Simulating Dynamic Pricing Algorithm Performance in Heterogeneous Markets

Hans Olav Østbø Hilsen

Industrial Economics and Technology Management
Submission date: June 2016
Supervisor: Einar Belsom, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Problem Description

Develop a model for simulating different dynamic pricing algorithms in various heterogeneous market conditions. Evaluate the algorithms' performance and how the different consumer behaviors impact the overall market and revenues.

Preface

In front of you now, lies the thesis ”Simulating Dynamic Pricing Algorithm Performance in Heterogeneous Markets”, which is an overview and simulation-based evaluation of dynamic pricing by machine learning algorithms. This master’s thesis was written to fulfill the graduation requirements of Industrial Economics and Technology Management at the Norwegian University of Science and Technology (NTNU). I was engaged in writing this thesis from mid-January 2016 to mid-June 2016. The majority of Chapter 2 through 6, stems from my project thesis, ”Dynamic Pricing by Machine Leaning: An Overview”, written during the fall of 2015.

The idea for this thesis started out after I read the book ”Automate This: How Algorithms Came to Rule our World” by Christopher Steiner. In his introduction, Steiner elaborates on how the price of a book on Amazon.com came to sell for $23,698,655.93 due to an ongoing upward price war between two established Amazon sellers’ pricing algorithms. I have previously been aware that online stores change their prices regularly, though it had never occurred to me that behind the fluctuating prices lies sophisticated algorithms utilizing artificial intelligence (AI).

Presumably, although you might not be fully aware of it, it is my belief that the concept of dynamic pricing is already somewhat familiar to you as well. For instance, you might have noticed that prices for airplane fares tend to vary and that prices for goods sold on the internet might change over time. As will be evident later in this thesis, these are great examples of dynamic pricing utilized by companies today.

When I first started working on this thesis, I had little knowledge of dynamic pricing and had no insight into the world of machine learning and AI. Working with this thesis, I have learned more about the intriguing aspects of pricing, programming, and how learning algorithms can be used to optimize pricing decisions. My goal has been to gain insight into a field of research that I found to be interesting, and this I have achieved.

I sincerely hope that by reading this thesis you too will gain an appreciation of the fascinating underlying dynamics of modern pricing algorithms. My personal measurement of success relies on that you will enjoy reading this thesis as much as I have enjoyed writing it.

Trondheim, 2016-06-09

Hans Olav Hilsen

Hans Olav Østbø Hilsen

Acknowledgment

First of all, I would like to express my gratitude and appreciation to my supervisor Einar Belsom for the useful remarks, comments and engagement he has provided throughout the learning process of this master's thesis. Furthermore, I am also grateful to all of those who have expressed their interest in my work, and especially Erlend Fasting and Christopher Lange for our discussions on the topic. Also, I would like to thank Haakon Hals Rød for lending me his books regarding AI and machine learning. Finally, I express out appreciation to my friends and family who have supported me during this semester.

Hans Olav Østbø Hilsen

Abstract

This thesis investigates how sellers in e-commerce can maximize revenue by utilizing dynamic pricing decisions by machine learning algorithms in a market consisting of multiple learning agents and a heterogeneous consumer base. Employing a novel approach we elucidate how a population of agents adapt and perform in the presence of other adaptive agents when faced with a mixed composition of myopic and strategic consumers in a finite market.

By analyzing similarities and variation in current research, we find that a variety of different machine learning approaches have been applied to dynamic pricing problems, but there seems to be no unifying best algorithm for solving these complex problems. Furthermore, we find that all the presented literature evaluates the performance of machine learning algorithms in simple simulated environments. Also, we shed light on the literature's failure to compare the performance of different machine learning approaches under equal conditions. Perhaps our most important discovery is that there seems to be no empirical evidence justifying the added value of dynamic pricing by machine learning algorithms for real-world sellers.

Our approach studies dynamic pricing decisions by simultaneously learning, Q-learning and neural network algorithms, and find that despite the non-stationary nature, the algorithms provide robust performance in our moderately realistic markets. Furthermore, the agents show tendencies to collude implicitly, keeping average prices above marginal cost without any means of communication. Neither of the algorithms proves to be the overall best achiever, as their performance depends on the underlying market's consumer and seller composition, and the extent to which the agents are trained. We find that Q-learning presents the most reliable approximation of optimal future price paths, but at a cost of a long and tedious training period. The suggested neural networks show promising results and provide a more balanced approach to training time and performance. However, their limited network size precludes them to comprehend the consequences of their actions, and consequently, show less implicit cooperation than Q-learning.

We find that the agents' ability to capitalize on fluctuations in consumer valuations, not only improves the sellers' profitability but may also increase consumer surplus, compared to a fixed-price policy. However, it seems that the dynamics between the sellers' price policies have the greatest impact on revenues, as the algorithms are collectively incapable of exploiting increases in consumer valuations. Furthermore, in the presence of strategic and myopic consumers, lower price paths may be beneficial for maximizing revenues, depending on the discrepancy between the consumers' willingness to pay. Consequently, we find that competition can increase output and seller surplus because it induces an earlier lower price path. In a monopoly, the algorithms' more gradually learn that a lower price path can generate more revenue.

Sammendrag

Denne oppgaven undersøker hvordan selgere i elektroniske markeder kan bruke maskinlæringsalgoritmer til å utføre dynamiske prisbeslutninger i et konkurranseutsatt marked, bestående av flere læringsalgoritmer og en heterogen kjøpergruppe. Vi tar en ny tilnærming til problemet, ved å belyse hvordan en gruppe algoritmer tilpasser seg og yter i et simulert tidsbegrenset marked, bestående av både myopiske og strategiske konsumenter.

Ved å studere eksisterende litteratur på området, finner vi at det eksisterer atskillige algoritmer og markedsmodeller brukt i denne sammenhengen, men at ingen av de foreslåtte algoritmene viser seg å være den absolutt beste til å løse disse komplekse problemene. Samtlige artikler vurdert, undersøker hvor gode algoritmene er til å prise varer i simulerte markeder. En svakhet ved litteraturen er at ingen har vurdert hvor effektive maskinlæringsalgoritmene er til å prise varer i forhold til andre avanserte algoritmer under like forutsetninger . Vårt kanskje viktigste funn fra litteraturstudiet, er at litteraturen har vist en manglende evne til å bevise at dynamisk prising utført av maskinlæringsalgoritmer kan skape verdi for selgere i den virkelige verden.

Vår tilnærming til problemet utvider eksisterende litteratur ved å studere hvordan flere simultant lærende Q-learning og neurale nettverk kan utføre automatiske prisbeslutninger. Vi viser at de foreslåtte algoritmene fungerer godt, til tross for vårt markeds ikke-stasjonære art. Algoritmene viser tendenser av implisitt prissamarbeid, og holder prisene godt over marginalkostnad uten noen form for kommunikasjon. Ingen av algoritmene viser seg å være den absolutt beste for alle tilfeller, da ytelsene i stor grad avhenger av markedets konkurrent- og kjøperadferd, samt behovet for "læring". Q-learning produserer de beste prisingsreglene, men krever en svært lang treningsperiode. De neurale nettverkene viser lovende resultater, og tilbyr en mer balansert tilnærming til læringstid og ytelse. Dog er deres evne til å forstå konsekvensene av sine handlinger begrenset av størrelsen på nettverkene, og således viser de mindre implisitt prissamarbeid enn Q-learning.

Vi viser at algoritmenes evne til å kapitalisere på variasjoner i kjøpernes betalingsvillighet, ikke bare er en fordel for selgerne, men også for konsumentene, sammenlignet med en fast-pris strategi. Samtidig finner vi at dynamikken mellom selgerne har størst betydning for lønnsomheten, da algoritmene er inkapable til å utnytte økninger i konsumentenes betalingsvillighet på grunn av markedskonkurransen. Videre ser vi at når et marked består av både myopiske og strategiske kunder, kan det være fordelaktig for selgerne å følge en noe lavere prisutvikling for å maksimere inntektene. Dette er dog avhengig av avstanden mellom kjøpernes betalingsvillighet. Som et resultat, finner vi at konkurranse kan øke produksjonen og selgernes lønnsomhet fordi det sørger for lavere priser tidligere i markedet. I et monopol vil algoritmene mer gradvis lære at en lavere pris kan generere større inntekter.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "We envision a world a decade or two hence in which billions of software agents will act as economic players in their own right, exchanging information, goods, and services, with humans and with other agents" - Sairamesh and Kephart (1998, pp. 28)

In recent years, we have witnessed an increased adoption of task automation and decision making using algorithms in retail and many other industries (Steiner, 2012). Adhering to this trend, recent research has begun to explore the use of smart algorithms for pricing goods and services in electronic markets. Determining the "right" price to charge a customer for a product is a complex task, which not only involves evaluating the company's operating cost, but also how much the current consumer values the product and what future demand might be. In addition, the increased price transparency enabled by price comparison providers has further complicated competition and dynamics between competing retailers. As a result, to succeed at pricing products "right", companies require a wealth of information about their customer base and competitors, and must be able to adjust their prices at minimal cost (Elmaghraby and Keskinocak, 2003).

We are all affected by bounded rationality, and our ability to extract knowledge from large amounts of information is limited. As a result, singlehandedly relying on managers deciding what prices should be on a day to day basis might not be the best approach. Therefore, as predicted by Sairamesh and Kephart (1998), we are currently experiencing an ever increasing outsourcing of pricing decisions to smart and sophisticated algorithms. Consequently, the need for developing such algorithms and understanding what implications they might have on a market has gained traction in recent years.

In accordance with this trend, this thesis will expand the current literature of dynamic pricing performed by algorithms and provide its reader with an overview of previous research. The following chapter will in greater detail explore the background and motivation for dynamically pricing goods using algorithms, before presenting the main contributions and limitations of this thesis.

1.1 Background

The primary goal for any retailer is to maximize profits and shareholder value. This is a challenging task which includes many strategic evaluations regarding factors as the company's business model, value chain and marketing efforts. Ultimately, it is the retailer's ability to widen the gap between its costs and revenues that dictates how profitable the firm will be. Thus, one approach for maximizing profits would be to find the optimal price to charge, and consequently, the optimal number of goods to sell to consumers in order to maximize revenue.

In a simple world with perfect information, finding this optimal price is straightforward. Anyone with basic knowledge of microeconomics can calculate what the optimal price must be to maximize revenue given some demand curve in uncomplicated constructed markets. Simplifying assumptions and the introduction of homo-economicus, abstract pricing decisions into simple algebraic equations. However, as uncertainty and more realistic assumptions are introduced, finding the optimal price is more of a challenging task. The real world consumers do not translate well into smooth, differentiable curves in a diagram. Uncertainty regarding demand and competitive landscapes imposes some serious challenges.

Pricing is a strategic decision, and for any retailer, it has a profound effect on their business and profitability. Price is one of the most efficient parameters a company can utilize to influence or control the demand for their products. It enables companies to manipulate their consumer's preferences to maximize their revenue and helps regulate inventory and pressure on production. Because pricing is such a fundamental component of operating a service or manufacturing company, there exist an enormous amount of research trying to find the best approach.

Before the introduction of the Internet and electronic commerce, pricing was a rather static task. The retailer would perform simple calculations and set its prices according to a fixed pricing tactic. These days, however, the shear information available and our ability to process data and perform digital tasks, has led the science of pricing to a whole new level. Sellers are now able to take advantage of fluctuations in demand and customers' willingness to pay by altering the price of their products dynamically. Today retailers can respond to changes in the marketplace almost instantly, and the reduced cost of posting prices in a digital world has opened up a whole new area of research, namely that of dynamic pricing.

Dynamic pricing is commonly defined to be a pricing strategy where the price of the same unit of a product or service can change over time (Dimicco et al., 2003; Chan et al., 2004). As prices become more dynamic, the need for developing systems for automating price reconsideration techniques increases. Manually plotting in prices and considering changes, seems far less efficient than having an algorithm perform the job.

Today, nearly all online retailers of some size use algorithms to price their products. Beneath their smart looking websites, lies sophisticated software programs constantly monitoring competitor's prices and market trends, and evaluating whether prices should change or not. With a growing number of dynamic pricing optimization software providers (DPOSP), such as Boomerang Commerce and Wiser, the e-business market is evolving into a battlefield where whoever has the best algorithm wins.

Having an algorithm making decisions for the seller can be an incredibly powerful tool when used correctly, however, engineering such algorithms is a highly complex task. Because factors like competition and customer behavior are uncertain, changing and mostly unknown, we need to find algorithms that can deal with this complexity and uncertainty, by learning the "parameters" of the market as they go.

The science of artificial intelligence was born in 1956, and colloquially, the term is applied when a computer uses cutting-edge techniques to competently perform or mimic "cognitive" functions that we intuitively associate with human minds, such as "learning" and "problem solving" (Russell and Norvig, 2010). It is from within this highly technical and specialized field of research that we find one possible approach for the dynamic pricing of products, called machine learning.

Machine learning can be described as a field of study that gives computers the ability to learn without being explicitly programmed, and it explores the construction of algorithms that can learn from and make predictions on data (Simon, 2013). Machine learning algorithms should be able to efficiently automate pricing decisions to maximize profits, as they can perform pricing decisions using sophisticated calculations and predictions, by putting all available data into perspective, and change their pricing strategy to best adapt to a dynamic environment. Consequently, a considerable amount of research trying to solve these hard pricing problems by machine learning has been performed, and many predict that pricing done by machine learning algorithms is the future of pricing (Sairamesh and Kephart, 1998).

However, some issues arise when considering machine learning for dynamic pricing. How can a seller have confidence that the algorithm is succeeding with its learned price policy, and what might happen when multiple sellers start employing machine learning algorithms? These questions are not easily answered. Therefore, a thorough understanding of the algorithms and how they adapt, interact, and perform in a market is needed to address these issues. Obtaining this knowledge from real-world implementations, however, might not be the best approach. A retailer is likely to be reluctant at utilizing an algorithm that has not previously been extensively researched. Hence, we need another method that allows us to test, evaluate, and predict algorithm behavior without real-world implementation.

Real-world marketplaces are often hard or even impossible to represent using a fully analytical approach, due to their complexity, dependencies and uncertainties. To succeed with an analytical approach, overly simplifying assumptions have to be made, which deprive us of the possibility of replicating an actual market. Consequently, numerical analysis using simulated marketplaces have revealed themselves as a very useful approach in that they enable the modeling of more realistic diverse and complex markets. Several researchers within the revenue management literature have made significant contributions to the problem of dynamic pricing by producing tangible numerical results (Dimicco et al., 2003), and especially in the case of machine learning models, the use of simulations has been widely adopted. Furthermore, simulation allows us to rapidly test, improve and comprehend different algorithms, and although we are not capable of fully replicating real-world markets, discoveries made from simulations are vital for understanding the implications and possibilities of algorithm-based price policies.

1.2 Main Contributions

The main goal of this thesis is to provide a numerical analysis and evaluation of how dynamic pricing algorithms perform when faced with a heterogeneous market using a simulation-based approach. To achieve this goal, we first need to present an overview of dynamic pricing and an introduction to dynamic pricing by machine learning. Focusing on existing literature, this thesis aims to compress the fundamentals of dynamic pricing, including its history, different models, and conditions for succeeding with dynamic pricing. Then by examining different machine learning approaches to dynamic pricing problems, we hope to gain insight regarding their value and applicability in real world situations.

Studying previous research on this topic, it seems that a conclusive review of the literature has not been performed. Researchers working with machine learning and dynamic pricing use different models, methods, assumptions and performance metrics, making it hard to compare different papers (den Boer, 2015). Still, although variations are present, we believe there exist similarities waiting to be found. By developing a generalized framework for analyzing literature, focusing on the market models used for simulating machine learning algorithms, we aim to generalize, find trends and identify weak spots in the existing literature. Traces of such a framework is present in the literature, as many researchers build on the works of others, yet to our knowledge, no one has attempted to define a generalized framework to comprehend the literature better.

We aim for an original approach for simulating dynamic pricing performed by machine learning algorithms in a market consisting of both myopic and strategic customers. Furthermore, we strive for a realistic market representation by considering demand uncertainties and limited information. Also, by developing a model that enables competition between multiple machine learning agents utilizing different pricing strategies, we seek to gain a numerical understanding of how these algorithms might adapt, interact and affect each other's dynamics and performances when employed in an electronic market. Hence, the main contributions provided by this thesis are:

- A framework for generalizing machine learning literature focusing on marked models used for simulations
- A non-exhaustive list of research is grouped and analyzed within the framework
- A model for simulating dynamic pricing algorithms performance in heterogeneous finite markets with non-linear stochastic demand
- An analysis of how a market consisting of both strategic and myopic buyers might affect sellers' prices and revenues
- A numerical analysis of how Q-learning and neural network pricing algorithms perform when faced with the same underlying market, and in competition with each other
- An evaluation of the real-world application of machine learning algorithms for dynamic pricing from an economic perspective, by considering its added value for sellers

1.3 Limitations

This thesis will primarily focus on dynamic pricing in an electronic marketplace with demand uncertainty performed by machine learning algorithms. However, with the introduction of electronic shelf pricing, some of the material presented might apply to modern brick and mortar stores as well. Furthermore, our intention is not to provide general insights into all current approaches for solving dynamic pricing problems, consequently, many exciting areas of research has been excluded. The available research regarding dynamic pricing is vast, and we will only focus on a small part of this literature. For an excellent in-depth review of available literature concerning dynamic pricing, readers are referred to den Boer (2015).

Posted price mechanisms will be the overall focus of this thesis, and as such, price-discovery methods as auctions will not be discussed. No attempt has been made to evaluate the literature according to its applicability in specific markets or industries, such as telecommunication or electricity pricing. Also, our goal is not to present the full complexity of machine learning algorithms, but rather to introduce some of the most important aspects needed to comprehend how the commonly used algorithms function. Readers looking for a thorough understanding of machine learning and artificial intelligence are referred to Russell and Norvig (2010) and other sources.

1.4 Clarifications

Throughout this thesis the words algorithm, agent, and pricebot are commonly used to address the actual equations or methods applied for implementing machine learning algorithms. Furthermore, we make no attempt to differentiate between buyers, customers, and consumers, and likewise, sellers, firms, and companies, unless explicitly stated.

1.5 Structure of Thesis

To get a better understanding of the concept of dynamic pricing and how pricing can be performed using smart algorithms, this master's thesis includes a comprehensive literature study. Consequently, Chapter 2 through 6 aims at providing its reader with insights of current research and highlights some of the areas in which we find a potential for further research. The majority of this literature study stems from my project thesis "Dynamic Pricing by Machine Learning: An Overview". Having presented an extensive overview of current research and introduced the most important academic terms, we then present our approach for evaluating the performance of dynamic pricing algorithms in a simulated market. In more detail, this thesis is structured as follows.

Chapter 2 provides a historical overview of how the theory of dynamic pricing has evolved from the 19th century until today and gives an introduction to the basics of dynamic pricing. In Chapter 3, four categories of dynamic pricing models are presented to give a better understanding of the existing approaches used for modeling the problem. Chapter 4 presents a framework and introduction to understanding machine learning algorithms for dynamic pricing and how they are evaluated. Then in Chapter 5, some promising articles of machine learning are presented to provide a more in-depth explanation, ending with a categorization of the presented articles in the proposed framework. Chapter 6 contains a discussion and evaluation of the available literature and points towards future work and research.

Then, based on our findings from previous chapters, we look to expand the current literature and introduce our novel approach for simulating dynamic pricing algorithms in Chapter 7. Chapter 8 describes how we have chosen to model our approach, focusing on both seller and buyer behaviors. After that, we simulate a variety of different market scenarios in Chapter 9 to see how the algorithms adapt and perform under various scenarios. Chapter 10 summarizes the findings from Chapter 9 and draws new insights by relating our findings to reality and other literature, in addition to discussing the weaknesses with our approach. Finally, the thesis concludes with Chapter 11, stating the most important discoveries made from the literature study and market simulations.

Chapter 2

Overview of Dynamic Pricing

"Everything is worth what its purchaser will pay for it"
– Pubilius Syrus, First Century BC

The simple, yet powerful quote by Publilius Syrus captures one of the most important aspects of pricing, namely that the price of a product should equal what its buyer is willing to pay. However, as will be explored in this chapter, simply charging each and every customer their willingness to pay is an almost impossible strategy. Hence, simplified strategies have been developed, and by studying price discrimination and the emergence of dynamic pricing from the 19th Century, these issues and how they can be addressed will be exemplified throughout this chapter.

2.1 A Brief History of Price Discrimination

Odlyzko (2003, pp. 356) argues that the; "... incentives to price discriminate and the increasing ability to do so are among the key factors in the evolution of our economy". Price discrimination, sometimes referred to as personalized pricing or price customization, is an intriguing area of pricing in which identical or largely similar goods or services are transacted at different prices by the same provider in different markets and to different consumers. With the increased focus to find optimal pricing policies maximizing sellers' revenue, it is evident that price discrimination is somewhat unavoidable. Consumers tend, however, to be negative towards such pricing schemes and especially towards individual price discrimination. When Amazon.com experimented with charging consumers different prices for the exact same DVD purchased at the exact same time, they experienced an increase in customer rejection (Baker et al., 2001). However, price discrimination might not be all bad, nor is it illegal. Provided of course that prices are not discriminated based on factors such as gender, race and religion. In fact, considering microeconomic theory, perfect price discrimination can help increase firms output and thus maximize economic welfare and result in a Pareto optimal quantity. The following section will, based on Odlyzko's article and other important literature, examine some of the most important evolutions in price discrimination strategies from the 19th century and until today.

2.1.1 19th Century Railroad Pricing

Perhaps one if the greatest inventions of the 19th century, was the introduction of railways. Enabler of connecting geographically separated communities and the start of globalization. Like many modern industries, the railways faced very high fixed costs and low marginal costs. This "business model" creates strong incentives to price discriminate, as the cost of bringing an extra passenger or good is close to zero. Any new revenue gained will instantly help improve the company's average profitability. Despite lacking modern technologies, railways managed to price discriminate on a grand scale. Their solution was to price differentiate between different segments of consumers by versioning their services. In the words of Jules Dupuit (1849);

> "It is not because of the few thousand francs which would have to be spent to put a roof over the third-class carriages or to upholster the third-class seats that some company or other has open carriages with open benches. What the company is trying to do is to prevent the passengers who can pay the second-class fare from travelling third class; it hits the poor, not because it wants to hurt them, but to frighten the rich. And it is again for the same reason that the companies, having proved almost cruel to the third-class passengers and mean to the second-class passengers, become slavish in dealing with first-class passengers. Having refused the poor what is necessary, they give the rich what is superfluous."

This is a great example of the inefficiency created by versioning. Efficiency would be much greater if the railroad provided decent seats to all and instead charged passengers according to their willingness to pay. Needless to say, the railways had no way to determine their individual passengers' willingness to pay at that time. Frequent rider programs were not feasible, and the lack of proper passenger identification made it impossible to sell non-transferable advance purchase tickets with Saturday night stay-over restrictions. Although economically inefficient, versioning did, in fact, help increase revenue, and arguably enabled more people to travel by trains. In addition, many tickets were sold trough brokers who varied widely in price, which resulted in further price discrimination.

Versioning is predominantly a second-degree price discrimination technique, and it is mainly used when sellers are unable to differentiate between different types of consumers. This creates an opportunity for the seller to provide incentives for the consumers to differentiate themselves according to their preferences, usually done by quantity discounts or as in the case of the railways, offering different service quality.

Freight pricing on the other hand, was conducted by the use of explicit price discrimination. Complicated freight classifications, special deals for particular shippers and charging more for short hauls than long hauls became the norm. This created a dynamic market, close to what we consider as dynamic pricing today, that created no outsized profits and appeared to work very efficiently (Odlyzko, 2003).

However, the price discrimination of the railway companies aroused great controversy. Customers, and especially farmers and poor people were rebelling over high prices and terrible service. Further, the Chicago Board of Trade argued that their city was being handicapped by rates for transport to New York that were higher than those from Milwaukee, even though the trains from Milwaukee went through Chicago. As stated by Odlyzko (2003, pp. 363), "the pervasive price discrimination by railroad was undermining the moral legitimacy of capitalism". The United States Congress responded with the Interstate Commerce Act of 1887, enforcing regulations on the railroad's ability to price discriminate.

Regulation did not, however, help reduce average prices much. The consumers got a reasonably simpler pricing scheme, with predictable and seemingly fair prices, but the prices still remained high. As the anti-rail road agitation decreased in the late 1890s, so did the pressure on the railways to lower their prices. Therefore, it seems that it was not the actual price of the fare that caused rebellion, but it was more a matter of how those prices were imposed that mattered.

2.1.2 Airlines and the Introduction of Revenue Management

The development of price differentiation strategies evolved rather slowly from its introduction in the railway industry. While the economy grew and new technologies such as airplanes entered the market, the development of operations research on optimizing revenue and seat reservations was moderate. Nearly all quantitative research regarding ticket reservation control before the early 1970s focused on controlled overbooking. According to McGill and van Ryzin (1999) these calculations depended on predicting the probability distributions of passengers ready for boarding, and provided useful research on disaggregate forecasting of passenger cancellations, no-shows and go-shows. This method provided a moderate level of success and gave the researchers some degree of credibility of reservations control.

It was not until 1972 when Littlewood (1972) wrote his paper on forecasting and control of passenger bookings, that the pricing schemes of the Airline industry evolved significantly. His proposition, otherwise known as Littlewood's rule, said that discount fare tickets should be accepted as long as their revenue value exceeded the expected revenue of future full fare tickets. This simple, yet effective two-fare seat inventory control marked the beginning of yield management and evolved to the revenue management research of today.

Perhaps the greatest pioneer in yield management at that time were American Airlines and their Super Saver fares. Introduced in April 1977, the program allocated reservations to different classes of passengers who competed for space on the same flight. It introduced an effective way for airlines to divide passengers into business, personal and pleasure travelers. Much like the versioning done by the railroads. This allowed for effective adjustment of demand to match the supply of seats. Lower fares stimulated demand for low demand flights while maintaining high profits on popular flights by limiting the number of super saver fares offered (Smith et al., 1992). As stated by Smith et al. (1992, pp. 8) their overall goal was to "Sell the right seats to the right customer at the right prices"

Yield management flourished after the work presented by Belobaba (1987) and the success of its implementation at American Airlines. This provided the industry with a concrete example of the benefits yield management tools can have on the operations of a company (Smith et al., 1992). Still, however, as seen in the literature review conducted by Weatherford and Bodily (1992) the majority of research focused on capacity management and overbooking. Dynamic pricing was little discussed, and their pricing models were more or less fixed, with managers opening or closing different fare classes as demand changed.

Although yield management was not to people's liking, there was no united anti-airline rebellion. The airlines succeeded in presenting their pricing scheme in a somewhat understandable way, presumably having learned from the mistakes of the railways. Consumers were informed that booking a flight early was cheaper than waiting until the last minute. They might not like it, but at least, they understood what was going on, and that seems to have made all the difference.

2.1.3 The Revenue Management Revolution

After its success in airlines, yield management emerged into a variety of other industries and was further developed into what has become known as revenue management. At its core, revenue management tracks historical demand for products and establish future product availability based on demand forecasts to maximize revenue. In nearly all traditional revenue management applications, four main elements dominate the design of a system: (1) the inventory control mechanism, (2) optimization, (3) demand model and forecasting, and (4) interaction with users of the revenue management system (Boyd and Bilegan, 2003).

The boundaries between yield management and revenue management are often ambiguous. Netessine and Shumsky (2002) defines yield management as a somewhat simpler form of revenue management, in that it commonly only focuses on using price for allocating limited resources. Implying that yield management is a variable pricing strategy, based on understanding, anticipating and influencing consumer behavior to maximize revenue or profits from a fixed perishable resource. However, yield management can also be defined in broader terms more resembling revenue management, and as such, we will not pursue a strict use of these terms.

Today, revenue management techniques are used by a variety of different industries, ranging from railways, hotels, cruise lines, rental cars and even golf courses. It has also provided the underlying framework for what has become the principles of dynamic pricing. In a way, dynamic pricing can be regarded as one of the tools in the revenue management tool-box.

Many tend to misunderstand the relationship between traditional revenue management and dynamic pricing. To illustrate a practical example of this, if today you are searching for ticket prices going from Oslo to Trondheim by plane, you might be offered a price of 599 NOK. However, tomorrow the price might have jumped to 1199 NOK, and you could say that the fare price has doubled. However, what seems to be a change in price, might, in fact, be that the fare class selling for 599 NOK has closed.

Therefore, an important distinction to be made is the difference between dynamically adjusting the price of a single identical products and managing the availability of different products using the same underlying resource. This impacts how the problem is modeled and reveals how we will define the difference between dynamic pricing and revenue management. For instance, an airline might first estimate the demand for each class and then supply the price exogenously, meaning that price is not a variable (Boyd and Bilegan, 2003). After this, the airline might employ dynamic pricing on the specific class fare, and adjusting the price according to current demand.

After the introduction of information technologies (IT) and the Internet, the science of revenue management greatly improved. IT enabled greater collection of valuable information such as demand, inventory levels, and competitor strategies, while also being able to process all this data in real time. This made pricing strategies using IT a huge advantage and led the way for extensive development. Managers today are able to act and react dynamically to changes in the market, by adjusting any variable under their control and especially prices. Also, the introduction of web-based sales systems made the implementation of using dynamic prices much easier. In the pre-IT era, re-labeling of prices was a costly and time-consuming process. Today, algorithms and computers can change prices thousands of times each day for a cost of practically zero. An estimation provided by L2 Think Thank found that Amazon.com changes prices approximately 2,5 million times each day (McLean, 2014).

2.2 Introduction to Dynamic Pricing

Den Boer (2015) defines dynamic pricing as the study of determining optimal selling prices of products or services, in a setting where prices can easily and frequently be adjusted. Dynamic pricing includes two main aspects: (1) price dispersion and (2) price discrimination (Narahari et al., 2005). Price dispersion can be spatial or temporal. Spatial price dispersion occurs when several sellers offer a given good at different prices, whereas temporal price dispersion occurs when a given store varies its price for a given good over time. Aspects of price discrimination have been previously touched upon and are assumed to be known.

A considerable amount of research has been done on the topic of dynamic pricing from different scientific communities, including operations research and management science, marketing, economics, econometrics and computer science. Consequently, the methods and underlying problem definitions greatly vary. For instance, dynamic altering of prices can be used for learning demand and characteristics of consumer behavior, or as a tool in revenue management where learning demand using prices gets combined with capacity-based or inventory control problems. Readers looking for a more detailed review of current dynamic pricing literature are referred to den Boer (2013).

The methods and policies used in pricing greatly vary, however, they often fall into two broad categories: posted-price mechanisms, and price-discovery mechanisms. A posted-price mechanism involves selling goods at a “take it or leave it” price determined by the seller, whereas price-discovery mechanisms regard prices determined by processes such as auctions. In the past, companies would set a fixed price for their products that would last for a long period. This is often referred to as posted static prices. Dynamic posted prices are also “take it or leave it” prices, but the prices are dynamically altered by the seller over time, also known as dynamic intertemporal pricing (Elmaghraby and Keskinocak, 2003).

In their research, Gallego and van Ryzin (1994) analyses the problem managers face when selling a given stock of items within a deadline in a monopolistic market. They find an upper bound on the expected revenue for general demand functions by analyzing the deterministic version of the problem, and use this bound to prove that simple, fixed price policies are asymptotically optimal and outperform dynamic pricing as the volume of expected sales tends to infinity. However, a significant amount research has evolved to offer solutions to more realistic models since 1994, including product substitution effects, consumer inertia, competition and price uncertainty (Sen, 2013). Most of the current literature implies that the adoption of dynamic pricing policies can provide significant gains over fixed-price policies, especially when demand is uncertain. Also, the ever increasing adaptation of dynamic pricing policies in retail and other industries provides some evidence that the future of pricing is dynamic.

2.2.1 When Will Dynamic Pricing Succeed?

Dynamic pricing is arguably a price discrimination strategy (often referred to as price customization) aimed to maximize a seller’s revenue by charging different prices to end consumers based on a discriminatory variable. However, as explored in 2.1.1, price discrimination is not always welcomed by the consumers. In an attempt to explore when applying dynamic pricing makes managerial sense, Reinartz (2002) propose that there are five conditions that must hold if price customization is to work. These conditions are:

1. Customers must be heterogeneous in their willingness to pay
2. The market must be segmentable
3. The potential for arbitrage is limited
4. The cost of segmenting and policing must not exceed revenue increases due to customization
5. Violations of perceived fairness must not be committed

Condition 1: Customers must be heterogeneous in their willingness to pay

Perhaps the most basic condition is that the consumers must be willing to pay different prices for the same goods or services. If this is not the case, the gains of implementing complex pricing algorithms would be minimal as the prices are likely to be static. Differences in consumers' willingness to pay can be explained in numerous ways, including, the opportunity cost of time, the product risk different consumers are anticipating, the need to perform price-search efforts and varying perceptions of brand names and value.

In most markets heterogeneity seems to apply (Reinartz, 2002). It is especially evident in markets where branded and generic products co-exists, such as apparel, groceries and electronics. Further, markets incorporating a wide range between the lowest and highest price for a particular product, and those where price-search efforts are rewarded, tend to have heterogeneous consumers. However, in commodity markets, consumers are likely to be more homogeneous, and their willingness to pay is unlikely to vary much.

Condition 2: The Market Must be Segmentable

Effective price discrimination is hard to reach without the possibility of identifying different groups of consumers. Segmentation enables a firm to reduce complexity by specifying actions for a larger portion of consumers, and not just individuals. This is achieved by making certain assumptions about each group's willingness to pay. For instance, a consumer might sort products according to price before choosing (price sensitive customer) or might choose to make a decision based on non-price attributes like brand or quality (price-insensitive consumer). Commonly used segmentation techniques involve loyalty programs and self-selection.

During the last 20 years firms' ability to segment markets have greatly improved as a result of the increasing web-based commerce. This has enabled tracking of individual customer purchases through the Internet, and thus building up profiles over time, allowing for marketing campaigns directed to each consumer, or more likely, to segments having different willingness to pay. In addition, by examining consumers search behavior, sellers can infer consideration set marketing, which is the set of products that a consumer might consider to be nearly equal before purchasing (Reinartz, 2002). Knowing what alternatives a consumer evaluates, often reveals information about their willingness to pay. For instance, two customers looking for a jacket might end up buying a high-quality Arcteryx jacket. However, if one of them bought it because it is a premium brand, and the other evaluated several other jackets, but chose that particular one because it provided the highest quality/price tradeoff, then presumably, the first consumer has a higher willingness to pay than the latter one. Consequently, tracking the browsing path of consumers concede valuable information about their consideration set.

Condition 3: The Potential for Arbitrage is Limited

A firm's ability to charge different prices to its customers is dependent on limited arbitrage. Opportunistic consumers should not be able to buy a product at a lower price and then resell it to a consumer having a higher willingness to pay. If arbitrage is present, the firm would have no incentive to price discriminate customers as it would be more effective just to charge one static price to all its customers. Airlines solved this issue by imposing restrictions on cheap fairs, in that the ticket is made out for one specific customer, and the cost of changing the tickets exceeds the gains. Arbitrage is often regarded as limited because the cost of resale does seldom cover the benefits of doing so.

Previously, brick and mortar retailers were able to charge different prices according to geographical regions. A product might not cost the same in Oslo as in Trondheim, but customers typically do not know or search for prices in all regions, hence, arbitrage is not considered an issue. However, the Internet and shopbots (price comparison websites) takes price transparency to a whole new level, and thus online environments can help destroy existing price customization schemes (Reinartz, 2002).

Condition 4: The Cost Cannot exceed Revenue Gained

If there are to be any gains from price customization, the cost of doing so must not exceed the extra revenue gained from it. In the past, many price customization models had to be foregone for this reason. They were simply too expensive to implement (Reinartz, 2002). The airline industry took a great leap of faith when they started investing millions of dollars in their yield management systems, but they have as previously noted been highly successful.

Condition 5: Violations of Perceived Fairness Must not be Committed

The final condition concerns the fairness a consumer perceives when dealing with a seller. Perceived fairness is often thought of as the consumers' perception that both parties in a transaction have gained something from it. It is important for firms and managers to understand the risks of perceived unfairness, as this could greatly harm business as happened to Amazon.com when they experimented with price customization. A pricing program is doomed if the consumers do not feel they are being treated fairly regarding price (Reinartz, 2002). Ultimately, it is the consumers that have the power and any seller attempting dynamic pricing needs to be aware of the profound effects negative word of mouth amongst consumers has on its reputation.

How a consumer defines his or hers perceived fairness is to a large degree dependent on whether the supply of a good or service is limited or not. When buying airline tickets, customers are aware that a particular seat on a flight is in short supply, and acknowledge that the prices of individual seats are sold under varying conditions. On the other hand, a customer buying a computer or groceries would be less forgiving regarding price differentiations as these goods are theoretically available on an unlimited basis. This could invoke a feeling of being exploited by the sellers and that the sellers are profiting at the consumers expense.

Still, there are ways of avoiding the downsides related to perceived fairness even when supply is unrestricted. The most common approach keeps the transaction between the consumers and the firm a private matter. Other approaches include placing a time limit on the products availability or advertising it as a ”perishable”.

2.2.2 Forms of Dynamic Pricing

The applicability and variety in dynamic pricing are wide, and there exist several kinds of models and practices used today. Based on the five conditions of price customization Reinartz (2002) categorizes two main forms of dynamic pricing from a price discrimination perspective: the weak and the strong form.

Weak Form of Dynamic Pricing
The weak form of dynamic pricing involves changing prices over time, but not between customers. This makes the market for a product behave sort of like a stock market, where prices fluctuate according to supply and demand. In markets with a weak form dynamic pricing, the fact that prices fluctuate over time is explicitly and publicly stated, implying that the consumers are aware of what's going on. As a result, consumer bitterness is avoided by the fact that all customers pay the same, provided that they purchase the product at the same time. Reinartz (2002) argues that products in limited supply, perishable or versioned are best suited for this kind of dynamic pricing.

Strong Form of Dynamic Pricing
The strong form of dynamic pricing, on the other hand, involves price changes over time and between different customers. As a result, this extra dimension makes the situation for customers more complex and seemingly random. In this setting, firms tend to withhold their price changes from the consumer (Reinartz, 2002). The underlying reason is that bitterness and lack of fairness are expected if the customers are made aware that prices change not only over time but also across customers. Consequently, if a company is to succeed with a strong form of dynamic pricing it has to be extremely well executed.

