source: Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, pag.61

3.3.1 Optimal value function and optimal policy

For a finite MDP, the value function defines a partial ordering on the policies. A policy π, is defined to be better than or equal to a policy π', if its expected return is bigger than or equal to that of π' for all states:

$$\pi \geq \pi'$$

i.i.f:

$$v_\pi \geq v_{\pi'},\ \forall s \in S$$

There is, at least, one policy that is better than all the others. All the optimal policies are denoted as π^*. The optimal policies achieve optimal value function and optimal action-value function.

The optimal state-value function, $v_\pi(s)$ is the maximum value function overall policies, thus the maximum rewards that one can extract from a MDP:

$$v^*(s) \doteq max_\pi\ v_\pi(s),\ \forall s \in S \tag{3.12}$$

While the optimal action-value function $q^*(s,a)$ is the maximum action-value function overall policies, thus the maximum amount of rewards one can extract starting in state s, and taking action a:

$$q^*(s,a) \doteq max_\pi\, q_\pi(s,a),\ \forall s \in S,\ \forall a \in A \tag{3.13}$$

Bellman's equations (3.8) and (3.9) become the Bellman's optimal equations. These express the fact that the value of a state under an optimal policy must be equal to the expected return for the best action in that state:

$$\begin{aligned} v^*(s) \doteq & max_a\, q^{\pi^*}(s,a) = \\ & = max_a\, E_\pi[R_{t+1} + \gamma\, v^*(S_{t+1}) | S_t = s],\ \forall s \in S \end{aligned} \tag{3.14}$$

The Bellman optimal equations can be represented with the backup diagrams:

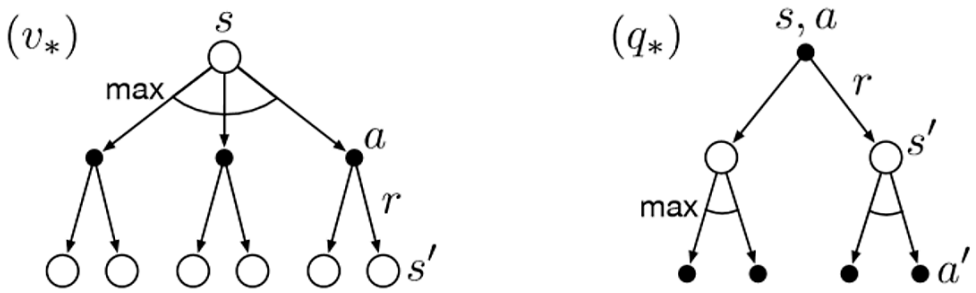

Figure 3.4: Backup diagram for v^*_π and q^*_π
source: Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, pag.64

For finite MDPs, Bellman's optimal equation has a single solution independent from politics.

If the reward function, $R(S_t, A_t)$ and the transition probability, $P[S_t, A_t]$ are known, the system of equations for v^* is often solved using one of the methods for systems of nonlinear equations.

Once v^* is obtained, the optimal policy can be established: for each state, the action (or actions) that gives the maximum in Bellman's optimal equation is (are) considered. Then, the probabilities associated with the other actions are set to zero. A policy that assigns non-zero probabilities only to such actions is optimal.

Having q^*, the search for the optimal policy is simpler because for each state s, the action that maximizes $q^*(s,a)$ has already been stored.

However, solving these equations is not always possible and is often impractical. Certain conditions are necessary:

- Understand the dynamics of the environment;
- Markov's property has to be respected;
- Have sufficient memory and computing resources available;

We can opt for iterative methods that approach q^* and v^*, such as the Dynamic Programming method (DC), the Monte-Carlo method (MC) or the Temporal Difference method (TD).

3.4 Iterative methods

The iterative algorithms in Reinforcement Learning, are used to approximate value functions and policies and to find the optimal points.

These algorithms, here, can be described as Generalized Policy Iteration (GPI).

GPI is based on two processes that interact simultaneously: one to make value functions consistent with current policy, *policy evaluation* (prediction problem), and the other to make policies greedy to the current value function, *policy improvement* (control problem). These two processes alternate and can be considered competitive and cooperating:

These pull in the opposite direction, but, in the long term, interact to

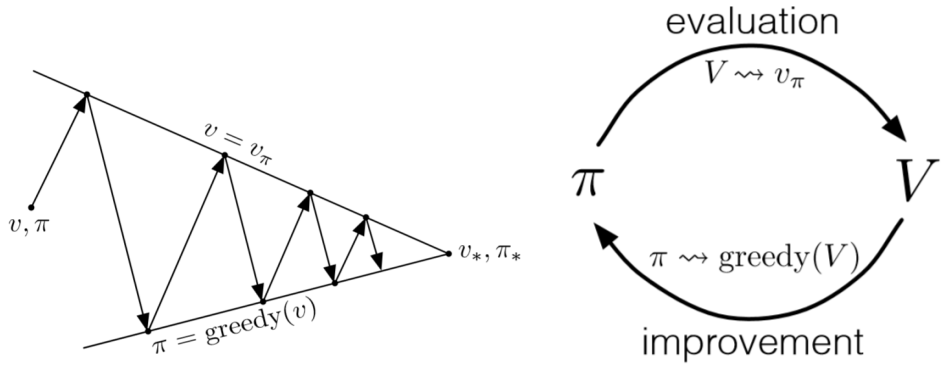

Figure 3.5: GPI process

source: Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, pag.86

find a single solution (optimal policy, optimal value function). The optimal point is found if both processes stabilize, so when no more changes are produced.

These algorithms can be classified into two main categories: *model-based algorithms and model-free algorithms.*

Model-based, are used for planning. In fact they are used to decide the goodness of an action, they consider possible future situations before actually being experienced. In general, during a model-based algorithm, the agent can potentially predict the dynamics of the environment, since it is an estimate of the transition function (and the reward function).

Model-free algorithms, estimate the optimal policy without using or estimating the dynamics (transition and reward functions) of the environment. In practice, a model-free algorithm estimates the "value function" or the"policy" directly from experience.

Regarding these two models, V. Pong, M. Dalal, S. Levine and S. Gu [24] report: *"Model-based algorithms tend to be substantially more efficient, but often at the cost of larger asymptotic bias: when the dynamics cannot be learned perfectly, as is the case for most complex problems, the final policy can be highly sub-optimal. Therefore, conventional wisdom holds that model-free methods are less efficient but achieve the best asymptotic perfor-*

mance, while model-based methods are more efficient but do not produce policies that are as optimal".

3.5 Dynamic Programming

The term "dynamic programming", Bellman [2], refers to a collection of algorithms that are used to calculate optimal policies in a MDP, using optimal value functions. These methods require complete knowledge of the environment: reward functions, value functions etc. The way to find optimal policies is through the policy iteration (PI), Howard [16].

3.5.1 Policy evaluation

The main idea here is to predict the expected total reward from any state, assuming that $\pi(s)$ given. Considering a sequence of functions of approximate value: v_0, v_1, v_2.., during which the initial value is arbitrarily chosen. v_0 and subsequent values are obtained using the Bellman's equation for v (3.8) as updating rule:

$$v_{k+1}(s) \doteq E_\pi[R_{t+1} + \gamma\, v_k(S_{t+1}|S_t = s)],\ \forall s \in S \tag{3.15}$$

Where the sequence v_k converges to v_π as $k \to \infty$.

The equation (3.15) is called "iterative policy evaluation": in each iteration, the algorithm updates the value of each state to provide new approximate value functions, v_{k+1}.

This algorithm is described in [37] and reported in alg.1.

Algorithm 1: Iterative policy evaluation for estimating V $\approx v_\pi$

Input: π, the policy to be evaluated;

Parameter: $\theta > 0$, a threshold to determine the accuracy of the estimation;

Initialize $V(s), \forall s \in S$

repeat
- $\triangle \leftarrow 0$
- **for** $s \in S$ **do**
  - $v \leftarrow V(s)$
  - $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} p(s', r|s, a)[r + \gamma V(s')]$
  - $\triangle \leftarrow \max(\triangle, |v - V(s)|)$

until $\triangle < \theta$;

3.5.2 Policy improvement

We know how is good to follow current policy, but *does it make sense to deviate from $\pi(s)$ at any state?*

In practice, if $q_\pi(s, \pi'(s)) \geq v_\pi(s)$, then policy $\pi(s)'$ must obtain a greater or equal expected return from all the states: $v_{\pi'}(s) \geq v_\pi(s)$.

If the inequality is strict, policy $\pi'(s)$ is better than $\pi(s)$. This is often called *"policy improvement theorem"* (Howard, [16]). The new greedy policy is given by:

$$\begin{aligned} \pi'(s) \doteq & argmax_a q_\pi(s,a) = \\ = & argmax_a E[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \end{aligned} \tag{3.16}$$

The greedy policy takes action that looks better in the short term in line with v_π, except when the original policy is already optimal.

3.5.3 Policy Iteration

Once π has been improved, using v_π to provide a better policy π, it is possible to calculate $v_{\pi'}$ and improve the whole process again. By combining policy evaluation and policy improvement, an improved sequence of policies and value functions is obtained:

$$\pi_0 \vec{E}\, v_{\pi_0} \vec{I}\, \pi_1 \vec{E}\, v_{\pi_1} \vec{I}\, \pi_2 \vec{E}\, v_{\pi_2} \vec{I} ... \vec{I}\, \pi^* \vec{E} v^*$$

where $\vec{E}$ stands for the evaluation process and $\vec{I}$ stands for the improvement process. From the policy improvement theorem, *each policy is destined to be a rigorous improvement over the previous one (unless it is already optimal).*

Sutton and Barto [37], present the entire algorithm (see 2).

Algorithm 2: Policy iteration for estimating $V \approx v_\pi$

1. Initialization: $V(s) \in \mathbb{R}$ and $\pi(s) \in A(s) \forall s \in S$;
2. Policy evaluation (cfr.alg.1);
3. Policy improvement;

```
repeat
    foreach s ∈ S: do
        oldaction ← π(s);
        π(s) ← argmax_a Σ_{s',r} p(s', r|s, a) + [r + γV(s')];
        If oldaction ≠ π(s), then π(s)stable ← false
until π(s)stable ← true;
```

3.5.4 Value iteration

One of the disadvantages of the policy iteration is that it is an iterative algorithm. For this reason, we must wait for it to converge. As previously seen, figure 3.5, the most cycle contains two steps: one iterative algorithm within another.

This process may take an extended time before the value function converges. It is reasonable to ask: *there is some point before convergence, such that the resulting greedy policy doesn't change?*

Indeed, some studies have shown that after just a couple of policy evaluation iterations, the resulting greedy policy is constant.

The algorithm to be considered is called *value iteration*. It combines policy evaluation and policy improvement in one step. The first step is to initialize V arbitrarily. Next, we enter the cycle and keep track of the most important change.

The cycle is interrupted when the most important change falls below some threshold, indicating convergence.

Algorithm 3: Value iteration for estimating $\pi \approx \pi*$

Parameter: $\theta > 0$, a threshold for determining the accuracy of the estimation;

Initialize $V(s) \forall s \in S$;

Output: $\pi \approx \pi*$;

repeat

  foreach $s \in S$: **do**

    $v \leftarrow V(s)$;

    $V(s) \leftarrow max_a \sum_{s',r} p(s', r|s, a) + [r + \gamma V(s')]$;

    $\triangle \leftarrow max(\triangle, |v - V(s)|)$;

until $\triangle < \theta$;

3.6 Monte-Carlo

The term Monte-Carlo (Stanislaw Ulam, 1946), usually refers to any algorithm involving a large random component. Within the reinforcement learning methods, the random component is the return.

What we always seek is the expected return from a certain state. By using the Monte-Carlo method, instead of calculating the real expected value, which requires probability distributions, this method use the sample mean. Therefore, one must assume that there are only completing episodic tasks. The rationale is that an episode must end before returns are calculated.

This also means that the Monte-Carlo method is not completely an online algorithms. In fact, the algorithm doesn't perform an update after every action but after every episode. Episodes are sequences of states, actions and rewards. Each episode is described as a triple: (S_t, A_t, R_t) and has to be complete. From these episodes, values are estimated.

For this reason, the MC method is a method that estimates values directly from experience. To learn the optimal policy, an idea similar to GPI is followed.

3.6.1 Policy evaluation

In the prediction step, the goal is to find an estimate value function of an unknown MDP. Hence, there is a need to learn v_π from experience under the policy π , meaning that we don't use a model to compute the value of each state but average returns that start in that state.

Recall that return is the total discounted reward, G_t (equation 3.7) and the expected return is $v_\pi(s)$ (equation 3.8).

MC method uses empirical mean return instead of expected return be-

cause of the unknowledge of the environment.

A state value function is calculated by returning to the same state multiple times and averaging the actual cumulative reward received each time.

But what if we encounter an equivalent state more than once in an episode? What will be the return for that state?

There are two methods of calculating the return for a state and both attend an equivalent response.