2.3 Examples of Dynamic Pricing

We conclude this Chapter by presenting some examples of how dynamic pricing has been implemented in various industries, in addition to presenting some of the dynamic pricing optimization software providers operating today.

Uber is a company that connects drivers and riders through a software application and their goal is to open up more possibilities for riders and more business for drivers (Uber, 2015). Since its introduction in 2010 Uber has used a dynamic pricing policy that they call ”surge pricing”. Put simply, this strategy lets Uber raise its prices, often sharply, during times of high demand, and lowering prices when supply is high. The implementation of surge pricing has been highly successful, predominantly, because it exploits demand while also helping to increase supply. It provides the drivers with additional monetary motivation for driving at inconvenient times when demand are high, like New Years Eve and at 2 a.m. on a Saturday night, thus increasing the service level for riders. However, many riders have complained, accusing the company of profiteering and exploiting its customers, and one customer even called Uber ”price-gouging assholes”. Still, their implementation of dynamic pricing is regarded as highly successful (Surowiecki, 2014).

In addition to using classic yield management techniques segmenting different types of customers, easyJet, like many airlines today, use dynamic pricing to price their seats. EasyJet has even published an explanation of their pricing strategy on their website, arguably to improve transparency between themselves and their travelers to avoid rebelling customers. The seats on a particular flight are priced differently according to the level of demand for seats still to be sold. Simply put, their strategy is; the higher the demand for seats, the higher the price (easyJet, 2015).

Jet.com is an online retailer selling all kinds of products from detergents to computers, and their goal is to “reinvent” e-commerce. They address the issue that, in general, the only way to pay less for an item is to buy in volume or choose the slowest possible shipping alternative available. Thanks to an innovative, dynamic pricing strategy, Jet.com allows shoppers to track how the prices of items in their online shopping basket change in real time. The prices the customer sees are based on their selection of goods and how the composition of items affect the actual cost of that particular transaction, such as product warehouse location, items in a shipment, payment method, returns and more (O’Meara, 2015)

There exist quite a few DPOSPs today, and the most known providers are companies such as Wiser, Boomerang Commerce, Upstream Commerce and 360pi. Here we will take a closer look at how Wiser presents their pricing software WiseDynamic. WiseDynamic has three main goals:

1. Estimate demand using regression models leveraging historic sales data to create a demand estimation engine
2. Maximize profits by a dynamic algorithm aggregating several pricing variables to set an optimal price for the product
3. Continuously recalculate variables to adjust to a dynamic marketplace using a machine learning algorithm

Wiser claim that retailers using WiseDynamic see a 22% increase in sales revenue, 18% increase in conversion rate and 32% increase in bottom-line profit (WiseCommerce, 2015).

Chapter 3

Dynamic Pricing Models

In a world of perfect information, pricing of goods would be more or less trivial. However, in reality, perfect information is a bold assumption and managers are bound to optimize their prices based on limited information. Firms do not know how consumers respond to different selling prices, and thus, the optimal price is hard to find. The problem of dynamic pricing is as such not merely about optimization; it also includes learning about the relationship between price and market response. Usually, this relation is modeled by a demand model or function depending on some number of unknown parameters, whose value can be learned by applying statistical estimation techniques on historical sales data (den Boer, 2013).

There exist a variety of approaches and mathematical models used for computing optimal dynamic pricing policies in a world of uncertainty. The majority of these models are formulated as optimization problems aiming to find the optimal prices to maximize profit. The following chapter will provide insight regarding the most commonly used dynamic pricing models in e-commerce.

In their research, Narahari et al. (2005) categorizes the existing dynamic pricing models into five different categories. These categories are in no way indisputable, and several models are often combined in a given pricing scheme, still they provide a simple and effective overview of current approaches relevant for this thesis. The categories are:

- *Inventory-based models*: Pricing decisions are primarily based on inventory and customer service levels
- *Data-driven models*: Collected data regarding customer preferences and buying patterns are used in a statistical manner to compute optimal dynamic prices

- *Game theoretic models*: Focuses on the interplay between and strategies used by actors in a multi-seller scenario, where the firms compete for the same customers and thus induce a dynamic pricing game among the sellers
- *Machine learning models*: By introducing algorithms that dynamically alters the price of a seller's products in an e-business market, seller's potentially can learn buyer preferences and buying patterns
- *Simulation models*: May use any of the four models above or a prototype system to mimic the dynamics of a more complex system, to provide useful insights about the underlying approximated market

The following sections will explore the top four of these dynamic pricing models in greater detail. We could have chosen to include simulation models as well in this overview, however, it has been excluded due to the following. Simulation models are simply a way of simulating market dynamics in a computer environment, and as such the simulation approach has few other common attributes. It is a commonly used method for evaluating and exploring the other dynamic pricing models, because it allows for numerical evaluation of problems that are far to advanced to be analyzed analytically. Consequently, we will explore the benefits of simulations models through the other dynamic pricing models.

3.1 Inventory-Based Models

Due to its applicability and increasing adoption across multiple industries and markets, inventory-based dynamic pricing models have been researched extensively. Chan et al. (2004) and Elmaghraby and Keskinocak (2003) present a thorough review of inventory-based pricing models for traditional retail markets in their research. A brief overview of some of their most important findings is presented here.

Elmaghraby and Keskinocak (2003) suggest that there are three main market characteristic that influence the type of dynamic pricing problem a retailer faces in the presence of inventory considerations, namely replenishment or no replenishment of inventory, dependent or independent demand, and myopic or strategic customers. We shall now examine these characteristics in further detail.

Replenishment vs. No Replenishment of Inventory (R/NR)

A seller's ability to replenish its inventory during the price planning horizon has a profound effect on its prices. If replenishment is not possible, as is the case for some short life cycle products such as Christmas decorations or seasonal fashion apparel, inventory decisions have to be made up front before the selling season starts, implying that the retailer must make pricing decisions given a fixed amount of inventory. On the other hand, when replenishment is available, additional units are accessible to supply the observed demand and thus a more flexible pricing strategy can be utilized.

Dependent vs. Independent Demand Over Time (D/I)

The consumers demand for a product may change over time, and is often categorized as being either dependent or independent. When considering multiple selling periods, demand is dependent if the product is a durable good or if the customers' knowledge about the product influences their buying decision. By definition, a durable good is a product whose lifetime is longer than the time horizon over which a seller can make price changes. The total demand for a durable good, although it maybe unknown, can be modeled as being fixed with no or limited repeat purchases throughout the selling horizon, implying that "...a sale today, is one less possible sale tomorrow" (Elmaghraby and Keskinocak, 2003, pp. 1289). The customers' knowledge about the product may also affect their buying decision and willingness to pay making demand dependent. For instance, when a new product is released, customers are less informed about the product and its value. Thus customers might be more cautious buying products they have no knowledge about or they might not even know the product exists. As a result, retailers can try to influence future demand at the start of a product's life cycle, using advertising and word of mouth from previous sales to inform customers about its products value. Under independent demand, on the other hand, current sales do not have a negative impact on future sales. This is the case for most non-durable goods, such as necessity items like groceries, where consumers frequently repeat purchases.

Myopic vs. Strategic Customers (M/S)

Consumer buying behavior is commonly classified as being either myopic or strategic. Myopic customers make a purchase immediately if the price of the product is below their valuation (reservation price), and do not consider future fluctuations in prices. Strategic customers account for the future price path before making a purchase and thus induce complexity for the seller's pricing decisions. A seller facing myopic customers does not have to account for undesirable effects of future price cuts on current customer purchases. Whereas, when dealing with strategic customers, the seller needs to consider the effects of future and current prices on its customers' buying decisions.

Moving on, Elmaghraby and Keskinocak (2003) found that the bulk of existing inventory-based pricing models can be partitioned into two categories:

- NR-I-(M/S): No Replenishment, Independent Demand, Myopic (Strategic) customers
- R-I-M: Replenishment, Independent Demand, Myopic customers

When a seller faces a short selling horizon, for instance due to short-life-cycle products, the market is classified as an NR-I market. In this setting, the retailer has a fixed inventory and must decide on the best pricing policy for its selling horizon. Further, NR-I markets are divided into two subcategories, according to whether customers are modeled as strategic or myopic. While in R-I-M markets, replenishment of inventory is available, goods are non-durable, and the customers immediately buy products when the price is lower than their reservation price. The paper by Elmaghraby and Keskinocak (2003) provides an in-depth review of all three categories, and most of their findings are applicable to single seller monopolistic markets.

3.2 Data Driven Models

The social media revolution and our ever-decreasing privacy online has lead to enormous opportunities for revenue enhancing measures. Sites as Facebook, Google, Amazon, and the online retail environment, accumulate enormous amounts of data about its customers that they can leverage to improve their revenues and profits. More and more sophisticated data-mining algorithms are being developed for retailers, enabling them to make use of what has come to be known as "big data". In his survey, Raghavan (2005) presents how web mining applies to improving the services provided by e-commerce based enterprises. The literature regarding data-driven models often focuses on building processes that contribute to delivering value to the end customers, by altering what information is available, predicting click streams and customizing marketing efforts. Also by learning customer buying behaviour from past web-logs, sellers are able to segment their customers more efficiently. As claimed by Adnan et al. (2011, pp.174) "...analysis of the user's behavior would allow e-commerce website developers to increase the user experience and, in return over time, an increase in revenue and overall customer satisfaction.

In regards to data-driven approaches to dynamic pricing, several efforts have been made to transform user data into valuable pricing algorithms. Revenue or yield management in the airline industry are traditionally driven by customer data. Boyd and Bilegan (2003) review successful e-commerce models of dynamic, automated sales enabled by central reservation and revenue management systems. They find that collected data could be used to infer customer purchasing habits and provide a way to better control product sales once these habits were understood. Rusmevichientong et al. (2006) developed a non-parametric, data-driven model to determine optimal dynamic prices for General Motor vehicles, by leveraging data gathered from the Auto Choice Advisor website.

Firms as Facebook and Google are continuously gathering data from its users, trying to monetize on their enormous databases. As for now, their focus has been to provide advertising services to its clients. However, it is likely that it is only a matter of time before these two giants find ways of analyzing and using this data for determining better ways of pricing for its clients.

3.3 Game Theoretic Models

When determining its selling prices, a company often has to consider the selling prices of its competitors. Therefore, it seems only natural to study pricing and learning policies in a competitive market, as neglecting the competitive aspects of the market could have negative consequences (den Boer, 2013). A market consisting of multiple competing rational and selfish agents can be modeled in a game theoretic framework. In regards to dynamic pricing, both non-cooperative game theory and cooperative game theory are relevant for modeling the e-business markets (Narahari et al., 2005). Especially interesting is the area of repeated games with incomplete information, since it should provide a natural framework for studying dynamic pricing in a competitive environment. However, long-term dynamics of repeated games can be very complicated, even without incomplete information. This fact is presumably one of the reasons why the literature on pricing and learning with competition in a game theoretic framework is rather scarce (den Boer, 2013).

3.4 Machine Learning Models

In most markets, demand and supply fluctuate, creating a constantly changing market environment. Predicting all possible future states of such markets is impossible, and available information is limited. As a result, a considerable stream of literature on dynamic pricing has emerged from the computer science and artificial intelligence community. These models enable firms to put available data into perspective and change their pricing strategy to best adapt to the market environment (Narahari et al., 2005). In general, these papers do not attempt to provide a mathematical analysis of the performance of pricing policies. Instead, they are aimed at designing realistic models for electronic markets and subsequently apply machine learning techniques (den Boer, 2015).

The advantages of using machine learning include the possibility to model many of the factors that influence a market, such as competition, fluctuating demand and strategic buyer behavior. However, the main disadvantage of this approach is that the models are often too complex to analyze analytically, and as such, insights on the behavior of different pricing strategies can only be obtained by numerical experiments (den Boer, 2013).

Machine learning models are also often combined with data-driven approaches to determine optimal dynamic pricing. The literature on machine learning pricing is commonly classified into models using a single or multiple learning agents, and it offers a new and exiting approach for automatic pricing decisions. Solving dynamic pricing problems using smart machine learning techniques will be our main focus for the remaining chapters.

Chapter 4

Understanding Machine Learning Models

The following chapter will continue the discussion regarding machine learning models, and go into greater detail by presenting common attributes and methods in the literature. We start by presenting an introduction to the concept of machine learning and the most used algorithms, focusing on approaches used for dynamic pricing. We then conclude with some of the commonly used attributes for replicating a marketplace in this literature. The presented material will later be used in Chapter 5 for classifying and generalizing the studied literature, in addition to providing the foundation for the development of our own machine learning model.

4.1 Overview of Machine Learning

The topic of machine learning originated from the computer science community, and has evolved from the study of pattern recognition and computational learning theory in artificial intelligence. In a way, machine learning is the science of getting computers to act without being explicitly programmed (Coursera, 2015). Since the 1990's substantial amounts of resources and research have been devoted to developing sophisticated learning algorithms, and they have been applied to various fields including spam-filtering, search engines, chess and computer vision.

This thesis will only provide a brief introduction to the science of machine learning, enabling us understand the basic underlying principles. There exists a variety of different models and approaches within this topic, but it is common to categorize machine learning algorithms into three broad categories, based on the nature of the learning signal or feedback available to the algorithm Russell and Norvig (2010).

- **Supervised learning:** Example inputs and their desired outputs are given to the computer by a "supervisor" and the computer's goal is then to learn a general rule that maps inputs to outputs. Commonly though of as function approximation, where x and y observations are supplied and the algorithm has to find a function that approximates $y = f(x)$
- **Unsupervised learning:** No examples are given to the computer and, thus, it is left on its own to find structure in the input data. As opposed to supervised learning, the algorithm is only supplied with x observations, and its job is to find an appropriate function $f(x)$ representing the data, sometimes referred to as clustering.
- **Reinforcement learning:** The computer interacts with a dynamic environment and its task is to reach a certain goal without a supervisor telling it explicitly if it is close to its goal or not. The algorithm is supplied with observations x and z, and will use those to find y and $f(x)$, when $y = f(x)$

The core of machine learning algorithms is that they can generalize from their experience. Consequently, they are often able to perform accurately on new and unseen tasks after experiencing a learning data set. In general, learning data sets are given from historic data or an unknown probability distribution that aims to represent the space of occurrences. The algorithm then has to build a general model of the given space to produce adequately accurate predictions in new cases (Russell and Norvig, 2010). When learning sets are finite, and the future is uncertain, the performance of algorithms can not be guaranteed, thus, their performance is commonly given by probabilistic bounds and expected values.

Furthermore, there are two main approaches for training an algorithm, namely online and offline learning. Offline learning implies that the algorithms only learn from data sets consisting of i.e. historical data, and consequently, do not change their approximation of the target function when the initial training phase has been performed. Whereas online learning implies that the algorithms sequentially update their prediction of the target function when new data becomes available. The two approaches are commonly combined, resulting in algorithms that can be trained, or initialized, according to historical data or data-estimations, before being "released" in an application and subsequently update their initial prediction of the target function as they go.

When considering dynamic pricing problems using machine learning, some important considerations can be made. First, we should represent the market in which the algorithms are supposed to operate using some probability distribution, or model, that aims to capture realistic market dynamics. Then, provided that the designer has understood the environment, different machine learning algorithms can be developed, tested and trained based on this understanding. Typically, the performance of the algorithms is evaluated by numerical simulation of a market aiming to represent reality, here referred to as a market model.

Based on previous research in this field, the following section will provide a general introduction to machine learning algorithms used for the dynamic pricing problem. Then a general framework for analyzing the different market models and assumptions used in the majority of literature is presented. This chapter will thus provide us with the necessary understanding for finding similarities and variations of the literature in Chapter 5.

4.2 Machine Learning Framework

In order to understand the basics of machine learning, this chapter will present some of the overall approaches and assumptions used for the different types of machine learning. Our goal is to provide sufficient information enabling us to explore relevant features of machine learning for dynamic pricing in the literature, without going into to much detail. We will pay most attention to reinforcement learning, as it is a frequently used approach, but we will also present some more details of other common approaches.

4.2.1 Reinforcement Learning

A supervised learning agent needs to be told the correct move for each state it encounters, but such feedback is seldom available (Russell and Norvig, 2010). Reinforcement learning (RL), however, is a method used by machine learning agents to learn what to do in the absence of labeled examples of what to do. Consequently, RL-agents are more sophisticated and closer to what some consider to be artificial intelligence. The majority of literature regarding machine learning for dynamic pricing uses some form of RL-algorithm, and as such, we will present some insights on what they are and how they work.

Decision Processes

In order to employ a reinforcement learning algorithm, one has to represent the world by some model in which calculations can occur. This section will present the three most common representations of “reality” used in machine learning.

Markov Decision Process
A Markov Decision Process (MDP) is a sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards. Simply put, it is a way of simplifying reality by assuming that only the present matter. It consist of a set of states s and an initial state s_0, a set of actions $A(s)$ in each state, a transition model $P(s'|s, a)$, and a reward function $R(s, a, s')$. The transition model is Markovian on the basis that the probability of reaching state s' from s depends only on s and not on the history of earlier states. $P(s'|s, a)$ is the probability of reaching state s' if action a is performed in state s (Russell and Norvig, 2010). Relating to our introduction to RL in section 4.1, one can think of the parameters x, y, z and $f(x)$ as s, a, r and π^* respectively.

The solution to an MDP must specify what the agent should do for any state that the agent might reach, and is commonly referred to as a policy. Policies are usually denoted by π and $\pi(s)$ is the action recommended by the policy π for state s. If regardless of the outcome of any action, the agent always knows what to do next, the agent is said to have a complete policy.

A given policy is always started from an initial state and each time a given policy is executed, the stochastic nature of the agents environment may lead to a different environment history. Consequently, the quality of a policy is measured by the expected utility of the possible environment histories generated by that policy. In this setting, an optimal policy, denoted π^*, is the policy that yields the highest expected utility or optimize the long-term expected rewards. Utility in this setting is the immediate reward in s plus the future expected rewards when following policy $\pi(s)$.

MDPs are assumed stationary with an infinite horizon, in that the transition model, or the rules, does not change with time. Hence the optimal policy $\pi(s)$ will always return the same action a, regardless of what point in time state s is evaluated. Because the time horizon is infinite, the value of the sequences will also be infinite. To cope with this issue, when designing an algorithm for an MDP, a discount factor $0 < \gamma < 1$ is introduced enabling us to find a finite value to an infinite sequence. MDPs can be solved to optimality by linear or dynamic programming in polynomial time.

Partially Observable Markov Decision Process
The MDP assumes that the environment is fully observable, that is the agent always knows which state it is in, whereas in the Partially Observable Markov Decision Process (POMDP) the agent does not necessarily know which state it is in. As a result, it can not execute the action $\pi(s)$ recommended for that state. Moreover, the utility of state s and the optimal action in s depend not just on s, but also how much the agent knows when it is in s. Hence, the POMDP is usually viewed as considerably more difficult to handle than MPDs, but they are again much more similar to our real world (Russell and Norvig, 2010).

The POMPD shares many of the same elements as an MDP, the transition model $P(s'|s,a)$, actions $A(s)$, and reward function $R(s,a,s')$, but in addition it has a sensor model $P(e|s)$. The sensor model specifies the probability of perceiving evidence e in state s, which is used for estimating which state it is in.

A key concept for solving partially observable problems is the belief state, which represents the agent's current belief about the possible physical states it might be in, given the sequence of actions and percepts up to that point. Considering the POMDP, the belief state b becomes a probability distribution over all possible states, and $b(s)$ gives the probability assigned to the actual state s by belief state b. Hence, the agent can then calculate its current belief state as the conditional probability distribution over the actual states given the sequence of percepts and actions so far.

The fundamental insight required to understand POMDPs is that the optimal action depends only on the agents' current belief state. Or put differently, the optimal policy can be described by a mapping $\pi^*(b)$ from belief states to actions, thus it does not depend on the actual state the agent is in. However, actually finding the optimal policy for a POMDP is virtually intractable (Braziunas, 2003)

Markov Games
Markov games, or stochastic games, represent a generalization of both Markov decision processes and repeated games. As with MDPs, we model our world by states, actions, transitions and rewards. However, because we now are introducing multiple players, we have to adjust some of our previous assumptions. Each player $i = 1, 2, ..,$ choose their own actions $A_i(s)$ and may or may not have the same set of actions available. The transition function depends not only on their actions, but also the actions of others, that is, $T(s, a_{(1,2,..)}, s')$. So does their rewards, now given by $R_i(s, a_{(1,2,..)}, s')$. This strategic game between agents adds another dimension of complexity and greatly increase the difficulty of finding an optimal policy.

Further, for repeated games, it is common to include some discount factor to represent the probability of meeting opponents again, and this can easily related to the expected number of rounds, which, is arguably the same mechanism used in the MDP to find a finite value to an infinite series. In a way, an MDP is a narrowing of a stochastic game, where the players are fully unaffected by the action and transitions of others. The concept of fully and partially observable processes, translate to the game-theoretic terms of perfect information and imperfect information. Markov games are generally very hard to solve.

Algorithm

Before presenting the literature regarding reinforced learning, a quick intro to the general algorithm principle is provided. Readers looking for a more thorough review and explanation of the procedure in reinforcement learning algorithms are referred to Schwind (2007) and Russell and Norvig (2010). There exists many approaches to reinforcement learning and arguably all RL-approaches have the ability to solve an MDP by learning. The notion of solving an MDP relates to finding the optimal policy, often when the transition function and reward function is unknown. Hence, an RL-algorithm are able to estimate the reward and transition functions of an MDP by using observed states and rewards. Schwind (2007) provides the following explanation:

- The reinforcement learning agent is connected to its environment via sensors
- In every step of interaction the agent receives a feedback about the state of the environment s_{t+1} and reward r_{t+1} for its latest action a_t
- The agent chooses an action a_{t+1} representing the output function, which changes the state s_{t+1} of the environment and thus leads to state s_{t+2}
- The agent receives new feedback from reinforcement signal r_{t+2}
- The objective of the agent is to optimize the sum of the reinforcement signals in the long run

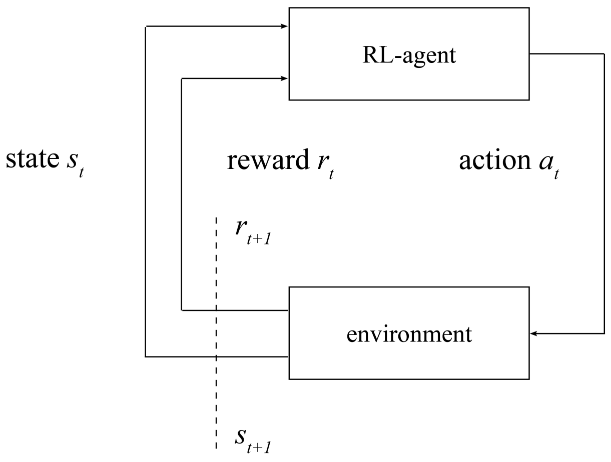


Figure 4.1: An illustration of a how a reinforcement learning agent interact with its environment

The overall procedure of the reinforcement learning can be abstracted to a learning process where the algorithms are rewarded for favorable moves and punished for unfavorable moves. By keeping track of what moves give rewards by a action-utility function, the algorithm navigates in its solution space trying to maximize its reward. The most used RL-algorithms in dynamic pricing, are the family of Q-learning algorithms, which have been proven to converge to the optimal policy for an MDPs if the state-action pairs are visited infinitely often (Littman and Charles, 2015).

Exploration vs. Exploitation

An issue of machine learning that often reveals itself is the trade-off between exploration and exploitation. During the training period, the algorithms are supplied with information and builds rules to approximate this data. However, after learning from a training set of data, the algorithm now has a trade-off to make, should it exploit what it has learned so far, or should it perhaps explore more of the solution space and avoid being trapped in a local maximum.

This problem of exploration and exploitation is often exemplified by the multi-armed bandit problem (Xia and Dube, 2007). Imagine a gambler facing multiple slot machines, and he has to decide what machines to play, how many times to play each machine, and in which order to play them. Each machine has its own distribution of random rewards, and the objective of the gambler is to maximize his sum of rewards earned from his choice of lever pulls. Choosing what machines to play for optimizing the gambler's expected payoff is not straightforward.

In regards to dynamic pricing, the issue of exploitation and exploration is important when considering profits and learning. Should the algorithm maximize it profits given what it know now so far, or should one look to delay rewards and go for choices that are not immediately optimal, but are assumed to improve long-term expected profits. These questions are not easily answered, but they are a fundamental part of designing a reinforcement learning algorithm.

4.2.2 Supervised Learning

The approach of supervised learning is much wider than that of reinforcement learning, in that it is a general method of function approximation. Hence, there exists a large variety of different supervised learning algorithms, many of whom might easily be converted for use in unsupervised learning. However, the approach of neural networks is an interesting one that is purely supervised learning, and that has considerable potential for the dynamic pricing problem.

Neural Networks

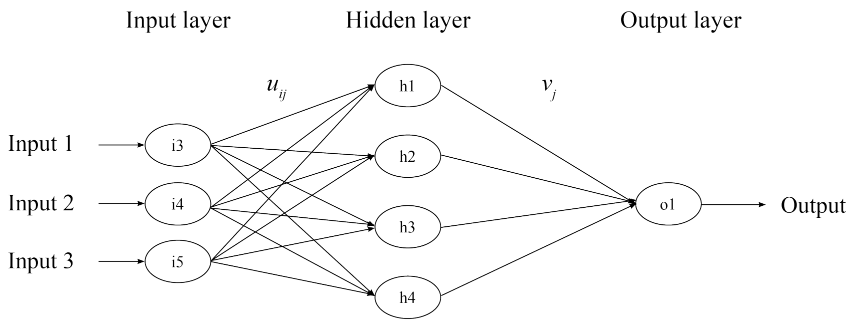


Figure 4.2: The structure of a simple neural network

The modeling approach of Artificial Neural Networks (NN) is inspired by the way biological nervous systems like our brain process information. NNs consist of a collection of units connected together, whose properties are determined by their topology and their "neurons". Neural networks are predominantly used to estimate or approximate functions that depend on a large number of inputs and that are generally unknown. They are commonly presented as systems of interconnected "neurons" exchanging information between each other. Numerical weights are applied to each connection of nodes and tuned based on the algorithms experience, thus making neural networks adaptive to inputs and capable of learning. The networks always consist of an input and output layer, and may incorporate a single or multiple hidden layers, depending on the data and complexity to be approximated. An illustration of a neural network with one hidden layer is presented above in Figure 4.2.

4.2.3 Other Algorithms

The following algorithms have all been used for solving dynamic pricing problems, but to a lesser extent than the previous mentioned ones. These can be classified as unsupervised or supervised learning, depending on the training set or input provided.

Evolutionary Algorithms

The methods of evolutionary algorithms are inspired by biological evolution, in that they use mechanisms of reproduction, mutation, recombination and selection to generate new and better solutions. Typically, an evolutionary algorithm starts by randomly generating a population of solutions. Then each solution is evaluated, and a subset of the better solutions are used to generate a new population, commonly referred to as a child population. The child population can be generated by different techniques, depending on what type of evolutionary algorithm one chooses to use (Shakya et al., 2012). Perhaps the most common algorithm is the Genetic Algorithm (GA), which uses a crossover and mutation approach. This involves combining two random parent states hoping to create an even better solution, and then with a small probability performing some random incremental mutation on the new solution. This process is continued until a satisfactory solution is found, or some termination criteria is met (Russell and Norvig, 2010).

Derivative Following & Goal Directed

The Goal Directed and Derivative Following algorithms represents a simple form of unsupervised machine learning, and have been especially developed for the dynamic pricing problem. They both make incremental exploratory price adjustments in an attempt to learn the demand in a market. These adaptive algorithms make no assumptions regarding the buyers behavior or the type of buyers in the marketplace.

The Goal Directed strategy adjust prices in order to reach a goal provided by the seller. In the presence of constrained product availability, this goal is commonly defined as selling the entire inventory by the last day of the time period, and not before. It does this by decreasing prices when sales are low, and increasing prices when sales are high, with the goal of selling to the highest paying buyers on each individual time step (Dimicco et al., 2001).

The Derivative Following strategy adjusts its price by simply evaluating the revenue gained from the previous time step as a result of the previous time step's price change. If the price change in the previous time step gave more revenue per good, then the algorithm makes a similar change in price. However, if the opposite is true, then the algorithm makes an opposing price change. Consequently, the seller goal is to sell at the highest price each time step that still generates sales (Greenwald and Kephart, 2000)

4.3 A Generalized Market Framework

When applied to a dynamic pricing problem, it is hard and often impossible to prove analytically that the above mentioned algorithms result in an optimal policy. Therefore, the standard method of evaluating their performance is to simulate a market environment and perform numerical experiments. Only then can we evaluate how the algorithms perform. If we are to grasp the algorithms better, we need understand these market models. Hence, the following section will provide an overview of the assumptions and properties used when simulating markets in the machine learning literature.

It should be noted that some important market properties like those of profit models, costs and demand experienced by each seller in competition will not be presented, as it has been proven difficult to find common ground in the studied literature regarding these factors.

4.3.1 Number of Players

The number of players determines the overall structure of the market. In general, we have two categories of players in these market models, namely sellers and buyers. One usually considers an economy in which a good is offered for sale by S sellers and is of interest to B buyers. In nearly all models, the number of buyers greatly outnumbers that of the sellers, that is $B \gg S$. The number of buyers may be given explicitly or presumed infinite.

By altering the number of sellers, various levels of competition are introduced. In accordance with economic terminology, $S = 1$ implies a monopoly market, $S = 2$ represents a duopoly market, and $S > 2$ is known as a oligopolistic market. At some point the number of sellers may transition a market from oligopolistic to perfect competition. However, no research regarding perfect competition and dynamic pricing has been found, hence we will regard all markets with $S > 2$ as oligopolistic markets.

In addition, one needs to consider the number of learning agents, or price algorithms, in the market denoted by L. Thus, a market in which a single seller employs a price algorithm is a single learning agent market, whereas markets with $L \geq 2$ implies that there are multiple learning agents present. It worth noticing that the number of sellers and learning agents does not need to be equal, and that one can compare the performance of other more rudimentary pricing strategies to those of machine learning.

4.3.2 Time & Arrival

By definition, dynamic pricing involves changing prices over time, thus, one needs to model a somewhat realistic dynamic market in which the actions of the players occur at some point in time.

The set of players each has their own processes relating their action to the time frame. In the case of buyers, we are often interested in modeling the arrival rate of buyers. By which we mean, at what rate ρ_b is the buyers arriving at our "shop" with a goal of buying a good. The process that has been widely adopted throughout the literature is a Poisson process. This is a commonly known process which has the property that each point is stochastically independent of all the other points in the process. The rate of arrival in a Poisson process is generally denoted λ, and the expected number of arrivals N in a given time interval t is given by $E(N(t)) = \lambda t$. Poisson processes may be homogeneous or in-homogeneous, depending on whether λ is fixed or dynamic. Other used approaches for modeling arrival include having all customers arrive in each time step or drawing the number of arrivals from a uniform probability distribution.

On the other hand, the sellers are always staying in the process and thus have no property of arrival. However, when modeling sellers one has to take into account at what rate ρ_s the sellers are able to reconsider (and potentially reset) the prices of the goods they are offering. Moreover, in markets with multiple sellers, one needs to represent the interplay between sellers. Should all sellers have the possibility to update their prices simultaneously, or do they update sequentially or randomly distributed throughout the interval? If sellers update their prices sequentially, the common assumption is that sellers are able to infer information regarding its competitors current prices before posting their own.

4.3.3 Demand & Utility

Perhaps the hardest factor to present realistically in a model is the behavior of the buyers and their buying patterns. In section 3.1, the properties of dependent and independent demand were discussed, however in markets where multiple goods are offered one might also consider the interdependency of demand. Interdependent products can be defined as those whose demand are affected by the prices of other products and services (Rana and Oliveira, 2015). For instance, changes in the price of a pair of shoes might affect the demand for a matching belt or shoe polish. Consequently, the literature consists of models considering either a single or multiple different goods for sale that might be homogeneous or heterogeneous.

In addition, some models attempt to segment the market by applying some preference to each segment and providing some distribution of the various segments. Others treat the market as a single unity. Examples of such segmenting will be further explored in the Shopbot and Shoppers & Captives models below. The properties of myopic and strategic buyers have been discussed in section 3.1, and will not be repeated here.

Readers familiar with the literature regarding dynamic pricing and machine learning will note that the our definition of the Shopbot, Price-Quality, Shoppers & Captives and Price Information models, is not analogous with the way they are presented by their original authors. We have found that most of the literature implements only parts of these models, and hence we have extracted the most relevant aspects for our analysis. Implying that some aspects originally part of the actual models, like those representing experienced demand at each seller has been neglected to make the overall assumptions of the models more widely applicable.

Shopbot Model
In a Shopbot model, first presented by Greenwald and Kephart (2000), a single homogeneous good is offered for sale by S sellers. The utility provided by the good for buyer b, is a function of price commonly given by:

$$u_b(p) = \begin{cases} v_b - p & \text{if } p \leq v_b \\ 0 & \text{otherwise} \end{cases}$$

implying that a given buyer has positive utility if and only if the seller's price is less than the respective buyer's valuation of the good. In all other cases, the buyer has a utility of zero. The Shopbot model does not assume buyers to be utility maximizing, instead, they use a set of search rules taken from a fixed sample size when selecting which seller to buy from. Buyers are divided into i types, where $0 \leq i \leq S$, and searches for the lowest price among i sellers chosen at random from the set S. Buyers of type $i = 1$, $i = 2$, and $i = S$ are referred to as employing Any Seller, Compare Two or Bargain Hunter strategies. The fraction of buyers utilizing a given strategy is exogenously determined and satisfies $\sum_i w_i = 1$. The Shopbot model is as such a simplistic representation of buyer behaviors, where factors as seller quality, delivery times, and other preferences are not considered.

Price-Quality Model
The Price-Quality model was first described by Sairamesh and Kephart (1998), and considers a market in which each seller offers a single product or service. The good may have a number of different attributes, each with several discrete possible values or a continuous range of possible values. However, for simplicity it is usually assumed that the preferences of buyers are correlated in such a way that there exists a universally agreed-upon mapping that transforms a good's set of attributes and values into a simple

scalar named quality. This is often referred to as a form of vertical differentiation, exemplified by: given two distinct products, if they where sold at the same price, then all consumer would choose the same one (the "higher quality" product) (Shaked and Sutton, 1987).

The buyers are informed of the price p_s and quality q_s of goods offered by seller s, and the sellers are assumed to report their quality honestly. Each buyer b has a utility function dependent on both price and quality $u_b(p, q)$ that might be given by:

$$u_b(p,q) = \begin{cases} \gamma_b(q - \bar{q_b}) + (1-\gamma_b)(v_b - p) & \text{if } v_b \geq p \text{ and } q \geq \bar{q_b} \\ 0 & \text{otherwise} \end{cases}$$

Based on its utility function, each buyer will select the single seller s that maximizes their utility, provided that the utility is positive, and then purchase a unit at the price of p_s. The parameter v_b is the buyer's valuation and $\bar{q_b}$ its quality floor, γ_b represents the buyer's bias towards price or quality and ranges between 0 and 1. For instance, a buyer with $\gamma_b = 1$ is at the extreme limit of quality sensitivity, meaning that he will choose the seller with the highest quality as long as its price is below the price ceiling $\bar{p_b}$.

Shoppers & Captives Model
Raju et al. (2006) presents a model that somewhat combines features of the Shopbot and Price-Quality model. They argue that in an e-business market retailers can learn buyer segmentation from their preferences over price-quantity packages. By utilizing non-linear pricing, sellers are able to segment the buyers by self-selection, and thus employ a form of second-degree price discrimination. For simplicity, the sellers are usually providing two options: one price for a single unit and a buy-two-get-one-free option.

Arguably, one can infer that the consumers selecting the single unit price can be assumed to be not so price sensitive, and that they are willing to pay the high price if any additional service offer (quality) is part of the package. In this setting, this additional service is regularly some lead time commitment in the case of no stock being available. These consumer are classified as captives, denoted b_c, and by realizing over time the additional service benefits from the seller, they will adhere to the same seller for future purchases. Those buyers taking the three for two option are named shoppers, denoted b_s and they are assumed to be willing to bear with inconveniences imposed by the sellers. Their respective utility functions can be represented by:

$$u_{b_c}(p,q) = \begin{cases} (v_{b_c} - p)^{\gamma_{b_c}}(q - \bar{q_{b_c}})^{(1-\gamma_{b_c})} & \text{if } v_{b_c} \geq p \text{ and } q \geq \bar{q_{b_c}} \\ 0 & \text{otherwise} \end{cases}$$

$$u_{b_s}(p) = \begin{cases} v_{b_s} - \frac{2p}{3} & \text{if } \frac{2p}{3} \leq v_{b_s} \\ 0 & \text{otherwise} \end{cases}$$

and following the same parameter definitions as the two previously presented models. The fraction of captives and shoppers experienced by each seller is exogenously determined and satisfies $w_c + w_s = 1$

Price Information Model
The model presented by Kephart et al. (1998) assumes an information-filtering economy consisting of a source agent that publishes information goods (news articles, audio files etc.), C consumer agents that are hoping to buy goods that interest them, and B broker agents that buy selected goods from the sellers and resell them to the consumer agents. In contrast to the vertical differentiation of the Price-Quality model, the information goods sold in this simulated economy is horizontally differentiated. Horizontal differentiation implies that a product that is worthless to one consumer might be priceless to another (Tirole, 1988). Horizontal attributes can include color, size, and design. This model is a bit more complex than the previous models and seemingly less used in the literature, consequently readers looking for an introduction are referred to Kephart et al. (2001).

4.3.4 Product Availability

Arguably, in many cases a customer's willingness to pay is dependent on the good's availability. In some cases the availability of products may also be abstracted to some level of quality for the customers. Further, as mentioned in section 3.1, the impact of product availability has a profound effect on sellers' prices.

Constrained Product Availability

Constrained product availability occurs due to scarce resources and is often materialized as some limited inventory or capacity. The implications of replenishment and no replenishment inventories have already been discussed, nevertheless, we shall present a standard replenishment policy commonly used the literature to gain further insight.