The first method is called *"first visit"*, during which the average performance is calculated only for the first time the state is visited in an episode (unbiased estimate); the second method is called *"each visit"*, during which the average performance is calculated each time the state is visited in an episode; (biased but consistent estimate). Usually the first method is the most used.

S. Sutton and A.G. Barto [37] illustrate this method at pag.92:

Algorithm 4: First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated;

Initialize: $V(s) \forall s \in S$;

Returns(s) $\leftarrow$ an empty list $\forall s \in S$;

1 **repeat**

  Generate an episode following $\pi : S_0, A_0, R_1, S_1, A_1, R_2 ... S_{t-1}, A_{t-1}, R_t$;

  G $\leftarrow 0$;

2 **foreach** *episode t = T-1,T-2..* **do**

3 $G \leftarrow \gamma G + R_{t+1}$;

until S_t *appears in* $S_0, S_1, .., S_{t-1}$;

Append G to Returns(s);

$V(S_t) \leftarrow average(Returns(S_t))$

By following the "first visit" method, $V(s)$ is the empirical mean return of the first occurrence in the state s. It is updated incrementally[6] after each

[6]Incremental mean: the mean $\mu_1, \mu_2..$ of a sequence $x1, x2..$ can be computed incrementally: $\mu_k = \frac{1}{k} \cdot \sum_{j=1}^{k} x_j$

episode:

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)} \cdot (G_t - V(S_t)) \tag{3.17}$$

where

$$N(S_t) \leftarrow N(S_t) + 1 \tag{3.18}$$

is the *"increment counter"*, that is the number of visit in each state.

Notice that the increment in $V(S_t)$ is determined at the end of the episode when G_t is known. The target for MC update is G_t.

3.6.2 Policy improvement

Without a model, it is impossible to derive an optimal policy, π^*, from the optimal expected return V^*.

In fact, the value function is indexed only for S. Therefore, we do not know which actions will lead to a better policy. It is useful to estimate the action-value functions rather than the state-value functions.

$Q(S,A)$ is the average return (here empirical mean return) when starting in state s, taking action a and following the policy π. The updating rule for $Q(S,A)$:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \frac{1}{N(S_t)} \cdot (G_t - Q(S_t,A_t)), \vee s \in S, \vee a \in A \tag{3.19}$$

Once having the action-value function, it is possible to compute greedily the optimal policy:

$$\pi^*(s) = argmax_a Q(S,A) \tag{3.20}$$

if we act greedily we end up choosing one action forever.

To prevent the agent to ending up in a non-optimal policy and ensuring continuous exploration, the greedy policy is performed.

The idea is that epsilon, ε, is used to guarantee that each action has the possibility to be selected.

All m actions are tried with non-zero probability, the greedy action is performed with a probability of (1-ε) and a random action is performed with a probability ε of exploring the action space:

$$\begin{cases} \frac{\varepsilon}{m} + 1 - \varepsilon, & \text{is } a^* = argmax_a Q(S,A) \\ \frac{\varepsilon}{m}, & \text{otherwise} \end{cases} \tag{3.21}$$

The GPI of the Monte Carlo algorithm can be summarized as:

$$\pi_0 \vec{E} q_{\pi_0} \vec{I} \pi_1 \vec{E} q_{\pi_1} \vec{I} \pi_2 \vec{E} q_{\pi_2} \vec{I} ... \vec{I} \pi^* \vec{E} q^* \tag{3.22}$$

One of the main problems with using $Q(S,A)$ is that, as discussed in section 3.5.4, given the presence of many values to consider when updating $Q(S,A)$, the policy iteration algorithm may be inefficient.

The solution is again to adopt the same approach of the value iteration (3.5.4). In this way, we continue to update the same $Q(S,A)$, and to improve policies after every single episode.

3.7 Temporal difference

The Temporal Difference Learning, or simply TD, is one of the most important learning techniques.

It is a combination of Monte-Carlo and Dynamic Programming. Like MC, it can learn from experience, without a model of the environment. Like

DP, it updates estimates on the basis of other learned estimates, without waiting for an episode to end (bootstrapping).

In this section, first the prediction problem will be examined and next the control problem will be viewed. In the control problem two different ways of dealing with it, *Sarsa* and *Q-Learning*, will be explored.

3.7.1 Policy evaluation

TD(0)[7] method is one of the simplest TD methods. In these methods, the true expected return G_t is not available, as in MC, and it is approximated with the TD-target.

The TD-target is an approximation of equation 3.14. It is an estimate of the return and it can be derived by taking the sum of the observed return and the discounted expected value of the next state: $R_{t+1} + \lambda V(S_{t+1})$.

The updating-rule becomes:

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \lambda V(S_{t+1} - V(S_t)), \vee s \in S \tag{3.23}$$

where α is a constant step-size parameter(learning rate); $(R_{t+1} + \lambda V(S_{t+1} - V(S_t))$ is an error, called TD-error.

The TD-error, γ_t, measures the difference between the TD-target and estimated values of $V(St)$.

[7]There are also other methods: TD(1), TD(λ), that we shall not discuss here.

The algorithm described below is taken in Sutton and Barto [37]:

Algorithm 5: TD(0) for estimating v_π

Input: a policy π to be evaluated;

Initialize: $V(s) \forall s \in S$;

repeat

  foreach *episode t = T-1,T-2..* **do**

    A $\leftarrow$ action given by π for S;

    Take action A, observe R, S';

    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$;

    $S \leftarrow S'$;

until *S is terminal*;

TD(0) under batch updating converges to optimality [37], finding Maximum Likelihood estimates.

In general, one can say that batch TD(0) converges to the *certainty-equivalence estimate* (Goodwin and Sin, [14]).

3.7.2 Policy improvement

As for Monte Carlo, this method uses the action-value function, because it allows to choose an action based on the *argmax* of Q.

Thus, the first step, as in MC, is to learn the action-value function because it is not possible to derive an optimal policy, π^* from the optimal expected return V^*.

The updating rule is the same as the one of $V(s)$ (equation 3.23):

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)), \vee s \in S, \vee a \in A \quad (3.24)$$

To improve the policy, there are two different algorithms that one can use: *Sarsa* and *Q-Learning*.

Sarsa (State-Action-Reward-State-Action), is an online learning method, meaning that it uses the same policy to choose the next action.

It can be described with the quintuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$.

The algorithm is illustrated in 7.

Algorithm 6: Sarsa (on-policy TD control) for estimating $Q \approx q^*$

Initialize: $Q(s,a) \forall s \in S$, $\forall a \in A$;

repeat

  foreach *episode t = T-1,T-2..* **do**

    A $\leftarrow$ action given by π for S;

    Take action A, observe R, S';

    Choose A' from S' using policy from Q (e.g ε-greedy);

    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$;

    $S \leftarrow S'; A \leftarrow A'$;

until *S is terminal*;

To better understand: the agent starts in an initial state (S1), acts (A1) and gets a reward (R1), then it goes into a new state (S2). In the new state (S2) it performs a new action (A2), then it updates the first action (A1) performed in S1.

Q-Learning (Watkins, 1989), is an off-policy method, meaning that it uses the greedy policy to choose the best successive action. This method estimates the action-value function that selects the action of the highest value, equation 3.24 becomes:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \, max_a Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)), \vee s \in S, \vee a \in A \tag{3.25}$$

And the algorithm is:

Algorithm 7: Q-learning (off-policy TD control) for estimating $\pi \approx \pi^*$

Initialize: $Q(s,a) \forall s \in S$, $\forall a \in A$;

repeat

  foreach *episode t = T-1,T-2..* **do**

    A $\leftarrow$ action given by π for S;

    Take action A, observe R,S';

    Choose A' from S' using policy from Q (e.g ε-greedy);

    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma max_a Q(S',A') - Q(S,A)]$;

    $S \leftarrow S'$;

until *S is terminal*;

The agent starts in an initial state (S1), acts (A1) and gets a reward (R1), then it goes into a new state (S2). In the new state (S2) it looks for the maximum reward and then it updates the first action (A1) performed in S1.

3.8 Summary

In section 3.2, the Markov decision-making process was defined. In the following sections of this chapter, the main methods for finding solutions to this process have been examined.

The first method examined was that of Dynamic Programming which tries to find the value function given to a policy.

Two ways to look for optimal policy and optimal value function have been examined. The first is called policy iteration and involves an iterative algorithm within another iterative algorithm. This could be inefficient.

Subsequently, another algorithm was considered (value iteration algorithm).

It allows to "truncate" the policy evaluation phase and combine this with

the policy improvement phase in a single step. A known feature is that the Dynamic Programming model requires a complete model of the environment, in particular, the transition probabilities of the state. In the real world these probabilities may not be measurable, especially when there is an oversized number of states. Another feature of this method is that an initial estimate is required (bootstrapping). Both, for policy iteration and value iteration, estimates from other estimates are calculated, going back and forward between the value function and the policy.

The Monte-Carlo method was subsequently examined.

The MC method is able to learn from experience. The main assumption made is that the value function is the expected return given a state. By probability theory, the expected value of something can be approximated by a sample mean.

To use Monte Carlo, there is a requirement to generate episodes and then to calculate returns for every episode. In this way we have an inventory of pairs of states and returns and we can, therefore, calculate the average of these returns to find the value for each state.

There are two methods for average returns: the *first visit* method and the *every visit* method.

It has been shown that both converge on an equivalent result.

Usually, the first visit method is the most used because it is simpler. The main difference with the previous method, Dynamic Programming, is that MC uses $Q(S,A)$ rather than $V(S)$, given the fact that a model for the environment is not present.

Another peculiarity of this method is that, it needs many samples to get accurate estimates.

The policy iteration makes it very inefficient, within the sense that it might take too long time in terms of computation. Here, too, the value iteration approach is adopted. Morever, another thing that has been said

during this paragraph is that, in order to obtain a complete measurement of $Q(S,A)$, it is necessary to use the epsilon-greedy strategy. This enables to continuously explore all states within the state space.

The last section deals with the Temporal-Difference method. TD method has been reported in this section. This method, as has been seen, combines aspects of both MC and DP.

To use TD, instead of taking the sample mean of returns, we take the sample mean of the estimated return, which is based on the current reward, and the next state value.

For the control problem, as in MC, the action value function is used instead of the state value function. There are two methods for dealing with the control problem: SARSA (online method) and Q-Learning (offline method).

In this thesis a Dynamic Programming approach will be used, in particular that of value iteration, for the construction of the dynamic pricing algorithm.

CHAPTER 4

PROBLEM DESCRIPTION

This chapter, investigate the problem of dynamic pricing in an online marketplace under competition. The simulations of the market were made by participating in a competition (Dynamic Pricing Competition, Haensel AMS).

As the authors said [40], the competition was designed to resemble an online sales market. Competitors have no a priori information on the sales mechanism or on the behavior of the other competitors.

During the competition it is required to publish a price before each (discrete) time period. Participants had to design an algorithm that would accept their sales and historical prices of all competitors as input, and then the algorithm would return their price for the period ahead.

4.1 Scenario

The competition scenario involved the sale of a product in a time period of $T = 1000$, indexed by $t = 0, 1, 2..T$. The merchant compete in each period with n competitors. In each time period $t \in 0, 1..T$, the merchant could set price $p_{i,t} \in 0, 0.001, 0.002..100$ and no costs were involved. After each period, sales, $s_{i,t}$ are generated.

The merchant could observe his sales in $t-1$ and the prices of competitors in $t-1$.

Once sales are generated, merchant earns a revenue, $p_{i,t}s_{i,t}$.

A difficulty is that merchants do not know the customer behavior. Consumers have unknown characteristics, but we know that there is a sort of horizontal differentiation in the market. The objective is to find pricing policies that maximize the expected profit.

The mechanism of the simulations is represented in the figure below:

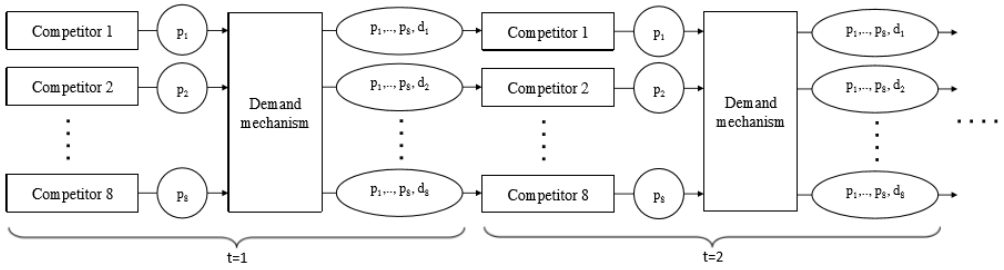

Figure 4.1: Overview of one simulation

source: [40] Ruben van de Geer and Arnoud V. den Boer et al. (2019)

4.2 Sales generating mechanism

In this competition, sales are generated by an unknown mechanism. The sales generation mechanism is built in a way that resembles a competitive market.

Unfortunately the above mechanism remained unknown even after the end of the competition.

Despite this, this section reports the sales generating mechanism built during the competition that took place in 2017.

This will allow us to have an idea on the methodology with which the sales processes are built within these kind of competitions. Obviously, the process that will be described below will be different from what we found in this year's competition.

As reported in [40], an arrival process and a demand mechanism have been assumed time-independent within a simulation. In addition, a market consisting of three segments was inferred: loyal customers, shoppers and scientists. Each of them has its own parameterized demand functions.

In this competition it was assumed that Poisson customer arrivals over time with average arrivals over time period equal $\lambda \sim U(50, 150)$.

The arrival process - which is described in [40] at page 9 - is reported below:

Algorithm 8: Arrival process

foreach $i \in \{1, ..., 500\}$ **do**
    Sample arrival rate $\lambda \sim U(50, 150)$;
    Sample segment shares $\theta_i^{sho}, \theta_i^{loy}, \theta_i^{sci}$;
    Sample subsegment shares $\phi_i^{phd}, \phi_i^{prof}$;
foreach $t \in \{1,..1000\}$ **do**
    Sample arrivals $n \sim Poisson(\lambda_i)$;
    Sample segment arrivals $n^{sho}, n^{loy}, n^{sci} \sim Multinom(n, (\theta_i^{sho}, \theta_i^{loy}, \theta_i^{sci}))$;
    Sample subsegment arrivals
    $n^{phd}, n^{prof} \sim Multinom(n^{sci}, (\phi_i^{phd}, \phi_i^{prof}))$;

During the competition in 2017, scientists were divided into two subsegments: PHDs and professors. Each of them with respective shares ϕ_i^{phd} and ϕ_i^{prof}.

As reported from the authors, the WTP of shoppers was assumed to be exponentially distributed with mean $\beta_i^{sho} \sim U(5,15)$. In each time period the organizers sampled a WTP for each arriving shopper from the exponential distribution with mean β_i^{sho}. Then, the WTPs were compared to the lowest price offered in the market. Each shopper for whose WTP had exceeded the lowest price offered, bought from the competitor that offered the lowest price and otherwise left without buying anything.

For the loyal customers, they assumed their WTP is exponentially distributed but with mean $\beta_i^{loy} = \mu \cdot \beta_i^{sho}$ where $\mu \sim U(1.5,2.0)$.

In this way the loyal customers were relatively price-insensitive compared to the others. In each simulations, a loyal customer was assigned randomly to a competitor. Then, the organizers sampled for all loyal customers a WTP and compared this to the price offered by the competitor that they were loyal to. In case their WTP exceeded the price offered, the loyal customers buy, otherwise they leave without buying anything.

Instead, it was assumed that the scientists' demand follow a finite mixture logit model, in which the mixture included professors and graduate students. The probability that a PhD buyer purchased from competitor k was:

$$q_k^{phd} = \frac{exp(\lambda_i^{phd} - \beta_{n,i}^{phd} \cdot p_{k,t})}{1 + \sum_{j=1}^{n} exp(\lambda_i^{phd} - \beta_{n,i}^{phd} \cdot p_{j,t})}$$

To be realistic, they set λ_i^{phd} equal to a β_i^{sho}, to have optimal prices for PHD's within 50% of optimal prices for shoppers. This was achieved throughout:

$$\lambda_i^{phd} = \beta_i^{sho}$$

$$p_i^{phd} := \beta_i^{sho} \cdot \mu, \mu \sim U(0.5,1.5)$$

$$\beta_{n,i}^{phd} = \frac{W(ne^{\lambda_i^{phd}-1}+1)}{p_i^{phd}}$$

Where W is the Lambert function (i.e $W(xe^x) = x$). The parameters are set as:

$$argmax_{p\in\mathbb{R}^+} \sum_k p_k q_k^{phd}(p) = (1)_n p_i^{phd}$$

This means that if the market consisted only of PHDs, then overall revenue was maximized if all competitors was set p_i^{phd}.

$(1)_n p_i^{phd}$ was an n-vector with each element equal to p_i^{phd}.

Professors also had a similar probability.As reported at page 11, they also set:

$$\lambda_i^{prof} = \lambda_i^{phd} \cdot \mu, \mu \sim U(1, 1.25)$$
$$p_i^{prof} := p_i^{phd} \cdot \mu, \mu \sim U(1, 1.5)$$
$$\beta_{n,i}^{prof} = \frac{W(ne^{\lambda_i^{prof}-1}+1)}{p_i^{prof}}$$

In this sense, a market consisting only on professors was higher than a market consisting only of PHDs.

By summarizing, in all the simulations, in expectation, there were: $\lambda_i \cdot \theta_i^{loy}$ arriving loyals; $\lambda_i \cdot \theta_i^{sci} \cdot \phi_i^{phd}$ arriving PHDs and $\lambda_i \cdot \theta_i^{sci} \cdot \phi_i^{prof}$ arriving professors; $\lambda_i \cdot \theta_i^{sho}$ arriving shoppers. Each of these customer chosen according to its own demand function.

The sales-generating mechanism proposed in [40], is a good example for understanding better the complex system behind the competition structure.

CHAPTER 5

EXPERIMENT DESIGN

This chapter presents the model built for the competition. This model aims to solve the dynamic pricing problem.

To estimate the demand an approach presented by R. Schlosser and M. Boissier [30] was followed. The demand is estimated by using a polynomial regression model trained on historical data. The estimated demand is then used in Dynamic Programming to generate optimized pricing.

The first section 5.1 describes the elaboration of the elements necessary for the formalization of the pricing problem: sales probabilities, and estimated average sales.

In the next section the problem is formalized as an MDP and then the implementation process is explained.

At last, some time improvements are made in the last section.

5.1 Demand learning

The first step in this analysis regards the analysis of the historical data. As mentioned in section 2, the exploration of historical data permit to the firm to find out many details and information.

This knowledge allows decision-makers to make the right pricing decision.

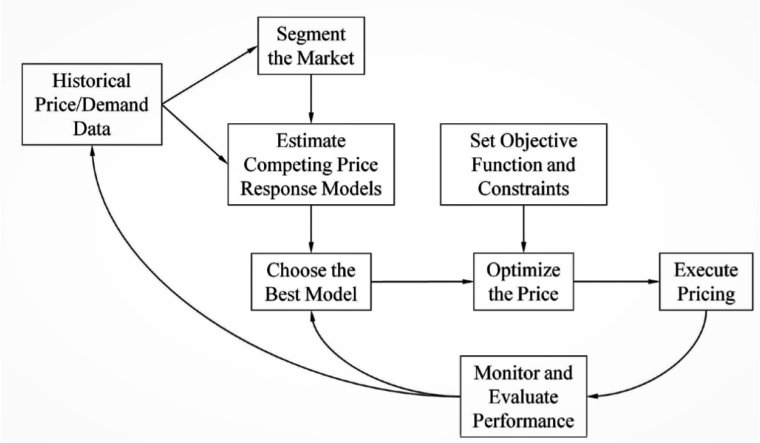

Figure 5.1: Pricing analytics process

source: Pricing segmentation and analytics, Bodea T. and Ferguson M., pag.4

As illustrated in the figure above (5.1), through historical data it is possible to understand how much demand occurred in each period.

Essentially, this permit to extract data on customers, that usually are unknown.

Firms, tend to not have data on customers' behavior unless they used surveys, consumers' research, particular metrics or indexes, or, as in this case, historical data.

One of the most important assumptions that one can make, is about the WTP (willingness-to-pay), as mentioned in T. Bodea and M. Ferguson [5].

WTP is the maximum amount of payment that the consumer will pay for a product or a service. Knowledge about WTP plays a crucial role in the market. As Breidert C. et al. [6] report, missing adequate knowledge of

WTP comports for companies the failure in pursuing pricing strategies.

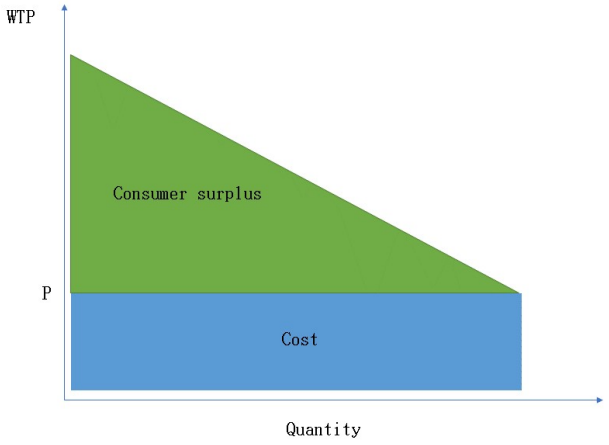

Figure 5.2: Customer's willingness to pay
source: `https://careonline.it`

The WTP varies across many factors, such as personal preferences, the state of the economy, the quality of the product/service, the rareness of the product etc.

5.1.1 Regression model

Through historical data it is possible to estimate the demand function. Our demand function is described by a polynomial regression.

Also, the polynomial regression tries to simulate the sales generating mechanism behavior.

To build this model, in the first 150 periods, a random price from a uniform discrete distribution is set. By using a uniform discrete distribution in the range [1,100], one can assign the same probability to be extracted to all the prices within the range. In fact, the probability that a price is draws out is $1/n$.

The choice of a uniform discrete distribution, permit to collect more information (cf. exploration part 2.4) by selecting randomly all the possible

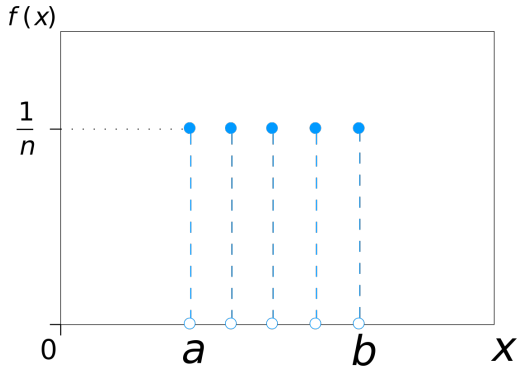

Figure 5.3: Uniform discrete distribution with a=1 and b=100

prices. In fact, in this way the algorithm is able to recognize better which prices are good for the simulation and which not.

After 150 periods, the polynomial regression model is trained. We use 150 historical data as the first training set, then, adding periods, we increase the training set time by time.

With the available data -history of competitors prices, history of our sales data and history of our prices- following the approach proposed by R. Schlosser and M. Boissier [30], we build the explanatory variables as follow:

1. x_0: intercept;
2. x_1: our price;
3. x_2: $min(p_{1t}, p_{2t}, p_{3t}..p_{nt})$, minimum of competitors prices;
4. x_3: $mean(p_{1t}, p_{2t}, p_{3t}..p_{nt})$, mean of competitors prices;
5. x_4: $max(p_{1t}, p_{2t}, p_{3t}..p_{nt})$, maximum of competitors' prices;
6. $x_2 \cdot x_3$, interaction between minimum and mean of competitors' prices;
7. $x_2 \cdot x_3$, interaction between minimum and maximum of competitors' prices;
8. $x_4 \cdot x_3$, interaction between mean and maximum of competitors' prices;

Given the four regressors, the polynomial regression model becomes:

$$\hat{Y}_{sales}(a, \vec{p}_t, \vec{\beta}) = \hat{\beta}_0 + \hat{\beta}_1 x_1^3 + \hat{\beta}_2 x_2^3 + \hat{\beta}_3 x_3^3 + \hat{\beta}_4 x_2 \cdot x_3 + \hat{\beta}_5 x_2 \cdot x_4 + \hat{\beta}_6 x_3 \cdot x_4 \quad (5.1)$$

where $\hat{Y}_{sales}$ represents the mean sales per period, $\vec{p}$ describes the market situation (i.e. competitors' prices) and $a = [1;100]$ is our price.

Within the model, the optimal beta's are estimated thanks to the following formula:

$$\hat{\beta}^* = (X^T X)^{-1} X^T Y \quad (5.2)$$

Each period has a different estimate of betas because, period by period, more data are included in the training set. As a matter of fact, in any period, the last T-2 historical data are used to train the model, while the data in the last period (i.e competitors' prices) are used as variables to fit the model 5.1.

The use of a regression model, as mentioned in [30], permit to consider the following effects:

- Customer buying behavior;
- Customer arrivals;
- Competitor pricing strategies;

The choice of a regression model instead of another type (such as decision tree, XGBoosting, Neural Network, etc.) was dictated by the time requirements of the competition.

For this competition it was necessary to develop an algorithm that would provide the answer in 0.1 seconds.

A regression model was a good compromise between accuracy and timing.

Once selecting a regression model, another important aspect that we considered is the selection of a linear vs one non-linear. The preference for

a polynomial regression instead of a linear regression is due to the desire to capture non-linear behavior within the data. The possible non-linear behavior is due to the presence of a degree of horizontal differentiation within the market.

From the problem statement, section 4, we know that there is a degree of horizontal differentiation in the market.

The presence of horizontal differentiation comports that at the same price some customers will buy one product and other customers will buy other products. This is due to the unknown preferences of the consumers.