The (q,r) Replenishment Policy
Assuming that each seller has a finite inventory capacity given by I_{max}, they follow a fixed reorder policy for replenishment. When the current inventory level plus the quantity ordered falls below a level r, the seller orders a replenishment of size $I_{max} - r$. The replenishment lead times of the sellers are exponentially distributed with a mean of $1/\mu$. Moreover, each seller incurs

an inventory holding cost rate of H_I per unit good per unit time, and a cost of H_q per each backlogged request. The sellers buy the product at a price of p_c and incurs a reorder cost of p_k.

Unconstrained Product Availability

In the case of unconstrained product availability, I_{max} is considered to infinite, and as such there is no need to consider the amount of products available.

Chapter 5

Examining the Literature

Now that we have an understanding of machine learning and market models, we will embark on more detailed descriptions on how artificial intelligence has been applied to the pricing problem. Based on the literature review conducted by den Boer (2015), the machine learning-principles are classified into nine categories, and the articles listed by den Boer within each category are studied. Further, for each category, the most promising articles (in the author's view) will be discussed in more detail. Each section ends with a brief overview of available literature in each respective "field", and the chapter concludes with an overview and classification of the literature based on the framework presented in Chapter 4.

5.1 Reinforced Learning

The majority of research regarding machine learning models in dynamic pricing problems is focused on some form of RL-algorithm. A possible explanation of why it is widely adopted could be that RL-algorithms have the potential to solve a Markov Decision Process (MDP). RL-algorithms have the benefit of not needing knowledge about the MDP's transition function and reward function, and thus, prove valuable in large MDPs were exact methods become infeasible. Rana and Oliveira (2015) further claims that reinforcement learning is an ideal method for solving the pricing problem in situations where both the probability distribution of demand and the expected revenue gain for taking a pricing action is unknown. Below, two different approaches using reinforcement learning to solve a dynamic pricing problem are presented.

5.1.1 Multiseller Reinforcement Learning

Vengerov (2008) presents a reinforcement learning algorithm that can tune the parameters of a seller's dynamic pricing policy in a gradient direction, in the absence of complete information of the sellers environment. The algorithm is evaluated by simulation of a Grid market environment, where customers choose a seller from multiple Grid Service Providers (GSP) for computing jobs based on properties of price and expected delay, or queue. Utilizing differentiated pricing, the author presents a form of second-degree price discrimination, but instead of reducing the unit price for large quantities, they present two pricing options. One static price for a standard service level, i.e. those who are willing to wait their turn, and one dynamic price for the premium level, i.e. those willing to pay more to be first in line. This resembles the Shoppers & Captives model, apart from the property that the captives in the original model have seller preferences. The problem also shares some of the properties of inventory based models, in that the seller has some limited resource (CPU time), and aims to learn the optimal price for the available CPU time to maximize revenues. Replenishment in this setting could be thought of as when a customer's job is finished and frees the CPU. Given a starting function, the RL-algorithm experiments with the premium price over time, to learn in which states a higher or lower price is more beneficial, and thus employing a weak form of dynamic pricing.

The simulation is performed in a multi-agent Partially Observable Markov Decision Process (POMDP) in continuous time and continuous state-action space. Customers are modeled as myopic, and given a utility function dependent on price and waiting time, with an exponential factor drawn from a uniform distribution. Customers arrive according to a homogeneous Poisson process with rate λ.

Their simulations show that each GSP can significantly increase its profits by offering a premium service level, in addition to a standard one. Further, they show that if all GPS charges the optimal symmetric static price X^* for its premium level, no Nash equilibrium is present. However, if all (7) GSPs uses the presented RL-algorithm to tune their premium prices, the process converges to a Nash equilibrium, which is more profitable than the optimal static price in a GSP duopoly and slightly less profitable in the case of four or more static price GSPs.

The algorithm only converges to the optimal policy if the environment remains stationary, meaning that the probability distributions do not change. Thus, before releasing it in a real deployment, they suggest implementing parameter tuning in a meta-framework, that observes the important customer characteristics and resumes tuning of the pricing policy if a change in these characteristics is detected. By saving previously learned characteristics, the seller can start tuning for a previously observed state if the environment returns to a previously learned state.

5.1.2 Q-Learning and Interdependent Demand

Many retailers offer a variety a products or services that are interdependent, meaning that the demand for one good is affected by the prices of others. Rana and Oliveira (2015) explore dynamic pricing policies for such markets, by considering a dynamic pricing problem of multiple interdependent perishable products, in the presence of no-replenishment inventory constrains over a finite sales horizon. They propose a Q-learning algorithm with eligibility traces $Q(\lambda)$, and models the decisions in a large scale Markov Decision Process (MDP). The policy makes no assumptions about the functional relationship between price and demand, instead the model learns the relationship explicitly using the observation of realized demand to derive an optimal dynamic pricing policy.

Demand varies throughout the day, with some periods of excess demand (peak hours) and some with low demand (off-peak). The customers choose if they want the service during peak or off-peak hours, and are considered myopic. Customers arrive according to a inhomogeneous Poisson process, and their willingness to pay increases exponentially as their decision time approaches expiry. Provided the prices for peak and off-peak service are equal, customers prefer the peak option. Their goal is to find the optimal price for the two scenarios that maximizes revenue, by adjusting prices at each time step, consequently, modelling a weak form of dynamic pricing.

The Q-learning algorithm is first tested by simulation in which the service is delivered at time T, and the selling period ranges from $0...T$. By varying the distribution of peak and off-peak time slots, they analyse the algorithm's performance by comparing it to an individual learning algorithm, i.e. one that does not account for interdependencies. As expected, the interdependent learning policy outperforms the individual learning algorithm.

Perhaps more interesting is their example of a weekly pricing scheme attempting to manage the demand and control "overbooking" using dynamic pricing. By considering a company operating from Monday to Friday with a fixed number of employees, they aim to determine whether dynamic pricing can be used as a mechanism to decrease the cost of having employees working overtime, and keeping the lead time at an acceptable level of quality. Customers may place orders during the weekend, and if the quality is to be maintained, employees might have to work overtime at the beginning of the week. In this case, the sellers objective is two-fold, improve allocation of resources to maximize expected revenue, and meet pre-determined levels of quality. They observe that the optimal prices for the services using interdependent learning is generally higher than those derived using individual learning.

Although the algorithm provides promising results in terms of increasing profits, there are some considerations to be made. When considering five interdependent products and given an initial state, the number of learning episodes needed to reach an optimal policy is close to 8000. However, the authors claim that in the worst case scenario, that is without any prior training, a reasonable policy for the problem can be achieved within 6000 episodes.

5.1.3 Available literature

- Kutschinski et al. (2003) examine several RL pricing strategies and their learning behavior in a co-learning (multiple learning agents) scenario with different levels of competition.
- Sridharan and Tesauro (2002) study the use of a Q-learning algorithm with regression tree function approximation to learn pricing strategies in a competitive marketplace. They compare their Q-learning algorithm's learning ability when using a regression tree function approximation, and evaluates how their approach compares to function approximation done by a neural network.
- Han et al. (2008) develop a multiagent Q-learning algorithm capable of integrating the observed objective actions with the subjective inferential intention of the opponents, implying that the algorithm becomes adaptive, by establishing the decision model of other agents and predicting their next move in advance. Han (2010) later explores a different approach by considering a Bayesian model aiming to account for the general problem of exploration vs. exploitation trade-off. In simulations, the Bayesian RL outperforms the multiagent RL.
- Cheng (2008) examine a Q-learning approach in a POMDP model to find the optimal price for a single-seller selling a given stock of perishable items in a finite sales horizon. In his later work (Cheng, 2009), a revised Q-learning algorithm for finding optimal prices for seasonal and style products is proposed.
- Jintian and Lei (2009) look into a duopoly market in which a simulated annealing Q-learning and win-or-learn-fast policy hill climbing algorithm compete under partial information.
- Raju et al. (2006) use RL techniques in a monopolistic market with inventory replenishment consisting of captives and shoppers with limited available information.

- Schwind (2007) provides a thorough review of RL algorithms for dynamic pricing and automated resource allocation.
- Dogan and Güner (2015) analyze simultaneous ordering and pricing decisions in a multi-seller environment over an infinite horizon. They use an RL-algorithm to analyze the effects of competitiveness and performance in monopolistic and duopolistic markets.
- Könönen (2006) evaluates two competing brokers selling identical products and models the dynamic pricing problem as an asymmetric Markov game. The problem is then solved in simulation by two different reinforcement learning algorithms.
- Collins and Thomas (2012) performs a comparison of RL approaches for solving game theoretic dynamic pricing models.
- Chinthalapati et al. (2006) study different RL algorithms used for multi-seller markets with inventory replenishment, and evaluates the performance of the different approaches under the cases of no and partial information.
- Collins and Thomas (2013) examines how different consumer model's impact the airline pricing game, and solve the various games using an RL algorithm.

5.1.4 Discussion

Overall, the literature provides a fair amount of evidence that reinforcement learning is suited for finding optimal, or close to optimal, pricing policies in simulated environments, and can outperform simpler pricing polices such as static pricing, the Derivative-Following and the Goal-Directed strategies. However, since their performance is measured in a simplified simulated environment, it is hard to evaluate how they would perform when deployed in a real market. This issue will be further discussed in chapter 6

To converge to an optimal solution, the RL-algorithms have to be trained and learn about their market environment. When analysing their performance by market simulation, these training periods are trivial to perform in the thousands. In a real market environment, however, one needs to consider not only the time until which an optimal policy is reached, but also at what cost. If optimality is to be achieved, exploration by the algorithm must be done in order to learn the markets demand curve. Hence, there will be periods when the algorithm needs to explore its solution space, and thus, revenues might be lost. To exemplify this, an algorithm might believe that raising its price in the next period will increase revenues when, in fact, the opposite might be true if the price then exceeds the reservations price of its customers.

Many authors argue that companies operating online today already have considerable amounts of data regarding their customer and competitors behavior. This helps to create a somewhat better initial state for the RL-policy, by letting the algorithm learn from historical sales data. However, if the data provided stems from a fixed price strategy with few or no price changes, the quality of offline learning is limited.

Another question to be made is what information the sellers are able to infer about the market from an RL-algorithm. Do the various states and actions experienced by the RL-agent, transform info valuable information for the company, or is it just kept hidden? Or put differently, is it the company or the software agent that ultimately learns? These questions seems to only be vaguely addressed in the literature.

Also, the majority of literature focuses on using some underlying Markov Decision Process. This might be justified by considering that it is an somewhat appropriate way to model reality, and because it is possible to solve them to optimality in polynomial time. However, most real-world problems do not have the Markov property as they are often non-stationary, history-dependent and/or not fully observable. In order for RL-methods to be more generally useful in solving such problems, they need to be extended to handle these non-Markovian properties, which is arguably the reason for the increasing focus on using POMDPs and Markov games for dynamic pricing. It should be noted that the assumption of stationarity needed to prove convergence of a normal Q-learning algorithm will be violated in POMDPs and Markov games. A Q-learning agent would face an MDP only of the buyers and all opponent sellers use stationary policies, which is a bold assumption. Still as noted by Tesauro and Kephart (2002) this does not mean that all Q-learning algorithms are expected to behave badly, it simply implies that theory can not say how well the algorithms are expected to work.

5.2 Neural Networks

The approach of using neural networks in the setting of dynamic pricing is rather new, but still there are some research available that explores the problem. Neural networks are mostly used for learning demand functions in complex and unknown market environments, and seemingly less used for automating pricing decisions. The common approach is to estimate demand using a NN, and then apply some other pricing algorithm based on this demand.

5.2.1 Modelling Demand as a Neural Network

Shakya et al. (2012) were among the first researchers to propose using a neural network based demand model for dynamic pricing. Traditional models of demand usually require some assumption on its functional form, i.e. linearity or non-linearity, whereas in the real world these assumptions may not hold and the model is unlikely to correctly represent the true distribution of the data. When modeling demand as a neural network, one does not make any assumptions about the relationship between different factors in advance, and so, demand is modeled more realistically. Neural networks learn the underlying relationships, thus inferring demand from the data itself. Consequently they are able to derive meaning from complex relationships that are too difficult to be handled by humans or other computer techniques.

In their paper Shakya et al. (2012) propose using evolutionary algorithms (EA) to optimize the dynamic pricing problem based on a neural network demand model. The dynamic pricing model used focuses on a single firm aiming to maximize its profits given some production quantity constraint. Demand for a good in a period is assumed to be dependent on the price in that period and also the prices for the product in the other periods in the planning horizon. In order to evaluate the performance of modeling demand in a neural network, they employ the same set of evolutionary pricing algorithms to more traditional demand models such as a linear, exponential and multinomial logit model.

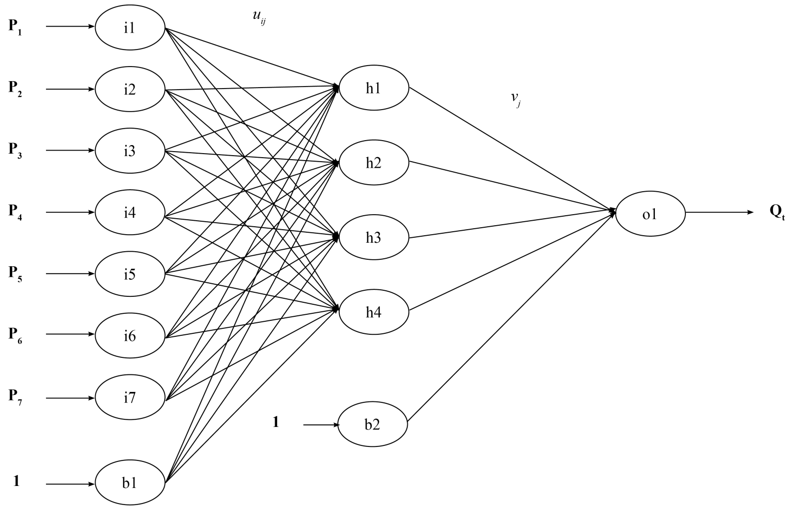


Figure 5.1: The structure of the neural network presented by Shakya et al. (2012)

Considering a fixed topology network with three layers, an input layer, a hidden layer, and an output layer, they build a set of T neural networks that each represents the demand in a given period t. The parameters of the neural networks are the weights associated with the connections, and the authors estimate these parameters using back-propagation from historical sales data. This method takes the prices from the data as inputs to the current model, and then makes an estimation of what production should be. By evaluating the squared difference between its estimation and the actual production given by the data, it slightly adjust its weight vector to minimize the difference. This process continues until a termination criteria is met.

The authors show that by using evolutionary algorithms together with the neural network demand model one can successfully optimize the pricing policy. Further they prove that in contrast ot the traditional demand models, the neural network is more consistent over a wide range of data generated from different sources and provides results closer to optimal in a range of different scenarios.

5.2.2 Available Literature

- Kong (2004) applies a dynamic pricing strategy using the Sales-Directed Neural Network (SDNN) to the Learning Curve Simulator presented by Dimicco et al. (2003) and proves that his approach exhibits superior performance to the other competing strategies included in the simulator (Goal Directed and Derivative Following). The SDNN superiority stems from its ability to predict and account for the long-term consequences of its actions.
- Liu and Wang (2013) propose a sequential feed-forward neural network model to capture the demand information from real-time data and predict the dynamic demand curve online.
- Ghose and Tran (2009) use a feed forward neural network to determine the product price dynamically, but unlike the approaches of Kong (2004) and Liu and Wang (2013), their model does not require the sellers to figure out the demand curve for the products.
- Brooks et al. (1999) present a comparison of neural networks and direct search methods across price schedules of varying complexity in a monopoly price information market.

5.2.3 Discussion

With its ability to learn relationships between different factors from a given or experienced dataset, neural networks seems to be a robust approach for modeling demand. They enable the modeling of complex environments, like those consisting of a mixture of strategic and myopic buyers in addition to demand interdependencies and seller competition, and can approximate any market demand function.

However, aspect like those of competition, price-quality trade-offs and buyer preferences are only indirectly captured through the observed demand. Consequently the implications on demand by, for instance, multiple competing firms are hidden in the network, thus making it hard for the seller to observe the strategies of others and infer information. The presented NNs only rely on the indirect effects from the other agents' actions. Subsequently, a seller is not able to infer what underlying causes effect the demand curve, and thus analyzing and evaluating the different factors implications is hard to achieve.

5.3 Goal Directed and Derivative Following

The Goal Directed and Derivative Following algorithms are computationally cheap and have proved to be effective for real time learning when processing power was a scarcer resource than it is today. They are also commonly used in the literature for comparing the performance of more complex algorithms.

5.3.1 Learning Curve Simulator

Dimicco et al. (2003) present a market simulator designed for analyzing agent pricing strategies in markets under finite time horizons and fluctuating buyer demand, to demonstrate the strength of a simulation based approach for understanding agent pricing strategies. They develop a platform for running different dynamic pricing algorithms in simulated markets, called the Learning Curve Simulator.

According to the McKinsey Quarterly (Baker et al., 2001) sellers should pursue dynamic pricing on-line by running different pricing experiments, by making small price adjustments to discover the demand levels of their buyers. Dimicco et al. (2003) expands on this idea, but propose that sellers simulate the effects of price changes prior to enrolling such a strategy in a real market. They believe that a more sophisticated implementation strategy can be found, if the sellers have an intuitive understanding of the theoretical aspects beforehand.

The Learning Curve Simulator aims to address the complexities of e-commerce buying behavior, and does so by providing a variety of behavior parameters. For instance, variance in buyer reservation price, different buyer distribution curves, number of buyers who employ comparison shop and buyer preferences for sellers. The user is able to run simulations for different values of the parameters, and then simulate how different pricing strategies perform in monopolistic or multi-seller scenarios, as well as considering a single or multiple learning agents.

They demonstrate that by observing market conditions, a seller can take advantage of fluctuations in buyer demand to earn more revenue and sell more inventory. Using the Goal Directed and Derivative Following algorithms, they evaluate their performance compared to a fixed pricing strategy and show that they outperform static pricing strategies in a variety of different market situations.

5.3.2 Available Literature

- Vázquez-Gallo et al. (2014) further develop the derivative following strategy proposed by Dimicco et al. (2001). They add a scale factor of price variation enabling them to adjust prices based on increasing average revenues, final revenues, recent revenues and comparative revenues, and not just the previous time steps revenue.

5.3.3 Discussion

The Goal Directed and Derivative Following strategies are simple to implement and require limited processing power, still they are rather effective dynamic pricing strategies. They perform well in simulations when sellers have little knowledge of buyer behavior or the other sellers prices, provided that there is only a single learning agent present. There is no option of training the algorithms offline, but as they are able to quickly adapt to the demand curve, this of little concern.

However, when multiple learning agents are imposed, these simple strategies are unable to predict and account for the long term consequences of their actions. As a result, they often create price wars when competing with agents utilizing the same strategy as them-selves. Arguably this is one of the most important lessons learned from the simulations, if they had been employed in a real setting without any previous studies these algorithms would be trapped in cyclic price wars, and thus performing suboptimally. Further when faced with more sophisticated algorithms such as reinforcement learning, they are easily outperformed.

5.4 Evolutionary Algorithms

Evolutionary algorithms are less used than the above mentioned approaches, but might still be an interesting approach for solving a dynamic pricing problem.

5.4.1 Adaptive Strategies

Ramezani et al. (2011) design an adaptive dynamic pricing strategy and optimize its parameters with an evolutionary algorithm. They consider a duopoly market supplying a single type of good for a limited time period, given no-replenishment inventory restrictions. They consider two cases of demand, one in which the average of the customer's stochastic valuation for each good is constant throughout the selling horizon, and one where the average valuation is changed according to a random Brownian motion.

Customers behave according to the bargain hunter strategy of the Shopbot model and demand is assumed independent, but the distribution of the customers' valuation may change in time. However to introduce another dimension of uncertainty, they assume that with some probability λ, a customer might choose not to buy the good, even though having found the seller providing them with the highest positive utility.

They develop an Inventory Based Adaptive Heuristic Strategy that uses an evolutionary algorithm to compute its parameters and compare it to other strategies as the Derivative Following and Goal Directed in a simulated environment. They prove that their suggested strategy yields consistently higher revenues than the previously mentioned ones.

5.4.2 Available Literature

- Shakya et al. (2012) show how evolutionary algorithms can be used for analyzing the effects of demand uncertainty in dynamic pricing. They find that the reliability of evolutionary algorithms for finding accurate policies could be higher when there are higher fluctuations in demand and that a generic algorithm is the most reliable algorithm for finding optimal dynamic prices in their market model.

5.4.3 Discussion

Evolutionary algorithms have been used for a variety of different problems and seems to be of value to dynamic pricing problems as well. Perhaps the most notable feature of the literature on EA is that they are focusing on simple markets without segmentation and problems with no replenishment inventory. Generic algorithms, which is the EA principle used in both articles, are known for having issues dealing with high degrees of complexity, and as such more sophisticated market models might have been rejected.

The technique of evolutionary algorithms is suitable for black-box optimization, where no derivatives are known, and should therefore be a promising approach to the dynamic pricing problem. One should try to evaluate how an EA would perform in a multi-attribute market, and when considering more than two competitors, to see if the algorithm efficiently can approximate a good solution.

5.5 Other Approaches

In addition to the above mentioned algorithms, there exists a few less used approaches that will be presented next. Because of their variability and uniqueness, the different algorithms will only be presented briefly.

5.5.1 Simulated Annealing

Xia and Dube (2007) propose an approach to solving a dynamic pricing problem that combines a simulated annealing algorithm with Bayesian learning. They show that the trade-off between exploration and exploitation in a Bayesian Markov Decision Processes with Gittins Indices leads to spatial price dispersion. That is if several players were to experiment with the same initial belief, they might end up choosing different offerings in the long run. The authors suggest borrowing techniques from simulated annealing to avoid this problem.

They simulate their algorithm in a market closely resembling a Price-Quality model with multiple goods and independent demand. By performing numerical experiments, they show that by combining simulated annealing with learning, price dispersion is avoided. Further, when combined with shrinkage techniques, their algorithm results in complete learning and also generates the optimal average reward in the long run.

5.5.2 Particle Swarm

Mullen et al. (2006) compare a Kalman Filter and existing particle swarm adaptations for dynamic and/or noisy environments with a novel approach that time decays each particle's previous best value. They argue that this provides a more graceful and effective transition between exploitation and exploration, which is a necessity in the dynamic and noisy environments of the dynamic pricing problem.

5.5.3 Markov Chain Monte Carlo

Chung et al. (2012) develop a new demand learning algorithm for firms in monopoly or oligopoly markets using Markov Chain Monte Carlo methods to estimate the model parameters, unobserved state variables and functional coefficients.

5.5.4 Aggregated Logarithmic

Levina et al. (2009) study the problem a monopolistic seller faces when selling a perishable product to a market in which the buyers know that prices are dynamic and may time their purchases strategically. They derive the demand model using a game-theoretic consumer choice model and propose applying an aggregating algorithm to predict the demand. The seller's pricing policy is optimized using a simulation based approach integrated with the aggregating algorithm.

5.5.5 Direct Search

In their work, (Brooks et al., 2002, 2001; Kephart et al., 2001), demonstrate the trade-off between complexity and profitability for some common price schedules for an information economy. An information economy, is defined as a market where information goods such as news articles and audio files are traded. They propose that by explicitly considering both the learnability and the generated profits extracted by different price schedules, a producer can extract more profits while learning, than if it naively choose models that only perform well after a substantial learning period. The authors examine the problem under different information assumptions, ranging from supplied complete information to cases where information has to be learned.

5.6 Classification of Literature

Now that we have an understanding of some of the underlying components of dynamic pricing problems solved by machine learning, we can try to categorize some of the literature to see if there are any similarities among the different research approaches and papers. This categorization can be seen in Table 5.1 and 5.2 below. The same papers are listed in both tables, and splitting was necessary to increase readability. It should also be noted that the following categorization is not indisputable; still it provides a general overview of the listed articles.

Of all the 31 articles evaluated, all but one was testing their algorithms performance in a simulated market environment dealing with myopic customers. Levina et al. (2009) provided the only approach explicitly dealing with strategic customers. However, Shakya et al. (2012) Liu and Wang (2013) and Ghose and Tran (2009) can be said to indirectly consider the implications of strategic customers, in that they make no assumptions about their customers' behavior, and just consider estimating a demand curve from sales data.

Further, it seems that, for the majority of literature, the number of buyers has little significance when designing an algorithm, and as a result, the common approach is to assume that there are no finite number of consumers. The number of sellers and learning agents, however, has a considerably greater impact on how the problem is evaluated and modeled.

A total of eight articles combines a machine learning approach with a game theoretic model to gain insights regarding competition and co-operation of multiagent pricing algorithms in such a framework. Those considering multiagent markets in a non-game-theoretic framework, usually evaluate how the overall market demand and profitability changes due to competition, and does not consider the algorithms to be strategic in their evaluations.

Unsurprisingly demand is considered to be stochastic in most of the literature, as demand is generally uncertain for real-world sellers, and because deterministic demand can often be solved to optimality using more traditional methods. The article by Collins and Thomas (2013) dealing with deterministic linear demand, focuses on modeling a pricing game in a multiagent economy and for simplicity assumes the demand to be fully known to all players. Further, the stochastic demand is often considered to be non-linear.

The majority of literature modeling buyers arrival rate using a Poisson process, implements an arrival rate that is inhomogeneous. These papers assume that the arrival rate of buyers changes linearly with time. This is a more realistic assumption than the homogeneous Poisson process which assumes the arrival rate of customers to be stationary throughout the period.

Relating to the classification presented by Elmaghraby and Keskinocak (2003), we see that, as is also the case for their review, the bulk of the literature studied in this thesis consider constrained product availability can be defined as being either NR-I-M or R-I-M models. Fourteen of the considered papers assumes no-replenishment-independent-demand (NR-I-M), whereas only six evaluates the replenishment-independent demand (R-I-M). Dependent and interdependent demand is generally harder to evaluate, because, as the number of interdependencies among products increase, the speed of convergence decreases exponentially (Rana and Oliveira, 2015).

We further see that the Shopbot model of demand, that is, a market in which consumers are only focused on price and perform various levels of price comparison tactics, is the most commonly used. Three of the articles experiments with their proposed algorithm in the Shopbot, Price-Quality and Price-Information approach, to see how they cope with changes in demand models and how different assumptions affect market dynamics.

The majority of literature on machine learning and dynamic pricing is focused on some form of reinforcement learning. Undoubtedly one can find many other articles than the ones presented here, but its likely that the ratio between different approaches is rather accurate. Hence, we are able to evaluate which approaches seems to be the most popular. Further, all reinforcement learning literature presented use some form of Q-learning. There has been no attempt aiming to separate the different Q-learning techniques from each other, for instance, some use a gradient approach, where others are combining Q-learning with simulated annealing. However, we find that the common approach is to use a look-up table for storing learned Q-values, as Sridharan and Tesauro (2002) is the only one considering using function approximation instead. Nevertheless, one should expect to find some interesting differences in how the algorithms are developed and how they perform in different markets.

Finally, many researchers on this topic try to prove their machine learning algorithms performance by comparing it to other pricing algorithms or strategies. Consequently, we have also listed the various algorithms used for performance measure, enabling us to evaluate the how the authors validate their results. Chapter 6 will in greater detail discuss the findings from the evaluated literature.

Table 5.1: Classification of literature according to the previously presented framework, part I

Article	**Sellers**	**Learning Agents**	**Buyer Arrival**	**Seller reconsider**	**Segmenting**	**Dependency**
(Vengerov, 2008)	Oligopoly	Multiple	Poisson [H]	Simultaneous, ETS	Yes	Independent
(Rana and Oliveira, 2015)	Monopoly	Single	Poisson [I]	ETS	Yes	Interdependent
(Kutschinski et al., 2003)	Oligopoly	Multiple	All in ETS	Simultaneous, ETS	Yes	Independent
(Sridharan and Tesauro, 2002)	Duopoly	Multiple	Poisson [H/I]	Alternately, ETS	Yes	Independent
(Li et al., 2006)	Duopoly	Multiple	-	Simultaneous, ETS	No	Independent
(Han et al., 2008)	Oligopoly	Multiple	-	Simultaneous, ETS	No	Independent
(Cheng, 2008)	Monopoly	Single	Poisson [I]	ETS	No	Independent
(Cheng, 2009)	Monopoly	Single	Poisson [I]	ETS	No	Independent
(Jintian and Lei, 2009)	Duopoly	Multiple	Poisson [H]	Simultaneous, ETS	Yes	Independent
(Han, 2010)	Monopoly	Multiple	-	Simultaneous, ETS	No	Independent
(Raju et al., 2006)	Monopoly	Single	Poisson [H]	Random interval	Yes	Independent
(Tesauro and Kephart, 2002)	Duopoly	Multiple	Poisson [H/I]	Alternately, ETS	Yes	Independent
(Dogan and Güner, 2015)	Duopoly	Multiple	Poisson [H]	Simultaneous, ETS	No	Independent
(Könönen, 2006)	Oligopoly	Multiple	Poisson [H/I]	Alternately, ETS	No	Independent
(Collins and Thomas, 2012)	Duopoly	Multiple	One in ETS	Alternately, ETS	No	Independent
(Chinthalapati et al., 2006)	Duopoly	Multiple	Poisson [I]	Random interval	Yes	Independent
(Collins and Thomas, 2013)	Duopoly	Multiple	One in ETS	Alternately, ETS	Yes	Independent
(Xia and Dube, 2007)	Monopoly	Single	Bernoulli	ETS	No	Independent
(Mullen et al., 2006)	Monopoly	Single	-	ETS	No	Independent
(Kong, 2004)	Oligopoly	Multiple	Uniform	Simultaneous, ETS	Yes	Dependent
(Shakya et al., 2012)	-	-	-	ETS	-	-
(Liu and Wang, 2013)	-	-	-	ETS	-	-
(Ghose and Tran, 2009)	-	-	-	ETS	-	-
(Brooks et al., 1999)	Monopoly	Single	-	ETS	No	Independent
(Chung et al., 2012)	Duopoly	Multiple	-	Simultaneous, ETS	No	Independent
(Dimicco et al., 2003)	Several	Multiple	Uniform	Simultaneous, ETS	Yes	Dependent
(Vázquez-Gallo et al., 2014)	Monopoly	Single	-	ETS	No	Independent
(Ramezani et al., 2011)	Duopoly	Multiple	Poisson [H]	Simultaneous, ETS	No	Independent
(Shakya et al., 2009)	Monopoly	Single		ETS	No	Interdependent
(Brooks et al., 2002)	Monopoly	Single	-	ETS	No	Independent
(Levina et al., 2009)	Monopoly	Single	Poisson [I]	ETS	No	Dependent

"-" : Unclear/Not relevant/Other

Abbreviations:

ETS: Each time step
H: Homogeneous
I: Inhomogeneous

Table 5.2: Classification of literature according to the previously presented framework, part II

Article	**Demand**	**Demand/Utility**	**Supply***	**Process**	**Algorithm**	**Performance Measure**
(Vengerov, 2008)	Stochastic NL	Shoppers & Captives	R	POMDP	Q-learning	Static Pricing Policy
(Rana and Oliveira, 2015)	Stochastic NL	Price-Quality	N-R	MDP	Q-learning	Individal Q-Learning
(Kutschinski et al., 2003)	Stochastic L	Shopbot	R	MDP	Q-learning	DF, MO, PP
(Sridharan and Tesauro, 2002)	Stochastic L/NL	SB, PQ & PI	UC	MDP	Q-learning	-
(Li et al., 2006)	Stochastic L	Shopbot	N-R	MG	Q-learning	Nash Q/Best Responce Q
(Han et al., 2008)	Stochastic L	Shopbot	UC	MG	Q-learning	-
(Cheng, 2008)	Stochastic NL	Shopbot	N-R	POMDP	Q-learning	Static Pricing Policy
(Cheng, 2009)	Stochastic NL	Shopbot	N-R	MDP	Q-learning	Static Pricing Policy
(Jintian and Lei, 2009)	Stochastic NL	Shoppers & Captives	R (q,r)	MDP	Q-learning	Static Pricing Policy
(Han, 2010)	Stochastic L	Shopbot	UC	MDP	Q-learning	Multiagent RL
(Raju et al., 2006)	Stochastic L	Shoppers & Captives	R (q,r)	MDP	Q-learning	-
(Tesauro and Kephart, 2002)	Deterministic L/NL	SB, PQ & PI	N-R	MG	Q-learning	Myopic Optimal
(Dogan and Güner, 2015)	Stochatic NL	Shopbot	R	MDP	Q-learning	Adaptive Pricing
(Könönen, 2006)	Stochastic L	SB, PQ & PI	UC	MG	Q-learning	Singleagent RL
(Collins and Thomas, 2012)	Stochastic L	Shopbot	UC	MG	Q-learning	Myopic Optimal
(Chinthalapati et al., 2006)	Stochastic NL	Shoppers & Captives	R (q,r)	MG	Q-learning	Derivative Following
(Collins and Thomas, 2013)	Stochastic NL	Shopbot	UC	MG	SARSA	-
(Xia and Dube, 2007)	Stochastic NL	Price-Quality	N-R	Bayesian MDP	Simulated Annealing	-
(Mullen et al., 2006)	Stochastic NL	Shopbot	UC	-	Particle Svarm	-
(Kong, 2004)	Stochastic L/NL	Price-Quality	N-R	-	Neural Network	DF, GD
(Shakya et al., 2012)	Neural Network	-	N-R	-	Neural Network / EA	Other demand models
(Liu and Wang, 2013)	Neural Network	-	N-R	-	Sequential Learning	-
(Ghose and Tran, 2009)	Neural Network	-	UC	-	Back Propagation	-
(Brooks et al., 1999)	Stochastic NL	Price-Information	UC	-	Neural Network / DS	-
(Chung et al., 2012)	Stochastic NL	Shopbot	UC	Markov Chain	Simulated Annealing	Static Pricing Policy
(Dimicco et al., 2003)	Stochastic L/NL	Price-Quality	N-R	-	GD & DF	-
(Vázquez-Gallo et al., 2014)	Stochastic L	Shopbot	N-R	-	DF	DF
(Ramezani et al., 2011)	Stochastic NL	Shopbot	N-R	-	Evolutionary Algorithm	Static Pricing Policy, DF
(Shakya et al., 2009)	Stochastic NL	Shopbot	N-R	-	Evolutionary Algorithm	SA, DEUM, GA
(Brooks et al., 2002)	Stochastic NL	Price-Information	UC	-	Direct Search	Static Pricing Policy
(Levina et al., 2009)	Stochastic NL	-	N-R	MG	Aggregating Algorithm	-

*Product Availability has been substituted by "Supply" to enable compressing of the table
"-" : Unclear/Not relevant/Other

Abbreviations:

L: Linear	SB: Shopbot	UC: Unconstrained	DF: Derivative Following	MO: Myopic Optimal
NL: Non-linear	PQ: Price-Quality	NR: No replenishment	GD: Goal Directed	PP: Price Point
	PI: Price-Information	R: Replenishment	EA: Evolutionary Algorithm	SA: Simulated Annealing
				GA: Genetic Algorithm

Chapter 6

Evaluation of Current Literature

By now, we have an understanding of dynamic pricing, machine learning and how these topics can be combined. Considering the literature studied in Table 5.1 & 5.2, we will in this chapter point to some general and interesting observations, relate the aspects of dynamic pricing to real-world consumers and point towards future directions for research regarding dynamic pricing by machine learning. Furthermore, this evaluation will highlight some of the issues we address in our own market simulator, presented in Chapter 7.

6.1 Observations

In this section, we will perform an evaluation of what value machine learning can have for companies, by considering our findings and discussing some of the general assumptions and methods used in the literature.

6.1.1 Lack of Evidence and Performance Metrics

Perhaps the most interesting observation is the lack of empirical evidence regarding the revenue increases different approaches are argued to create in a real-world application. The majority of the algorithms presented are claimed by their authors to be suitable for more or less direct implementation in a real market setting, but none of the above-mentioned articles have provided any empirical evidence of their performance in an actual market. All algorithms are evaluated in a purely simulated and simplistic market. Considering the fact that, one of the benefits of machine learning is its adaptability and ability to relate to uncertainties, it seems contradictory that these methods are not tested in a proper marketplace. Hence, the need for proper evaluation of machine learning principles are an important area for future research.

Some research has been performed to analyze the impact of shopbot use amongst customers on prices and price dispersion (Tang et al., 2010), but proof of seller's increased profits still lacks in the literature. Other approaches to the dynamic pricing problem have to some extent validated its value for sellers. For instance, Fisher et al. (2016) test a best-response pricing algorithm through a carefully controlled live experiment, and document an 11% increase in revenue. However, they derive this algorithm using constrained optimization, and not machine learning. Nevertheless, DPOSPs such as Wiser that utilize machine learning advertise that their product can help increase bottom line profits by 32% (WiseCommerce, 2015). However, this statement is lacking some additional information in regards to what machine learning algorithm they apply, and also fails to specify what previous price strategy they use to calculate this increase in revenue.

Another key observation is that the majority of literature, in addition to skipping empirical evidence, compare their suggested improved algorithm to the simplest of algorithms. The static price strategies are proved to be beatable over and over again, but few seems to consider how the machine learning algorithms perform when evaluated against each other. Some have evaluated their algorithms to the Derivative-Following and Goal-Directed strategies, but these are very rudimentary adaptive algorithms. Research in which the author(s) develop multiple algorithms commonly compare the algorithms to each other, but we are missing comparisons of the suggested algorithms performance to other researcher's advanced algorithms. It would be especially interesting to see for instance how an RL-algorithm would perform when faced with a competing neural network. Arguably, this lack of proper comparison methods might be a result of variations in underlying market assumptions, but still, objectively evaluating different approaches should be possible. A comprehensive study evaluating the impact on market dynamics from multiple different smart agents would be welcomed.

6.1.2 Relating to the Model Assumptions

First of all, there is a wide adoption of myopic consumers in the studied literature. Of the 31 articles investigated, the paper by Levina et al. (2009) was the only one dealing explicitly with strategic consumers. It is assumed that the matter of strategic consumers has been excluded as it introduces additional complexity in already complicated matters. Furthermore, Elmaghraby and Keskinocak (2003) claim that the assumption of myopic buying behavior can be appropriate in several settings, such as for; (1) necessity items where consumers cannot wait for the price to drop, (2) the prices or price changes are small, thus the value of waiting is also small, (3) a large customer pool so that one single customers purchase has no effect on prices, (4) a market where customers do mostly impulse purchases.