If we had used linear regression, we had supposed that a decrease in price could comport an increase in sales and vice-versa, figure 5.4, but the horizontal differentiation implies that not always smaller prices have higher sales or higher prices have smaller sales. It really depends on the customers' preferences.

Figure 5.4: Linear demand function

In this case, the polynomial regression permits to fit better the data, by catching possible non-linear trends. Furthermore, a degree 3 polynomial was chosen to increase the flexibility of the model, while maintaining a certain logic. As G. James et al. said: [13], *"Generally speaking, it is unusual to use d greater than 3 or 4 because for large values of d, the polynomial curve can become overly flexible and can take on some very*

strange shapes".

5.1.2 Sales probabilities

The regression model permit also to predict sales probabilities. Indeed, mean sales per period can be used as the mean of a probability distribution.

The probability distribution used in this experiment for describing the demand distribution is the Poisson distribution (cfr.[30]).

The Poisson distribution enables us to find the sales probability in each period. It is also considered a good distribution for representing the customers' arrival process [42].

For each period, so for each market situation $\vec{p}$ the sales probabilities are formalized as:

$$P(i,a|\vec{p},\vec{\beta}) = e^{-\hat{Y}_{sales}(a,\vec{p},\vec{\beta}^*)} \cdot \frac{\hat{Y}_{sales}(a,\vec{p},\vec{\beta}^*)^i}{i!} \tag{5.3}$$

where $i = [0,100]$ [8] is the number of maximum sales per period. Given this formulation one can observe that for each price there will be 100 different sales probabilities (one for each k, and for each price there will be 100x100 different sales probabilities.

As mentioned at the beginning of this chapter the sales probabilities are important in this experiment because allow us to use the Dynamic Programming approach.

[8] The number of sales was decided with respect to the simulations we participated in, before finalizing the algorithm. It was noted that on average the sales made are around 50-60, so it was decided to consider a maximum of 100 sales.

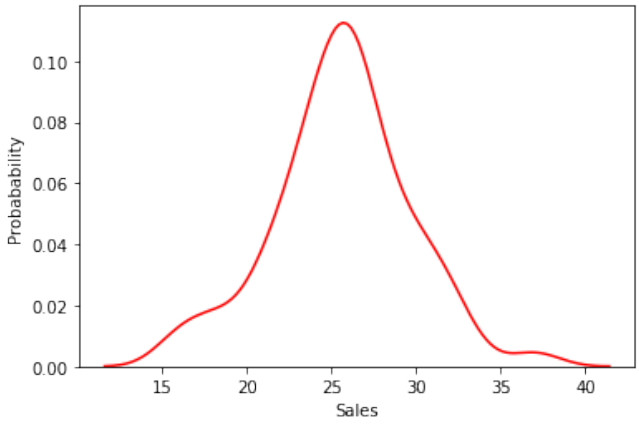

Figure 5.5: Sales probabilities for $a = 25$

5.2 Model formulation of Dynamic Pricing problem

"The dynamic pricing problem can be formulated as a MDP because pricing is a real-time decision-making problem in a stochastic environment". R. Rana, F. Oliveira [26].

To use a Dynamic Programming approach our problem has to be formalize as a MDP. Having said this, in our model the components are:

- The state space: $\vec{S}$, is represented by the possible sales for each period;

  $\vec{S} = \{0, 1, 2, 3..100\}$;

- Time horizon: $t \in T = 0, 1, 2...1000$;
- Actions: A_{it} is the set of prices the merchant can choose when he is in the state i at time t;
- Sales probabilities: $P_i(i, a|\vec{p}_t, \vec{\beta})$, is the probability to sell i items at price a given the market situation $\vec{p}$ at time t;
- Reward: $R(a|\vec{p}_t, \vec{\beta})$ is the profit of the merchant, gained for executing price a;
- Transition probability: $P_{s,s'}(s, s', a|\vec{p}_t, \vec{\beta})$, is the probability to shift from state i to state j at time t given the action a;

Because of the randomness (competitors behavior is not predicted in this experiment), the merchant pricing decision is always influenced by the market situation. The pricing decisions at $t \geq 0$ can depend on $\vec{p}$.

Optimal pricing policies are found using the Dynamic Programming approach described in section 3:

$$V(s) = max_{a \in A} \sum_{s=0}^{100} [P_{s,s'}(s,s',a|\vec{p}_t,\vec{\beta}) + (a \cdot \hat{Y}_{sales}(a,\vec{p}_t,\vec{\beta})) + \gamma V(s')] \tag{5.4}$$

s	$a^*(s)$
0	99
1	96
2	95
3	93
4	91
5	88
..	..
99	43
100	52

Table 5.1: Example of optimal pricing policies for each state

Thanks to the above equation 5.4, it is possible to find the optimal pricing decision (cf. tab.5.1) for each state as follows:

$$a^*(s) = argmax_{a \in A} \sum_{s=0}^{100} [P_{s,s'}(s,s',a|\vec{p}_t,\vec{\beta}) + (a \cdot \hat{Y}_{sales}(a,\vec{p}_t,\vec{\beta})) + \gamma V(s')] \tag{5.5}$$

At the end, the action that obtains the greatest reward (revenue) will be chosen.

5.2.1 Step-by-step process

In each period the expected sales and the sales probabilities are estimated.

If one is in one state, all possible actions (prices) for that state will be evaluated.

Using regression (5.1) allows to have as many expected sales as there are prices. In fact, by changing the price, the expected sales will modify.

The price represents one of the variables in the model, x_1. Each price, x_1, will correspond to a different expected sale while the competitors' prices, x_2, x_3, x_4 will remain fixed to the previous period.

Moreover, at each price corresponds a different reward (profit), which will be given by the price multiplied by the expected sale for that price: $a_i \cdot \hat{y}_i$. Furthermore, each price will coincide to a probability of sale, as these are calculated through the expected sales, as mentioned in 5.3.

To be clear, a numerical example is reported in table 5.2.

As can be seen from the table, there are a hundred of prices. If one take $a_{10t} = 1$, the expected sales $\hat{y}_{sales} = 4$ are calculated with: $\hat{\beta}_0 + \hat{\beta}_1(1)^3 + \hat{\beta}_2 x_2^3 + \hat{\beta}_3 x_3^3 + \hat{\beta}4\, x_4^3 + \hat{\beta}_3 x_2 \cdot x_3 + \hat{\beta}_5 x_2 \cdot x_4 + \hat{\beta}_6 x_4 \cdot x_3$; the reward $R_{10t} = 4$ is calculated through: $a_{10t} = 1 \cdot \hat{y}_{sales} = 4$ and the sales probability $P(10, a_{1t}, \vec{p}, \vec{\beta}) = 0.0073$ is derived from: $e^{-4}\,\frac{4^{10}}{10!}$.

State 10			
a_{10t}	$\hat{y}_{sales}$	$R_{10}(a, \vec{p}_t, \vec{\beta})$	$P(10, a_{10t} \mid \vec{p}, \vec{\beta})$
price:1	4	4	0.00529
price:2	8	16	0.09926
price:3	11	33	0.11938
price:..	..	..	..
price:100	2	200	$3.8189e^{-5}$

Table 5.2: Numerical example for state 10

To pass from one state to another, transition probabilities (cf. 3.2) have to be introduced.

The transition probability from a state i to a state j is defined as:

$p_{ij} = P\{j|i\}$.

The equation for p_{ij} reminds to the Bayes theorem.

Mathematically, given two events A and B the Bayes' rule provides the connection between $P(A)$ and $P(B)$ and the conditional probability of A given B and viceversa.

The common form is:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)} \tag{5.6}$$

In this way, it is possible to calculate the probability of transition, given a price, between one state and another by using the Bayes theorem. From what written up to now, with reference to equations 5.3 and 5.6, the transition probability from state s to state s', in our experiment, can be described as:

$$\begin{aligned} P_{s,s'} = P[(s,a|\vec{p},\vec{\beta})|(s',a|\vec{p},\vec{\beta})] = \\ = \frac{P[(s,a|\vec{p},\vec{\beta})|(s',a|\vec{p},\vec{\beta})] \cdot P(s')}{P(s)} = \\ = \frac{P[(s,a|\vec{p},\vec{\beta})|(s',a|\vec{p},\vec{\beta})] \cdot e^{-\hat{Y}_{sales}(a,\vec{p},\vec{\beta})} \frac{\hat{Y}_{sales}(a,\vec{p},\vec{\beta})^{s'}}{s'!}}{e^{-\hat{Y}_{sales}(a,\vec{p},\vec{\beta})} \frac{\hat{Y}_{sales}(a,\vec{p},\vec{\beta})^{s}}{s!}} \end{aligned} \tag{5.7}$$

For simplicity, only the probabilities of passing from one state to another forward and then back are considered, but the probabilities of passing from one state to all the others and back are never considered. This means that, one can pass from state 0 to state 1 and from state 1 to state 0, but not from state 0 to state 2 and viceversa.

To be clear, if the agent is in the situation in which have 0 sales it can only calculate the probability to be in that state and to shift from that state to the state in which it will have 1 sales.

This is a strong assumption. Either way, it was necessary to assume so for three reasons:

1. Curse of dimensionality;
2. Timeliness problem;
3. Lack of information;

The first and second points can be explained with some simple calculations. In our case there will be a hundred 2x2 transition matrices for each price. If one had considered all the possible connections between the states, there would be as many 100x100 transition matrices as there are prices. In terms of computation one can note that in our case, there will be 1000000 calculations, while in the second case there will be 40000 calculations.

Taking the third point, it can be noted that the conditional probabilities were not present. By using a 2x2 matrix the transition probabilities are deduced through:

	state 0	state 1
state 0	P(0)	1-P(1)
state 1	1-P(0)	P(1)
	state 1	state 2
state 1	P(1)	1-P(2)
state 2	1-P(1)	P(2)

Table 5.3: Example of transition matrices

Where P(0) is the sales probability of state 0: $P(0,a,\vec{p},\vec{\beta})$; and P(1) is the sales probability of state 1: $P(1,a,\vec{p},\vec{\beta})$.

Obviously, the sales probabilities change according to different p.

The image below, shows the mechanism in which each state is updated:

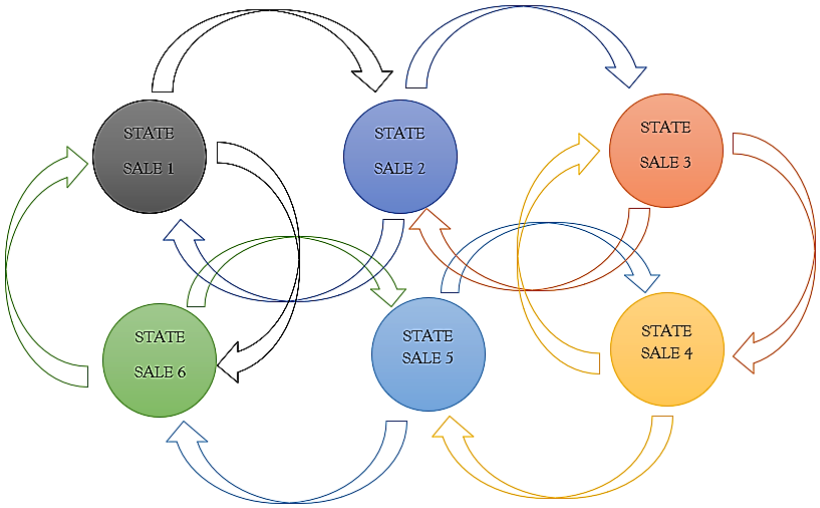

Figure 5.6: Example of transition probabilities for six states

5.3 Runtime efficiency

In this experiment, the most important requirement concerns the speediness of the algorithm. In fact, the algorithm needs to return a price response on average in 0.1 seconds per iteration and the maximum time for a single price response is 10 seconds [9].

5.3.1 Prices

Given the great dimension of the problem, 100 states and 100 prices, we decided to reduce the number of possible actions for each iteration.

In the beginning, we considered all the prices sampled from a uniform discrete distribution in the range 1 to 100. In a second moment, we cut the range of actions by taking 20 random prices (without replacement) from our past historical prices.

[9] See the requirements on: `https://dynamic-pricing-competition.com/`

This method permits to save the computational time by reducing the number of iterations. The runtime results are shown in the table above.

Method	Average runtime in ms
Price range (1,100)	116,0698161
20 historical random prices	11,51037022

Table 5.4: Computation time improvements (in avg.) of DP approach by changing price configurations

Applying the second method, it was possible to reduce the average time per iteration from 1.16 secs to 0.011 secs.

In this way, one could think, that the algorithm would take always the same prices, but using all the historical prices, the algorithm also chooses the prices in the first 150 periods that are randomized.

5.3.2 Theta values

The theta value (3) is decided by the programmer. The more theta is higher the faster the algorithm is. In the same way, the more theta is higher the more difficult is to reach the convergence.