However, there is reason to believe that a substantial portion of consumers act more strategic in most situations. For instance, some people might withhold purchases until after Christmas when the season sales start. Moreover, although the assumption of myopic customers might be justified in some special situations, most markets are bound to have some portion of strategic buyers.

Many authors have studied the issue of pricing when customers behave strategic, and perhaps the most famous of which is Ronald Coase and the Coase conjecture. In his work, Coase (1972) considers a monopolist selling a consumer durable facing strategic consumers. He proposes that if the seller is unaware of its customers willingness to pay, then trying to separate consumers by different price levels in different periods will fail, as the consumers will anticipate the declining price path. Consequently, the seller is in competition with itself, and if consumers are willing to wait, those with the highest valuation will be able to buy the product at the lowest price. The seller is arguably left with no other choice than to simply offer a competitive price in the first selling period. To gain further insight and a better understanding of reality, a possible future direction for the machine learning approach should be to include some portion of strategic buyers in the market.

Second, all examined literature employ some form of weak dynamic pricing. The papers utilizing a Shoppers and Captives model, do encourage self-selection by second degree-price discriminating, but so far research refining such systems is lacking. Due to the massive collection of data available to retailers, and possibilities for consideration set evaluations, we believe that there are opportunities for finer segmentation of the market. Developing new differencing value propositions by offering additional levels of service or quality might be an approach to increase the supplier's portion of the consumers' surplus. The exploration of strong dynamic pricing models is assumed to have been neglected due to the lack of consumers fairness perception. Still, it would be interesting to explore the benefits of such an approach by simulation. Needless to say, one has to address the problem regarding fairness, but shedding light on the possible increased profits and efficiency might encourage creative ways of avoiding this issue.

Third, a large portion of the assessed literature assumes that customers arrive according to a Poisson process. The Poisson process is commonly used for estimating demand arrivals, and consequently, it is assumed to present some representation of reality. There seems to be an even amount of papers considering homogeneous and inhomogeneous Poisson processes. Arguably the adaptation of inhomogeneous Poisson processes is more representative of real life, as i.e. the number of arriving customers is likely to be dependent on price or time. Evaluating the assumptions regarding the Poisson process, there are two assumptions that might not hold in the real world:

1. The probability of an event within a certain interval must not change over different intervals
2. The probability of an event in one interval is independent of the probability of an event in any other non-overlapping interval

Assumption one might be overruled by the fact that for some goods, demand might be higher during the afternoon, or on Fridays. This can to some extent be addressed by the inhomogeneous Poisson processes, where the rate of arrival can change according to time. The second assumption is well suited for markets dealing with independent arrivals, but if more realistic models are to be made, dependencies of arrivals need to be included, and as such, this assumption might not hold. It is easy to imagine that the arrival rate of consumers are dependent, i.e. by considering that a consumer having made a successful purchase previously is likely to return to the same seller for future purchases or browsing. Interestingly the wide usage of Poisson processes and its implications are seldom discussed, and we would like to see some level of theoretical or empirical justification to further enhance the literature.

Fourth, the bulk of literature simulates their algorithms in rather simplistic market environments. The models representing customers buying behavior seems to be overly simplistic, and the simulated buyers are far from replicating a real world scenario. This is also evident from the popular assumption that sellers reconsider their prices simultaneously, thus forgoing some aspects of a real world strategic interplay between agents. Since there is a lack of real-world performance measures, one might assume that the simulation models would be more realistic to help justify the performance of the algorithms. Neglecting dependencies and behavior might be analytically beneficial but at the cost of forgoing important information. Especially when considering that the goal of many simulation models is to learn about how one expects the real market to respond, the need for a sufficiently accurate model of the market is fundamental.

Fifth, there are many different machine learning approaches, and although the literature on reinforcement learning seems to be the most promising, considering the amount of research performed, there seems to be no unified consensus regarding which approach is the most valuable. A modeler who assumes that the world can be represented using an MDP would prefer a reinforcement learning algorithm, whereas if one were to make few or no assumptions about the market, a neural network might be a good approach for learning demand. It seems that it all boils down to what the researcher finds to be interesting and how they assume their market to behave. Presumably, because of its multidisciplinary nature, the problem of dynamic pricing should to be evaluated by economists, computer scientists and marketers if a comprehensive model is to be made.

Sixth, the supervised and reinforcement learning algorithms, all require some historical data and/or a lengthy training period to reach a level of sufficient performance. Although understandable, this issue of learning imposes some important questions for the sellers that they might not be able to comprehend. For instance, the algorithm might be able to learn from past sales data, but if those data points have been generated by a static price algorithm, then the data represent more of a forecast of demand given a fixed price, rather than representing a demand curve dependent on the product price. If the data points are of little learning value, then the trade-off between exploration and exploitation, especially for RL-algorithms, has to be considered to a larger extent, as it then has limited input to base initial pricing decisions on. This trade-off is generally implemented in the algorithm themselves, but the sellers, or users, should be able to understand the implications of this issue which might not be that easy.

6.2 Relating to the Consumer

One aspect of the machine learning literature is that although one tries to construct an algorithm that is suitable for implementation in a real world application, few articles from the economic and computer science community consider how customers in the real world are assumed to react to these new pricing schedules.

By now, people are used to the fact that prices fluctuate. It has been part of our lives for a long time, and classical examples of prices changing with time include the happy-hour phenomenon, and that buying shorts is almost always cheaper in September. There is a need to study how the general community will react as more and more companies utilize some form of dynamic pricing. Research regarding this issue has been performed in the marketing literature:

Haws and Bearden (2006) evaluate dynamic pricing and consumer fairness perception, when prices vary over time, consumers and/or circumstances. They find that differences in price between consumers resulted in the greatest perception of unfairness. Also, customers view price changes within a short time period as more unfair than changes over a more extended time period, particularity when exposed to lower prices. The studies were however performed in a lab involving students and purchase scenarios, and is only an approximation of how people are expected to react. They argue that the effects of dynamic pricing might be even more pronounced when varying prices are encountered in realistic shopping environments. Garbarino and Lee (2003) find that experiencing dynamic pricing reduces mean trust in the benevolence of firms and that people seemingly are equally displeased whether they got the higher or lower price. Apparently, people feel that all customers should be offered similar prices. Still, this study is also based on a constructed environment, and as such, might not include the full picture.

From the literature evaluated in this thesis, it is clear that weak dynamic pricing is the preferred form, and hence the need to evaluate fairness is limited. However, economic theory tells us that first-degree price discrimination can maximize economic welfare if output is increased, the drawback of course is that the firms captures all available surplus. Arguably, the consumers could hold shares in the company and get dividends to indirectly increase their surplus. Still such an approach is bound to be met with some resistance. Regardless, the tempting aspect of strong dynamic pricing has to be evaluated accordingly. Perhaps the benefits of possible increased profits will significantly outrun the additional risk exposure to consumer trust?

6.3 Future Directions

Based on the findings in this thesis some interesting suggestions for further research can be explored. As previously stated, there is a need for unbiased evaluations of the performance of machine learning algorithms in real world dynamic pricing applications. These results might be readily available at the different DPSOPs operating today, however, they seem to be kept as company secrets. Efforts should be devoted to gain insight regarding the benefits of dynamic pricing, as ultimately, the real measure of success for this literature would be to justify its revenue-maximizing promises.

The Learning Curve Simulator presented by Dimicco et al. (2003) provided a solid framework for experimenting with simple machine learning algorithms in the early 21st century. However, as processing power and our ability to model more complex relations has increased, the development of a new and improved market simulator would be valuable to the scientific community. Predominantly, such a simulator should enable the benchmarking of different algorithms in the same environment, and as such different approaches could be justified based on the same assumptions and basis. Providing a framework for enabling the evaluation and comparison of different approaches, in order to study which approaches might work better than others under given assumptions. Suggestions for such a simulator would include the opportunity to model strategic and myopic consumers, defining multiple different market segments, heterogeneous goods, interdependencies of demand and different arrival models. Addressing more complex buyer models and the effect they might have on market price development and learning of pricing strategies is a commonly stated area of future research in the literature.

The issue of a company selling multiple heterogeneous products with interdependent demand might also be an interesting exercise for dynamic pricing algorithms. Perhaps one could construct a policy that improves profit allocation amongst goods by considering multiple goods with different marginal cost and prices. For instance, a simple example of such a strategy is to sell a TV-screen at a completive price, while selling add-ons and extra equipment as HDMI-cables with a high premium. Dynamically altering the prices of such interdependent products, implementing discrete choice theory, and more advanced buyer behaviors, can be an exiting topic for future studies.

Ultimately, theoretical economic efficiency is reached when whoever values the good the most, gets the good. In an efficient economy, excess demand for events, concerts and iPhones is non-existent. Still, as our technology progresses, the rate at which events is sold out is steadily increasing. For instance, it is not uncommon for a concert to sell thousands of tickets within minutes of launch. The problem of pricing such perishable goods, has been considered by Vázquez-Gallo et al. (2014) and others, and is an interesting area for research. Arguably because of the limited sales horizon and perishable property, pricing tickets for popular events can enable close to first-degree price discrimination without imposing varying prices between consumers. This might be achieved by a Dutch auction like approach, in which posted prices start high and decrease with time. Hence, one might get the benefits of strong dynamic pricing, without having to consider the downsides.

Although it might not be considered fair, the idea of perfect price discrimination is an intriguing one. One might not need to employ methods that would encourage an Orwellian economy (Odlyzko, 2003), where a package of aspirin might cost a \$1 if the purchaser could prove to be indigent, but \$1000 if he was Bill Gates, but perhaps only slight adjustments in prices between consumers can lead to large profits. Thus, studying the possibilities of strong dynamic pricing, while maintaining consumer fairness (or do so in secrecy) might be an interesting approach to the dynamic pricing problem.

We end our discussion by expanding on an interesting suggestion by Surowiecki (2014) that might help efficient allocation of goods, but not necessarily improve a seller's surplus. This idea is to dynamically price goods under short supply according to demand, but instead of giving the difference between a "standard price" and the dynamic price to the seller as profits, it is given away to charity. This might be utilized for many applications, for instance, one could imagine that during rush hour, commuters driving cars could be able to buy access to the bus lane by paying money to a charity and this "gift" is determined dynamically by the amount of traffic.

Chapter 7

Our Approach

The previous chapter pointed us towards some interesting new fields of study when considering machine learning and dynamic pricing. Unfortunately, we will not be able to address all of these findings, however, by focusing on three of the main findings, namely the lack of strategic customers, consumer heterogeneity, and machine learning comparisons, we can expand the current literature and provide new insights. The remaining chapters of this thesis will, therefore, be used to present and analyze our approach for simulating dynamic pricing algorithm performance in heterogeneous markets.

7.1 Defining the Scope

Our overall goal with the coming simulations is to get an understanding of the market dynamics when one introduces different machine learning algorithms in a heterogeneous market, and thus be able to discuss the opportunities and obstacles of the algorithms. As previously explored, there exists a vast number of possible machine learning algorithms, many of whom have never been applied to the problem of dynamic pricing. Therefore to limit our scope we have chosen only to focus on previously tested techniques. From our literature study it seems clear that Q-learning and neural networks are the two most used approaches, and as such, we will be concentrating on these two families of algorithms. Moreover, the algorithms we use, will not be at the forefront of artificial intelligence, due to its incredible complexity. Instead, we focus on implementing functional and basic algorithms; that will provide us with a sufficient foundation for analyzing general behavior and implications of Q-learning and neural networks when faced against each other.

In addition, since our interest lies in the algorithms performance and implications on a market in an economic sense, we have chosen to omit discussions regarding the underlying mechanics of the algorithms. Therefore, topics such as overfitting, Q-learning convergence, and evaluation of different training techniques will only be touched upon briefly. There is without any doubt many interesting discussions to be had on these topics, but we will leave those out to the science of artificial intelligence. Additionally, our goal is not to find ways of optimizing the algorithms to best suit our simulated market. A future addition to this thesis could include ways of modifying the suggested algorithms to make them perform better, but due to the time constraint, we spend the majority of time analyzing their current performance, and not how they can be improved.

In regards to the market dynamics, we have chosen not to reinvent the wheel, but implement previously known market models from the literature. However, we take a novel approach in that we combine several of the previously presented methods in a way that has not been done before, at least to our knowledge. We follow the same principle used for the machine learning algorithms; that is we aim for a simple, yet functional market representation. First and foremost, we think of our model as a starting point for future more advanced simulations. In a sense, our model represents the minimum viable product for analyzing machine learning performance in heterogeneous markets. Consequently, we derive our model from the widely used model of weak dynamic pricing, with independent stochastic non-linear demand and no replenishment of inventory. Furthermore, we chose to use the Shopbot model as the premise for the implemented consumers, and will not be evaluating how the algorithms would perform using a different market representation.

7.2 Areas of Interest

The subsequent sections will present the three areas of interest for our model, and where needed, provide some new theory necessary to understand how other literature has dealt with the issues at hand.

First of all, we have the intention to learn more about how different machine learning algorithms compare to each other in terms of performance when faced with the same underlying market. Furthermore, we want to develop an understanding of how these algorithms affect each other and the distribution of welfare when machine learning algorithms compete for revenue in a market.

Secondly, since simulation is our preferred approach, we have the opportunity to implement a diverse and heterogeneous composition of consumers which would have been almost impossible using an analytic approach. Consequently, our goal is to develop a model that enables us to see what could happen in a market where consumers behave differently, and especially, how the algorithms would adapt to such a market.

Finally, in Section 4.3 we took a closer look at how the majority of machine learning models have been representing a marketplace, however as noted there is a lack of research regarding machine learning models and the use of strategic customer behavior. Therefore, we aim to gain a better understanding of how these algorithms perform under rational customer behavior. Hence, we need to pivot outside the machine learning literature to learn more about how customers buying behavior has been modeled in other dynamic pricing literature.

7.2.1 Multiple Learning Agents

The issue of having multiple learning agents interact and compete lies in the intersection between machine learning and multiagent systems. We have previously discussed machine learning, but the topic of multiagent systems has not been touch upon. The idea of multiagent systems very simple, but their implementation is complex and challenging. A multiagent system is one that consists of multiple agents, which interact with one another, typically by exchanging messages through some computer network infrastructure. At its core, the agents in a multiagent system will be acting or representing on behalf of their users or owners with very different goals and motivations. Therefore, to successfully interact, these agents require the ability to cooperate, coordinate, and negotiate with each other, in much the same ways that we do in our everyday lives (Wooldridge, 2009). The science of multiagent systems then, is about addressing the two following problems:

- How do we build agents that are capable of independent, autonomous action in order to successfully carry out the tasks delegated to them?
- How do we build agents that are capable of interacting (cooperating, coordinating and negotiating) with other agents in order to successfully carry out the tasks that we delegate to them, particularly when the other agents cannot be assumed to share the same interest/goals?

The first problem, relates to that of agent design, whereas the second issue is that of society design. These problems are not independent, for instance, in order for us to build a society of agents that work together effectively, it would help if we give the members of the society models of the other agents in it. Furthermore, the science of machine learning revolves around agent design, where on the other hand, researchers in multiagent systems may predominantly be concerned with engineering systems. However, the two are closely interconnected.

The majority of literature presented in Chapter 5 evaluates a market with multiple learning agents. However, all of these articles are focused on agent design, and none includes a system in which the agents can interact in a multiagent society model. It is bound to be many articles that explore the issue of system design and dynamic pricing of goods, though these seems to take more of an approach of dynamic pricing and automated resource allocation, like that of Schwind (2007). Consequently, our model will not implement interactions between agents, but instead focus on the implicit interactions between agents when they have no means of communicating amongst each other.

7.2.2 Market Heterogeneity

Market heterogeneity is often analogous to consumer segmenting, or the splitting of a large population of consumers into smaller unique groups. In Section 2.2.1, we learned that customer heterogeneity and the possibility of segmenting, are two of the conditions needed to be fulfilled for dynamic pricing to succeed. Still, we find that the majority of current literature does not deal with highly heterogeneous consumers. Common practice seems to be separating consumers according to their evaluation of sellers, willingness to pay and seller preference, but authors often ignore the possibility of having more complex variations between the segments, like those of myopic or strategic behavior. Arguably, this stems from the issue that the more heterogeneous a market is, the more complicated it is to find the best pricing policy.

Furthermore, it is our belief that the literature examined predominantly aims to provide insights regarding the engineering and value of algorithms when pricing goods. Hence, it would be natural to assume that more attention has been devoted to the development of suitable algorithms, than what has been devoted to constructing realistic replicating marketplaces. That is not to say it has been neglected, but it seems that since this area of research is still in its early stages, the major concern has first and foremost been to discuss the applicability of smart algorithms. Therefore, we expect to see more complex markets and increased customer heterogeneity in future research.

In our model, we aim to take a step towards increased heterogeneity by allowing for multiple consumer segments having different behaviors, and mixing myopic consumers with strategic consumers. This should allow for some additional knowledge of how algorithms can be expected to work when faced with a real marketplace, and provide some indication of what issues arise in such settings.

7.2.3 Strategic Customer Behavior

Trying to capture how we as humans make decisions is a complex and little understood process. Despite years of psychological and medical research, we only have limited knowledge of why we choose what we do. A common practice in operations management is to characterize customer demand exogenously (Shen and Su, 2007). Often, as is the case with machine learning models, market size is represented by a demand distribution, price sensitivity is captured by a demand curve, and customer arrivals are commonly modeled using some stochastic process. The commonality of all these models is that the consumers are passive, and do not engage in any decision-making process. They are simply governed by the demand profile specified initially. This is surely not how markets function, as all customers to some degree actively evaluate alternatives and make choices. Customers need to assess which product to buy, when to buy it, and how much they are willing to pay for it. When considering building a market simulator, one of our main focuses has to be to represent real consumers as best as we possibly can. Therefore, we turn our focus to strategic customer behavior and present some possible approaches suitable for modeling consumers.

In their review Shen and Su (2007) present how customer behavior has been represented in the revenue management literature. They classify customer behavior into two groups, intertemporal substitution, and multi-product choice. Multi-product revenue management considers discrete choice probabilities and tries to model how customers choose which product to buy, or which seller to buy from. Examples of this include when sellers offer similar goods for sale, and the customer has to choose what to buy from a set of products. These problems include complicated demand dependencies that arise due to substitution or complementary effects across products. Anderson et al. (1992, pp. 1) claims that "...understanding product differentiation is crucial to understanding how modern market economies operate". Furthermore, they argue that the wide array of products available in the marketplace is a response to the wide diversity of consumer tastes. Still, we find that only a few articles, like those of Dogan and Güner (2015) and Levina et al. (2009), include the problem of multi-product choice. Consequently, the added complexity and lack of previous research have led us to neglect this issue from our model.

Intertemporal substitution, on the other hand, considers how customers may choose to postpone their purchase because of an anticipation that prices will be lower in the future. This implies that individuals may choose when to buy a particular product in response to firms' dynamic pricing practices, especially when they anticipate price reductions. The practice of delaying a purchase to a future point in time is known as intertemporal substitution, and is also commonly referred to as strategic customer behavior. Since our goal is to implement strategic behavior in our model, the remainder of this chapter will focus how this has previously been achieved in revenue management literature.

Intertemporal Substitution

Take a minute and consider; when was the last time you passed on an immediate purchase because you had a feeling that the product you were considering might be sold at a discount in the future? Arguably the delaying of purchase can be a smart choice as intertemporal price fluctuations are common across industries. Prices for high-tech gadgets with short life cycles tend to dip soon after release, fashion items are marked down at the end of a season, and storable goods are periodically put on sale (Li et al., 2014). There are of course many cases where delaying a purchase is inconvenient, and customers may choose to ignore, intentionally or unintentionally, the possible benefit of delaying a purchase. Customers might not be aware of the possibility that prices may decrease, they experience high waiting costs or are not willing to risk immediate gains for the uncertainty of a possibly larger future gain. However, in many markets, it is safe to assume that at least a portion of the buyer population exhibits strategic behavior. In a recent study, Li et al. (2014) provide evidence that in air travel industries the amount of customers behaving strategic varies between 5,2% and 19,2%.

Until recently, revenue management literature has almost entirely ignored this matter, and just assumed that demand arriving at each instance in time is either realized or lost forever. This is an issue because, as stated by Shen and Su (2007, pp. 714), "there is no opportunity for demand to lie dormant in the market in anticipation of future purchase opportunities". Neglecting these issues cause us to lose an additional dimension that can have a significant impact on revenues, and by realizing that the assumption of a homogeneous myopic market is unrealistic, recent work has begun exploring the impacts of inter-temporal substitution. Next, we will explore three ways customer rationality has been previously modeled.

The work done by Su (2007) was among the first to study the impact of strategic customer behavior while allowing prices to increase or decrease freely over time. Previous work tended only to consider the option of markdown pricing, and the fact that prices could also increase was mostly neglected. Perhaps one of the most interesting insights from their article is that they demonstrate that, optimal prices over time may be increasing, decreasing or even non-monotone over time depending on the customer composition. Customers wish to maximize individual utility, and at each point in time they may choose to purchase a product at current prices, remain in the market at a cost to purchase later or exit the market altogether. Their customer population is heterogeneous along two dimensions; they may have different valuations for the product and various degrees of patience or waiting-costs. Their analysis concludes that increasing prices are optimal when customers with a high reservation price are proportionally more strategic, whereas decreasing prices are optimal when high reservation price customers are more myopic. They also, contrary to intuition, highlight two ways in which strategic waiting by customers may benefit the seller. Firstly, intertemporal substitution implies that when prices are high initially, demand is not immediately lost and hence may increase revenues when prices are eventually lowered. Second, when customers with low reservation prices wait, they compete for product availability with high reservation prices customers and thus their valuations may increase as their fear of not getting the product can increase their willingness to pay. The authors make some unrealistic assumptions in that the sellers must post their pricing schemes $p(t)$ and rationing policies $r(t)$ at the start of the market, and that there must not exist any time t when it is to the sellers' advantage to deviate to different policies than the ones first posted.

Ahn et al. (2007) introduce strategic customer behavior in a different and simpler way. Arguably "their" customers do not behave properly strategic in the sense that they do not actively try to find the best deal. Instead, they remain in the market for a fixed number of time periods and purchase if the price is set below their valuations within this time-frame. As a result, demand faced by sellers in a particular period depends on prices over multiple periods in the past. Effectively demand in each period is divided into two groups, current demand, and residual demand. Current demand is the demand by customers who enters the system at that period, and residual demand is that portion of demand resulting from customers who entered the market in previous periods, but who still have not made a purchase or left the market. Within this setup, the authors explore joint pricing and inventory decisions and identify structural properties and develop effective heuristics for some special cases.

Levin, McGill, and Nediak (2010) explore customer rationality in a series of papers where they assume that customer valuations are volatile and may stochastically vary over time. The degree of strategic behavior among customers is represented using a discount factor, in which a discount factor of zero implicates myopic behavior and a factor of one implies that future purchases are as valuable as current purchases and as such consumers are fully strategic. Given the customer's realized valuations and the seller's price, customers choose a purchase probability or shopping intensity λ. The problem for the sellers is then to set prices dynamically to maximize their revenues under customers response to these prices. The authors examine two special cases, one where all customers are myopic and one where they are all strategic, and derive dynamic equilibrium conditions and structural properties for these special cases. They further extend their analysis to an oligopolistic setting in Levin et al. (2009). Following this, they incorporate the use of demand learning and introduce seller uncertainty for some of the parameters of the demand process in Levina et al. (2009). The introduction of imperfect information makes the customers unable to predict future prices via rational expectations. Consequently, consumers are assumed to believe that prices follow a certain stochastic process and set their demand intensity accordingly. To analyze this problem, the authors use simulation-based techniques to find optimal pricing policies.

We end this chapter by briefly introducing the concept of capacity rationing, which is how sellers can use pricing and rationing to extract maximal revenue. Seeing that capacity, or inventory, is scarce, it is quite natural for firms to use inventory control as a strategic tool in the face of strategic customers. The common approach amongst researchers is to adapt a two-period model, where prices in the first period are higher, and then prices get lowered in the second period, but at the same time, there may now be limited product availability (Shen and Su, 2007). Broadly speaking, the majority of capacity rationing models investigate how rationing affects strategic demand by making customers more inclined to purchase earlier at higher prices. This is an intriguing addition to seller strategies, and as a result, it is our ambition to include some form of capacity rationing in our model, which is to be presented next.

Chapter 8

Model Description

As seen from the previous chapters, simulating different market scenarios can provide us with valuable insights regarding how algorithms can be expected to perform when applied in a real marketplace. We aim to gain a better understanding of how the introduction of strategic customers and multiple machine learning algorithms might affect a market, and we analyze their performance using numerical analysis on a variety of different market scenarios. The following chapter will go into detail on how the market simulator is constructed and how it works.

Our simulator originates from the Learning Curve Simulator presented by Dimicco et al. (2003) but has several extensions which improve its representation of a realistic market. We chose the Learning Curve Simulator as a starting point because it captures important market dynamics and allows for the construction and testing of multiple scenarios within the same model. We expand the work done by Dimicco, Maes, and Greenwald (2003) and by implementing more sophisticated buying behaviors, the possibility of having more than two heterogeneous segments and implementing advanced machine learning agents, new insights can be gained. We start off our model description by looking at the overall market parameters, before embarking on how we have set up the buyers' and sellers' behaviors.

8.1 Market Parameters

Our simulated market is a finite market lasting for a specified number of time steps, or days, in which sellers are given an initial amount of goods to sell, and they have no option of replenishing their inventory in the case of a stock out. At each time step, the sellers post their prices simultaneously, and customers arrive hoping to purchase goods if the price is "right". The market ends when the specified number of periods has been run, and any inventory left is lost with a salvage value of zero.

This setup can be related to a variety of different real e-commerce scenarios. For instance, seasonal fashion items or tickets to a sporting event or concert have a finite selling horizon, as do some short cycled consumer electronic products. Furthermore, since adding more seats to a concert can be rather inconvenient, and seasonal goods might have to be ordered well in advance, there are few options for replenishing inventory.

We use the following market parameters to initialize a marketplace.

Table 8.1: The market parameters used in the model

Simulator input	Input values
The total number of buyers B	Integer value
The number of sellers S	Integer value
The number of selling periods T	Integer
Vector describing the ratio of consumers in each segment C	Real value [0,1]

By altering the number of sellers, we can simulate various degrees of competition amongst sellers, such as monopoly, duopoly and oligopoly markets. There is no upper bound on the number of possible sellers. However, we find that a large number of sellers utilizing machine learning algorithms drastically increase the run time of the program. The number of selling periods dictates for how long we want our selling season to last, or how many periods of sale we allow. Conceptually, the discretization of time into decision periods can be related to days, in the case of seasonal products or recurring one-time events, such as concerts or airplane tickets. There could also be an option to increase the number of selling periods to be sufficiently large so that any continuous-time counting process can be approximated by its discrete-time analog, as done by Talluri and van Ryzin (2004). However, such an approximation would require the probability of more than one event occurring in a decision period to be relatively small so that it is safe to assume that at most one event can happen per decision period. In order to achieve this, our model would need some additional adaptation, but this should be possible in a future version.

The length of vector C dictates the number of segments present in the market, and each scalar represents the initial ratio of buyers within each segment. The sum of the elements in vector C equals one, and the number of buyers in any given segment is then a product of the segment's given ratio and the total number of buyers. The total number of buyers dictates the maximum possible sales available in one simulation run, and we assume that this figure stays fixed during the whole selling period. The remaining number of latent buyers at any given time step is then equal to B less the number of arrivals so far.

8.2 Buyer Behavior

After initializing the market scenario, we need to model how we want each of the segments, or buyers, to behave in our market. Our simulations differ from previous work in this area, in that we enable the introduction of multiple segments with different buying behaviors. This allows us to create a more realistic marketplace, where consumers might have different valuations over time, behave strategically or prefer one seller above the others. Conceptually, we assume that the market can be divided into multiple buyer segments and that buyers from within each segment behave identically. Hence, one could choose to think of the segments as one single buyer and let the total number of buyers equal the number of segments. However, this would drastically increase the size and run time of the simulator. Each segment is initialized by its own set of the following parameters:

Table 8.2: The buyer behavior parameters used in the model

Simulator input	Input values
Maximum valuation V_{max}	Integer value
Minimum valuation V_{min}	Integer value
Standard deviation per day σ_{std}	Real value
Daily price variance σ_D^2	Real value
Buyer tactic	BH, CT, AS, FL
Preference for certain sellers β_s	Real value
Lifetime of buyers	Integer value
Buyer valuation over time	Increasing, Decreasing, Mid-dipping, Mid-peaking
Daily price distribution	Normal distribution

The following subsections will further explain the use of these parameters and the buyer behavior setup.

8.2.1 Product Valuation

Over the course of the market, the collective behavior of the buyers is defined by the four variables, minimum and maximum valuation, type of valuation curve and lifetime. The minimum and maximum valuations, together with the valuation curve choices, specify how the consumers' underlying valuations change over time. The value curve selection determines how the segments average reservation prices changes as time progress, whereas the minimum and max valuations functions as upper and lower bounds for the curve. We have implemented four different valuation curves, namely the increasing, decreasing, mid-dipping and mid-peaking, as illustrated by Figure 8.1. Our motivation for having a selection of different value curves is that we want to see how the algorithms cope with, i.e. a product that increases in popularity/valuation due to market trends over time, and if they can exploit increases in consumer willingness to pay.

To account for further differences in valuations we introduce some daily fluctuation from the given mean. At each time step, the value resulting from the value curve is modified by adding a sample value from a normal distribution with a mean of zero and standard deviation set to σ_{std}. Additionally, to account for seller preferences, irrespective of buyer tactics, the parameter β_s is used to add an additional level of seller preference. A segment's seller preference tries to represent real world differentiations among products and sellers. This could be due to factors such as better quality products, easier checkout or a better shopping experience, product features or brand loyalty.

A segment with no particular preference for seller s would have a β_s of one whereas segments which have certain seller preferences would have a β_s of greater than one. A beta less than one would, in this case, indicate a segment's negative attitude towards the given seller, resulting from perhaps a previous bad customer experience or lack of trustworthiness. Note that a buyer using, for instance, a bargain hunter tactic, may prefer a given seller and as such be willing to pay a premium over the lowest available price in the market. The resulting curves for a given segment, one for each seller, is then the final valuation curve for that segment.

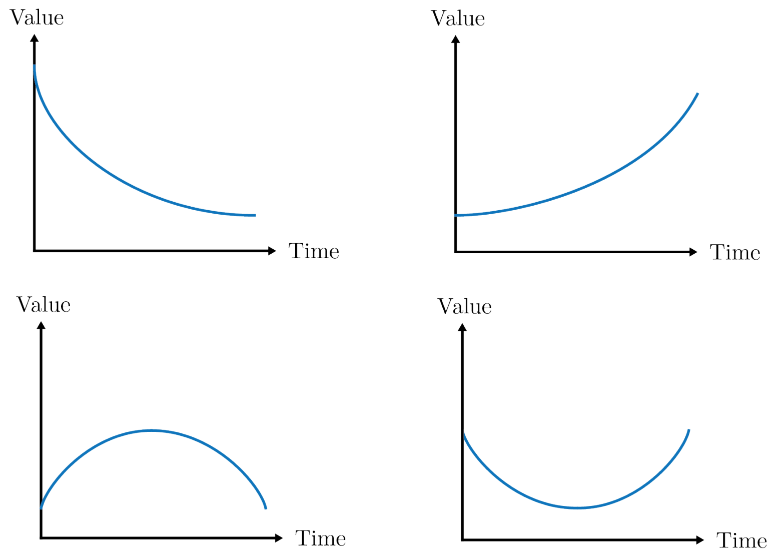


Figure 8.1: Illustrations of the different value curves utilized by customers

To generate the value curves we modify a quadratic expression to fit with the parameters of the segment. To exemplify this, the following equation represents the decreasing value curve:

$$V_{mean}(t) = \frac{V_{max} - V_{min}}{T^2} * (t - T)^2 + V_{min} + \beta_s N(0, \sigma_{std}^2) \tag{8.1}$$

8.2.2 Product Demand

The value curves map the collective relationship between time and mean valuations amongst the segments. However, we also need to implement a relationship between price and demand. To do this, we introduce the variable σ_D^2, which represents the daily spread of valuations. By taking the value curves output as a valuation mean, V_{mean}, and the realized number of arrivals μ for the given segment at time t, we can generate the demand curve $p(q)$ by equation 8.2.

$$p(q) = \frac{2\sigma_D^2}{\mu} * (q - \mu)^2 + V_{mean} - \sigma_D^2 \tag{8.2}$$

Since we are looking to find the demand for the posted price by the chosen seller, we need to invert Equation 8.2. Doing this gives us Equation 8.3, and by solving it for the posted price, we obtain the demand for the good at time t for the chosen segment. Note that the demand curve is decreasing and that we have only defined a range of values in which there is an explicit relationship between price and unit demand.

$$q(p) = \begin{cases} 0 & \text{if } p \geq V_{mean} + \sigma_D^2 \\ \mu & \text{if } p \leq V_{mean} - \sigma_D^2 \\ \mu - \sqrt{p - (V_{mean} - \sigma^2)}\sqrt{\frac{\mu^2}{2\sigma^2}} & \text{otherwise} \end{cases} \tag{8.3}$$

8.2.3 Customer Arrival

To represent the customers' arrival to the marketplace at each point in time, we employ a homogeneous Poisson distribution with arrival intensity λ. This is a commonly used distribution for representing customers arrivals in the literature. In our case, we set the arrival rate λ to be constant throughout the selling period and the total number of arrivals at each time step is then sampled from this distribution. The number of arrivals from any given segment is then calculated by taking the product of the realized number of arrivals μ and the segment's ratio. A future extension to our model would be to introduce an inhomogeneous Poisson process with some dependency among the segment's utility and arrival rate, to account for the likely event that unusually low or high prices should increase or decrease the number of arriving customers to the marketplace.

8.2.4 Strategic Customer Behavior

As explored in Chapter 7, previous research has presented a variety of ways for representing strategic behavior with consumers. We follow the approach of Ahn et al. (2007) where we assume that customers exerting strategic behavior stay active for a given number of days and return to the marketplace at later times to check whether prices have changed in their favor. We use the parameter lifetime of buyers for representing the "degree" of strategic behavior. We assume that buyers from a segment having a lifetime of more than zero will return to the market at later randomized times drawn from a uniform distribution if they were not able to buy when they first entered the market. This would happen if they receive a utility of less than zero from doing the transaction or if their seller of choice was in stock out. The "unsuccessful" strategic buyers are then stored as residual demand, keeping their original valuations fixed throughout their lifetime, and their number of random returns to the marketplace is given by the respective segment's lifetime. It follows that buyers arising from a segment having a lifetime of zero, will adopt a myopic strategic, and thus leave the market altogether if they are unsuccessful in purchasing an item at their arrival. This simple, yet quite effective way of introducing strategic customer behavior enables us to represent the customers utility using the following representation from the Shopbot model presented in Section 4.3.3:

$$u_b(p) = \begin{cases} v_b - p & \text{if } p \leq v_b \\ 0 & \text{otherwise} \end{cases} \tag{8.4}$$

8.2.5 Choosing a Seller

Despite the introduction of online shopbots and the general decrease in search costs in online retailing, empirical evidence reveals that not all buyers are actively looking for the lowest price possible. For instance, in their article regarding the impact of shopbot use and price dispersion Tang et al. (2010) found that in the case of online book retailing in the years 1999 - 2001 the share of shopbot users were about 25%. The share of shopbot users has likely increased in later years, still, arriving customers might behave differently and to incorporate this we have added several options controlling how consumers choose their sellers. To cope with this factor, we introduce buyer tactics to represent how consumers choose between the available sellers. Each segment can be initialized to following a certain seller evaluation strategy, and in agreement with other literature, such as Greenwald and Kephart (2000), we have defined these tactics as:

- Bargain Hunters (BH): These customers will check prices from all available sellers using i.e., a shopbot service and buy from whichever seller has the lowest price.

- Compare Two (CT): Customers following this strategy will choose two sellers at random and compare their prices and choose to buy from the one that has the lowest price.
- Any Seller (AS): These customers have an equal probability of choosing any one of the sellers, and choose one at random. This would be consumers without any store loyalty or limited knowledge of possible sellers.
- Full Loyalty (FL): Customers using this strategy have strong customer loyalty and will only buy from their preferred seller.

Note that Equation 8.4 applies for all of the above-presented buyer tactics. We further assume that the buyers who are not bargain hunters, only check prices with their selected sellers. If they gain no surplus from purchasing the product from these sellers, they either exit the market in the case of myopic behavior or stay in the market hoping for a future decrease in price in the case of strategic behavior. In the case where a seller reaches stock out before meeting the demand at a given time step, buyers that are unable to purchase are either lost or kept as residual depending on whether they are myopic or strategic. Furthermore, the buyers only evaluate active sellers, that is sellers that have goods available for sale at the beginning of the period, and in the unlikely event that a buyer receives the exact same utility from purchasing from different sellers, they make their choice randomly.

An alternative approach could be to let the buyers using "compare two" or "any seller" tactic iteratively search for prices at different sellers until one is found below its reservation price if their selected sellers provide them with no positive utility. Moreover, one could also let the buyers who have chosen a seller but finds themselves unable to purchase because their seller of choice reached stock-out some time during that time period, re-evaluate the active sellers and make a new seller choice.

8.2.6 Economic Welfare & Consumer Surplus

To better understand the allocation of welfare in the marketplace we need to find the economic welfare and consumer surplus. We find the economic welfare by integrating the demand curve from 0 to κ as represented by equation 8.5. In this case, κ is the number of goods sold to a specific segment at a given time.

$$\int_0^{\kappa} p(q)dq = \frac{2\sigma^2}{\mu}(\frac{1}{3}\kappa^3 - \kappa^2\mu + \kappa\mu^2) + (V_{mean} - \sigma^2)\kappa \qquad (8.5)$$

After calculating the economic welfare, we can then find the consumer surplus for that given segment by subtracting the seller surplus, given by $\kappa * p$. Consequently, doing this for all active segments allows us to add together the values to find the total economic welfare, consumer surplus, and seller surplus for any given time step.