In fact, theta could be viewed as a stopping-rule: when the difference between the old value function and the new value function is less than theta, the algorithm stops. In this experiment, we tried different configurations of thetas' values. The model used is the one with actions equal to 20 random historical prices and the resulting improvements are shown in table:

Method	Average runtime in ms
$\theta = 0.01$	21,94614935
$\theta = 0.05$	12,31062007
$\theta = 0.5$	11,87338448
$\theta = 1$	10,56930212

Table 5.5: Computation time improvements (in avg.) of DP approach by changing the theta value

The value used is $\theta = 0.05$. This value showed a good compromise for both, speediness and accuracy of the algorithm. In fact, if we had selected a value higher than 0.05, we could face the problem of selecting a price that was far from convergence more frequently.

CHAPTER 6

RESULTS

In the previous chapter, the implementation of the algorithm was seen. Once implemented, the algorithm has competed with the other competitors' algorithms.

Through this chapter, the resulting data from the competition were analyzed and discussed. The main purpose of the analysis reported in this section is to understand the most relevant trends and behaviors of prices and, primarily, of our algorithm in the competition .

6.1 Simulations

The algorithm has participated into 20 simulations of 1000 period each.

Each simulation had a different sales-generating mechanism that might be different in each simulation. Moreover, the sales mechanism was statistically identical in different time period –within the same simulation– and conditionated on selling prices.

A final CSV file with all the results was been shared from the organizer of the competition to us.

In what follows, we focus our discussion on simulation 3, 9, and 20. The choice of these three has been establish according to the observed revenue's mean. The simulation 3 is the one with the lowest revenue's mean; simulation 9 is one of the simulation that present a value centered in the range between the minimum and the maximum of the revenue's mean; simulation 20 is the one with the highest revenue's mean.

6.1.1 The least profitable simulation

In this simulation, our algorithm shown the lowest mean revenues. Probably our model didn't fit well the sales generating mechanism.

Since the above mentioned mechanism is remained unknown, the only data that we can analyze are the one received.

The first data that we viewed are our sales. In this scenario, we got in mean 37.684 sales. By looking at the fig 6.1, one can note a peak in the period 42 in which the algorithm obtained 100 sales by using a price of 11. The revenue obtained in period 42 are equal to 1100.

By going forward, the algorithm decided to select higher prices, because the main purpose is to obtain high revenues and not high sales.

In figure 6.2 the trend of our price in this simulation is reported. One can

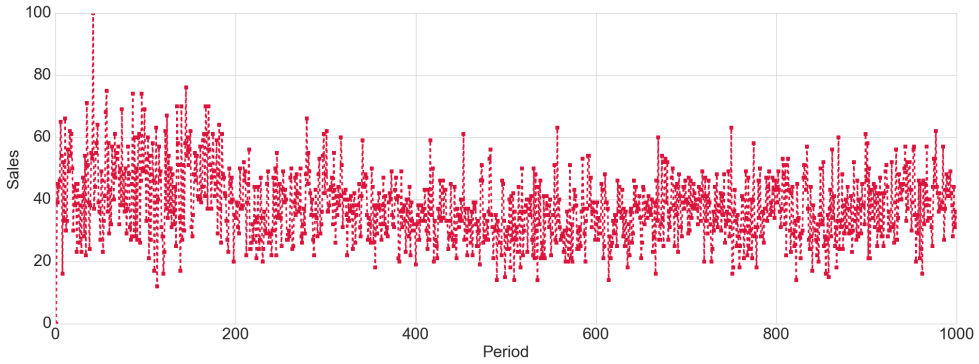

Figure 6.1: Sales trend in simulation 3

saw that the chosen prices are higher than 11. In fact, they range from 40 to 70. The observed mean price is 57.68.

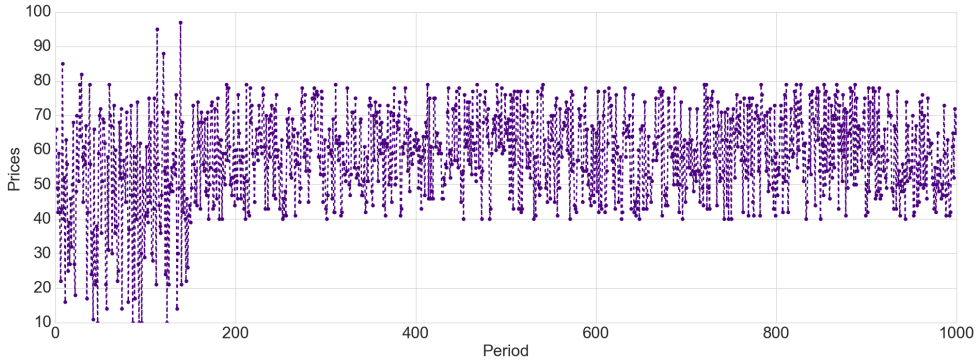

Figure 6.2: Prices trend in simulation 3

From image 6.2, the algorithm decided to maintain high-medium prices (40-70).

The most important objective is the maximization of the revenues. The highest revenue obtained in this simulation was in the period 279, in which the algorithm obtained a revenue of 4158. The chosen price for that period was 63 and the resulting sales were 66. Others high revenues are encountered in period 669, see fig. 6.3, in which the algorithm chosen a price of 64. The sales obtained are 60 and the consequent revenue is 3840.

By saying this, one can think that prices in the range 60-65 are optimal

in this simulation.

By analyzing better the situation, it can be noted that optimal prices depend on market situations (i.e. competitors' prices). As a matter of fact, the third highest revenue is obtained in period 174 with a price of 74, and the fourth is obtained in period 167 with a price of 48.

Overall, the average revenue in simulation 3 is equal to 2059.29, and, as mentioned above, it is the lowest of all the simulations.

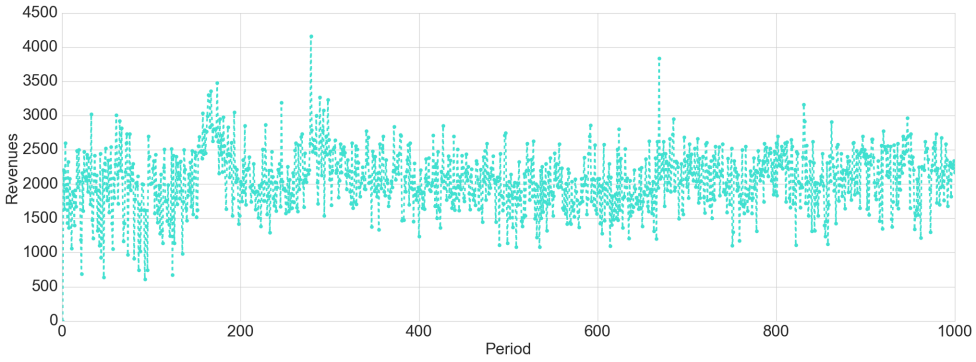

Figure 6.3: Revenues trend in simulation 3

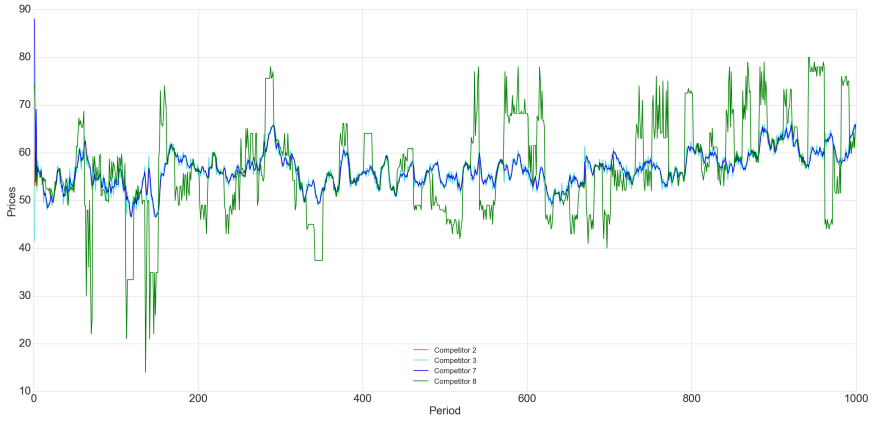

Figure 6.4: Prices trend for competitors 1, 3, 7, and 8 in simulation 3

Prices are mostly decided according to competitors' prices. These, are observable in fig. 6.5, in which the medium prices per competitor are reported.

Looking at the figure 6.5, the higher average price is the one of competitor

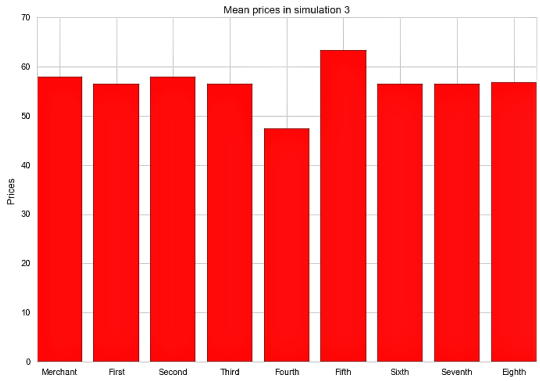

Figure 6.5: Average prices for all the participants in simulation 3

5, that is equal to 63.32.

Competitors 1, 3, 7, and 8 have a similar average price: 56, with some difference in the decimals (cf. 6.4); Competitors 2 has a medium price similar to us, around 58.

The lowest average price is the one of competitor 4: 47.41.

6.1.2 A good simulation

The second simulation analyzed, has been one of the simulation in which the algorithm obtained an average revenue in the range between the highest and lowest.

The highest revenue obtained is in period 809 with a value of 4292. The resulting sales are 58 with a price of 74.

The lowest, instead, is obtained in period 150: 500; the sales observed are 50 and the price chosen is 10.

From fig. 6.6 one can note a floating behavior of competitor 2. There are many periods in which the competitor set prices under 5. These periods are in the range of 400 to 700.

Another behavior that can be note by fig. 6.6 is the one of competitor 4. This competitor seems to have a fixed price of 39 from the period 4 to

period 657. There are some exceptions in this period (4-657) , in which the competitor used different prices, but, overall, 39 is the most used price for him.

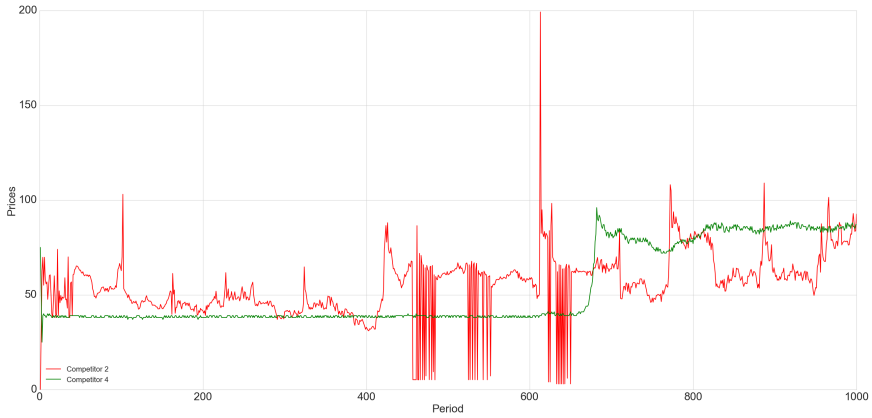

Figure 6.6: Prices trend for competitors 2 and 4 in simulation 9

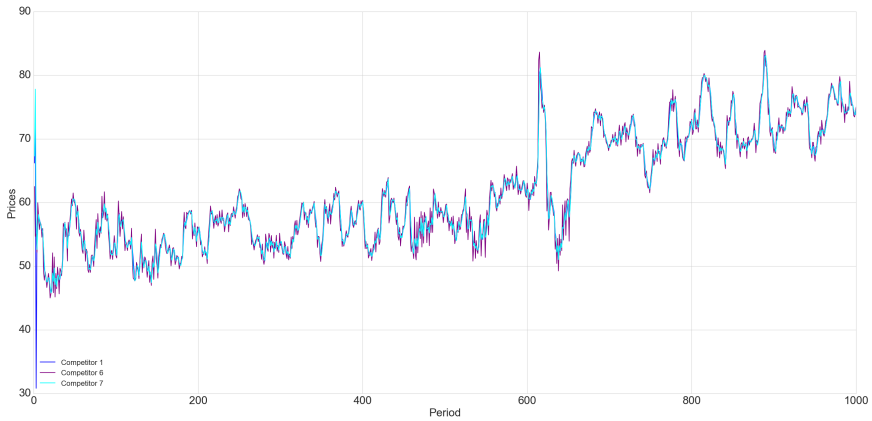

Figure 6.7: Prices trend for competitors 1, 6, and 7 in simulation 9

Analyzing the average prices of competitors, fig. 6.8, the twos competitors with lowest prices are competitor 2 (55.46) and competitor 4 (52.99). The motif of this has been explained above (cf. fig. 6.6). Competitors 1, 6, and 7 have a similar average price: around 61 (cf. 6.7). Competitor 5 has the highest average price: 68.96.

Regarding our medium price, it is equal to 58.605. By examining the trend of our price in this simulation, fig. 6.9, one can note that the prices chosen

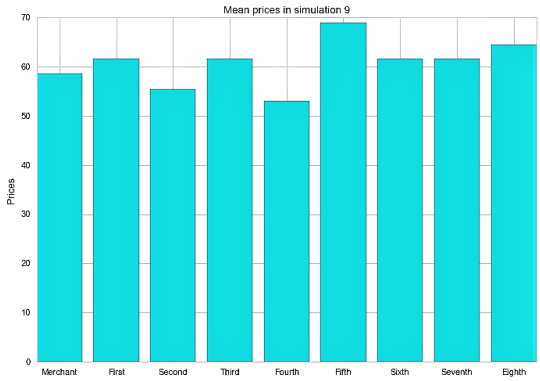

Figure 6.8: Average prices for all the participants in simulation 9

are in the range 40-80. If we inspect the revenues, the highest revenues are obtain with a range of prices between 60-77.

Given the fact that our average price is 58, maybe the algorithm don't perform well in this simulation. This can be due to the different sales generating mechanism. Anyway, the average sales obtained in this simulation are 43.329.

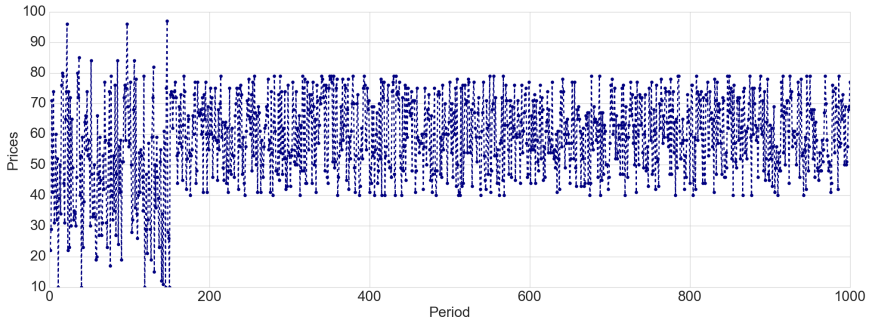

Figure 6.9: Prices trend in simulation 9

6.1.3 The most profitable simulation

In simulation 20, the algorithm reached the highest average revenue: 3,231.97.

By looking at fig. 6.10, one can note many peaks.

The highest revenue was obtained in period 564 with a price of 79. The resulting sales were 64 and the total revenue 5,056.

There are many other periods in which the algorithm performed well. To be precise, in 89 periods the obtained revenues are higher than 4,000.

Moreover, in fig. 6.10 is observable that all the other revenues -except those in the first 150 periods- are within the range of 2,000 and 4,000.

By analyzing the received data, it is noted that high revenues are not due to highest sales.

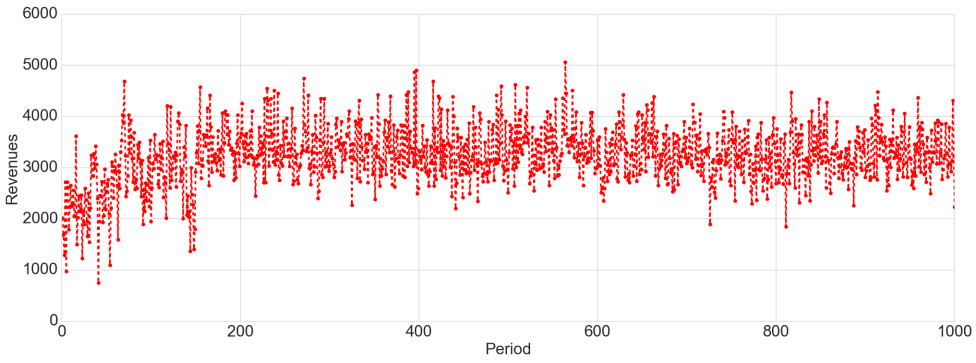

Figure 6.10: Revenues trend in simulation 20

In fact, speaking about sales, the topmost sales are equal to 113. These, were obtained in period 80 with a price of 24. The corresponding revenues are 2,712.

In periods 133, and 140, a similar situation happened: the chosen prices were lows (22, and 26) and the respective sales are 94 and 105.

Taking these three periods, 80, 133, and 140, it can be note that the algorithm obtained high sales because other competitors chosen high prices (between 78 and 90).

In this simulation, it is noted that using higher prices respect to our competitors, in many times imply that we obtain lowest sales. As a matter of fact, in period 9 a price of 93 is chosen and 24 sales are realized; others competitor had chosen prices in the range between 58 and 74. The average sales is 57.18 sales.

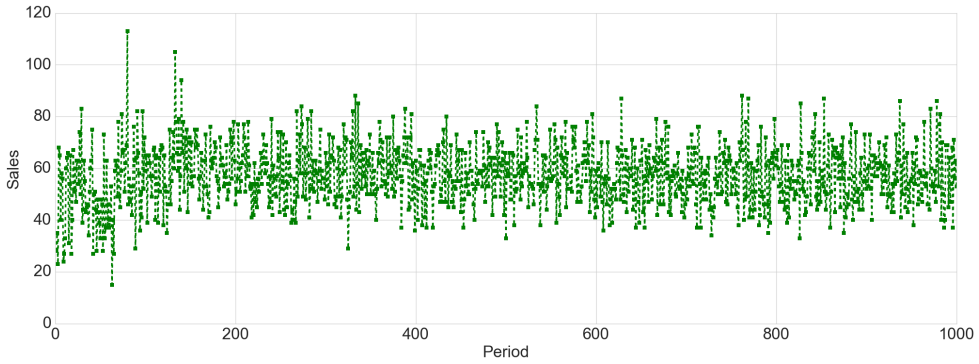

Figure 6.11: Sales trend in simulation 20

In figure 6.11 the sales trend is represented.

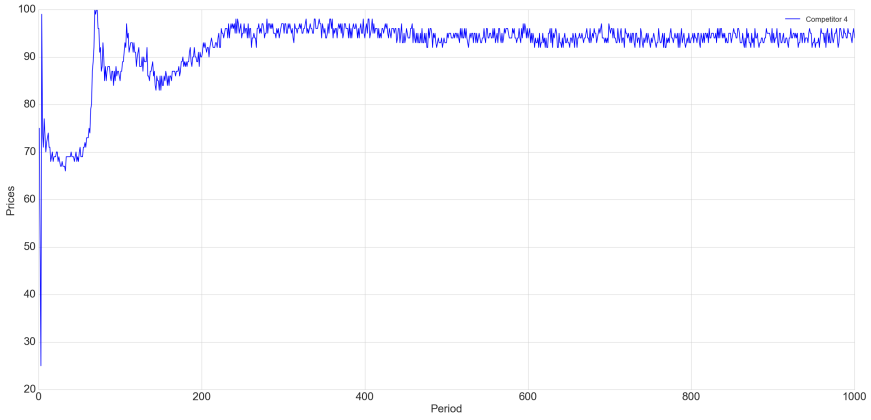

Figure 6.12: Prices trend competitor 4 in simulation 20

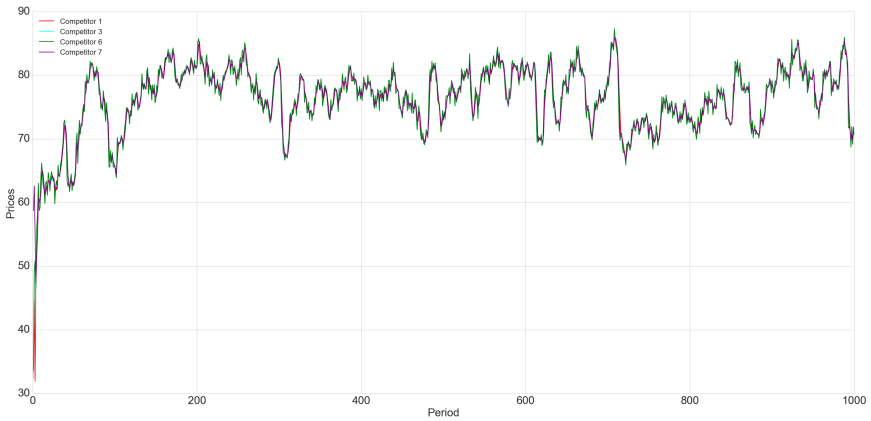

Figure 6.13: Prices trend competitors 1, 3, 5, and 7 in simulation 20

By exploring the competitors behaviour, competitor 4 is the one with the higher average price: 92.13 (cf. 6.12). Competitors 1, 3, 6, and 7 have a

similar pricing attitude: around 76 (cf. 6.13).

Competitor 5's average price is higher than the one of competitors 1, 3, 6, and 7: 79.05. While, competitor 8 is slightly less than competitors 1, 3, 6, and 7: 75.52. Competitor 2 has an average price of 68.80. Our medium price, instead, is the lowest and it is equal to 61.24.

6.2 Competitors' behaviour

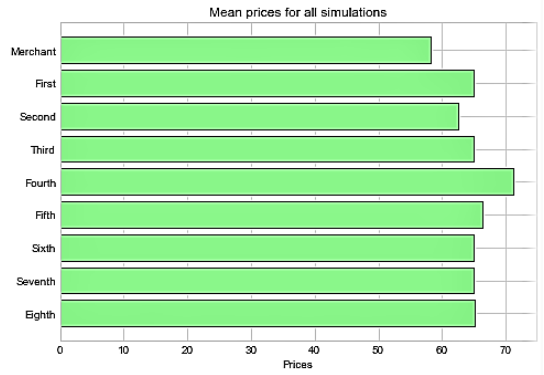

Figure 6.14: Average prices among all the simulations

Overall, analyzing all the simulations, it can be said that, in average, the prices are almost equal among all the competitors(fig. 6.14).

Competitors 1, 3, 6, 7, and 8, across all the simulations, show the same average price. Competitor 4 has the highest average price, while our merchant shows the least.

By going deeper:

- Competitor 1 (fig.6.15) presents prices that fluctuate between 56 and 77. In the third and in the eight simulation, he shows the smaller average price; while, in the twentieth simulation he has the highest average price.

  Moreover, according to the figure (6.15), competitor 1 has a positive trend: from simulation 3, with some fluctuations, he increases his

prices, until the last simulation in which - as said before - he chooses the highest average price;

- Competitor 2 (fig.6.16) has average prices within the range 51-70.

  Despite the first simulations, from the eleventh he has a steadily positive trend, starting by the smaller price and reaching to the highest in simulation 19.

  From simulation 10 to 11, he shows a large decline in prices: shifting from around 66 to around 51.

  Moreover, in simulation 8 and 9 he exhibits the same medium prices' value;

- If one look at figures 6.17, 6.20, and 6.21, competitors 3, 6, and 7, have an equal behavior to competitor 1 (cf. 6.15).

  In fact all of them, show the same average prices among all the simulations.

  This conduct can be viewed as "strange", but in many competitive markets is a common practice to follow the others competitors' pricing tendencies.

  This attitude will be discussed in chapter 7;

- Competitor 4 (fig.6.18) range's prices are within 48 and 92.

  This competitor depicts very variable prices. From simulation 4 to 7, the average prices are somewhat flat, while from simulation 12 to 17, they oscillate a lot. In the thirteenth and fifteenth simulation, he shows the lowest medium price. Instead, in the twentieth the highest;

- Competitor 5' average prices are between 53 and 79. The more elevated price is in simulation 20, whilst the lower price is in simulation 4.

  In fig. 6.19 it can be seen that in simulation 2 and 3 the competitor's algorithm chosen equal average prices.

Overall, from simulation 11 to the end, the prices fluctuate a lot;

- Competitor 8 shows average prices from 59 to 78. The highest price is in simulation 17, and the lowest is in simulations 3 and 4.

  Looking at fig.6.22, in the first three simulations, the algorithm follows a negative trend by reducing the prices. Instead, in the last simulations, it tends to increase the prices.