8.2.7 Optimal Value

Since we are looking to analyze the performance of different pricing algorithms, it would be convenient to know the best possible pricing policy for any market scenario to use as a benchmark for our suggested algorithms. Unfortunately, due to the stochasticity of the problem and the many dimensions, finding this best practice policy is far from trivial. As a result, we have chosen instead to calculate the maximum possible seller surplus that can be extracted from the marketplace, and use this value to compare how effective the pricing algorithms are. In a way, what we are calculating is the total value of the population of consumers. We find this optimal value for a given segment by Equation 8.6:

$$\mu * (V_{mean} - \sigma_D^2) \tag{8.6}$$

Then by summing for all segments and over the entire selling period, we can comment on how much value that can be extracted from the marketplace by the sellers. Note, however, that extracting the maximum value from the market might not be possible as it in some cases can only be achieved by offering different prices to each segment. Also, in special cases where the customers $V_{mean} - \sigma_D^2$ is very small, it might not be optimal to meet the entire current demand μ, however, for all our simulated scenarios, Equation 8.6 calculate the optimal value.

8.3 Seller behavior

The simulator allows multiple sellers to operate within the same market using the same or different strategies, enabling us to see how the various seller strategies compare to each other and what kind of impact they have on each other. We have chosen to omit the sellers fixed and marginal costs and only focus on achieved revenue, as the effects of cost heterogeneity amongst sellers is of less significance to our analysis. Consequently, we define a price of zero in this market to represent the sellers' marginal cost. Furthermore, we want to represent a market that has a realistic amount of information available to the sellers. Hence, we do not assume that the sellers have any information regarding the actual demand, arrival rate of consumers or buyer behaviors, but need to estimate these parameters from the observed market. Each seller gets initialized by the following parameters:

Table 8.3: The seller parameters used in the model

Simulator input	Input values
Seller strategy	GD, DF, SDNN, RBNN, APNN, QL
Initial price P_0	Integer value
Initial inventory I	Integer value
Available inventory per t	Integer

The initial inventory at each seller dictates how many goods a seller can sell during the selling period as there is no option to replenish the inventory. We have also imposed an option to restrict the number of goods offered for sale each day, to avoid having sellers selling out their full inventory on day one due to a large number of arriving buyers, or to impose a strategic capacity rationing tactic.

The sellers' initial price effects the first day of sales, and in the case of a fixed price strategy the fixed price equals the initial price. As will be noted in our analysis we see that the choice of initial price will have an effect on the seller revenues as the initial price directly or indirectly affects the future prices for the seller strategies.

In the next subsections, we will have a closer look at the different seller strategies we have chosen to implement. The adaptive algorithms Goal-Directed (GD) and Derivative-Following (DF) are computationally cheap and have proved to be effective for real-time learning when processing power was a scarcer resource than it is today. They are also often used in the literature for comparing the performance of more complex algorithms. Of more complicated algorithms the simulator supports the use of three online learning artificial neural networks and a model-free reinforcement learning algorithm from the Q-learning family.

8.3.1 Derivative-Following

The Derivative-Following strategy demands little information and can be used in the absence of any knowledge or assumptions about the seller's competitors or the buyer demand function. The strategy adjusts its price by looking at the amount of revenue earned on the previous day as a result of the previous day's price change. It works by experimenting with different prices using incremental increases or decreases, continuing in the same direction until the observed profitability level falls, at which point the direction of movement is reversed (Greenwald et al., 1999). As a result, if yesterday's price change gave more revenue per good than the previous day, then a similar change in price is imposed. However, if the previous change made less revenue per good the strategy makes an opposing price change. In this setting revenue per good is equal to the sales price, except when no goods are sold, and consequently, the seller will always adjust its price aiming to sell at the highest price that generates sales.

$$\begin{aligned}
price_{i+1} &= price_i + (change_{i+1}) \\
change_{i+1} &= price_i * (\beta + \frac{T-i}{(T+i)*\alpha}) * yestSuccess * yestChange \\
yestSuccess &= \begin{cases} +1 & \text{if } revenue_i > revenue_{i-1} \\ -1 & \text{if } revenue_i < revenue_{i-i} \\ -1 & \text{if } revenue_i = revenue_{i-i} \end{cases} \\
yestChange &= \begin{cases} +1 & \text{if } change_i > 0 \\ -1 & \text{if } change_i < 0 \\ +1 & \text{if } revenue_i = revenue_{i-i} \end{cases}
\end{aligned} \tag{8.7}$$

Equation 8.7 computes the price for a particular day, $price_i$, by adjusting yesterday's price change by a percentage change, $change_{i+1}$, scaled by a ratio based on its progress through the market. The scaling ratio ensures that the day of the market is accounted for and is an adaptation of the original DF strategy for a finite market (Dimicco et al., 2003). The constant β guarantees a minimum price change each day, whereas α counterbalances beta to ensure that the price changes are not too large at the beginning of the market. We have added an additional condition to the original algorithm presented by Greenwald et al. (1999); that ensures that if the revenue of the previous day equals the revenue of the current day, the price is decreased. This only happens if no goods are sold, and as such the seller is charging a price above the segments' valuations leading us to reduce the price in the next period.

8.3.2 Goal-Directed

The Goal-Directed strategy adjusts its price by attempting to reach the goal of selling its entire inventory by the last day of the market, and not before (Dimicco et al., 2003). Its aims at selling to the highest paying buyers on each day, and tries to achieve this by lowering prices when sales are low and raising prices when sales are high, hence it tries to pace its sales over the entire selling period.

$$
\begin{aligned}
p_{t+1} = p_0 + p_0 * \frac{\sum_{n=1}^{t} goodsSold_n - expGoodSold_t}{expGoodSold_t} * scale_t \\
expGoodsSold_t = t * \frac{initalInventory}{T} \\
scale_t = \frac{T}{2 * (T - t)}
\end{aligned}
\tag{8.8}
$$

The price posted on a given day is calculated by adjusting the price the seller offered at the beginning of the market by the ratio of the number of goods sold thus far in the market by the number of goods it had expected to sell by day t. The scaling factor improves the strategy's ability to make price adjustments at the end of the market and enables more dramatic price changes during the final days when sales are most important if the seller is to reach stock-out just in time (Dimicco et al., 2003).

8.3.3 Neural Networks

In our model, we have implemented three different neural networks, namely the Sales-Directed Neural Network (SDNN), the Aggressive Pricing Neural Network (APNN) and the Revenue Based Neural Network (RBNN). Our motivation for using neural networks for pricing is that they can earn complex relationships from experience, and enable quite effortless offline training. Also, previous research has gained solid results from demand learning using neural networks. Furthermore, we find potential in implementing the pricing decision directly in the network, following the approach of Kong (2004) and Ghose and Tran (2009).

All neural networks act by building a complex curve out of simpler ones in order to fit the data presented to it. They consist of minimum three layers, one input layer, one output layer, and any number of hidden layers. Each layer has a given number of nodes used to represent values, and each node apart from the input node(s) forms a weighted sum using a simple curve. Typically this transformation function is a sigmoid function represented by equation 8.9, and this will also be the transformation function we use for all our neural networks.

$$
f(x) = \frac{1}{1 + e^{-x}} \tag{8.9}
$$

Figure 8.2 illustrates the basic structure of our three-layered neural networks. The matrix X consists of our input variables, the matrices $W^{(1)}$ and $W^{(2)}$ contain the weights, and the matrix y stores the output parameters.

When calculating the values, each neuron, or node, receives a signal from the previous layer, and each one of those signals is then multiplied by a separate weight value. Then the weighted input to the given node is passed through a sigmoid function, which scales the output to a fixed range of values. The resulting output is then passed on to the neurons in the next layer. A forward pass in a three-layered network can be represented by:

$$\begin{aligned} z^{(2)} &= XW^{(1)} \\ a^{(2)} &= f(z^{(2)}) \\ z^{(3)} &= a^{(2)}W^{(2)} \\ \hat{y} &= f(z^{(3)}) \end{aligned} \tag{8.10}$$

The networks intelligence lies in the weight values between the neurons, and by adjusting these weights, the network can adapt to the underlying data. Perhaps the most common approach for adjusting these weights is using a learning algorithm called backpropagation which will be presented next.

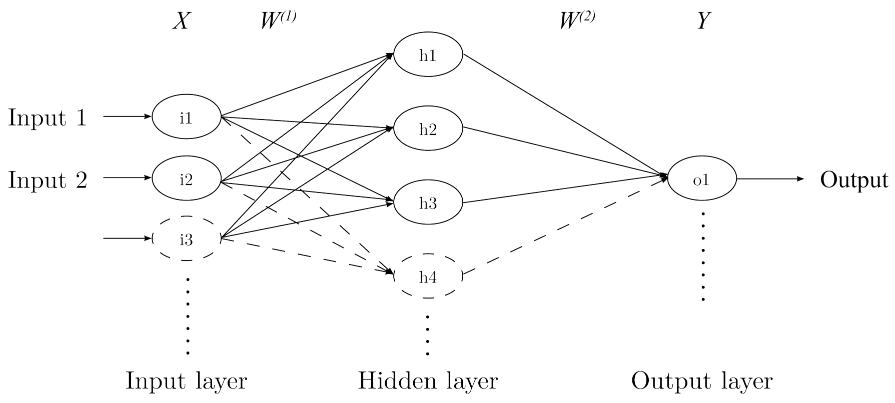


Figure 8.2: The structure of a three-layered neural network

Backpropagation

Back Propagation is a form of gradient descent in that we are looking to find in which direction we should move to decrease our networks error, or cost, given by:

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \tag{8.11}$$

Training our network is then a matter of minimizing our costs by adjusting the weights trying to find the combination of weights that makes our costs as small as possible. This can be achieved by using batched gradient decent and partial derivation of our cost function J, represented by Equation 8.12:

$$\begin{aligned}
\frac{\partial J}{\partial W^{(1)}} &= X^T \delta^{(2)} \\
\frac{\partial J}{\partial W^{(2)}} &= (a^{(2)})^T \delta^{(3)} \\
\delta^{(2)} &= \delta^{(3)} (W^{(2)})^T f'(z^{(2)}) \\
\delta^{(3)} &= -(y - \hat{y}) f'(z^{(3)})
\end{aligned} \tag{8.12}$$

A method is batch styled if we are using all of the input-output examples at once. After performing a backpropagating step, we need to update our current $W^{(1)}$ and $W^{(2)}$ according to the calculated $\delta(2)$ and $\delta(3)$.

Although gradient descent is a clever method, it provides no guarantees of convergence to a good solution; that we will converge within a certain amount of time; or that we will converge to a solution at all. There exists a wide choice of methods suitable for training or updating a neural network from the mathematical optimization research community, ranging from simple heuristics to rigorous theoretical methods. LeCun et al. (2012) analyze the convergence of back propagation learning, and evaluates a variety of different methods. For our networks we will start by using the simplest learning procedure for standard backpropagation learning where our weights are iteratively adjusted as follows:

$$\begin{aligned}
W_t^{(1)} &= W_{t-1}^{(1)} - \eta \frac{\partial J}{\partial W^{(1)}} \\
W_t^{(2)} &= W_{t-1}^{(2)} - \eta \frac{\partial J}{\partial W^{(2)}}
\end{aligned} \tag{8.13}$$

Data normalization and Overfitting

Before we can employ our neural networks, we also need to account for the differences in the units for our data. Intuitively, we need to make sure that our neural network does not compare apples to oranges, but rather apples to apples, and as such we need to scale our data. As all of our data are positive, we simply chose to normalize our data by dividing the input and output values by their maximum possible value, to scale them to a number of 0 and 1. In our case we normalize the current inventory by dividing it by the initial inventory and the price is divided by a maximum price well above the customers maximum valuation.

Overfitting is one of the problems that can occur during training of neural networks. It happens when the error on the training data set is driven to a very small value, but when new data is presented, the network fails to represent it appropriately and thus a significant error is generated (Russell and Norvig, 2010). The problem of overfitting is most common when the model is excessively complex. Hence, one method for avoiding this issue is to design a network that is just large enough to provide an adequate fit. Since all our implemented neural networks are rather limited in size, we can assume that overfitting will not occur in our model.

Sales-Directed Neural Network

The sales-directed neural network implemented in our model is based on the neural network presented by Kong (2004) and consist of three layers with one input node, one hidden node, and one output node. In our simulated market, the price is the only parameter controlled by the seller agent. To find a suitable price, the SDNN-strategy uses observed sales to calculate the error between the observed sales and desired sales. In this setting desired sales are calculated based on the number of goods left in stock and the number of days left in the market. In other words, sellers using this strategy aim to sell an equal amount of goods each day for the remaining time period, closely resembling the GD-strategy. If there is a discrepancy between the desired sales and actual sales, the error is then backpropagated through the network and small changes are made to the weights in each layer. Based on the resulting network, which is then our "best guess" of the relationship between price and sales, we calculate the appropriate price based on our expected sales and desired sales which can bring about the desired sales quantity.

The functional relationship between price and sales quantity represented by this neural network can be represented by:

$$Sales = f(W^{(2)} * f(W^{(1)} * Price) \tag{8.14}$$

Since we are trying to find what price the seller should charge to reach its desired sales, we need to invert the neural network represented by Equation 8.14 and solve for the price. Inverting a Neural Network is, in general, an ill-posed problem because the mapping from the output space to the input space is a one-to-many mapping (Lu et al., 1999). In this case, where we only have one node in each layer, we can represent the inverted network using Equation 8.15. However, we need to be wary of the infeasible solutions and take some precaution when solving for the price. Note also that the matrices $W^{(2)}$ and $W^{(1)}$ for the SDNN is, in this case, 1-by-1.

$$g(sales) = \begin{cases} -\frac{W^{(1)} \ln(-1-W^{(2)})}{\ln(\frac{1}{sales}-1)} & \text{if } sales > 0 \\ Infitiy & \text{if } sales \leq 0 \end{cases} \tag{8.15}$$

Due to the issues of inversion, the SDNN strategy is quite sensitive in regards to the initial weights we provide. As an example, we see that for our normalized sales the denominator of equation 8.15 will always be positive. The numerator, on the other hand, is only defined in real numbers for values of $W^{(2)} < -1$, and consequently $W^{(1)} > 0$ or else the returned price would be negative, or we could be dealing with imaginary numbers. An exemplifying pseudocode for the SDNN-algorithm is provided in Algorithm 1

The inverted SDNN strategy takes the expected sales quantity as its only input, and the price is then its output. Consequently, the SDNN strategy needs no information about the buyer population or the competitors prices. It learns from the observed sales by building its own representation of expected sales from a range of prices. As a result, the relationship between market demand and competitor pricing strategies should indirectly be reflected in the estimated relationship between the sellers price and observed sales. By letting the network learn online, that is in real time; it should be able to adapt to the current market conditions.

Aggressive Pricing Neural Network

The Aggressive Pricing Neural Network tries to learn the lowest price offered by the other sellers at any given time, and aims to undercut their prices by a certain percentage δ. We have given the APNN one input node, three hidden nodes, and one output node. As its input, it takes the current time step, and the output is the network's best guess at what the lowest price in the market might be. Its values are then readjusted after the market closes at the given time step, and it updates its weights using backpropagation by looking at the difference between its estimated lowest price and the actual lowest price.

Algorithm 1 Pseudocode for the Sales-Directed Neural Network

```
procedure Run Neural Network
Intitialize:
    Starting time $t = 1$
    Predicted price $p^t$ = initial price
    Desired sales $s^t$ = inventory$^t/T$
    Weigth $W^{(1)}$ = constant
    Weigth $W^{(2)}$ = constant
Loop:
    for each time step $t < T$ do
        Post price $p^t$
        Observe current sales $c^t = inventory^{t-1} - inventory^t$
        if current sales $c^t$ = desired sales $s^t$ then
            No prediction error $\rightarrow$ No change in weights
            Set $s^{t+1}$ = inventory$^t/(T - t)$
            Set $p^{t+1} = g(s^{t+1})$
        end if
        if current sales $c^t! =$ desired sales $s^t$ then
            Set prediction error = $s^t - c^t$
            Adjust $W^{(1)}$ and $W^{(2)}$ according to backpropagation
            Set $s^{t+1}$ = inventory$^t/(T - t)$
            Set $p^{t+1} = g(s^{t+1})$
        end if
    end for
end procedure
```

$$price = f(W^{(2)} * f(W^{(1)} * t) * \delta \qquad (8.16)$$

The APNN is predominantly a price war inducing seller strategy, as it will aggressively price its goods to undercut its competitors' prices. Furthermore, since it has no way of learning the actual product demand, it relies on the intelligence of others. It is for instance rather useless in a monopoly setting, but can be an interesting addition to an oligopolistic market. Another interesting feature of the APNN-strategy is that it resembles some of the available real-world pricing algorithms. Companies like Wiser, offer sellers an automated software agent that prices their goods marginally below other sellers (WiseCommerce, 2015).

Revenue Based Neural Network

Our suggested Revenue Based Neural Network tries to learn what price it should be charging for it to reach its desired sales at any time step. It has two input nodes, three hidden nodes, and one output node. The input data X consists of the time step we are currently in and the number of desired sales with the seller. The desired sale is calculated in the same way as for the SDNN-strategy. The output of the network is the algorithm's best guess at what revenue we should achieve with the stated inputs. We then use these parameters to calculate the seller's price and then observe what revenue we actually achieve. By evaluating the difference between the desired revenue and the currently realized revenue, we can find the networks' error and update the weight matrices using backpropagation when necessary. Algorithm 2 provides a pseudocode for the RBNN-strategy.

The RBNN and SDNN strategy have some similarities, in that they both aim to sell an equal amount of inventory at each selling period. However, where the SDNN needed to be inverted in order for us to find the relationship between price and demand, the RBNN strategy tries to learn this relationship implicitly by evaluating the revenues. Since the RBNN knows the desired sales and can learn the desired revenue by evaluating realized revenue, we can then find the desired price for next period by dividing the desired revenue by the desired sales.

Like the SDNN, the RBNN strategy needs no information about the buyer population or the competitors prices. However, because it is not affected by the issues of invention, we can effectively train the RBNN offline, enabling us to initialize an expected relationship between time, desired sales, and revenues, utilizing very little information.

Algorithm 2 Revenue Based Neural Network

```
procedure RUN NEURAL NETWORK
Initialize:
    Starting time $t = 1$
    Desired sales $s^t$ = inventory$^t/T$
    Desired revenue $r^t$ = initial price $* s^t$
    Weigth $W^{(1)}$ = constant
    Weigth $W^{(2)}$ = constant
Loop:
    for each time step $t < T$ do
        Post price $p^t$
        Observe current revenue $Cr^t = (inventory^{t-1} - inventory^t) * p^t$
        if current revenue $Cr^t$ = desired revenue $r^t$ then
            No change in weights $\leftarrow$ No prediction error
            Set $s^{t+1}$ = inventory$^t/(T - t)$
            Set $X = [t + 1, s^{t+1}]$
            Set $r^{t+1} = f(W^{(2)} * f(W^{(1)} * X)$
            Set $p^{t+1} = r^{t+1}/s^{t+1}$
        end if
        if current revenue $Cr^t! =$ desired revenue $r^t$ then
            Set prediction error $= 0.5 * (s^t - Cr^t)^2$
            Adjust $W^{(1)}$ and $W^{(2)}$ according to backpropagation
            Set $s^{t+1}$ = inventory$^t/(T - t)$
            Set $X = [t + 1, s^{t+1}]$
            Set $r^{t+1} = f(W^{(2)} * f(W^{(1)} * X)$
            Set $p^{t+1} = r^{t+1}/s^{t+1}$
        end if
    end for
end procedure
```

8.3.4 Q-learning

Given the parametric nature of the model presented, it might appear that it is possible to explicitly define the transition structure of the underlying processes for finding an optimal price policy. However, the computation of transition probabilities is highly non-trival in this case, and it can be extremely challenging for realistic cases. Also, since the customers' demand functions are unknown to the sellers, we need an approach that would work with any form of demand function. These two reasons, in addition to current literature's focus on reinforcement learning algorithms, contend our motivation for exploring a model-free RL-approach for the pricing of goods.

The family of Q-learning algorithms is one of the most significant and actively investigated reinforcement learning algorithms (Watkins, 1989). They have the property that they do not need a model of their environment, hence the name model-free modeling, and such they are suited for online learning. Given a Markov Decision Problem, Q-learning has been proved to converge to exact optimal value functions and policies when lookup table representations of the Q-function are used, which is feasible in small state spaces (Watkins, 1992).

For our problem, we define the set of states $\mathcal{S}_t$ to be the number of remaining inventory left at the seller at time t. The set of actions $\mathcal{A}$ defines the allowable actions to be taken by the agent, and this is simply a discretized range of possible prices bounded by a lower and upper price bound. The reward function, as well as the transition function, are unknown to the seller. Hence we can only observe the immediate reward from taking action a in state s and the transition from $s \to s'$ given a. The reward the agent receives after a transition is simply the revenue gained from transitioning from state $s \to s'$ by taking action a as given by Equation 8.17.

$$r = (s - s')a \tag{8.17}$$

Further, we let $Q(s, a)$ represent the discounted long-term expected reward of an agent for taking action a in state s. The discounting of future rewards is done at a rate of γ, such that the value of an expected future reward at n time steps in the future is discounted by γ^n. The values of $Q(s, a)$ are stored in a look-up table containing a value for each possible state-action pair and is initialized to being zero for all entries. Then, the procedure for solving $Q(s, a)$ is to repeat the following procedure infinitely:

1. Select a particular state s and action a, observe the immediate reward r and the resulting next state s'
2. Adjust $\hat{Q}(s, a)$ according to equation 8.18

$$\hat{Q}(s,a) \leftarrow (1-\alpha_t)\hat{Q}(s,a) + \alpha_t(r + \gamma \max_{a'} \hat{Q}(s',a')) \tag{8.18}$$

This will cause $\hat{Q}(s,a)$ to converge to $Q(s,a)$ and solves the Bellman equation. However, we need to visit each state-action pair infinitely often if we are to have any chance at converging to the optimal policy. To cope with this issue, a common approach is to initialize $\hat{Q}(s,a)$ to some values believed to be good and then introduce a simulated annealing approach, in which we allow for some random action with the probability of Epsilon, ϵ. Implying that every once in a while, we will take a random action helping us to explore more of the solution space, and not just take an action that we initially believed to be good since we have not explored enough of the solution space. Using a greedy limit infinite exploration tactic should let $\hat{Q} \to Q$ and $\hat{\pi} \to \pi^*$ by decaying ϵ as time progresses, making the probability of taking a random action smaller and smaller as we progress and learn. Using this approach, we are continuously learning more about our solution space, while also using what we have previously learned. Implying that this should provide us with a good solution to the exploration-exploitation trade-off. However, finding good initialization values for $\hat{Q}(s,a)$ is not always easy.

Since we are simulating multiple agents in the same environment, some issues might arise when utilizing Q-learning. The problem of having multiple agents adapting simultaneously to the underlying market conditions is in general non-Markov, in that each agent then provides an effectively non-stationary environment for the other agents. This is problematic in that it breaks the stationarity requirement needed for the Q-learning algorithm to converge to its optimal policy. Consequently, we do not know whether any global convergence will be obtained, and if so, we have no way of telling if such solutions are optimal (Tesauro and Kephart, 2002). However, this does not imply that we expect the Q-learning algorithm to behave badly; it rather means that we have no theory enabling us to say how well the Q-learning algorithm is supposed to work. This absence of theoretical guidance motivates us to develop an empirical understanding of multiagent Q-learning performance in our simulations.

Chapter 9

Strategy Analysis

The following chapter is used for analyzing different market scenarios using the model presented in the previous chapter. Our ambition is to learn more about how the dynamics of the market changes and how the different algorithms perform under various circumstances. Hopefully, by examining this simulated market, we can gain new knowledge that can provide us with further insights regarding the opportunities and obstacles when using machine learning for pricing goods in an electronic marketplace. We programmed the model in Java 1.8, and all simulations were run on a MacBook Pro with 2.7 GHz Intel Core i7 and 16 GB of RAM. The Java code for the implemented model is attached as a .zip-file.

We start the strategy analysis by discussing some of the implications and issues we found needed to be addressed when implementing machine learning algorithms. Then we evaluate a "balanced" market scenario consisting of seven segments and four sellers, before performing a sensitivity analysis, enabling us to comment on the different market parameters' impact on the market. After that, we experiment with varied value curves, compositions of sellers and algorithms.

9.1 Machine Learning Insights

Before we embark on analyzing the market scenarios and strategies, there are some insights we need to address regarding the machine learning algorithms. Because machine learning algorithms try to learn underlying parameters of the data presented to them, we face the issue of training our algorithms. A naive approach would be to "release" the algorithms into the marketplace, and just let them learn online as time progresses. This method would imply that we need to perform a vast number of simulator runs in order for the algorithms perform satisfactorily. Consequently, we find that we have a lot to gain by initializing our algorithms using some training data and performing off-line learning. However, as will be evident, this might not always be straightforward.

9.1.1 Initializing the Neural Networks

The RBNN and APNN strategies are both originally supplied with randomized weight matrices at their initialization at the beginning of the market. As a result, we then have nine random weight elements for the RBNN and six random weight elements for the APNN. To help the neural networks perform better from the start, we can utilize offline training of the network using a small data set consisting of three input-output examples. This training data can be regarded as market insights acquired by the seller from some form of market research; that tells us something about what the seller initially believes the relationship between input and output to be. For both the RBNN and the APNN we have chosen to provide them with an estimate of what the lowest price or expected revenue could be at the beginning, mid and ending of the market. These training values enable the networks to get an understanding of what the weights should be before we start the actual simulations. Hence, if the sellers believe that prices will decrease with time, they would train the network using data conforming to this hunch, and thus, they have already initialized a curve with the preferred shape and in an appropriate region before they offer their products for sale. Initializing the networks in this way, makes them perform way better, as they now only have to evaluate whether the seller's initial guess was right, and the chances are that the initial guess is closer to the actual values than the values produced by the randomized weight matrices. To perform this offline training, we let the neural networks adjust their randomized weight matrices from the training data set using back propagation for 50000 iterations using a learning rate of 0.1.

The SDNN on the other hand needs to be initialized in a slightly different manner, mainly due to the issues of inversion. As a result, simply training the network using the same approach as for the RBNN and APNN could easily lead us to have weights that are outside the defined scope of the inverted function. This brings us to another approach, in which we initialize the weights in such a way that the initial desired sales $\approx f(W^{(2)} * f(W^{(1)} * \frac{initialprice}{pricescale})$. Doing this, we provide the network with a feasible starting point, and thus, points it towards what region we expect the weights to be in for the coming next periods. Given a feasible starting point, we find that the likelihood of the weights being adjusted into the infeasible region can be neglected.

9.1.2 Initializing Q-learning

Initializing the Q-learning algorithm, on the contrary, is a much more rigorous procedure in comparison with the neural networks initialization. This is mainly due to the use of a look-up table; that aims to store the $Q(s, a)$ value for all possible state-action pairs. Since we do not know the transition probabilities of moving from one state to another, initializing the look-up table with satisfactory values is hard. Despite our best efforts, we have not been able to find a suitable way of estimating the value of being in any particular state using only a few training examples. That is not to say it cannot be achieved. One plausible approach could be to formulate a best-guess transition function, but examining this procedure would be outside the scope of this thesis.

However, we have had some moderate success using two different approaches. Firstly, we can choose to initialize a starting price policy, and then use simulated annealing to update our policy as we progressively learn more about the market. The issue with this approach is that we are only initializing the policy and not the underlying Q-values. For instance, we could choose to initialize the Q-learning algorithm to perform pricing according to the Goal-Directed strategy, but this initial policy is quickly deviated from as we update the Q-values for each state we visit. The problem being that since all Q-values are zero to begin with, almost any visited state will be updated with a Q-value larger than zero, and thus be the new preferred state in the policy. Furthermore, after the first run of the simulator, whatever policy generated from the GD-policy during that specific simulation, will now be the policy in the next period. Hence, its quite possible that we update the policy with a suboptimal action since the algorithm has very limited data available for determining whether the initial GD-policy, the full GD-policy or the randomly chosen action is better. As a result, this approach can perform satisfactory for the first few simulation runs, but we are still dependent on a vast number of iterations in order for the algorithm to perform well.

A second approach is to initialize the Q-values and just neglect the fact that we do not know the transition function. Doing this we can set the values of $\hat{Q}(s, a)$ doing the following procedure presented in Algorithm 3.

This procedure places a value for each state-action pair, however, since we do not know the probability of the transitions, the Q-values do not reflect the actual dynamics of the market. Following this approach, we would always make the algorithm choose the highest action, or price possible. We have capped this price, to a “max price”, and as such we are then effectively initializing a fixed pricing policy. Consequently, we do not experience much improvement in the algorithms performance doing this, as such, we are better off just letting the algorithm perform random actions.

Subsequently, these issues make the Q-learning algorithm a bit more tedious

Algorithm 3 Pseudocode for initializing $\hat{Q}(s,a)$ values

```
procedure INITIALIZE
Loop:
    for each time step $t < T$ do
        for each stock of inventory $I < initialinventory$ do
            for each action $p < maxprice$ do
                $\hat{Q}(s,a) = (I - \frac{I*t}{T}) * p$
            end for
        end for
    end for
end procedure
```

to handle. The two suggested procedures would not be particularly interesting to evaluate, because they do not reduce training time much, nor do they have any significant impact on increasing revenues. The fact that we have no good method for off-line training implies that we need to let the algorithm learn everything online. We are able to produce rather good complete policies from the Q-learning algorithm despite this, but only after a few million simulation runs. Hence, we have chosen to omit Q-learning from the following sensitivity analysis, because the many market scenarios and long run-time, makes it too time-consuming to include. We will, however, return to Q-learning performance in Section 9.7.

9.2 Understanding Market Dynamics

In order for us to understand how the different agents and market parameters influence each other, we start of our analysis by performing a form of sensitivity analysis on a "balanced" base scenario. Then, by altering the parameters individually from this base scenario, we can gain insights regarding how each individual parameter affects the overall market.

For our base scenarios we have chosen to activate seven segments and four sellers. Our motivation for considering oligopolistic markets with four sellers stems from the increased price transparency in modern e-commerce, which suggests that prices should be close to marginal cost. Hence, these transparent competitive markets cannon substantiate investments, and thus, we can infer that only a few players can remain in the market. Furthermore, utilizing seven segments allows us to adequately represent a heterogeneous market.

The total number of buyers is set to 1000 and we define one run of the market to last for 100 time periods or days. We further want to ensure that virtually all 1000 buyers has arrived when the market ends, and an appropriate arrival rate for achieving this is $1.1 * \frac{NumberofBuyers}{NumberofPeriods}$. We will use this arrival rate calculation throughout all coming market scenarios. Table 9.1 contains the seven segments starting parameters.

Table 9.1: The simulator inputs used for the segments in the base scenario

Simulator input	Seg. 1	Seg. 2	Seg. 3	Seg. 4	Seg. 5	Seg. 6	Seg. 7
Segment ratio	0.2	0.2	0.1	0.125	0.125	0.125	0.125
Max valuation V_{max}	350	500	500	500	500	500	500
Min valuation V_{min}	100	100	100	100	100	100	100
Price variance σ_D^2	25	25	25	25	25	25	25
Daily price deviation σ_{std}	5	10	10	10	10	10	10
Buyer tactic	BH	AS	CT	BH	BH	BH	BH
Seller preference β_s	N/A	N/A	N/A	#1	#2	#3	#4
Buyer lifetime	10	0	0	0	0	0	0

For our base scenario we have a balanced mix of different segment characteristics. However, we have chosen to let all segments have a decreasing value curve. Our use of the decreasing value curve stems from our belief that in most markets, consumers willingness to pay decrease in time. Segment 1, represents 20% of the consumers, and as stated by a lifetime of 10, all buyers from within this segment behave strategically. As a starting point, we have chosen to keep the percentage of strategic buyers at 20%, relating to the study performed by Li et al. (2014), and consequently, the remaining six segments all utilize myopic behavior. Furthermore, we assume that because of their strategic behavior, these buyers would utilize the Bargain Hunter tactic and have a lower max valuation and daily price deviation than the average consumer due to their rational behavior and price awareness. Segment 2 to 7 all share the same max valuation, min valuation, price variance and daily price deviation as a starting point. However, they vary in terms of buyer tactic and seller preferences. Segment 2 also amounts to 20% of the market, but they choose their seller randomly and have no seller preference. Segment 3 constitutes 10% the market and choose their sellers by comparing the prices of two randomly selected sellers. Segment 4 to 7 share the exact same parameters, except for their seller preference. These segments have preferences for one seller each, and we have initialized the seller preference β_s to be 1.1 for their preferred seller and 1 for the others.

Altogether, these segments have been initialized this way with two things in mind. Firstly, we want to have a heterogeneous market consisting of multiple segments having different behaviors. Secondly, our base scenario should be balanced, meaning that each seller should have the same probability of success in the market, implying that we have no favoritism towards one seller or another. Because of this, we can evaluate which of the pricing algorithms that performs best.

In regards to the sellers, we have initialized them with the parameters stated in Table 9.2. Continuing our concept of a balanced scenario, we have chosen to provide all sellers with an equal amount of inventory, available inventory per day, and initial price. However, they all employ the use of different pricing algorithms. We implement the two adaptive algorithms, the Derivative-Following and the Goal-Directed, and two neural networks, the SDNN and the RBNN in our base scenario. Arguably, we could have chosen to use any of the implemented seller strategies for our base scenario, however, as previously noted the Q-learning algorithm is in need of a vast number of iterations to perform satisfactorily. The APNN's aggressive pricing strategy increases the price wars in the market, and will often cause prices to fall near marginal cost. Therefore, we chose to omit these two from our starting point, but we will return to these pricing algorithms later.

Table 9.2: The simulator inputs used for the sellers in the base scenario

Simulator input	Sel. 1	Sel .2	Sel. 3	Sel. 4
Initial inventory I	250	250	250	250
Initial price P_0	499	499	499	499
Seller strategy	GD	DF	SDNN	RBNN
Available inventory per t	30	30	30	30

The DF-strategy is initialized with a $\beta = 0.05$ and $\alpha = 5$, which proved us with a 5% minimum change in price each day, and the α counterbalances β to ensure that the price changes are not too large at the beginning of the market. As explored in Section 9.1.1, we can significantly improve the performance of the neural networks by performing some offline training. Using the same methodology as previously presented, we initialize the weights of the SDNN to be $W^{(1)} = [0.9]$ and $W^{(2)} = [-4.9]$. Whereas the RBNN strategy has been trained for 50000 iterations using the following normalized training data:

$$X = \begin{bmatrix} 10 & 2.5 \\ 50 & 2.5 \\ 80 & 2.5 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.05 \\ 0.5 & 0.05 \\ 0.8 & 0.05 \end{bmatrix}, Y = \begin{bmatrix} 450 * 2.5 \\ 230 * 2.5 \\ 120 * 2.5 \end{bmatrix} = \begin{bmatrix} 0.0225 \\ 0.0115 \\ 0.006 \end{bmatrix}$$

After initializing the agents and consumers, we run the simulator for 100 iterations, thus enabling us to see how the machine learning algorithms adapt to the market conditions and each other. At each run, all parameters are reset to their original values, except the neural networks' weights. Consequently, the neural networks can transfer previous learning to the next run, as opposed to the adaptive algorithms.

Figure 9.1 presents six figures of the market evolve during one simulation run. The first line of figures presents how the sellers' prices and customer segments' valuations evolve with time. The second line of figures, presents the development of each seller's current inventory, and the final line of figures map the revenue at each seller for each time step. The left figures represent data from the first simulator run, whereas the right figures represent the final, 100th simulation run. Note that for all figures presenting market price in this chapter, i.e. Figure 9.1a and Figure 9.1b, we plot the segments average valuation amongst all sellers, as it would be impractical to plot four value curves for each of the seven segments. In addition, the value curves represents the development of the segments' maximum valuation $V_{mean}+\sigma_D^2$.

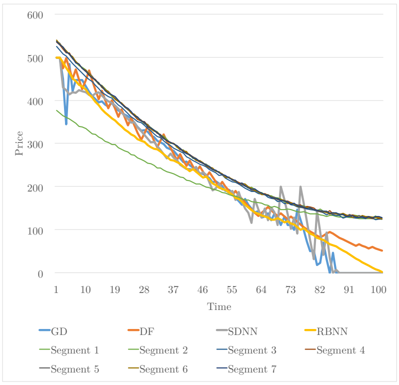
Table 9.3 contains the overall averages for all hundred simulator runs and contains the optimal value (OV), economic welfare (EW), consumer surplus (CS), seller surplus (SS), and the individual revenues for each of the sellers.

Table 9.3: The average surpluses and revenues for all simulator runs in the base scenario

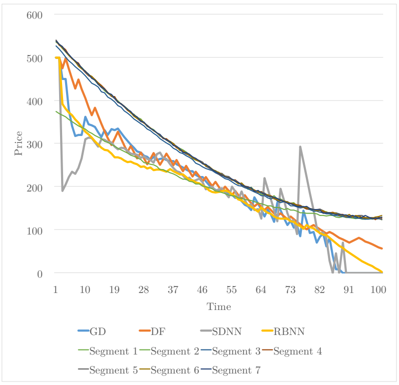
	OV	EW	CS	SS	GD	DF	SDNN	RBNN
Value	173416	163614	35556	128058	26425	15560	41762	44309

In general, we see that the best performing algorithm is the RBNN, with the SDNN coming in at a close second. The GD and DF performances are well below the performance of the machine learning algorithms, and especially the DF fails at competing for revenues in the market. From these simulations, we find that we are not able to extract the optimal value from the market, as the total seller surplus generated is below that of the optimal value. This is predominantly due the loss of sales resulting from prices being too high. Furthermore, we find that the sellers claim nearly 80% of the economic welfare.