  A significant rise is the one from simulation 15 to 17;

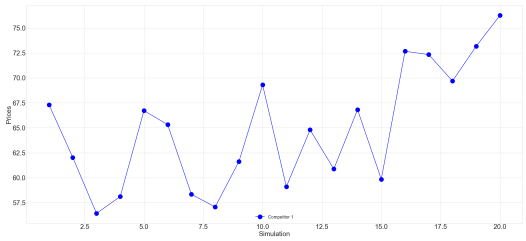

Figure 6.15: Competitor 1's average price for each simulation

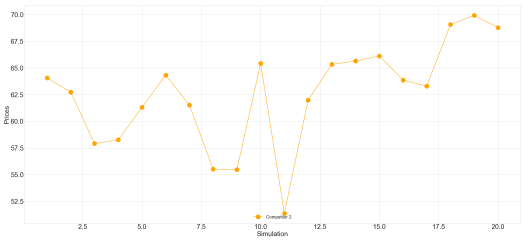

Figure 6.16: Competitor 2's average price for each simulation

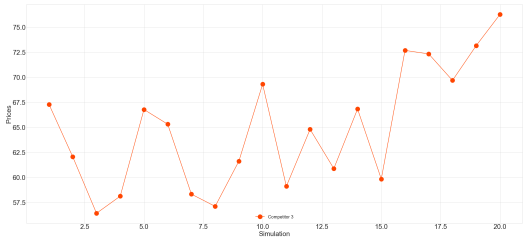

Figure 6.17: Competitor 3's average price for each simulation

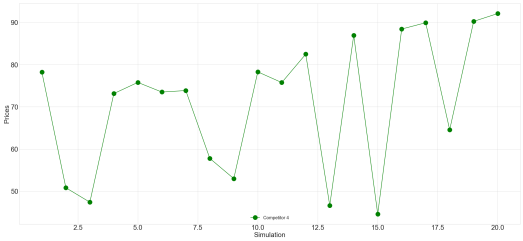

Figure 6.18: Competitor 4's average price for each simulation

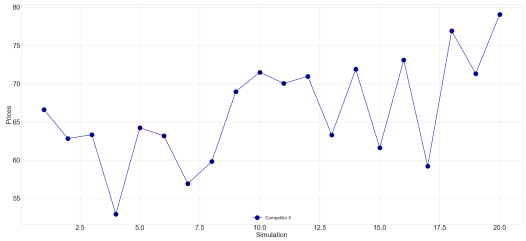

Figure 6.19: Competitor 5's average price for each simulation

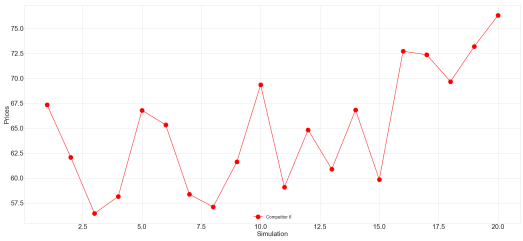

Figure 6.20: Competitor 6's average price for each simulation

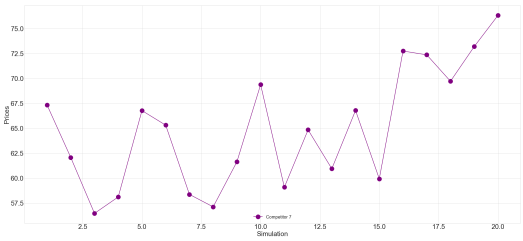

Figure 6.21: Competitor 7's average price for each simulation

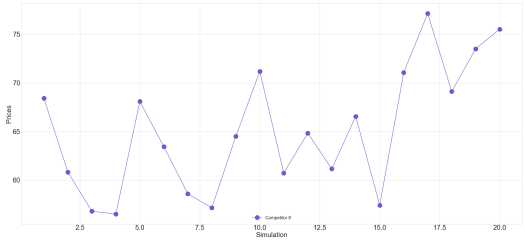

Figure 6.22: Competitor 8's average price for each simulation

6.3 Algorithm behaviour

The analysis carried out in the section above, is re-proposed for our algorithm. If one look at fig.6.23, it can be noted that the mean prices chosen by the algorithm are different among all the simulations, but they range in a small interval (55-60).

As a matter of fact, our algorithm is able to change prices and to fit in different scenarios; in each scenario, the mean of the prices differ but, in a certain way, all the prices chosen are not so different with each others.

This behaviour can be due or because the selected prices are recognize as the best to our algorithm or because the range of prices from which our algorithm select the best price is too narrow (cf. 5.3.1).

Since our competitors' sales and revenues are unknown, we cannot claim that the prices selected by the algorithm are the best. What we surely know is that reducing the range of possible prices don't allow our algorithm to evaluate all the feasible situations.

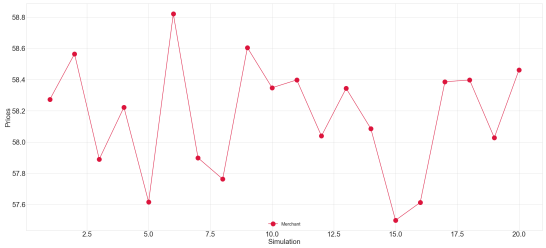

Figure 6.23: Merchant's average price for each simulation

6.4 Remarks

From what has been said in this chapter, we can make some statements:

1. The simulation with the more elevated prices is the simulation 20. This simulation is also the one in which we have obtained the higher revenues.

   If we look at the medium prices set by the competitors' algorithms and the one set by our algorithm, our price is the lowest. In this simulation, setting the lowest price permitted us to having more revenues;

2. There are some algorithms that set prices in highly variable way (i.e competitors 4 and 5), whilst others tend to maintain their prices in a smaller range (i.e our merchant and competitor 2);

3. There are some occasions in which some prices are set above 100 or below 50, but in mean, prices chosen by all the merchants in the market are in the top range, from 50 to 100;

CHAPTER 7

INSIGHTS

The next chapter describes some insights that will be deduced from the simulation results shown in the chapter 5.

Some insights will be obtained regarding our algorithm and others regarding the market.

Additionally, this chapter will try to understand the applicability and weaknesses of our approach.

7.1 Market insights

According to section 6, some market insights can be deduced.

The first thing to note is the one concerning the same behavior of certain competitors. In a real market, it is common to fix prices by following the actors involved. This conduct is called *"competitive pricing"*.

Competitive pricing is properly related to fixing prices at the same level as the other competitors. The motif to do so is connected to the fact the other actors have already tested their pricing strategy on the market. Moreover, if all follow the same price's tendencies, automatically, the market will find an equilibrium.

As written in an article by G. Grasset [10], this strategy is highly used in competitive markets because it implies low risks in choosing prices that are not convenient. On the other hand, there are also many disadvantages to using such a method. For example, competitor's prices can lead to choosing non-optimal prices. In addition, if all the actors, for a long period, continue to fix the same prices, the market can reach a sort of stationary.

It can happen, also, that if competitive pricing is very aggressive, then the merchants could experience a decrease in profits.

Another common behavior is one of choosing the lowest prices respect to those chosen by competitors.

The choice of low prices can be dictated by several reasons:

1. Increase market penetration;
2. Increase in market share;
3. Prevent the entry of rivals into the market;

According to the first point, Market penetration refers to increasing product sales through a pricing strategy. It is about a company's ability to be more attractive and cheaper to customers by setting low prices to increase

[10] www.lokad.com/it/definizione-competitive-pricing

customer demand. The goal is to build customer loyalty.

Economically speaking, if the customer is loyal, their demand has become less elastic over time. As demand tends to become inelastic over time, the company can increase profits by raising prices later. To better understand this mechanism, we refer to an article published by Lokad[11].

Initially, the choice of low prices leads to a short-term loss of economic surplus, but also to an increase in demand (cf. fig.7.1).

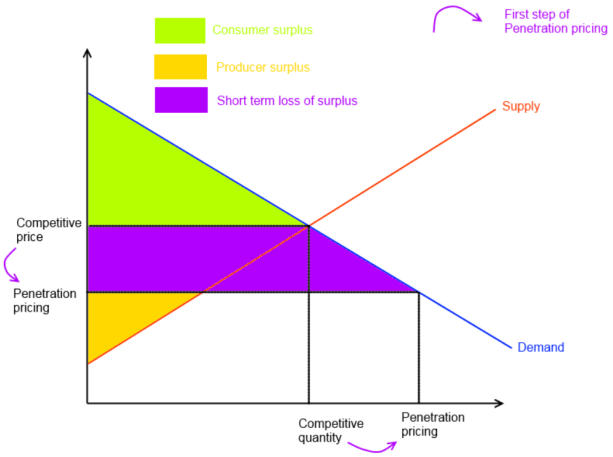

Figure 7.1: Penetration strategy first step

source: `https://www.lokad.com/it/definizione-penetrazione-del-mercato`

Later, when the price is raised, the economic surplus increases significantly, thanks to customer loyalty (cf. fig.7.2).

As Kotler and Armstrong [18] said: *"Market penetration is a low pricing strategy adopted by companies for new and existing products to attract a larger number of buyers and a larger market share"*.

The second point, market share, is closely related to the first point, market penetration.

The market share formula, in fact, includes the degree of penetration. To

[11] www.lokad.com/it/definizione-penetrazione-del-mercato

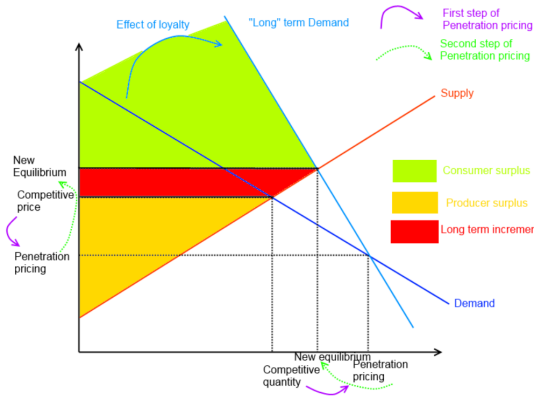

Figure 7.2: Penetration strategy second step

source: `https://www.lokad.com/it/definizione-penetrazione-del-mercato`

be clearer, we report the following formula below:

$$MarketShare = \frac{Q_i}{ACS_i} \cdot \frac{ACS_i}{Q} = \frac{Q_i}{Q}$$

Where Q_i is the quantity sold by the firm i, Q is the total quantity sold in the market and ACS_i are the total purchases of the customers.

Looking at the formula, it can be understood that market share refers to the percentage of sales that a firm is able to make out of total sales in the market in which it operates.

Market share indicates how the company can differentiate its products; in a certain sense it suggests the "originality" of the company. Increasing the market share means increasing the profits and the competitiveness of the company on the market.

Finally, the third point, the one that refers to the prevention of new rivals on the market.

New entrants to the market can erode the market share of existing firms resulting in a decrease in their future profits. Many times, therefore, the companies present on the market may decide to put real barriers to entry for new companies that could enter the market.

A common way to impose these barriers is to choose low prices, sometimes even below cost, to discourage entry.

In this simulated market, very low prices can sometimes be observed, however these are not continued for many periods. The third point therefore is not to be considered as a strategic choice of competitors present in this competition.

7.2 Algorithm insights

Overall, our algorithm has demonstrated the ability to set prices and generate revenue. It has also shown that it is able to compete in different situations and market contexts, despite the stochasticity in consumer choices.

In general, looking at the previous chapter (6), it can be seen that the prices chosen by our algorithm are, on average, always the lowest (cf. 6.14).

There are three reasons given. One concerns the choice of the reinforcement learning technique; another concerns the optimality of the chosen prices; finally, the last concerns the restrictiveness in the field of actions.

Starting from the first reason, the reinforcement learning algorithms (refer to chapter 3) are algorithms that try to maximize the discounted future rewards.

The peculiarity of these algorithms, in fact, is the use of a discount factor, γ, to give more or less importance to future rewards. For instance, $\gamma = 1$ only considers future rewards and does not consider current ones; conversely, $\gamma = 0$ considers only the current ones, not looking at future ones.

Within our algorithm, a discount factor, γ= 0.7 has been chosen.

Having said that, the proposed algorithm could choose prices according to possible future returns.

By processing the information on the simulated market, he could have learned that low prices can lead to loyal customers and that these prices could be raised later. Reference is made to the market penetration of the previous section (7.1).

This assumption cannot be tested, as the algorithm could only compete for 1000 periods in each simulation. A higher range of periods could have confirmed or denied this theory.

For instance, if the algorithm after a certain number of periods had started to significantly raise prices, then the above assumption could have been held.

The second hypothesis made is that on the optimality of prices. The choice of a narrow range of prices leads us to think that this is actually the range of prices that allow to obtain maximum revenues.

Despite this, as mentioned in section 6.3, there is no knowledge of other competitors' revenues, so it is not possible to assume that our prices are the best.

The fact that the algorithm still manages to vary prices during the various periods allows us, at least, to confirm that it does not choose prices in a myopic way but according to certain reasoning.

Below, some randomly selected data from the first of twenty simulations are reported to understand the reasoning the algorithm followed in setting prices most of the time.

427	1	426	60	44
428	1	427	62	58
429	1	428	66	41
430	1	429	51	59
431	1	430	44	61
432	1	431	40	51
433	1	432	54	49
434	1	433	75	48
435	1	434	57	50
436	1	435	70	48
437	1	436	58	50

Figure 7.3: Simulation 1 data from period 427 to 437

Looking at the figure 7.3, in the period 427 the algorithm has chosen a price of 60 obtaining a sale of 44, then a price of 62 is selected and the sales are increased by 14.

The algorithm decides to raise the price again, having experienced an increase in sales, going to a price of 66 in the period 429. Sales however decrease to 41. Raising prices again is not a good strategy.

In the following period, therefore, the price is reduced to obtain a new increase in sales. The chosen price is 51 and the sales increase to 59. In the period 431, the price of 44 is tested to be able to further increase the products sold; 61 sales are obtained.