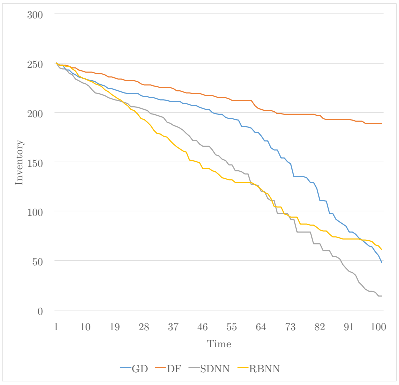
Considering the first simulator run, we see that the daily price dispersion between the different sellers is rather small. They are all quite good at following the myopic segments valuation curves, at least for the first 50 days. After this, the sellers fail at keeping prices up, due to more aggressive pricing initiated by the SDNN strategy. The SDNN-seller starts outperforming rather small price adjustments to meet its desired sales each day, but when the end of the selling period is near, its price changes are more dramatic. This happens as a result of its decreasing stock of inventory and fewer remaining days, which affects its desired sales and makes it more sensitive to larger inventory movements, which in turn affect its weight adjustments. We see that around day 60, the SDNN-seller drops its price more than "usual", predominantly because its weights are not optimally adjusted. Unfortunately for the SDNN-seller, the price drop are a bit too large, and as a result more inventory than desired gets sold. For the remaining days, the SDNN-seller's prices get increasingly more volatile alternating between overshooting and undershooting its prices, thus selling no or "too many" goods. Despite its effort to try and increase prices to make up for its previous suboptimal prices. The SDNN's bad judgment at day 60 has caused the rest of the sellers to decrease their prices to the same level, and ultimately they are all forced into a lower price region. As a result, they are collectively failing to realize



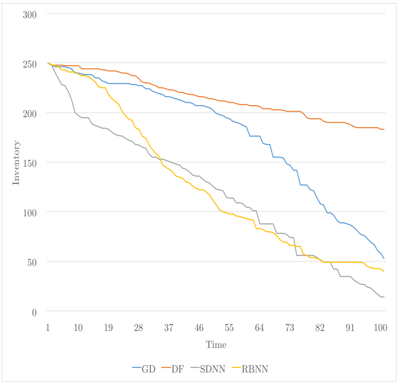
(a) 1st Run, Market Prices



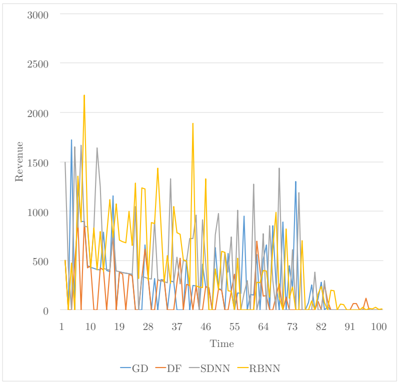
(b) 100th Run, Market Prices



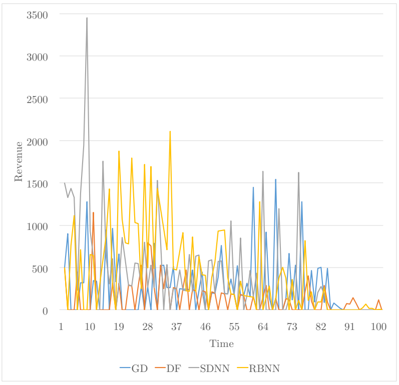
(c) 1st Run, Inventory Development



(d) 100th Run, Inventory Development



(e) 1st Run, Revenue



(f) 100th Run, Revenue

Figure 9.1: Market prices, inventory development and revenues for the balanced base scenario

that the SDNN-seller performed a bad move, and as such, revenues are lost to the benefit of the consumers.

Further, it is evident that the DF-seller has the highest average price throughout the selling period. Because its goal is to sell to the highest valuing customer at all times, it stays somewhat away from the price war between the others, resulting in a situation where it is only able to sell to the segment choosing a seller randomly for the final 20 days. Before this happens, the DF-seller is able to compete for some portion of market share. The GD-seller starts out by heavily undershooting its price, to make up for a loss of sales the first day. It then quickly raises its price again and stays rather competitive for the remaining periods. Whats interesting to note, is that as time progresses, the GD-sellers fear of not selling its entire inventory before the market closes, makes it lower its prices more aggressively towards the end of the market, and thus the GD-seller joins in at lowering overall market prices together with the SDNN-seller for the last quarter of the market.

The RBNN-seller, on the other hand, has the least volatile prices and is also the best performing seller, for the simple reason that it has the lowest average price. Because it has more hidden nodes, it can better represent the underlying data and generate a more complex curve, and we see that its price changes are more modest than for the SDNN-seller. For approximately the first half of the market it somewhat underprices its goods, losing out on some possible additional revenue, and as the market nears its end, it is drawn into the increasingly aggressive price war.

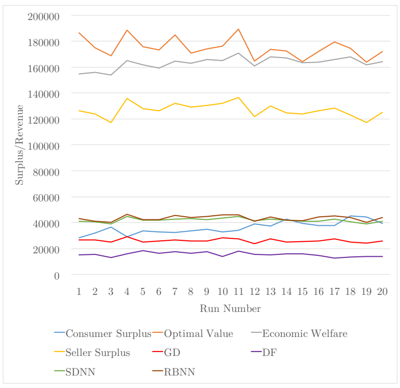
Another interesting thing to notice is the development of the current stock of inventory with each seller. We see that the DF-seller is left with 200 items in stock because of its poor pricing decisions. The inventory focused pricing algorithms, the SDNN and the RBNN, would prefer to sell an equal amount of goods each day, and as such their optimal inventory development would be a straight line from the initial inventory at day one to zero at day 100. The RBNN and SDNN seller are quite successful at generating a linear inventory development, and we see that the SDNN strategy almost reaches stock out at the end of the market. The GD-seller, on the other hand, starts out by selling rather few items per day, before increasing its sales quite dramatically around day 50. Arguably this happens as a result of the algorithms scaling ratio, enabling larger price adjustments as time progresses.

Furthermore, when evaluating the daily revenues of each seller, we see that as expected, revenues are higher at the beginning of the market due to the higher customer valuations and prices. As we reach the final 20 days of the market, almost no revenue is generated due the increasingly aggressive pricing by the sellers. From Figure 9.1e is quite easy to see the DF-seller's strategy in action. It adjusts its price upwards until no goods are sold, and then lowers its price the following period. This results in revenues commonly alternating between a positive value and zero. We further notice that the

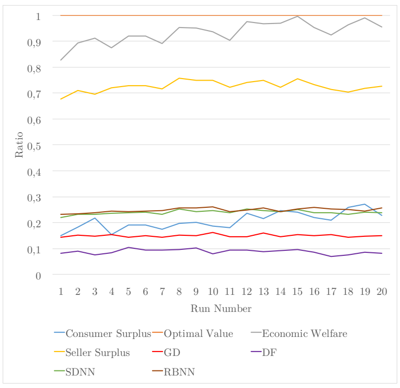
GD-seller gains a solid amount of revenue at the early days of the market, but then loses revenues to the SDNN and RBNN until day 50. After this, we see an increase in average revenues created by the GD-seller as a result of its lower prices and more sales. The SDNN-seller shares some of the same traits as the GD-seller, in that it forgoes revenues to the RBNN-seller in the second quarter of the market, but gains traction after day 50. Evidently, the reason for the RBNN-success comes from its ability to extract revenues at times when prices and valuations are high, and it earns the majority of its revenues in the first half of the market.

Moving on to the 100th run, we see that the situation is somewhat the same as in the first run. The pricing of goods get more aggressive towards the end, the SDNN-seller is still volatile, and the RBNN-seller continues to perform the best. However, the average prices of the market are in general lower than in the first run, resulting from the SDNN-seller's failure to price fairly at the beginning of the market. Presumably, this comes as a consequence of the SDNN algorithm's limited network size and few nodes. Its rapid changes in weights at the end of the previous run, has led it to start off the next run with poorly adjusted weights, causing it to perform suboptimally. The SDNN-seller quickly corrects its actions, but its early prices have consequences for the overall market and as such the average prices are brought down. We see that as a result of the lower prices, the strategic segment is now able to buy at their original arrival, and as such we see that the RBNN and SDNN-sellers' inventories drop more rapidly from the start in the final run than in the first one.

Overall, we find that the sellers, as expected, are sensitive towards each other's prices and collectively fail to seize the opportunities to increase prices. An increase in prices would require some form of collusion, but with no means of communicating with each other explicit collusion is unlikely. However, as the sellers' prices are kept well above zero, which in our model can be regarded as the marginal cost of the sellers, the dynamics of the market is far from the homogeneous goods Bertrand competition model. In such a Bertrand competition model, the competitive equilibrium would be to price the goods at marginal cost. In fact, what we are experiencing is a form of implicit collusion. The sellers are unable to raise average market prices when first lowered, but they are still able to keep prices high because of their rather modest price changes. Moreover, our model presents some form of a repeated game, or supergame. Thus, the Folk theorem could explain why we see this implicit cooperation amongst sellers. According to the Folk theorem, if the product of the probability of facing the same opponent and the discount factor $\frac{1}{1+r}$ is large enough, there exist multiple subgame perfect equilibriums in which the sharing of profits can range from perfect competition to monopoly pricing (Tirole, 1988). However, we find that the use of the Folk-theorem in our case has some issues. Since our model deals with stochasticity, it is hard for the sellers to evaluate whether an decrease in sales is a consequence of lower demand, or as a result of some other seller undercutting their prices. Hence, the use of, for instance, a grim-trigger strategy, could easily be suboptimal as the sellers lack information regarding the underlying reason for why they might experience lower demand.



(a) Evolution of surplus values



(b) Evolution of surplus share

Figure 9.2: The development of surpluses and revenues with increasing number of simulation runs for the base scenario

Figure 9.2 presents how the seller's revenues, consumer surplus, and economic welfare develops over the simulation runs. It plots the value from every fifth simulation run from the first to the last. From this, we find that the best performing seller overall is the RBNN-seller, but the competing SDNN-seller follows closely. They are both able to extract a modest increase in their relative revenues as more simulation runs are performed. The adaptive GD and DF strategies' revenues, on the other hand, stay rather constant. What is interesting to note is that the economic welfare is increasing with the number of runs. This happens as a consequence of the decreasing average market prices with every simulation run. As overall prices are lowered, there is an increase in demand. To sell an optimal number of goods the sellers should price their goods at $V_{mean} - \sigma_D^2$, which in Figure 9.1a and 9.1b is 50 units below the segments valuation curve. Furthermore, when prices are above the strategic segment's valuation, they are left to wait for their return. However, this residual demand might be lost if the strategic customers arrive before prices have dropped below their valuation. We further see that the seller surplus, stays somewhat constant, but the consumer surplus is steadily rising. In regards to the seller surplus, the increase in goods sold, makes up for the lower market prices, thus keeping their surplus level. Whereas for the consumers, this has a double effect, lower prices imply more surplus per sale, and more consumers purchasing further increase their surplus.

9.3 Buyer Behavior Sensitivity

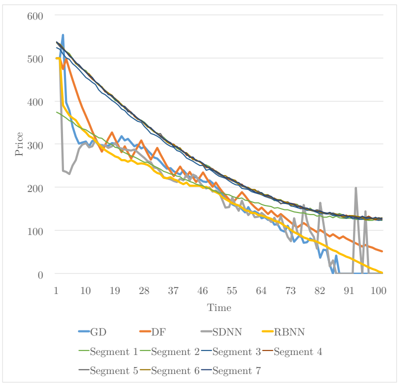
Having evaluated the balanced base scenario, our goal is now to see how the dynamics change as we adjust the different parameters. By increasing and decreasing parameters individually, all else being equal, we can gain insights regarding the given parameters impact, and use this new knowledge to see how they affect algorithm performance. We will continue keeping the initial idea of a balanced marketplace throughout the sensitivity analysis.

9.3.1 Sensitivity Towards Strategic Customer Lifetime

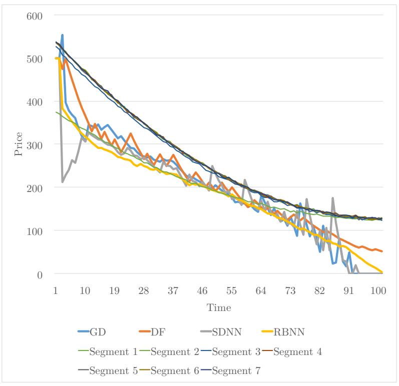
We start off by analyzing what impact the strategic customers lifetime have on the market. To do this, we have run the simulator 100 times for a short lifetime of 1 and a longer lifetime of 20. Our intuition is that as the customers lifetime decrease, the overall market prices should decrease, and likewise, if the buyers lifetime is increased, so should prices. This is a natural assumption, as when the buyers check in at fewer times, the likelihood that prices are below their valuation when re-entering is reduced. The more days we allow the customers to check prices, the more likely is it that they will eventually return on a day when their utility is above zero.

Analyzing Figure 9.3 we see that our intuition does not quite hold. If we compare Figure 9.3a to Figure 9.3b, we see that the prices, when lifetime is short, are close to the same prices as when lifetime is high. We find that the overall market prices deviate further away from the strategic segment's valuation curve at an earlier point in time, and as such the increasingly aggressive pricing initiated by the SDNN-seller and GD-seller starts earlier. To grasp this phenomenon, we need to evaluate the differences in inventory development between the two scenarios. We see that the inventory curves are in general smoother for the short lifetime scenario than the long lifetime scenario. This makes sense seeing that a short lifetime creates less residual demand, and consequently the realized demand each day is more stable. When lifetime is longer, the residual demand is naturally higher because the buyers have more days to stay active, and as such realized demand is more volatile. This effect is what's creating the more jagged inventory curves in the long lifetime scenario. Continuing this logic, we find that the volatility of the SDNN-seller and GD-seller is smaller for the short lifetime scenario, as there are fewer surprises in realized demand, caused by the residual demand.

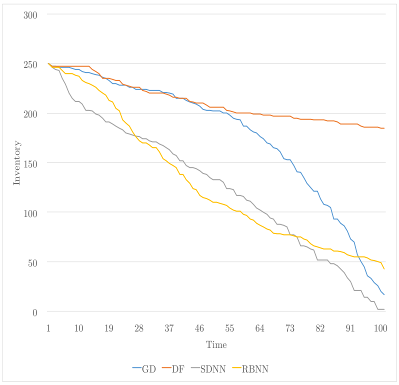
Another interesting fact to notice is the difference in surpluses and revenues for the two scenarios. We see that when customers have a short lifetime, the consumer surplus is generally about 0.1 points higher than for the longer lifetime. This also has an impact on the sellers surplus, which is slightly lower when lifetime is short. The reason behind this lies in the fact that more goods are sold in the short lifetime scenario.



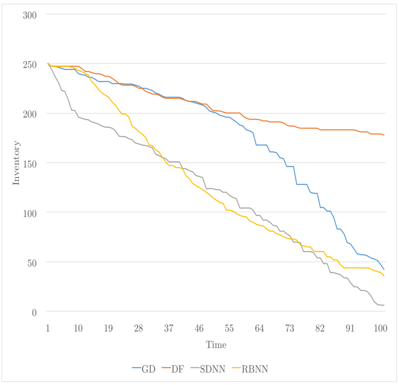
(a) Lifetime of 1, Market Prices



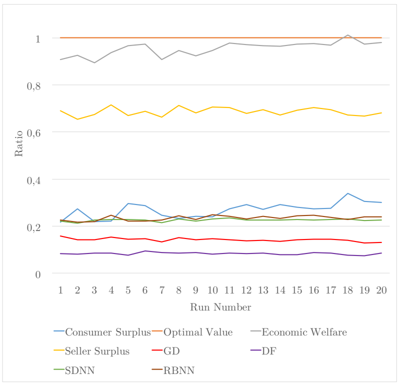
(b) Lifetime of 20, Market Prices



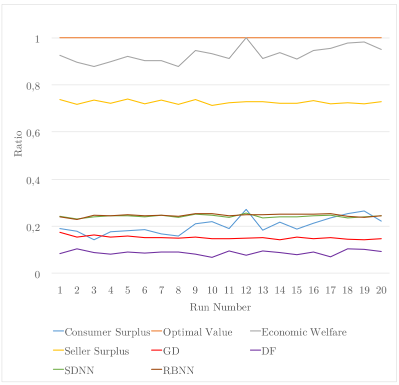
(c) Lifetime of 1, Inventory



(d) Lifetime of 20, Inventory



(e) Lifetime of 1, Surplus share



(f) Lifetime of 20, Surplus shar

Figure 9.3: Market prices, and inventory and surplus development when altering the buyer's lifetime. All graphs present results from the 100th simulator run

9.3.2 Sensitivity Towards Customer Price Variance

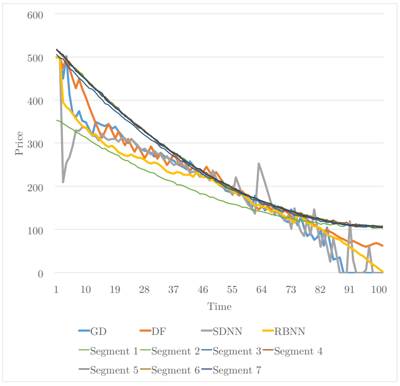
Next, we turn our focus to the customers' price variance, or the spread of valuations, that creates the quantity demand using Equation 8.2, by evaluating a low price variance of 5 and a high price variance of 50. From Figure 9.4 it is clear that the market prices is dependent on the customers price variance, as should be expected. If the sellers are to perform optimally, they should price their goods at $V_{mean} - \sigma_D^2$, and it follows that a high σ_D^2 would impose the sellers to lower their prices. Likewise, a low price variance should imply higher prices.

What is more intriguing is that in the case of low variance, we see a tendency of prices following a more concave path, as opposed to the convex path of the high variance scenario. Arguably, this is a result of the customers value ranges in the two scenarios. When the variance is low, the distance between the strategic segment's highest valuation and the lowest myopic segment's valuation is significant. This makes it harder for the sellers' pricing algorithms to reach a price low enough to serve the strategic segments immediately, as there is a "large" gap between the myopic and strategic segments willingness to pay, in which there is no additional revenue to be gained by lowering prices. Therefore, they find it more lucrative to increase prices and ignore the opportunity to experiment with lower prices. Furthermore, a low σ_D^2 also implies that it is possible to gain a higher revenue per sale, than if σ_D^2 is high. Consequently, the lack of sales is counterbalanced with a higher revenue per good, which is why we see only marginal differences in revenues between the two scenarios.

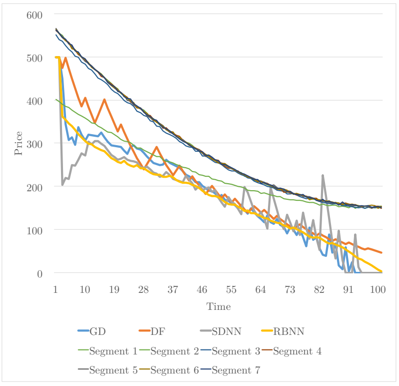
9.3.3 Sensitivity Towards Customer Daily Price Deviation

Moving on to the daily price deviation, we find that altering the standard deviation has little impact on the dynamics of the market. Overall, it is evident that when the standard deviation is small, the volatility of customers valuations decreases, and the value curves get smoother. This, in turn, means that the customers have little deviations in their evaluation of the sellers, and as such the sellers' prices get less volatile. The difference in revenues between the base scenario variation of 10 and 5 and 1 and 1, for the myopic and strategic segments respectively, is negligible.

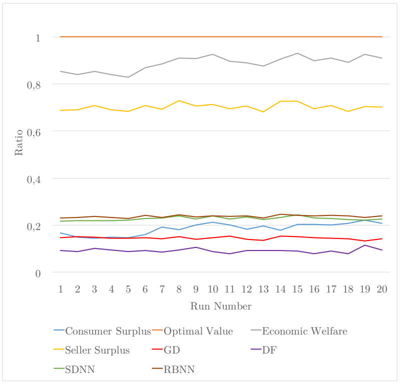
By increasing the deviation, it follows that valuations and prices increase in volatility. The tendency is that the more we increase the standard deviation, sellers revenues decrease and consumer surplus increase. This happens quite naturally because, as the spread between the segments different valuation and the day-to-day variations increases, the harder it is for the sellers to anticipate the "right" price. Consequently, the smarter algorithms, the



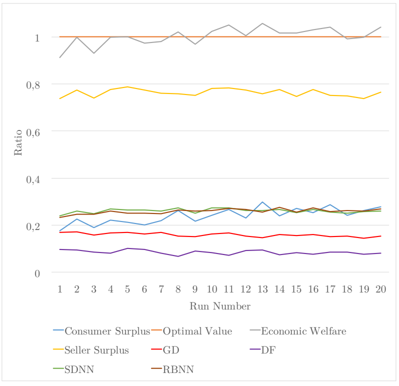
(a) Variance of 5, Market Prices



(b) Variance of 50, Market Prices



(c) Variance of 5, Surplus share



(d) Variance of 50, Surplus share

Figure 9.4: Market prices and surplus development when altering customers' price variance. All graphs present results from the 100th simulator run

SDNN-seller and RBNN-seller, and partly the GD-seller, find it better to price with an anticipation that valuations will be lower. The reason is due folded. Lower prices increase the probability of making any revenue at all, whereas higher prices decrease the probability of sale and the added surplus resulting from a possible higher revenue per sale, just can not compensate for the likelihood of no sales whatsoever. Moreover, since it is more optimal to decrease prices, this also affects the competition between sellers, and consequently, lower prices creates even lower prices because of the ongoing competition.

9.3.4 Sensitivity Towards Customer Tactics

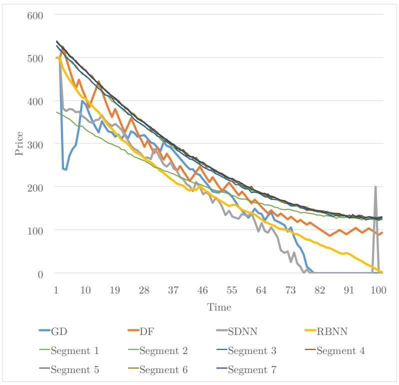
Lastly, we evaluate how the choice of customers tactics impact the market. Altering the tactics of the segments can have quite a substantial impact on profits and market dynamics, however, only making minor adjustments makes the effects harder to identify. Hence, we have chosen to alter all of the buyer tactics simultaneously to the same tactic, having all segments using either an any seller, a compare two or a bargain hunter tactic. Doing this means that we forgo some of the heterogeneity of the segments, but it allows us to make a more general analysis of what effects we experience when changing the tactics. Table 9.4 presents the average values from 100 runs of the simulator when all segments are using either AS, CT or BH tactics.

Table 9.4: Average surpluses and revenues for all simulator runs when varying buyer tactics

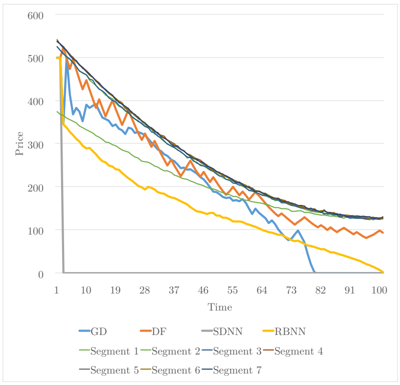
	OV	EW	CS	SS	GD	DF	SDNN	RBNN
Value (AS)	169628	141356	57303	84053	19376	17672	13261	33382
Value (CT)	169435	158881	35467	123413	24816	15756	39870	42970
Value (BH)	166785	161958	33827	128130	27363	13179	42327	45260

By examining Figure 9.5 together with Table 9.4 we see that the sellers' revenues depend on what tactic the segments are employing. If the segments all choose their seller randomly, we see that there is a dramatic decrease in seller surplus and revenues, but a substantial increase in customer surplus. This is rather counterintuitive, as one might believe that when customers pick a seller at random, we should not see any price wars in the market and as such revenues should be higher. Yet, this is only the case for the DF-seller. All seller strategies depending on desired sales of inventory perform consistently worse. In fact, the optimal pricing strategy, in this case, would be to charge as if one were in a monopoly market.

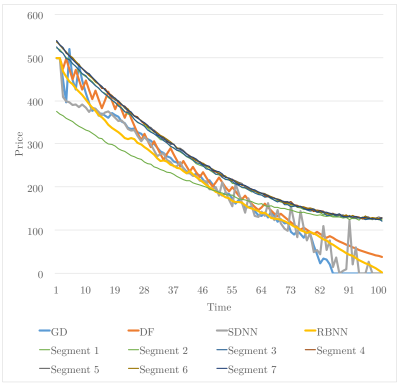
The reason for this malfunction is related to the fact that when all segments choose randomly, they will on average split the market and sell to 25% of the buyer population each. In itself, this should not be so problematic, especially



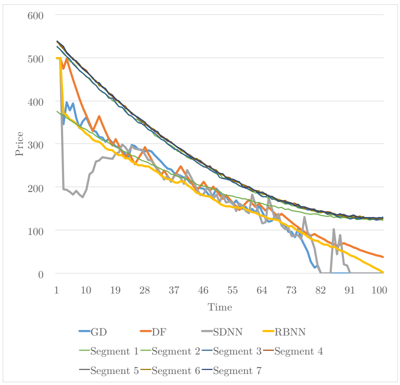
(a) 1st Run, Market Prices, AS



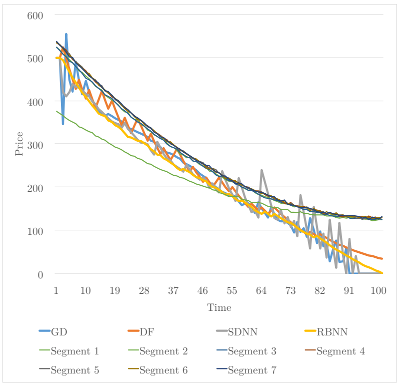
(b) 100th Run, Market Prices, AS



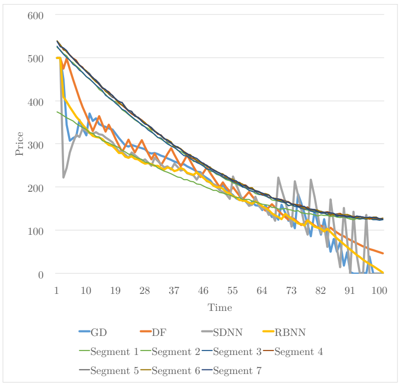
(c) 1st Run, Market Prices, CT



(d) 100th Run, Market Prices, CT



(e) 1st Run, Market Prices, BH



(f) 100th Run, Market Prices, BH

Figure 9.5: Evolution of market prices when altering the segments' tactic for choosing a seller

since the sellers all have just enough inventory to cover their implicit market share. Rather, it is the increased stochasticity of the market that causes problems for the SDNN-, RBNN- and GD-seller. The problem is that when a day of low demand occurs at anyone of these sellers, they decrease their price in the next period hoping to increase demand. However, since the demand is now given irrespective of the other sellers prices, lowering prices below the strategic segments $V_{mean} - \sigma_D^2$ will not have any effect, it will only result in the loss of revenue. It is their failure of realizing this fact; that makes them continuously decrease prices expecting to find a relationship that is not present in this market.

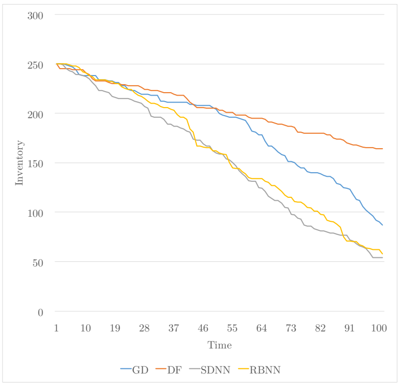
From Figure 9.5b, we see that this effect is especially obvious for the SDNN-seller, which ends up at pricing it is goods at zero or marginal cost in the 100th run, despite the fact that it is not really in competition with any of the other sellers. Arguably, this is a result of its overall strategy, limited network size and to some degree confirmation bias. The SDNN-seller starts out by pricing at a reasonable high level during the first simulator runs, as seen in Figure 9.5a, but gradually lowers prices with each run. As prices are decreased, the SDNN-seller experience an increase in demand until the price gets below the strategic segments $V_{mean} - \sigma_D^2$. Up to this point, the probability of sale is increasing, seeing that one are then able to sell directly to the strategic segment. This creates an anticipation that lowering prices lead to higher demand for the coming periods as well. This effect gets amplified during times of low demand because it then increases the number of desired sales for the next periods which in turn wrongly motivates further price reductions. We would expect the opposite to happen in periods of high demand, but the bias towards lowering prices dampens this effect.

The RBNN-seller is once again the best performing seller. However, it suffers from the same malfunction as the SDNN-seller. It fails to realize that there is a given threshold for which there is no longer any gains from lowering prices. This is evident in the overall decrease in prices between run 1 and 100, and in fact, if we let the simulator run for i.e., a 1000 times the RBNN's prices also gets close to zero. However, the RBNN-seller is more careful in its price adjustments which is why it is able to extract more revenue. This is a consequence of the larger network and offline training, which has trained the network using the initial market intelligence for 5000 iterations, and such a hundred online training examples are not enough to make it "forget" its initial values.

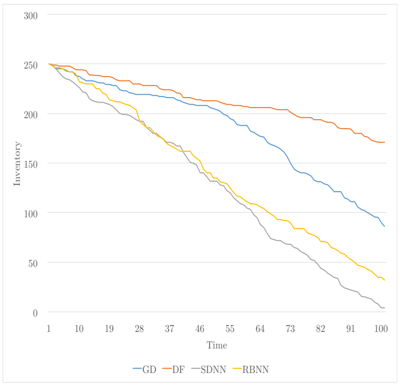
Furthermore, this random seller scenario makes it easy for us to identify one of the key weaknesses with the inventory reliant pricing algorithms. Because the algorithms' goal is not to maximize revenue, but rather sell an equal amount of inventory each day, they can be tricked into believing that they are performing well, when they are in fact performing suboptimally. If we consider the inventory development at the SDNN-seller and RBNN-seller presented in Figure 9.6, we see that they are actually, according to their own measure of success, performing close to optimal, as is evident from the almost straight line from their initial inventory to zero in Figure 9.6b.

The GD-seller, on the other hand, is able to keep its prices up for quite some time, but as the market nears its end, the scaling ratio of the algorithm enables more drastic price changes, and we see that from approximately day 50, and onwards, prices are more heavily decreasing. Unfortunately, this scaling ratio effect is implicitly delayed for too long, and despite its price of zero for the last 20 days, it is unable to reach its goal. Nevertheless, the GD-seller helps us emphasize another interesting fact. For the inventory dependent algorithms to reach their goal of selling an equal amount of products each day, their price should be below the valuations of the strategic segment. Looking at Figure 9.6, we see that it is only when the GD-seller prices its goods below the strategic segments valuation, that it is able to achieve the desired inventory development slope. As has become the trend, the DF-seller is performing the worst. Its fault lies once again in its pursuit of selling to the highest valuing customers, and consequently, demand is lost.

From Figure 9.5 we see that the differences between the CT and BH segment scenarios are rather small. There are some differences, in that the volatility of the SDNN and GD prices are a bit higher in the BH scenario than in the CT scenario. When the buyers only compare two sellers, some sellers are bound to be left out of the consideration. Because sellers have no means of determining whether the decrease in demand stems from more competitive pricing or just the stochasticity of the market, they wrongly always assume that a rival is undercutting their price, driving overall prices down.



(a) 1st Run, Inventory, AS



(b) 100th Run, Inventory, AS

Figure 9.6: Development of the sellers' inventory when all segments use the Any Seller tactic

9.4 Seller Behavior Sensitivity

Having evaluated how different buyer behavior parameters alters the market, we now turn to altering the behavior of the sellers. Note that this section will not include any changing of seller strategies or the number of sellers, variation of these parameters will be done later.

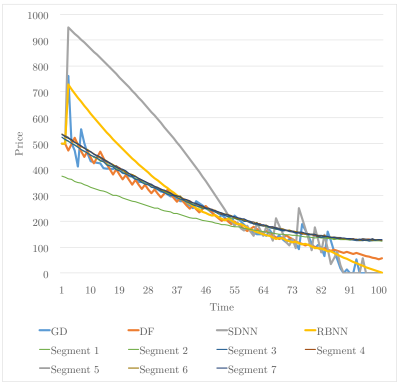
9.4.1 Sensitivity Towards Seller Inventory

We start off by looking at how the market changes when there is excess supply (ES) or excess demand (ED), by changing the number of available inventory at each seller. To evaluate this, we run two scenarios, one in which the sellers each have 150 goods to sell, and one where they have 350 goods to sell. Keeping the number of buyers constant, we are then faced with a shortage of 400 goods for the excess demand scenario, and an abundance of 400 goods for the excess supply case. Further, we have chosen not to alter the offline training data for the SDNN- and RBNN-seller, enabling us to evaluate how they adapt when the market is somewhat different than they initially expected.

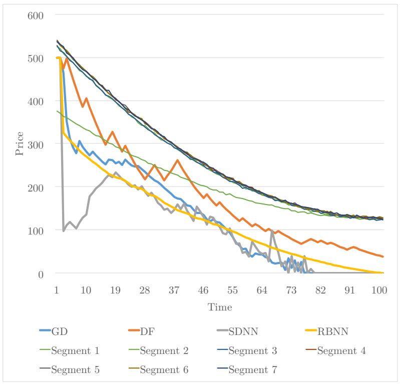
Table 9.5: Average surplus and revenues for all simulator runs when changing sellers' initial inventory

	OV	EW	CS	SS	GD	DF	SDNN	RBNN
Value (ED)	172434	97809	15499	82310	20089	20718	16989	24512
Value (ES)	170209	175218	111844	63374	11042	8690	21578	22063

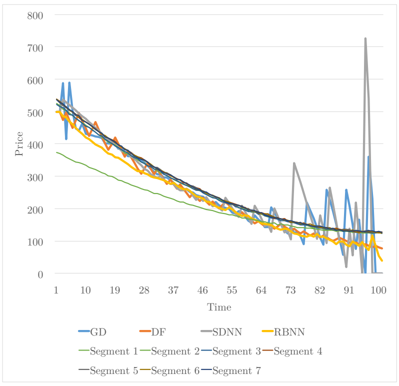
Figure 9.7 presents the development of prices and surpluses for the two scenarios. We begin by considering the scenario of excess demand, presented by the graphs on the left-hand side. Comparing the evolution of market prices from Figure 9.7a and Figure 9.7c we see that the SDNN- and RBNN-seller initially suffers from the suboptimal offline training. In the first simulator run this quite clear as they both charge a price above the highest valuing customer for the first 50 days. However, as more simulations are run, they adapt to the new market conditions and gradually increase their revenues as can be seen in Figure 9.7e. As would be natural in an excess demand market, we see that prices are generally kept at a higher level throughout the market. The SDNN-seller and GD-seller are still exaggerating their price adjustments towards the end of the market, causing more volatile prices. However, we see that this does not create the same price war tendency as in the base scenario, predominantly because they now tend to overprice their goods rather than underprice, as a consequence of having less inventory available than expected.



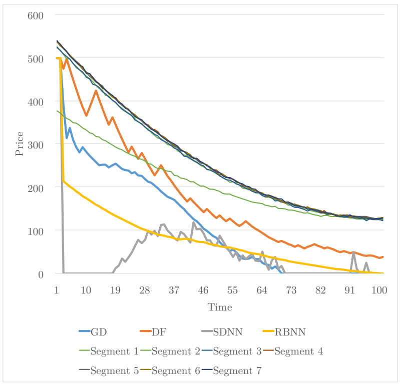
(a) 1st Run, Market Prices, ED



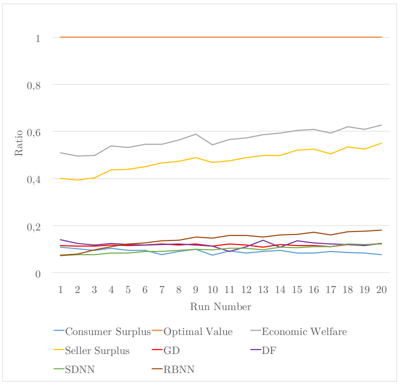
(b) 1st Run, Market Prices, ES



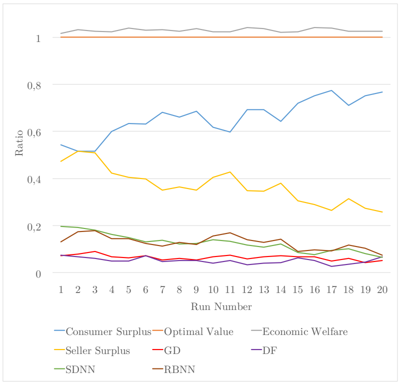
(c) 100th Run, Market Prices, ED



(d) 100th Run, Market Prices, ES



(e) Surplus share, ED



(f) Surplus share, ES

Figure 9.7: Evolution of market prices and surplus when we vary the sellers' initial inventory

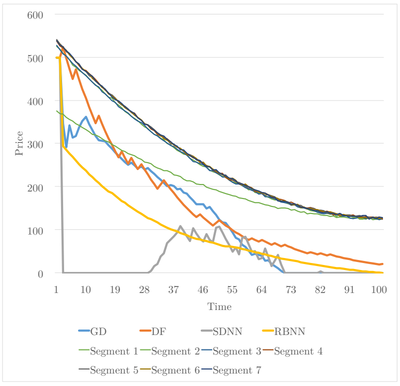
The GD-seller and DF-seller on the other hand, are rather stable, and perform well throughout the simulator runs, and they are in fact beating the SDNN-seller in terms of revenue generated. The RBNN-seller is able to double its share of the optimal value during the 100 runs, and consequently is a much better learner than the SDNN-seller, that is only able to increase its revenue share by 50%.

Turning to the scenario of excess supply, we see that as expected, prices are now generally lower. As the number of simulation runs increase, it is evident that the SDNN- and RBNN-seller lowers their price even more, and the SDNN-seller is, in fact, selling at marginal cost for about half the number of days. Consequently, they are both suffering from a loss of revenues with every simulation run and end up losing about half of their share of the optimal value. This happens because of the limitations of the desired sales strategies. By only focusing on selling out their inventories, they are forced to lower their prices and engage in more aggressive pricing, and as such revenues are lost to the benefit of the consumers.

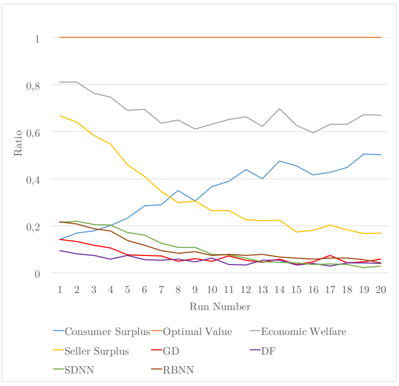
9.4.2 Sensitivity Towards Seller Capacity Rationing

The balanced scenario is initiated with a rather generous number of inventory available each day, and it is in fact, seldom restricting the sellers in any way. This implies that increasing available inventory each day does not have any noticeable effect on the market dynamics. However, if we decrease available inventory and let the sellers only sell a few goods each day, i.e. 5, the market drastically changes. Capacity rationing does, in fact, increase price wars in our model. From Figure 9.8 it is quite clear that with every simulation run the sellers forgo revenues and the seller surplus gets drastically reduced. The consumers heavily benefit from this. However, the economic welfare created is reduced.