The trend of decreasing prices continues, from 44 the algorithm passes to 40 but experiences a decrease in sales. In the following period it tried to increase the price to 54 (we are in the 433 period), the sales decrease again to 49. The last periods shown in the image show that this reasoning is not always followed.

In fact, from period 434 to period 437 there is no definite trend for the choice of prices.

If one think about revenues, however, the last periods show higher values than those previously reported in figure 7.3.

Finally the last hypothesis, concerning the choice of using a limited set of actions (section 5.3.1).

Due to the time requirements, it was decided not to use all 100 prices for each iteration but to choose 20 random ones without replacement from our historical prices.

From a logical point of view, the prices chosen by the algorithm vary within the range [1; 100]; the algorithm is therefore able to differentiate its actions at each period.

From a practical point of view, however, the algorithm could be limited in choosing prices: having 20 <100 actions available, the algorithm is forced

to evaluate fewer prices at each iteration.

It could happen that the chosen price turns out to be optimal among the range of available actions but not optimal among the entire range of these.

Obviously, in this way there is a loss of information with the consequent result that the algorithm is not always able to find the best price.

It has been observed (section 6.3) that on average the algorithm uses prices in a narrow range of actions: 55-60.

The question that arises is: "Would this range have expanded if we had considered the entire available range of actions (i.e 100 prices)?"

This question, like the other assumptions made above, cannot be answered. Surely limiting the actions involves a consequent loss of computational efficiency on the part of our algorithm.

7.2.1 Algorithm improvements

The algorithm proposed in this thesis has been designed to compete in a competition with particular participation requirements.

Not considering these requirements, many improvements could be made.

The first is definitely about better customer prediction.

In our algorithm, customer behavior is modeled through polynomial regression. This, as mentioned before (sec. 5.1.1), allows to predict the number of expected sales given a price and a given market situation (competitors' prices).

Expected sales reflect customer behavior. Perhaps different models with the available data could have predicted this behavior better.

With the availability of more time, it would have been possible to better investigate some methods, such as a Neural network, or it would have been possible to build real distributions capable of modeling the aforementioned behavior, as a Mixture model.

A second aspect to take into consideration is that which concerns the pricing strategies of competitors.

These strategies are, in our experiment, appropriate by Polynomial regression but are not predicted in any way, they are only elaborated by the model.

It would certainly be interesting to explore this aspect further.

For example, a time series method could be proposed, such as the one of Exponential smoothing, to predict various customer behaviors.

Having such a prediction would allow the algorithm to develop its own pricing strategies in response to the strategies of other competitors.

Moreover, having considered only a narrow range of prices had some repercussions on the efficiency of the algorithm.

The selection of random prices leads to the choice of random actions. Using such stochasticity in making pricing decisions, we realize, may not be reliable.

Certainly, other "safer" methods would have brought better results.

The need to have strict execution times to respect has made it necessary to use a shortcut which may not be ideal for choosing the best prices.

For what has been said so far, an improvement for our algorithm would be to choose the input actions in a different way, either considering them all or considering only a narrower range such as [30; 80].

Finally, one cannot ignore the possible improvement in the algorithm if one had chosen a lower theta value (cf. 5.3.2).

The theta value would have allowed a more certain convergence of the algorithm, the consideration of all the actions would have allowed an optimal search of these by the algorithm.

To conclude this section, it can be said that the proposed algorithm has been built to perform within a simulated market. This simulated market has an impact on pricing decisions and algorithm learning dynamics, in a

sense our algorithm may not reflect the complexity of a real market.

The insights reported are based on reflections and assumptions made within the experiment and cannot be generalized in reality.

CHAPTER 8

CONCLUSION

The analysis carried out in this thesis helped to understand the importance of using dynamic prices within the markets, with particular focus on the e-commerce markets, where there are greater dynamism and speed in pricing choices.

The main purpose of this work was, in fact, to understand how a seller can effectively use dynamic pricing techniques to set prices automatically, as well as to maximize their revenues.

In this study, a pricing model using a reinforcement learning method was proposed and estimated.

The implemented model made it possible to compete within the "Dynamic Pricing competition", resulting in the top half of the overall standings.

Although it was not possible to test our results and compare them with those of other competitors, being an online competition, we can confirm that our algorithm within the competition managed to perform well, being able to obtain high revenues and to adapt to all the different market situations.

The competition allowed the model to be used within a simulated e-commerce scenario where only some limited data was known. The first thing to note is that within simulated markets, machine learning can lead to significant revenue using the little information available.

However, real markets turn out to be very complex for the implementation of some algorithms. A real market has a multitude of information that the algorithm must be able to process quickly. It can be stated that the more complex the market, the greater the complexity required by the algorithm.

Also based on the literature reported in chapter 2, it can be seen that most scientific works are built on the basis of simulated market models in which strong hypotheses are made. This, precisely because the inclusion of realistic hypotheses would lead to greater complexity in the construction of such models.

Models built in simulated environments appear to be of vital importance for the development of theories and hypotheses, however these may not provide realistic results because what seems reasonable in a simulated market may not be in a real one.

Our work does not deviate from the trend shown in the literature. Within our model there are some hypotheses which, as mentioned in the previous chapter, may not allow the generalization of the algorithm within a real market. It could be true that if the improvements indicated in the previous chapter were applied, the algorithm would acquire greater applicability in a real market. Having said that, researchers are certainly required to study and develop more in-depth models that better reflect the reality of the markets, introducing more advanced effects and dynamics within the aforementioned models.

However, it is interesting to note that over the past decade there has been an increase in literature and an improvement in methods for dynamic pricing. The increase in research work on the subject was mainly dictated by the growing need of the company to introduce this method as a tool and pricing strategy.

In the future, companies in particular will be required to adapt more and more to different market situations. Surely the products will not have a fixed but highly flexible price, especially the prices of the products on the

online market.

In conclusion, this thesis analyzes how companies can benefit from learning and optimizing their pricing strategies by interacting in real-time with competitors and customers on the market, collecting data and managing to transform such data into market decisions, in this case in pricing decisions.

REFERENCES

[1] Adida E. and Perakis G. *Dynamic pricing and inventory control:uncertainty and competition*, Operations Research 58(2), 289-302, 2010.

[2] Bellman R. *Dynamic Programming*, Princeton University Press, 1957.

[3] Bernstein M.A and Griffin J. *Regional Differences in the Price-Elasticity of Demand for Energy*, 2006,10.2172/877655.

[4] Bertrand J. *Recherche sur les principes mathematiques de la theorie des richesses*, Journal des Savants, 499-508, 1883.

[5] Bodea T. and Ferguson M. *Pricing: Segmentation and Analytics*, Business Expert Press Marketing Strategy collection, 2012, DOI 10.4128/9781606492581.

[6] Breidert C., Hahsler M. and Reutterer T. *A Review of Methods for Measuring Willingness-to-Pay*, Innovative Marketing vol.1, 2015.

[7] Brown Z.Y.and MacKay A. *Competition in Pricing Algorithms.*, Harvard Business School Working Paper, No. 20-067, November 2019.

[8] Carvalho A.X. and Puterman M.L. *Dynamic Optimization and Learning: How Should a Manager set Prices when the Demand Function is Unknown ?*,Instituto de Pesquisa Econômica Aplicada - IPEA, no.1117, 2005.

[9] Chen W., Liu H. and Xu D. *Dynamic pricing strategies for perishable product in a competitive multi-agent retailers market*, Journal of Artificial Societies and Social SImulation 21(2)12, 2018.

[10] Cheng M. and Chen Z.L. *Recent developments in Dynamic Pricing research:*

multiple products, competition and limited demand information, Production and Operations Management 24(5), 704-731, 2015.

[11] Chung B.D, Li J., Yao T., Kwon C. *Demand learning and dynamic pricing under competition in a state space framework*, IEEE Transactions on Engineering Management 59(2), 240-249, 2012.

[12] Edgeworth F.Y. *The Pure Theory of Taxation*, Musgrave R.A., Peacock A.T. (eds) Classics in the Theory of Public Finance. International Economic Association Series, 1958.

[13] James G., Witten D., Hastie T., Tibshirani R. *An Introduction to Statistical Learning : with Applications in R*, Springer, 2013.

[14] Goodwin G.C and Sin K.S *Adaptive filtering, prediction and control*, Prentice-Hall, Englewood Cliffs. N.J (1984).

[15] Harrison K., Kumar S., Christa P.L and Santanello J. *Quantifying the change in soil moisture modeling uncertainty from remote sensing observations using Bayesian inference techniques*,Water Resources Research vol.48, 2012.

[16] Howard R.A (1960), *Dynamic Programming and Markov Processes*, MIT Press.

[17] Ito S. and Fujimaki R., *Optimization Beyond Prediction: Prescriptive Price Optimization*,10.1145/3097983.3098188, 2017.

[18] Kotler P. and Armstrong G. *Principal of Marketing*, 10th Ed. New Delhi: Prentice Hall of India Private Limited, 2002.

[19] Krämer A. and Kalka R. *How Digital Disruption Changes Pricing Strategies and Price Models*, Phantom Ex Machina: Digital Disruption's Role in Business Model Transformation, 10.1007/978-3-319-44468-06, 2017.

[20] Littlewood, K. *Forecasting and control of passenger bookings*, Proc. 12th AGIFORS Symposium, 1972, reprinted in Journal of Revenue and Pricing Management, Vol. 4 (2005).

[21] Martinéz de Albéniz V. and Talluri K. *Dynamic pricing competition with fixed capacities*, Management Science vol.57 no.6, 1078-1093, 2001.

[22] Phillips R. *Pricing and revenue optimization*, Stanford University Press,2005.

[23] Phillips R. and Ozer O. *Why are Prices Set the Way They Are?*,978-0199543175,10.13140/2.1.4933.2808, 2012.

[24] Pong V.H, Shixiang G., Dalal M. and Levine S., *Temporal Difference Models: Model-Free Deep RL for Model-Based Control*, ArXiv, abs/1802.09081, 2018.

[25] Raju C.V, Narahari Y., Ravikumar K. *Learning dynamic prices in electronic retail markets with customer segmentation*, Ann Oper Res vol.143, 59-75, 2006.

[26] Rana R. and Oliveira F.S. *Real-time dynamic pricing in a non-stationary environment using model-free reinforcement learning*, Omega, Elsevier, vol. 47(C), pages 116-126, 10.1016/j.omega.2013.10.004, 2014.

[27] Renzhi L., Hong S.H., Zhang X. *A demand response algorithm for smart grid: a reinforcement learning approach*, Applied Energy vol.220, 220-230, 2018.

[28] Rothschild M. *Search for the Lowest Price When the Distribution of Prices Is Unknown*, Journal of Political Economy vol.82, 1974.

[29] Sairamesh J. and Kephart J.O. *Price dynamics of vertically differentiated information markets*, Association for Computing Machinery, 28–36, 10.1145/288994.289000, 1998.

[30] Schlosser R. and Boissier M. *Dynamic Pricing under Competition on Online Marketplaces: A Data-Driven Approach*, KDD 2018: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 10.1145/3219819.3219833, 2018.

[31] Smith B.C., Leimkuhler J.F., and Darrow R.M. *Yield management at American Airlines*, Interfaces 22, no. 1 (1992): 8-31.

[32] Poole D. *Decision Theory, the Situation Calculus, and Conditional Plans*, Linkoping Electronic Articles in Computer and Information Science, Vol. 3(1998): nr 8.

[33] Porter M. *Strategy and the internet*, Harvard Business Review, 2001.

[34] Rangaswamy A., Pusateri M. and Shankar V. *The online medium and customer price sensitivity*, Working Paper, 1999.

[35] Reinartz W. *Customizing prices in online markets*, Emerging Issues in Manage-

ment, 10.4468/2002.1.05reinartz, 2002.

[36] Schlosser R. and Richly K. *Dynamic pricing strategies in a finite horizon duopoly with partial information*, SSRN Electronic Journal, 2016.

[37] Sutton R.S and Barto A.G, *Reincorcement learning: an introduction*, MIT Press, 2018.

[38] Talluri K. T. and van Ryzin G. J. *The theory and practice of revenue management*, Springer Science, 2004.

[39] Trovò F., Paladino S., Restelli M., Gatti N. *Improving Multi-Armed Bandit algorithms in online pricing settings*, International Journal of Approximate Reasoning, vol.98, 196-235, 2018.

[40] Van de Geer R., Van den Boer A. et al., *Dynamic pricing and learning with competition: insights from the dynamic pricing challenge at the 2017 INFORMS RM & pricing conference*, Journal of Revenue Pricing Management 18, 185–203, 2019.

[41] Van Den Boer A. *Dynamic pricing and learning: Historical origins, current research, and new directions*, Surveys in Operations Research and Management Science vol.20, 10.1016/j.sorms.2015.03.001, 2015.

[42] Wolff, R. W. *Poisson arrivals see time averages*, Operations Research 30,1982.