These effects stem from the inventory dependent strategies fear of not selling their entire stock of goods. Periods of low demand increases their desired sales for the next periods, but since they are now restricted in their ability to catch up with their desire in times of high demand, their "best" choice of action is to make sure they are not stuck with more inventory than they can sell. Once again it is the SDNN-seller that has the most aggressive prices, and we find from examining Figure 9.8a that for the majority of the selling period, its price equal marginal cost. The SDNN-increase its prices around day 30, because by that time, it has sold out more than 50% of its initial inventory, and consequently, sold "too many" goods. The RBNN's slower moving network is not able to keep up with the rapidly decreasing prices of the SDNN, and as a result it loses the fight for market share. However, the RBNN generates approximately the same revenue as the SDNN, despite only selling an average of 100 goods.



(a) 100th Run, Market Prices, CR



(b) Surplus share, CR

Figure 9.8: Evolution of market prices and surplus when sellers utilize capacity rationing

9.4.3 Sensitivity Towards Seller Initial Price

The sellers' initial price affects all of the pricing strategies, and to see how we have run two scenarios, one in which the sellers all have an initial price of 299 and one where the initial price is 699. We see from Table 9.6 for the adaptive algorithms, a lower initial price comes to their benefit, whereas for the neural networks, a lower initial price is a slight disadvantage. Arguably, the DF-seller performs better when prices are low, because it is then implicitly forced to sell at a lower price at the early days of the market, and thus, it takes more time for it to reach its suboptimal goal of only selling to the highest valuing consumers. The GD-strategy uses its initial price together with the scaling ratio when calculating the price for each day, and as such, a lower initial price will often lead the GD-algorithm to generally having lower prices. Which in this setting, results in more competitive pricing and thus an increase in demand. As a result, we see that a lower initial price, actually increases both the seller surplus and the consumer surplus, thus providing more economic welfare.

Table 9.6: Average surplus and revenues for all simulator runs when altering sellers' initial price

	OV	EW	CS	SS	GD	DF	SDNN	RBNN
Value (LOW)	171298	171972	44501	127470	31105	19045	38210	39109
Value (HIGH)	169853	152235	35325	116910	21748	11649	41223	42288

9.5 Altering the Value Curves

All simulations this far have been performed using a decreasing value curve. Therefore, we next examine how these algorithms perform when faced with the increasing (INC), the mid-dipping (DIP) and the mid-peaking (PEA) value curves. Note that we alter the sequence of market intelligence data fed to the RBNN's training process, and adjust the SDNN weight according to the standard procedure. As will be evident in this section, the way in which consumer valuations change over time does not have a profound impact on a competitive market dynamics. Table 9.7 presents the average revenues and their respective ratio of the optimal value for all 100 simulation runs.

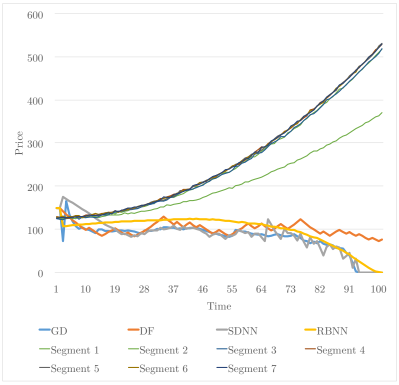
The first thing to notice is that compared to the original decreasing value curve, all the other value curves destroys value for the sellers. The sellers were able to extract approximately 70% of the optimal value when faced with a decreasing value curve, but as seen from Table 9.7 the other value curves cuts their surplus by more or less 50%. Furthermore, despite the fact that optimal value from the mid-peaking is much larger because of its higher average valuations, the sellers are unable to capitalize on this extra value. Note also that our ability to replicate strategic behavior in these scenarios is limited. In order for us to understand why we see these effects, we need to analyze the different value curves individually.

Table 9.7: Average surplus and revenues for all simulator runs when altering the segments' value curves

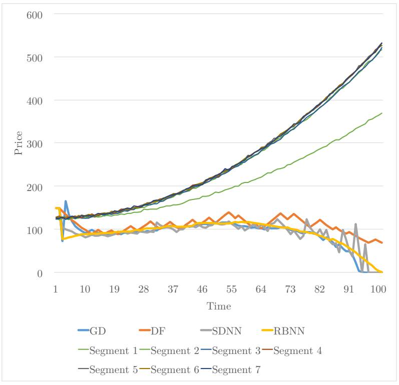
	OV	EW	CS	SS	GD	DF	SDNN	RBNN
Value (INC)	166899	170847	105957	64889	18432	11721	20832	13903
Ratio (INC)	1	-	-	0.389	0.110	0.070	0.125	0.084
Value (DIP)	171882	151602	92149	59453	8817	6442	23561	20631
Ratio (DIP)	1	-	-	0.345	0.051	0.037	0.137	0.120
Value (PEAK)	270896	279129	216336	62763	14516	11171	17585	19489
Ratio (PEAK)	1	-	-	0.231	0.053	0.041	0.065	0.072

9.5.1 Increasing Value Curve

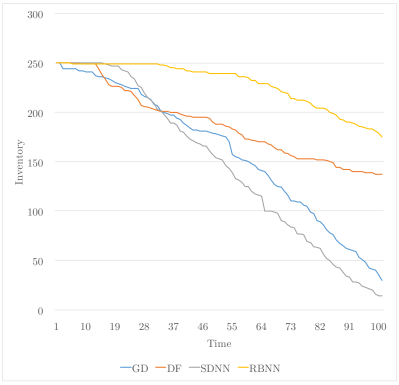
The increasing value curve can be related to products that increase in popularity over time due to marketing or market trends. To see how such a markets might affect the dynamics of pricing, we let the sellers start out with an initial price of 149 and observe the resulting effects from 100 simulations. From Figure 9.9 we can quickly identify the reason for why the sellers fail to generate a decent surplus in this market. The problem is that the competitive nature of the market makes it hard to increase overall prices, and thus, the sellers are incapable of benefiting from the segments increasing valuations.



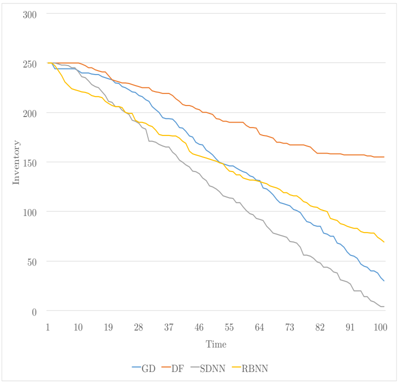
(a) 1st Run, Market Prices, INC



(b) 100th Run, Market Prices, INC



(c) 1st Run, Inventory, INC



(d) 100th Run, Inventory, INC

Figure 9.9: Evolution of market prices and inventory when the customers adhere to an increasing value curve

Since the offline training of the RBNN-seller adhere to an increasing price path, the RBNN overprices its goods for the first number of simulations. It rather quickly adapts to the market's actual dynamics of decreasing prices, but it loses out on some early revenues as seen by its large number of remaining inventory from Figure 9.9c. Because the difference between the initial price and the average market price is much smaller in this scenario than the decreasing scenario, the GD-seller and SDNN seller are now less volatile in their pricing. For the SDNN this happens because the difference between its weights at the ending of one simulation and start of the next are now smaller than previously. The GD-seller scales its initial price, and therefore, a closer starting point to the average prices, implies that prices should be more stable. We further see that the GD-seller now has a straighter inventory development curve, and thus it is able to better disperse sales throughout the selling horizon.

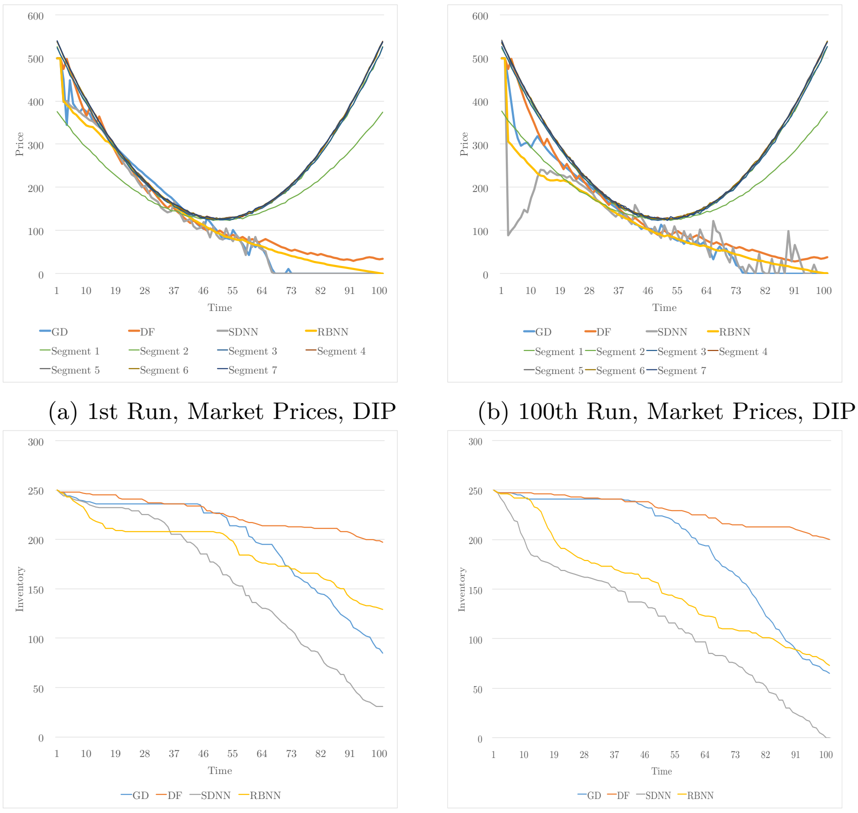
The only seller we would expect to have any benefit from an increasing value curve is the DF-seller. Because it is focused on high revenues per sale, and do not aggressively compete for market share, it should be able to approximate a rising value curve better. However, we find that the DF is also struggling with price increases, due to the stochastic nature of the random choosing seller segment. Nevertheless, it is able to keep its prices at a more constant level.

Looking at the final simulation run from Figure 9.9b, we see that for the first 50 days or so, the sellers are successful at marginally increasing average prices. After this, unfortunately, the GD- and SDNN-seller slightly undercut their prices for too many days in a row, and thus a more aggressive downward-sloping price war gets initiated.

9.5.2 Mid-dipping Value Curve

A mid-dipping valuation curve aims to represent a product that might decrease in popularity for some time, before rising to its original level. Figure 9.10 presents the market prices and inventory development for such a market, when the sellers' initial price are set to 499. Evaluating the graphs, we see that when customer valuations decrease, the sellers are capable of pricing according to the customers' willingness to pay. However, as valuations start to increase, the sellers are unable to increase their prices and only manage to reduce the slope of their overall prices.

Considering Figure 9.10a, we see that the algorithms price their goods close to the myopic segments' maximum valuation for quite some time, thus selling just a few goods at a high revenue per sale. This happens partly because the RBNN is struggling with adapting its weights to approximate the dramatic shape needed to conform to its offline training data. Hence, its initial price path is a more “shallow” bowl, than the segments valuation curves, which causes it to decrease its prices more slowly from the start. Also, the SDNN’s initial weights also adhere to a higher price path, and thus, since there is little aggressive pricing going on, prices stay close to the myopic segments’ value curves, and the sellers are not exposed to the strategic segment. Consequently, the high prices cause few goods to be sold, as is evident from Figure 9.9c. Moving on to the last simulation run, we find that prices are now generally lower, as the algorithms have adapted to the market and learned that more sales can be made if prices are lower from the start.



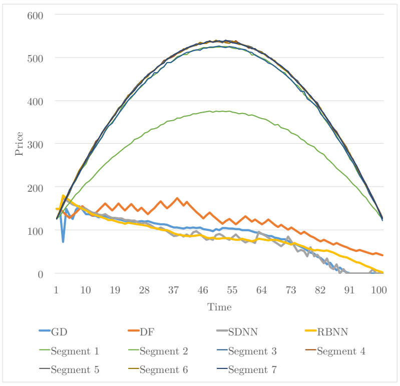
(a) 1st Run, Market Prices, DIP (b) 100th Run, Market Prices, DIP

(c) 1st Run, Inventory, DIP (d) 100th Run, Inventory, DIP

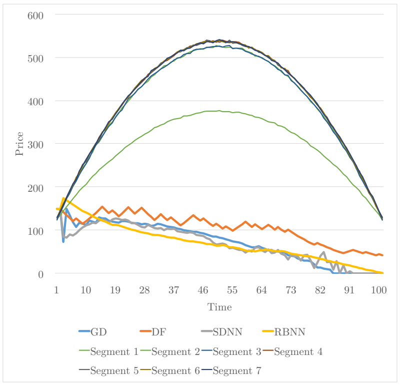
Figure 9.10: Evolution of market prices and inventory when the customers adhere to a mid-dipping value curve

9.5.3 Mid-peaking Value Curve

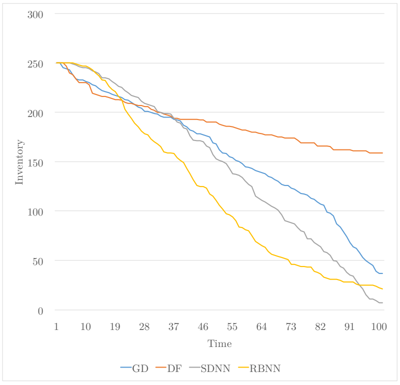
The final valuation curve we will be evaluating is the mid-peaking value curve, which relates to a product that has a peak popularity at the middle of its lifetime. Providing the sellers with an initial price of 149 we plot the dynamics of this scenario in Figure 9.11. Having evaluated both the increasing and mid-dipping value curves previously, we are easily able to predict that we will experience similar price paths in the mid-peaking scenario.



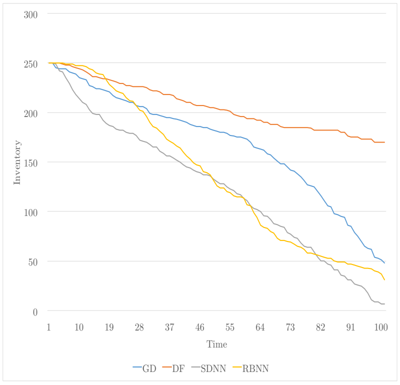
(a) 1st Run, Market Prices, PEA



(b) 100th Run, Market Prices, PEA



(c) 1st Run, Inventory, PEA



(d) 100th Run, Inventory, PEA

Figure 9.11: Evolution of market prices and inventory when the customers adhere to a mid-peaking value curve

We see that throughout the simulation runs, the RBNN-seller tries to stick with an increasing price path the first day of the market, but are forced to lower its prices to compete for any market share. Furthermore, the mid-peaking value curve drastically increase the consumers' surplus because of their higher average valuations, and the sellers low prices. As a result, the sellers are only able to secure 22% of the economic welfare in this market.

In summary, we find that it is the consumers that heavily profits from these types of value curves. Provided that the share of comparison shoppers is high and multiple sellers compete for market share, it seems clear that the sellers are bound to be stuck at competing for having the lowest price and thus destroy value for themselves.

9.6 Seller Strategies

The previous sections have given us an understanding of how the dynamics of the base scenario evolve. Next, we will alter the strategies used by the sellers to see how other seller compositions affect the overall market. We will continue using the same buyer and seller parameters as in the base scenario unless noted.

9.6.1 Introducing the Aggressive Pricing Neural Network

From our previous analysis, it is clear that for the scenarios tested, the DF-seller is the worst performing of them all. Hence, any seller using a DF-strategy in this market would be looking for a better algorithm to price its goods. So far we have not used the Aggressive Pricing Neural Network, and hence, we let the seller using the DF-algorithm switch to the APNN-strategy for the next scenario. We train the APNN network using the following normalized training data,

$$X = \begin{bmatrix} 10 \\ 50 \\ 80 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.5 \\ 0.8 \end{bmatrix}, Y = \begin{bmatrix} 450 \\ 230 \\ 120 \end{bmatrix} = \begin{bmatrix} 0.450 \\ 0.230 \\ 0.120 \end{bmatrix}$$

and set it to undercut its competitors prices by 10%. Evaluating Figure 9.12 together with Table 9.8 we see that the introduction of the APNN-seller has some dramatic implications for the market. Overall there is a significant decrease in prices, and the more runs we do, the lower the price gets. This is a natural result, as the APNN-seller constantly aims to undercut its competitors' prices and in order for them to stay competitive in the market, the all need to join in at lowering their prices. This destructive behavior greatly reduces the sellers' surplus, and as a consequence, the consumers end up with nearly 60% of the economic welfare available.

Considering the APNN-seller, we find that one of the reasons for its success in generating the highest revenue is the fact that it can sell its entire stock of inventory by approximately day 60. This means that it is capable of exploiting the consumers higher valuations at the beginning of the market, and thus, it drops out of the market before goods are sold at marginal cost. The other sellers fail to realize that the APNN-seller is in stock out and thus continue lowering prices, despite the fact that the largest promoter of the price war has left the market.

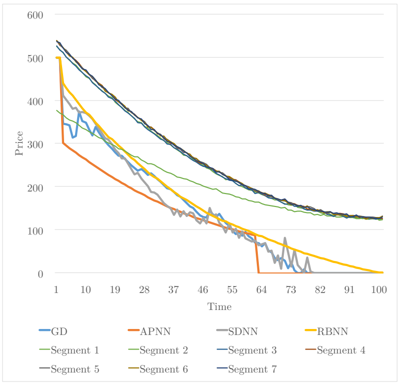
For the RBNN-seller, which in the previous scenarios has been the overall best performer, the tables have turned. Its fails at learning the new market conditions quickly enough, and as a result it is now one of the worst performing sellers. This happens because it is not able to rapidly update its weights, and thus the small and modest price movements that made it successful in the previous scenarios, are now causing it to be lagging behind the rest. Furthermore, the RBNN-seller is also left with the most inventory at the end of the market.

The situation for the SDNN-network is quite the opposite. In previous scenarios its nimble network and rapid changes, caused it to oscillate prices towards the end and initiate price wars. Now, however, its ability to swiftly alter its prices is beneficial. We see from Figure 9.12d that it is rather close to selling an equal amount of inventory each day.

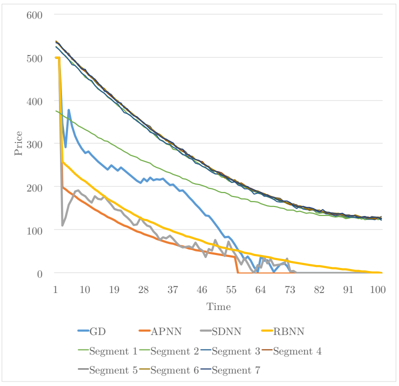
Table 9.8: Average surplus and revenues for all simulator runs when one seller utilize an APNN-algorithm

	OV	EW	CS	SS	GD	APNN	SDNN	RBNN
Value	169156	172311	102771	69539	8805	34088	17442	9204

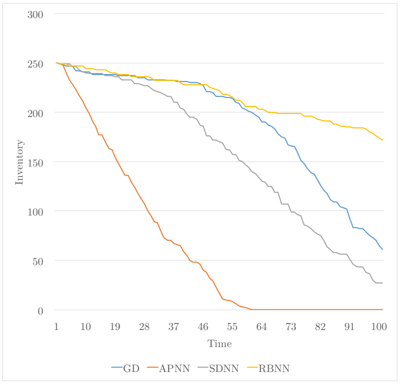
We have experimented with the APNN-algorithm following the same sensitivity analysis procedure, and for nearly all scenarios the seller using the APNN-strategy gets the vast majority of the sellers' surplus. However, overall there is a substantial decrease in the total seller surplus generated, but the economic welfare gets increased as more customers are able to purchase. Nevertheless, despite its success in these scenarios, the APNN-strategy is not without its flaws. First of all, the APNN-strategy is relying on the intelligence of other sellers, and as a result, it is only as smart as its competitors. For instance, if we were to let every seller use this algorithm, prices at marginal cost would be inevitable. Secondly, if we increase the number of segments choosing their seller randomly, the APNN-strategy performance decrease rapidly. We have previously touched upon the issue that all strategies perform rather poorly in this setting, as they fail to realize that they are no longer in competition, and because of its aggressive nature, this heavily affects the APNN's revenues.



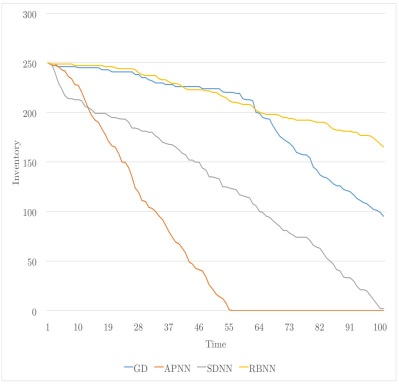
(a) 1st Run, Market Prices



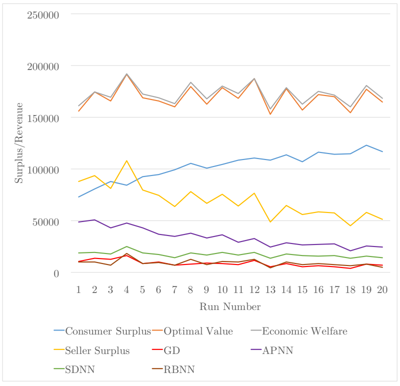
(b) 100th Run, Market Prices



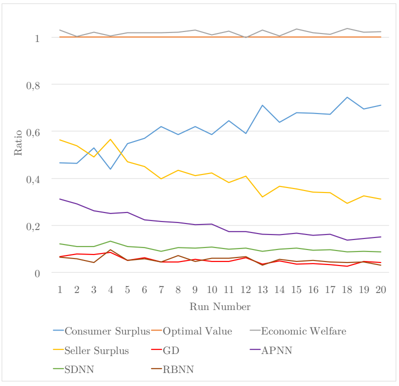
(c) 1st Run, Inventory



(d) 100th Run, Inventory



(e) Surplus values



(f) Surplus share

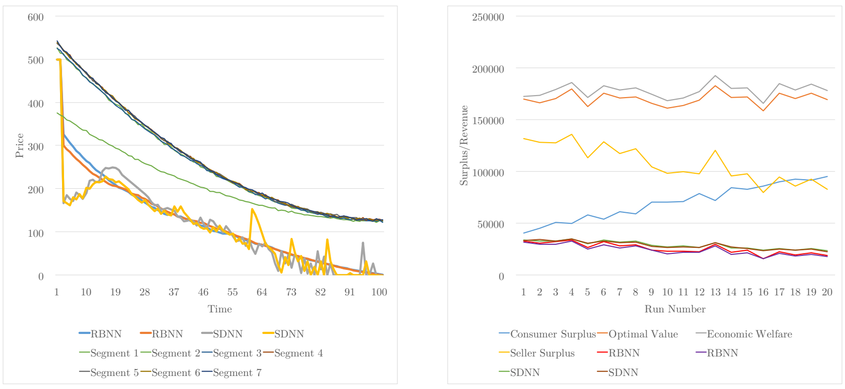
Figure 9.12: Development of market prices, inventory and surpluses when one seller utilize an APNN-algorithm

9.6.2 Mixing up the Strategies

Next, we turn to see how the different compositions of algorithms impact the market. Ignoring the Q-learning algorithm, we have five different algorithms, which implies that there are 120 possible unique seller compositions. Naturally, we will not be able to analyze them all, and besides, our previous analysis has made us able to make some generalizing notions regarding how the algorithms will perform, without analyzing all possible scenarios.

2 x SDNN & 2 x RBNN

Our first new seller composition consists of two sellers using an SDNN-strategy and two sellers using the RBNN-strategy. Figure 9.13 presents the market prices from the final simulation run and the development of surplus values. We see that as the number of runs is increased, the sellers' prices are lowered, and their revenues are steadily decreasing. If we repeat this market for more than 100 simulations, we eventually reach prices at marginal cost. Despite the differences in the smoothness of the price curves, the sellers generate a close to equal amount of revenue. Perhaps more interesting is the fact that the SDNN-sellers on average sells 100 goods each more than the RBNN-sellers, which in turn means that the RBNN-sellers can extract a higher revenue per good sold, at the cost of losing out on sales.

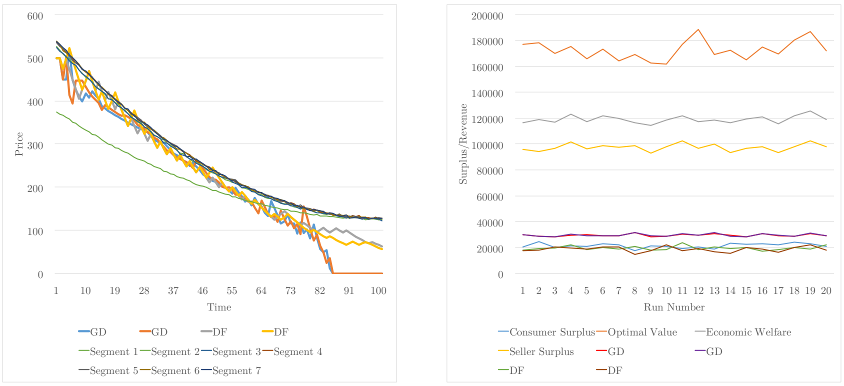


(a) 100th Run, Market Prices (b) Surplus value

Figure 9.13: Evolution of market prices and surplus when two SDNN-algorithms and two RBNN-algorithms compete

2 x GD & 2 x DF

In all of the previous scenarios, there has been one or more machine learning algorithm present. If we are to evaluate the implications of these algorithms, it would be interesting to see how the market would play out if the sellers only used the simpler adaptive strategies. Therefore, we have evaluated the use of two GD-strategies and two DF-strategies in the base scenario, to see what happens in such a setting. From Figure 9.14 is quite clear that the market is more stable when there are no learning algorithms present, as we are effectively repeating the same market over and over. We see that overall prices are generally higher and that the adaptive strategies are rather successful at mimicking the value curves of the myopic segments. However, we still get the same tendency as with the SDNN-strategy towards the end of the market. The GD-sellers' scaling factor and more sensitive desired sales, causes them to decrease their prices dramatically. The random seller choosing segments keep the DF-sellers out of the increased price war, but at the cost of lost sales. In fact, up until day 60 or so, they have all sold about the same number of goods, but for the remaining 40 days, the GD-sellers are each able to sell close to 90 items more than the DF-sellers.



(a) 100th Run, Market Prices (b) Surplus value

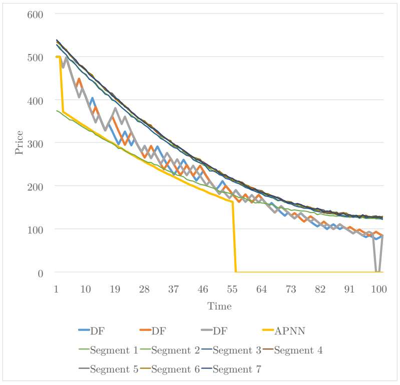
Figure 9.14: Evolution of market prices and surplus when two DF-algorithms and two GD-algorithms compete

Furthermore, we see that in comparison with our original seller composition presented in Figure 9.2, the total economic welfare is much lower, averaging about 40000 units below. If we consider the surplus generated by the sellers, we find that this is partly due the sellers surplus being approximately 25000 units below and partly due to the reduction in consumer surplus. Regardless of their high prices, the GD-seller and DF-seller are not able to extract the same value from the market as in the base seller scenario. The reason being that their higher prices have a negative impact on demand, and also the strategic segments willingness to purchase. This points us towards an understanding that the introduction of machine learning algorithms can increase the efficiency of our simulated market, and might, in fact, increase both the sellers and consumer surpluses.

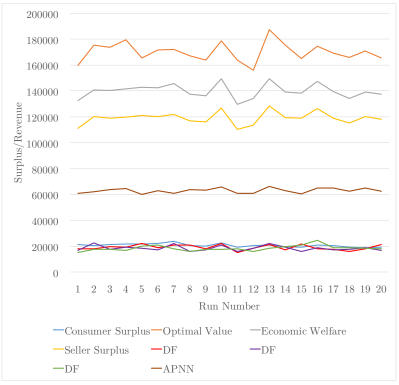
3 x DF & 1 x APNN

Finally, we put the APNN-strategy up against three DF-sellers in our base scenario. In previous literature, i.e. Dimicco et al. (2003), researchers have found the DF-strategy to induce price wars between sellers when the share of comparison shoppers is high. Therefore, it would be interesting to see what happens when three DF-sellers are faced with an, even more, price war inducing strategy, namely the APNN. In contrast, when evaluating Figure 9.15, we find that this seller composition is actually quite a stable market, with overall high prices. This happens because the DF-strategies' consistently fail to recognize that more goods can sold if prices are lowered when the market does not consist of 100% bargain hunter customers. Thus, the APNN-seller is able to undercut them all, and sell out its inventory by day 55, claiming close to half of the total seller surplus.

This scenario also lets us see the limitations of the APNN's network size. For the first ten days of the market the APNN-sellers is unable to only marginally undercut the DF-sellers prices, and thus loses some possible additional revenue. This is a consequence of having a small network, because this implicates a less complex curve, and consequently there is a trade-off of between matching the DF-sellers prices early in the market, or later.



(a) 100th Run, Market Prices



(b) Surplus value

Figure 9.15: Evolution of market prices and surplus when three DF-algorithms and one APNN-algorithm compete

9.7 Introducing the Q-learning Algorithm

Until now, we have ignored the Q-learning pricing algorithm, due to its dramatic impact on run time. One of the issues with the Q-learning algorithm implemented in our model is that, because it uses a look-up table, it is heavily suffering from the curse of dimensionality. To exemplify this, our base scenario with 250 units of initial inventory and 100 time periods, creates a state space of 25000 unique states. If we then allow the actions, or prices, to be defined between 500 and 0, with an interval of one, we are left with a $Q(s, a)$ table containing 12.500.000 instances. This is a huge inconvenience, seeing that we have no transition function, and thus cannot provide any good initial $Q(s, a)$ values by offline calculation. The lack of a transition function implies that if we are to get a complete policy and any decent performance from the Q-learning algorithm, we should visit each state-action pair at least once. This would require us to run some 20 million simulations, which would take us days to perform.

However, quite a lot of the states are unreachable, for instance, it is impossible to be in a state of zero inventory at time step 1, and as such there are many states that can never be visited in the base scenario. In addition, the effect of unreachable states is largest at the beginning of the market, leading us to see more optimal prices at the early days of the market, than the last. This happens because the number of reachable inventory states increases in time due to the buyers' increasing aggregated arrivals and demand. Nevertheless, we should decrease the size of the problem, and the easiest way to do this is by reducing the number of periods. Hence, all simulations in this section will be performed by using a total number of periods of 30, but we keep all else equal unless explicitly noted. Furthermore set the values of $\alpha = 0.1$ and $\gamma = 0.9$ for the QL-algorithm in all subsequent scenarios.

In all scenarios, we let the QL-algorithm search the state-action space by having it perform random actions, and store the $\hat{Q}(s, a)$ value from each simulation run. When we reach the final 100 simulation runs, we update the policy of the QL-seller using the learned $\hat{Q}(s, a)$ values to find the actions it believes to be the best, and let it choose actions according to its policy and not randomly. Consequently, all data presented from the QL-algorithm stems from the final 100 runs, meaning that we treat the previous runs up to this point kind of like offline training.

The increased run time of simulations has lead us to focus on just a few market scenarios. We start off by evaluating how the Q-learning algorithm performs in a monopolistic market compared to the other pricing algorithms. Then we introduce competition and see what happens when we introduce multiple QL-sellers in the same market before we analyze how the QL-strategy performs when opposed with the GD-, DF-, and RBNN-strategy.

9.7.1 Monopoly Performance

The basis for the monopoly market is the base scenario previously presented. However, since we only have one seller, we provide that seller with 1000 goods of initial inventory. Furthermore, we adjust the available inventory each day to be 100 and the strategic segments' lifetime to be three days to suit the reduced number of periods. Also, since there is only once seller, all segments are now effectively bargain hunters. We run the simulator for 10 million iterations for the QL-strategy, but keep our original run number of a 100 for the others. Table 9.9 presents the average revenue and surpluses created by the different seller strategies in a monopoly market.

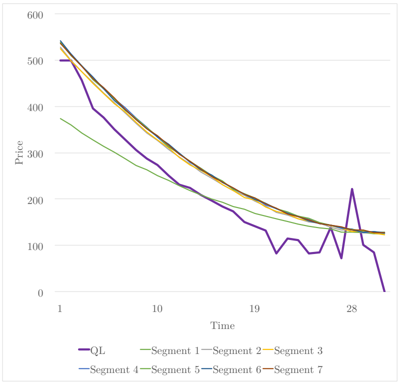
Table 9.9: Average revenues and surpluses all simulator runs when the sellers operate in a monopoly market

	GD	DF	SDNN	RBNN	QL
Revenue	103127	28364	23366	155113	154217
Share of Optimal Value	0.469	0.130	0.105	0.713	0.703
Optimal Value	219201	217662	220904	217315	218082
Consumer Surplus	39392	3432	181830	46551	29481
Economic Welfare	142519	15898	205179	201664	183699

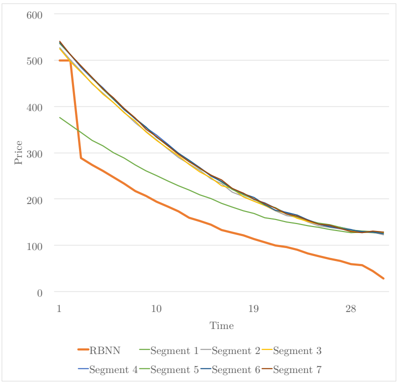
From Table 9.9 we can infer that the QL-seller and RBNN-seller, by far, are the ones able to extract the largest revenue from the market. The GD-strategy perform rather well, but we see that the DF-seller and SDNN-seller fail to generate revenue in this monopoly market. The SDNN-seller rapidly decreases its average prices with every simulation run and prices its goods at marginal cost by simulation-run 10. Consequently, the SDNN sells out its entire inventory but fails to realize that it could price its goods higher. However, the SDNN produces the largest economic welfare and consumer surplus. The DF overprices its goods and is only able to sell an average of 10% of its inventory, which drastically decrease economic welfare. Note that due to the shorter selling period; we cannot directly compare the optimal values and revenues from these simulations to those of the original selling period of 100 days.

If we compare the market prices of the final simulation run for the RBNN-seller and the QL-seller from Figure 9.16, we see that despite their similar revenues, the market prices of these two algorithms are quite different. The RBNN-seller sells more of its inventory at a lower price, whereas the QL-seller sells less inventory at a higher price. Furthermore, considering the last 25 days of the market, the QL-sellers prices are more volatile, because of the larger number of reachable states and the subsequent "small" number of iterations, prevents the algorithm from visiting every possible state-action pair often enough. As we get closer to the market's end, the number of reachable states increases, and thus more iterations are needed. Moreover, we see that at day 28, the QL-seller raises its prices above that of the myopic segments, because it has previously gained more revenue from selling to some realized residual demand caused by the strategic segment.

Overall, we find that the Q-learning algorithm can perform well in this monopoly market. In fact, because the monopoly market satisfies the notion of stationarity and the Markov property, it will eventually converge to the optimal policy if we let it visit every state-action pair infinitely often. The policy we derive using only 10 million iterations is not optimal, but still provide us with adequate evidence that there is a lot of potential in using Q-learning for pricing goods.



(a) 100th Run, Market Prices, QL



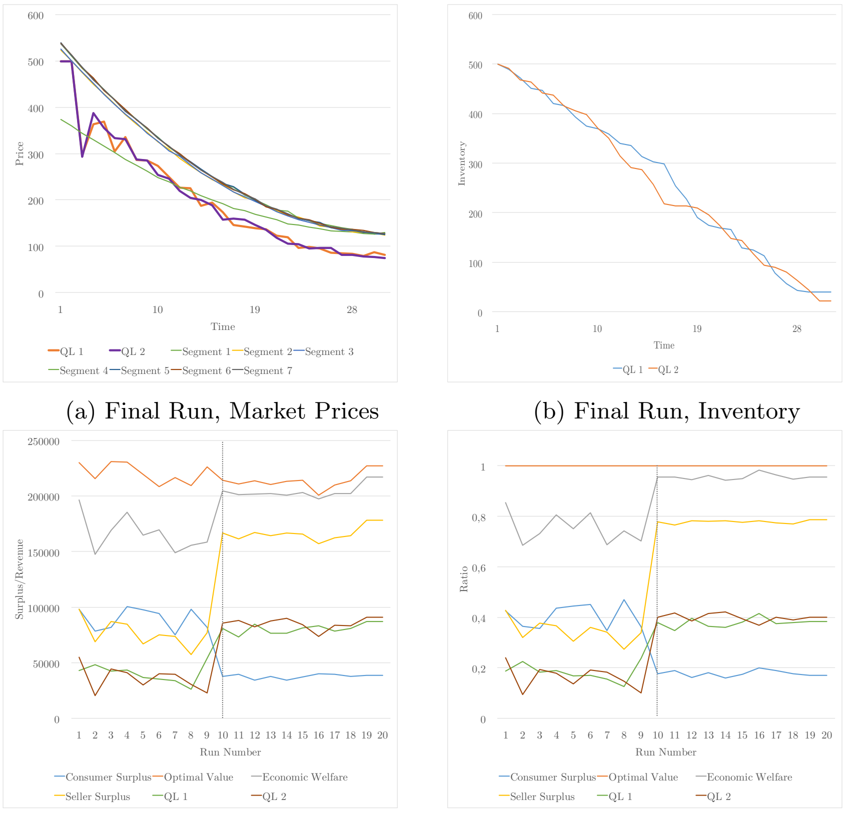
(b) 100th Run, Market Prices, RBNN

Figure 9.16: Development of monopoly market prices for the QL- and RBNN-algorithm

9.7.2 Multiple Q-learning Sellers

Next, we move away from the notion of stationarity and introduce a duopolistic market with two sellers using the Q-learning algorithm. Having multiple Q-learning algorithms compete is in general an unknown territory, as we have no theory that can ensure any convergence towards an optimal solution. Therefore, it is interesting to see how the algorithms adapt to each other, and how much revenue they can extract. To initialize the market, we let each seller get an initial inventory of 500 goods, and adjust the available goods for sale each day to be 50. We then run the simulator for 10 million iterations. The final run in which the QL-sellers use the learned policy is graphed in Figure 9.17

The Figures 9.18e and 9.18f first nine values represents the overall average values from the nine first million simulation runs. The 10th value, plots the average for the last 100 simulations, and the remaining ten values plot



(a) Final Run, Market Prices

(b) Final Run, Inventory

(c) Evolution of market prices and surplus

(d) Evolution of market prices and surplus

Figure 9.17: Evolution of market prices, inventory and surplus when two QL-algorithms compete in a duopoly

the values from every tenth simulation run for the last 100 simulations. From these figures, we can easily spot the increase in sellers' revenues, and resulting decrease in consumer surplus when the QL-algorithms starts pricing their goods according to their learned policy. Also, we find that the total economic welfare approximately increase by 20% as a result of this.

It is clear that despite the non-stationary environment, the algorithms perform great. In fact, we see from Table 9.10, that the two QL-sellers can extract a larger share of the optimal value, close to 80%, as opposed to the single QL-seller in the monopoly market which was only able to extract 70% of the optimal value. Furthermore, the two QL-sellers closely follow each other's prices, and as opposed to the neural networks, they do not engage in any aggressive pricing. Instead, they alternate between having the lowest and highest price, effectively sharing the market by performing implicit cooperation, implying that they are capable of properly evaluating the consequences of their actions. Thus, we can also assume, that if two QL-sellers operate in a duopoly market where consumers' willingness to pay increase in time; they should be able to capitalize on this increase given enough training.

The fact that the duopoly QL-sellers extract a higher surplus than the monopoly QL-seller, leads us to an interesting find. When in competition, the QL-sellers have a smaller state-space because they do not have to serve the entire market themselves. Consequently, it takes less training to converge to a solid price policy. The monopoly QL-seller should be able to generate the same seller surplus, however, because its state-space is twice as large, this would require an estimated 20 million training iterations. Also, when the market consists of strategic and myopic segments, the optimal price policy might be closer to the strategic segments' valuations, depending on the value dispersion between the myopic and strategic consumers. Comparing Figure 9.16a and 9.17a, we see that the reason for the duopoly QL-sellers' success lies in the fact that they capitalize on the strategic segments higher valuations early in the market, in addition to having a more efficient price policy towards the end of the market.

Table 9.10: Average surplus and revenues for the final 100 simulator runs when the two QL-algorithms compete in a duoploy

	OV	EW	CS	SS	QL 1	QL 2
Value	213991	204330	37454	166875	80897	85978
Share of Optimal Value	1	-	-	0.779	0.378	0.401

9.7.3 Neural Network vs. Q-learning

Our final run with the Q-learning algorithm includes putting it to the test against the RBNN-, GD- and DF-algorithms, allowing us to get an understanding of how they compare to each other. We run the simulator for 10 million iterations, and provide all sellers with the same parameters as in the original base scenario, except the available inventory each day which are adjusted to 70 to account for the shorter selling period. Because it is necessary to run the simulation millions of times to get decent results from the Q-learning algorithms, we chose competing algorithms that do not drive prices towards marginal cost. Consequently, we find that there is little knowledge to be gained from putting the QL-algorithm up against the more aggressive APNN and "unstable" SDNN, since prices would then be drawn to marginal costs long before the first million simulations. The QL-strategy would, of course, learn to price its goods at zero in time, but we find that it is more interesting to see what happens when aggressive price wars are not an issue because then we can explore more exciting price policies.

Table 9.11 presents the average values from the last 100 simulations, implying that we have then performed 9.999.900 iterations before this, in which the QL-seller has had the opportunity to estimate the $Q(s,a)$ values by random action selection. It seems clear that the QL-seller can infer a good policy from the simulations, despite the non-stationary environment. Furthermore, the QL-seller is the best performing algorithm and claims 40% of the total seller surplus.

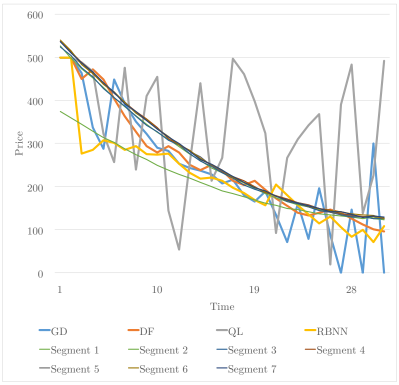
Table 9.11: Average surplus and revenues for the final 100 simulator runs when the GD, DF, QL and RBNN algorithm compete

	OV	EW	CS	SS	GD	DF	QL	RBNN
Value	214879	182000	43656	138343	26735	9532	56564	45511
Ratio	1	-	-	0.643	0.124	0.044	0.263	0.212

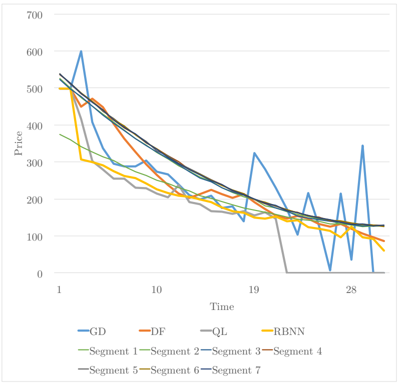
Figure 9.18 presents the development of market prices, inventory, and surpluses for these simulations. The Figures 9.18e and 9.18f first nine values represents the overall average values from the nine first million simulation runs. The 10th value, plots the average for the last 100 simulations, and the remaining ten values plot the values from every tenth simulation run for the last 100 simulations. By examining Figure 9.18b and 9.18d we see that the reason for the QL-sellers success is the fact that it can slightly undercut its competitors' prices, and since it has no desire to restrict its sales, it sells out its entire inventory by day 20.

The GD-seller's volatility of prices dramatically increases in the last 12 days, which is caused by its low price in day 18 and consequently the selling of 50 goods in one single day. Arguably this is an abnormally large demand, caused by the realization of some residual demand stemming from the strategic segment. The sudden large reduction of inventory, causes the GD-seller to heavily increase its prices, and because prices are kept too high for too long, it then continues to alternate between underpricing and overpricing its goods.

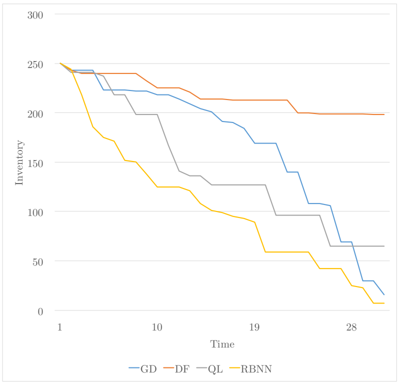
Figure 9.18a and 9.18c presents what happens in the very first simulation. From this, we can see the QL-seller's learning actions, and rather surprisingly, we find that despite choosing price randomly, the QL-seller is able to almost sell out its entire inventory. Looking at Figure 9.18e, we find that just setting the price randomly in this market scenario is a better strategy than using the DF or GD algorithms. Furthermore, it is evident that when the QL-seller starts acting according to its learned policy, we see an increase in seller surplus and decrease in consumer surplus. However, the reduction of revenues with the GD, DF, and RBNN seller is rather modest, implying that when the QL-seller prices according to its policy, it does not destroy value for its competitors, rather, it extracts its additional revenue from the consumers.



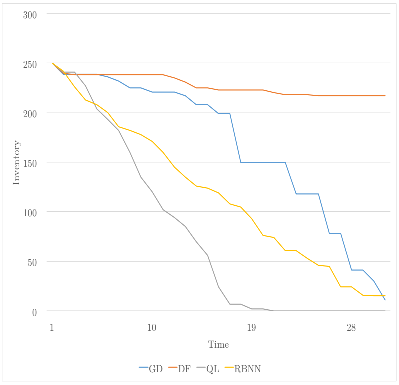
(a) 1st Run, Market Prices



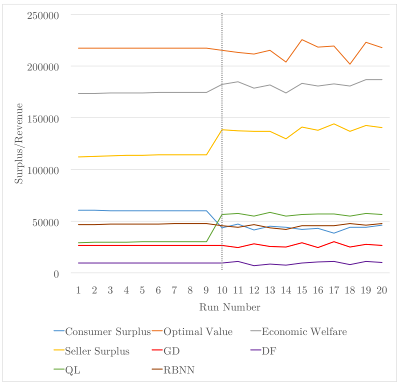
(b) 100th Run, Market Prices



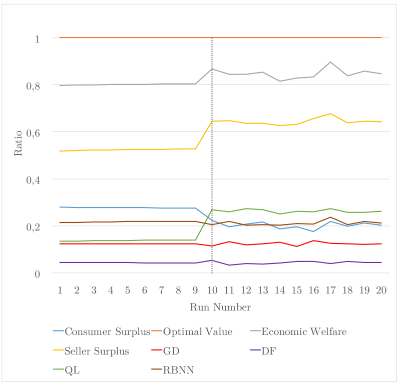
(c) 1st Run, Inventory



(d) 100th Run, Inventory



(e) Surplus values



(f) Surplus share

Figure 9.18: Evolution of market prices, inventory and surpluses when the GD, DF, QL and RBNN algorithm compete

The RBNN-seller is still able to generate substantial revenue, despite being superseded by the trained QL-seller. According to its own measure of success, the RBNN-seller is performing close to optimal in this market, indicated by its straight line from its initial inventory to zero. We further see from Figure 9.18a that the large spatial price dispersion caused by the QL-seller "learning the market dynamics", results in more volatile prices than normally for the RBNN-seller. However, when the QL-seller starts acting according to its learned policy, the RBNN's price path smoothens. Also, if we had averaged the values from all 10 million simulations, the RBNN would be the overall best performer because of its offline learning ability. Hence, evaluating which machine learning algorithm performs best depends on what assumptions and measures of success we utilize.

Chapter 10

Insights

The coming chapter will be used to derive insights regarding the simulations and strategy analysis performed in the previous chapter, and generalize our findings. Our model presents a novel approach, and as such the possibilities of relating our discoveries to other literature is limited. However, we find that there are some commonalities between our findings and that of others, and where appropriate we will discuss how our discoveries relate to more general literature. Furthermore, we use this chapter to evaluate the real world applicability of the algorithms presented and aim for a better understanding of how the assumptions in our model compare to actual markets. The chapter ends with a discussion regarding some of the weaknesses with our approach.

10.1 Algorithm Performance

Overall, we find that all the pricing algorithms implemented are able to generate revenues and price their goods fairly well. Despite utilizing no, or very limited, information, they are adaptable to various market conditions, and do not require any knowledge of the true buyer demand, the number of other sellers or agents, their competitors' current inventory or prices (except the APNN). Evaluating which of the algorithms are the overall best performer is not easy, as they all have their own strengths and weaknesses.

In general, it seems clear that in terms of revenue generated, the adaptive algorithms are the worst performers. Their inability to learn market parameters and their rather simple price tactics, makes them easily outperformed by the more sophisticated machine learning algorithms. Nevertheless, these adaptive algorithms are surprisingly robust. They require no lengthy training period, nor information to price their goods appropriately, and although better pricing algorithms will ultimately grab a larger market share, they still have their use-cases.

Utilizing artificial intelligence by machine learning is arguably a much more solid approach for pricing goods in complex markets. We have found substantial evidence that machine learning algorithms can extract higher revenues than the adaptive algorithms, as they can learn and adapt to a dynamic market. Despite the fact that the machine learning algorithms are not strategic in their evaluation, they succeed to some extent at implicitly coordinating prices. A similar discovery has been found by Kutschinski et al. (2003). Our approach is the first one, to our knowledge at least, that compares a Q-learning algorithm to neural networks, and also let them interact in the same market. We find no major technical issues of having multiple competing algorithms in a market, apart from the variation in offline training time, and consequently our results prove that QL and NNs can interact and adapt to each other without any special precautions.

10.1.1 Derivative-Following

Throughout our analysis, the DF-algorithm has consistently been the worst performer when faced with competition in a heterogeneous market. The problem with the DF-strategy in this setting is that it is simply not suited for a market where buyers differ in their valuations of a product. By always aiming to sell to the customers that value the good the most, it forgoes generating more sales and rather focuses on keeping revenue per good high. The DF algorithm can easily be related to a form of greedy heuristic, in that it tries to greedily climb the value curve of consumers. In a way, the DF-algorithm can be said to be a price skimming price tactic, by cherry picking the less price sensitive consumers.

Price skimming is a commonly known pricing strategy, often utilized for only a limited duration of time, to recover the majority of investments needed to build a product (Monroe, 1999). For a price skimming seller to gain further market share, it needs to use other pricing strategies. Moreover, the largest setback of using this tactic is that it often leaves the product at a high price compared to its competitors, which is exactly the effect we see in our simulations. Faced with other DF strategies in a market of purely bargain hunters, the DF-algorithms would create an aggressive price war. However, as long as there is some stochasticity in consumer choice, the DF's would reach a common price setting extracting high revenues per sale from the buyer population, as long as no other conflicting pricing strategy is introduced to the market.

Dimicco et al. (2003) found that the DF strategy excels in a market with an abundance of buyers and a peak valuations early in the market. We can confirm that the DF algorithm performs its best when there is an abundance of buyers, which is evident from our simulation of excess demand. Furthermore, we can also infer that the DF-strategy benefits from peak valuations early in the market, as it has no intention of maintaining available inventory throughout the selling period. However, the DF-algorithm suffers in a heterogeneous market, and we can thus also add to the conclusion of Kong (2004) that the DF-seller is a poor performer when faced with algorithms such as GD and SDNN.

Thus far, we have implicitly assumed that the best measure of performance is revenue generated. However, if we expand our view, we find that the DF-algorithm has some features that could make it beneficial in a real market. First of all, by skimming the market, the DF-seller can be a reasonable pricing algorithm for a premium seller aiming to sell goods at a high price, and that does not have a keen interest in obtaining a large market share. Second, the DF is incredibly easy to implement and understand. Any real-world seller wanting to experiment with their prices could easily implement this pricing strategy, however, they need to comprehend the limitations of this algorithm fully.

10.1.2 Goal-Directed

The GD-algorithm is the best performing adaptive algorithm in our simulated market scenarios, yet its performance is far behind the more advanced machine learning algorithms. We can confirm the conclusion by Dimicco et al. (2003), that the GD consistently sells the majority of its inventory given any combination of competition and buyer behavior, at the expense of drastically overshooting or undershooting the valuation curve early and late in the market. It is, therefore, most suited for use in markets in which inventory liquidation is essential.

The majority of our simulations has been done using a decreasing value curve. When there is a large difference in max and min valuations of the segments, the GD algorithm fails to recognize that more revenue can be made by trying to sell more goods in a section of the days. Consequently, the GD strategy performs its best for slower moving market, where the difference between the average valuations of the first and last customers is not too large. The scaling ratio implemented in the GD allows for larger price adjustments towards the end of the market. However, although the intentions behind this scaling are reasonable, we have found that the GD's more volatile pricing late in the market, helps induce price wars, and commonly drives prices towards marginal cost for the final days of the market.

The GD-strategy implements a very rudimentary form of yield management because it involves strategic control of inventory. However, the strategy of pacing sales throughout the selling horizon might not always be the best tactic. For some sellers, there might be an additional value to always having inventory at hand to serve arriving customers, but there might often be a trade-off between keeping inventory and generating more immediate revenue.

It seems, that any seller who wishes to dispose of all its inventory at the cost of losing revenues should consider the GD-algorithm. Furthermore, the concave inventory development curve created by the GD, suggests that the GD's pricing strategy does not quite manage to pace its sales throughout the selling horizon, as too few goods are sold early in the market, and "too many" at its end. As a result, we believe that there is a lot potential for increasing the performance of the GD-strategy by developing a more adaptable scaling ratio. Like the DF, the GD is a simple and understandable pricing algorithm. By fine-tuning its scaling ratio and parameters, it should be potential in using the GD for pricing goods in electronic markets, if the seller consider machine learning to be too complicated to implement.

10.1.3 Sales-Directed Neural Network

Despite its limited network size, the SDNN-algorithm is a solid performer due to its agile and adaptable pricing strategy. It does not require any previous training like the APNN and RBNN, expect for thoughtful initialization of its weights. By learning the demand curve, it implicitly accounts for other sellers strategies, and as such does not adapt to specific opponent strategies.

Because it uses its inventory as the basis for its price decisions, the SDNN's prices rely heavily on what initial inventory we provide it. Given an abundance of inventory, the SDNN will commonly under-price its goods and drive prices towards marginal costs. Often its price will be well below the nearest competitors price, causing large spatial price dispersions. Without any supervision or comparison of its competitor prices, the SDNN has no way of determining whether a decrease in sales occurs due to lower demand caused by stochasticity, or another seller undercutting its prices. This effect makes the SDNN implicitly competitive, as its "standard" response is to lower prices and thus it may unwillingly initiate price wars.

The puzzling trait of the SDNN of commonly undercutting its prices early in the market is arguably caused by suboptimal weights. Since desired sales tend to vary a lot, when there are only a few days left in the market, the SDNN starts the next selling period with weights adjusted to these desired inventory values. Therefore, when inventory gets replenished at the next simulation run, the SDNN has weights confirming to a different region than it would like to be in, and thus its prices are in general too low before it has time to adapt to the newly restarted market. Hence, we can infer that the SDNN performs best when there is only a small difference between its initial price and the overall average market prices.

The creator of the SDNN, Kong (2004), claims that the SDNN learns to predict and account for the long-term consequences of its actions, and shows its superiority over the GD and DF strategies. We can confirm that the SDNN is a better performing strategy than the adaptive strategies. However, we find evidence that the SDNN's ability to predict and account for the long-term consequences of its actions is somewhat vaguer. For instance, the increasingly volatile prices during the final selling periods often intensify price wars, and had the SDNN been able to predict this; it should have been warier with its price adjustments.

Although not being the best performer, the SDNN is the best strategy for sellers' needing to sell out their entire inventory in our finite market. The RBNN simply is not adaptable enough to take part in the final days rapidly decreasing prices, and therefore the SDNN excels in disposing of its inventory. Its limited network has some disadvantages, but these disadvantages can also be beneficial because it allows the SDNN to adjust quickly to market realities and lowering its prices to match its competitors. This is especially evident when faced with the APNN-strategy.

Despite of being a neural network, the SDNN is rather computationally cheap to use. Still, some improvements or additions should be included in the SDNN algorithm before considering trying out the SDNN in a real market. First and foremost, the SDNN algorithm should be expanded to include a mapping of its competitors' prices, thus enabling it to infer more information about market prices and remove the issues of heavily underpricing its goods. Secondly, efforts should be devoted to reducing the problems caused by suboptimal weights early in the market. Finally, the issues of inversion reduce the SDNN's applicability and narrows its weights' range of values, and as such, methods for using the same approach, but without having to invert a neural network should be explored.

10.1.4 Revenue-Based Neural Network

The underlying goal when we designed the RBNN-algorithm was to engineer an algorithm that could expand on the idea of the SDNN network, without having to invert the network. From our results, we can infer that the RBNN, in general, is a great performer, and thanks to the possibility of offline learning using very limited information, it has an ability to perform great from its initialization.

There seems to be a tradeoff when designing pricing algorithms using neural networks regarding the size of the network. The minimalistic network of the SDNN arguably causes more fluctuation in prices as it does not have the "memory", or ability to create the complex pricing curve needed to represent the market. The RBNN's larger network makes it movements more controlled and thus it moves more slowly than the SDNN. Commonly this is a benefit, as massive disruptions in prices increase price wars and creates a more unstable market. However, we saw that when faced with the APNN-algorithm, the RBNN was simply unable to act quickly enough to secure sales.

In terms of achieving their goal of selling an equal amount of inventory each day, the RBNN is inferior to the SDNN. However, the RBNN is better at generating revenue. This highlights the biggest issue with using these types of strategic inventory control algorithms and lead us to the conclusion that their goal is simply suboptimal. For the majority of the scenarios simulated, selling an equal amount of inventory each day is not the best strategy. This can cause a conflict of interest between a seller wanting to maximize revenue and an algorithm that wants to sell out its last item during the final day, and not before.

If we had to choose one algorithm for direct implementation in a real market today, it would have be the RBNN. It is consistently a solid performer when trained offline, because of its ability to adapt to the actual market conditions without dramatically destroying value. Also, sellers employing the RBNN are more likely to have a price that is a bit too high, than way too low. Hence, it can be well suited for retailers that aim to sell out their entire inventory, but not at all costs.

An opportunity for further tuning the RBNN strategy is to implement a more sophisticated evaluation of desired goods sold each day. This could include the shifting of focus to selling more inventory early or late in the market. One should also, as with the SDNN, include an ability to track competitor prices.

10.1.5 Aggressive Pricing Neural Network

The APNN strategy is great at capturing market share in a market where the majority of consumers comparison shop, but does so at the cost of destroying value for sellers at the consumers benefit. When faced with other implicit or explicit aggressive pricing strategies it drives prices towards marginal cost. Since it has no intention of keeping inventory throughout the selling period, the APNN can benefit from high customer valuations early in the market, but also forgo revenues if prices were to rise later in the market due to for instance an upward sloping value curve. Faced with a large number of random shoppers, the APNN will continue to assume that lower prices generate more sales when this might not be the case. Consequently, the

APNN will excel in some markets, and be inferior in others.

The APNN-algorithm can be considered as a mild form of a predatory or cutthroat pricing strategy. The algorithm does not price its goods below cost, but its aggressive nature causes products to be priced at marginal cost. Predatory pricing strategies are often intended for driving out competitors from a market, and as confirmed by our simulations, this form of pricing strategy will cause revenues to fall. Cutthroat competition may affect the rivals', or competing firms' prospects, as the rivals cannot raise enough resources to carry on. According to the "long purse" story, a firm with substantial financial resources, or the "long purse", can prey on its weaker rivals because the strong firm can sustain losses for a longer period of time (Tirole, 1988). When the smaller financially constrained firms are forced out the market, the initiator of an aggressive pricing strategy aims to increase its prices and thus extract more profits. Hence, predatory pricing is mostly suited when entry costs are high because this makes it harder for new competitors to enter the market. However, firms will not benefit from this strategy in the long term, as the probability of another seller utilizing the same tactic is high, and as such competition intensifies and major losses occurs. Also, Tirole (1988) argues that the issue of why the prey faces a financial constraint needs to be addressed because imperfections in capital markets are central to comprehend the implications of the "long purse" story.

Therefore, any seller considering implementing an APNN strategy should do so with caution. In a way, employing an APNN strategy can be related to a grim-trigger strategy, because if one seller starts using it, then the others are likely to do the same, and as such price will be equal to marginal cost and seller will be stuck in a homogeneous good Bertrand competition model. Nevertheless, many DPSOPs already offer price algorithms that undercut specific retailers' prices, but we assume that these real-world implementations include a lower boundary which prevents continuous aggressive cutthroat prices.

Using only a very limited amount of market insight for offline training, we can initiate the APNN in such a way that it becomes a quick learner, and its prices follow a smooth path. However, we find possibilities of further increasing the APNN's performance, by expanding its network to get even finer tuned prices that can adapt to more dramatic changes in day-to-day prices.

10.1.6 Q-learning

The Q-learning algorithm is the most comprehensive pricing algorithm implemented in our model, and it seems that its complexity pays off in terms of performance, at the cost of a lengthy training period. Our simulations provide evidence that regardless of no theoretical proof of convergence, Q-learning is a valuable approach for pricing goods. It would, of course, be nice to find the optimal pricing policy, but the world is seldom optimal, and our simulations show that an optimal policy is not necessarily needed to excel in a market. By only observing the market, the QL can understand the market dynamics, without any information what so ever.

In their paper, Greenwald and Kephart (2000) performed simulations of a model free Q-learning algorithm facing other equal QL-algorithms and amongst others also the Derivate-Following strategy. As we have, they found that Q-learning performs well, despite the non-Markovian nature of the opponents' strategies. Chinthalapati et al. (2006) also, rather unsurprisingly found a similar QL-strategy to be superior to the DF in their simulations. One of the differences between their approach and ours is that they analyze a simpler market, enabling them to find an optimal policy analytically and thus are able to more accurately evaluate Q-learning convergence towards an optimal policy. We operate in an unknown territory, where the optimal policy cannot be found by an analytical approach.

The majority of literature studying dynamic pricing using QL follow the same approach of letting the algorithm learn, and in the case of stationarity, converge to the optimal policy, before introducing it to the market. This is a somewhat unrealistic assumption, as it might be troublesome to perform in a real world implementation. Nevertheless, the value of simulating a market, is that one can gain further appreciation of what might happen in a real market. Hence, if realistic simulated marketplaces can be constructed, one could use a simulation-based approach to train a QL-algorithm offline, by replicating how one believes one's competitors and market behave. Furthermore, an interesting fact found in our simulations is that randomly selecting a price, may not be a terrible approach, and we find evidence that in some markets, randomly selecting a price can outperform adaptive algorithms, leading us to believe that the trade-off between exploration and exploitation is not that critical in our model since substantial profits can be made even though all focus is on exploration. In a real market, randomly choosing prices would not only be troubling for the seller, but the consumers might have an issue with prices not conforming to some form of pattern. Haws and Bearden (2006) found evidence that consumer distrust can increase if the day-to-day prices are heavily fluctuating.

If we were to select a winner in terms of which algorithm is best for pricing goods in our market, neglecting the issues of training, the QL-strategy would be it. Due to the long run time caused by QL, we have not extensively studied how the algorithm performs in various market conditions, yet, we have no reason to believe that QL would perform badly in any of our markets, provided that it has had enough training. However, we can infer that if a market is unstable, meaning that if its competitors' prices in the upcoming simulation run, largely differ from the previous runs, the slow learning QL would struggle at pricing its good properly. Implying that if the other sellers are aware of one seller using QL, they should vary or alter their pricing strategies from time to time, to confuse the QL-seller. As a result, the QL algorithm is an incredibly powerful, but vulnerable approach.

The Q-learning strategy is the only true profit maximizing algorithm in our market since the other implemented algorithms act more like greedy heuristics or are inventory centered. Furthermore, since the QL can learn market dynamics and predict future outcomes, its pricing strategy is adaptable. It will implicitly evaluate which pricing tactic would be the best at its current state, and thus, it can decide when it is beneficial to hold off inventory for later days, when to initiate aggressive cutthroat pricing, when to back out of a price war, or when to implicitly cooperate.

Overall, there are two major drawbacks with the model free QL implemented. The long and tedious training period and the problem of adapting to a highly non-stationary market. One suggested approach to the issues of training is to develop an algorithm that uses function approximation, instead of a look-up table (Greenwald and Kephart, 2000). Sridharan and Tesauro (2002) adheres to this method, and were able to produce solid improvements in learning time, while maintaining profits close, or equal, to look-up table policies. Function approximation seems to enable more rapid learning, and with the possibility of altering the function approximations if a market suddenly changes might also improve the QL's adaptability to an unstable market. In addition, the shear size of a look-up table trying to capture the full complexity of a realistic market would drastically increase an already lengthy training period. Addressing these issues in more detail is a highly complex task, far beyond the scope of this thesis.

10.2 Market Insights

If we assume that the underlying dynamics of the simulated market is somewhat realistic, we can infer some intriguing market insights from our model.

Sellers seem unable to benefit from customers increasing valuation curves when in competition. The problem relates to the dynamics of competition, which hampers the sellers' ability to raise prices. When the number of comparison shoppers is high, and there is little variation between customers valuation of sellers, the greatest number of goods will be sold to the seller with the lowest prices. Thus, the rational decision for any seller wishing to maximize market share is to keep prices low. This is a classic game-theoretic problem. If the sellers collectively and trustfully agreed to increase prices, they would all have a better possibility of increasing their profits. However, if any one seller starts behaving opportunistic, underpricing its competition, it would take the market share majority. Thus without any ability to cooperate and signaling trustworthiness, the sellers supposedly best option is to create a slow moving downward sloping price curve. We find this disability to monetize on increasing valuation and prices to be realistic. There are many reasons why it is uncommon to experience the average price of a consumer good raising in time. One of which is that, if sellers were able to sell their good for a lower price and still make a profit, the nature of competition would make it inherently hard to increase prices.

Furthermore, we find evidence that when faced with a heterogeneous mix of segments, a slightly aggressive pricing tactic is beneficial for the sellers. When sellers face a market of myopic and strategic customers, a tradeoff between utilizing the myopic or the strategic segments higher valuation arise. Evidentially, it would seem reasonable to keep prices above the strategic segment for some time, as strategic consumers can return to purchase later when prices are lower. However, the problem with this strategy is twofold. Firstly, there is a chance that the full residual demand will not be realized since the strategic customers might not return when prices are low enough. Second, the probability for the sellers to capitalize on the strategic segment's early higher valuation falls, as the customers are more likely to return when prices are well below their valuation than when it is close to their maximum valuation. Therefore, we find that more sales and revenues are generated when the sellers price their goods close to the strategic segment. This is also the reason for why a monopolist in a heterogeneous market might have issues extracting the same seller surplus as in an duopolistic market. Without competition, an uninformed and unexploratory seller might be unaware of a strategic segments presence in the market and naively enjoying its high revenue per sale, when in reality more profits can be made from slightly lower prices.

10.3 Critique

The way in which we have implemented strategic customer behavior has several issues. First of all, it is a rather rudimentary approach for replicating some of the effects caused by a rational consumer, by which we imply, that the model's strategic consumers do not fully conform with the definition of properly strategic behavior. A more correct approach would be to have the consumers predict future prices, and evaluate their purchasing decision using an expected value incorporating factors as future price paths, product availability and the cost of delaying a purchase. Secondly, we find that our approach makes the most sense when customer valuations are decreasing. Cases where customers' valuations increase in time, makes our strategic segment inherently un-strategic, as, by definition, strategic customers evaluate the option of delaying a purchase. Implying that if prices are rising, immediate purchasing is optimal. Thirdly, we are not able to replicate the effect arising from consumers competing for product availability and their risk profiles, because their willingness to pay is exogenously given and not dependent on expected product availability.

In continuation, all implemented buyers immediately purchase a good if the price is below their willingness to pay. However, it would be more realistic to assume that the probability of a consumer purchasing simply increases as prices drop below their valuation. Consequently, we find it likely that there might be times when consumers might not buy a product despite the possibility of gaining an immediate positive utility. Likewise, we also miss out on more realistic stochastic effects in how the consumers choose their sellers, by not including aspects of the more sophisticated discrete choice theory models.

Our model implements a higher degree of customer heterogeneity than other literature. Still, to properly replicate a market, one can assume that the number of segments and variations between them are far greater than we have explored. Therefore, simulations of even more dispersed value curves and variations of segments' behavior should be performed. The assumption of a constant arrival rate is also unrealistic, as it to some degree ignores the likely effect that low prices stimulate a higher demand. Since demand each period is capped by the segments arrivals, efforts should be made to implement a dynamic arrival rate using, for instance, an inhomogeneous Poisson process.

Furthermore, we have disregarded the sellers' cost, thus forgoing an important dimension of the sellers and also their heterogeneity. As a result, we miss out on effects caused by factors like inventory holding costs, and changing marginal costs. Furthermore, by not including any salvage value or cost for unsold inventory, we cannot fully represent realistic seller behavior, as there is bound to be some implications, good or bad, from unsold inventory. In addition, we have not explored seller rationality or interplay between strategic sellers to any extent, which also might a profound effect. We have further been unsuccessful in replicating probable market dynamics when sellers utilize capacity rationing, as prices in most markets would not decrease if supply becomes limited. The problem we experience is mainly caused by the inventory centered algorithms, and the fact that we have no method for inclining customers to purchase early when prices are high.

Our findings stem from repeating the same identical finite market over and over, and although the algorithms causes some non-stationarity, aspects of changes in buyer behavior over time are lost. Realistically, it is unlikely that consumers do not adapt to the market and continuously adjust their decision-making process and demand.

To improve confidence in our findings, we should to a greater extent have averaged the results from our simulations. Optimally each simulated scenario should be averaged for hundreds of runs. Our approach of mostly running 100 simulations, while letting the machine learning algorithms learn, can lead us to see effects that may be coincidental and thus preventing us from understanding the most likely dynamics. However, we found averaging hundreds of different "hundred simulation runs" to be too time-consuming, and thus our focus has been to evaluate a greater of variety market scenarios, and not to marginally improve our confidence in just a few scenarios. Furthermore, the same also applies to our graphs of the development of market prices and inventory. Since they are mere "snapshots" of a single simulation run's market, they only present what happened in that specific simulation, which may or may not be the common effect. Implying that some of our findings might stem from an unordinary situation, and thus might not reflect the actual common effect.

Furthermore, we have neglected discussions of technicalities regarding machine learning, and as such, we cannot evaluate issues such as Q-learning convergence, or finding the best method and learning rate for our algorithms. Also, the implemented algorithms are rather elementary, and might not reflect the full complexity and possibilities of their respective approach.

All this combined, somewhat deprives us the opportunity of confidentially concluding how machine learning will perform in reality. Our simulated market lacks important market characteristics that are likely to have a profound impact on underlying dynamics, pricing decisions and algorithm learning. Hence, we can only infer how these algorithms perform in a constructed simplistic market, and as such any insights beyond this are merely based on qualified guesses and reflections.

Chapter 11

Concluding Remarks

By studying the literature regarding dynamic pricing by machine learning we were able to find some similarities and generalize the papers in our framework, despite their apparent variation. Every paper considered in this context presented a machine learning algorithm whose performance was evaluated by simulation using some constructed market model. As a result, efforts were devoted to present the various assumptions used for simulating the demand and behaviors of consumers in these simulations.

From our results, we find that the majority of literature is considering rather simplistic market models. Assumptions like those of myopic customers and independent demand are used by almost every paper considered, predominantly due to the complexity introduced by adding more realistic assumptions. Further, it seems to be no consensus amongst researchers in the literature pointing towards which machine learning approach is best suited for solving dynamic pricing problems. Nor have we found any unified evidence that some approaches are better performing than others under various market assumptions. However, the bulk of literature considering dynamic pricing by machine learning develops reinforcement learning algorithms using the principles of Q-learning. Still, there is a lack of research justifying reinforcement learning as opposed to other methods like those of i.e. neural networks and evolutionary algorithms.

Our most conclusive discovery from the literature study is that, so far, the literature has proven to be unsuccessful in evaluating the performance of dynamic pricing using machine learning in real-world applications. Of all the 31 papers reviewed, not a single one presented any empirical evidence of actual increases in revenue or profitability for sellers. Consequently, we are unable to confidentially state the added value by these methods for real-world sellers.

What the literature present, however, is that in simulated markets incorporating simplifying assumptions, machine learning can provide solid increases in sellers' revenue by exploiting variations in consumers' willingness to pay, utilizing little or no a priory information. Further, dynamic pricing by machine learning is numerically proven to be superior to static pricing policies in these simulations. Also, many authors prove that more advanced approaches like Q-learning can generate close to optimal price policies for sellers in stationary markets. Lastly, because of different assumptions and methods used for simulations, it seems to be no standard practice for developing and evaluating machine learning methods; hence, it is difficult to assess their performance against each other.

By focusing on some of the weaknesses of the literature, our proposed model enabled us to consider how machine learning algorithms perform compared to each other in heterogeneous markets. Utilizing a simulation-based approach allowed us to evaluate how algorithms might learn, adapt and perform in a stochastic market. Perhaps our most conclusive discovery from our simulation is that what algorithm performs the best, is highly dependent on what market the sellers are operating in. None of the suggested machine learning algorithms are able to generate the highest revenues in all market scenarios. Thus, we emphasize that choosing which algorithm to pursue is a tactical issue, requiring the seller to understand both its current competitors' and customers' behavior, while also considering the future consequences of their choice of algorithm.

What we can infer is that the Q-learning algorithm present a more solid approximation of what optimal future price paths might be, but at a cost of a seriously lengthy training period. The neural networks show promising results in providing a more balanced approach to training time and performance, but they are unable to fully comprehend the consequences of their actions and consequently show a more aggressive price path than Q-learning.

Another valuable finding is that pricing performed by machine learning seems not only to create value for sellers, but can in many cases also increase consumer surplus and economic welfare. A static price strategy, cannot take advantage of fluctuations in consumer willingness to pay. Evidently, sellers using a fixed price strategy will either underprice their goods, forgoing surplus to the consumers, or overprice their products, thus reducing output and economic welfare. Hence, it seems conclusive that fluctuations in price may be good for maximizing economic welfare.

Our results further provide us with an understanding of that, when in competition, it is the actual dynamic between sellers' price policies that are of most importance. Variations in customer valuation over time is subordinate to seller composition because the nature of competition makes it inherently hard to capitalize on increasing customer valuations. The inventory oriented algorithms especially have a willingness to sacrifice margins for high sales, and in competition, these inventory focused algorithms might induce aggressive price wars.

However, competition might increase economic welfare in heterogeneous markets, because its lower prices persuade customers with lower valuations to purchase early, thus increasing current demand. Capitalizing on strategic segments early in a market will also benefit the myopic segments, by ensuring a price below their valuation. Hence, it seems that when faced with both strategic and myopic segments, sellers might be better off by reducing the price of their goods, as the added demand can generate higher revenues. Still, this relies on the share of strategic consumers and the dispersion between their valuations and those of the myopic segments. Another benefit of strategic customer presence is their ability to lie dormant in the market, and thus, sales are not instantly lost. However as time progress, the sellers ability to monetize on early arriving strategic customers decrease if sellers conform to a declining price path.

Contrary to common belief, our results show that a high percentage of comparison shoppers might actually create additional value for sellers. In stochastic markets where firms have no way of determining why current demand is not as expected, we find that inventory oriented sellers wrongly assume that competing firms are undercutting their prices, and thus they may initiate price wars. Seeing as markets with a high share of comparison shoppers reduce stochasticity, the relationship between price and demand is more stable, and thus, our algorithms can better adhere to the actual market conditions.

We find evidence that in a duopoly, two competing Q-learning algorithms can price their goods close to the monopolistic price and thus effectively share the market, despite lacking methods of communication and cooperation. Therefore, it seems that the $Q(s, a)$ values can implicitly capture the negative consequences of aggressive competition, and thus the QL-algorithms collectively and independently decide that cooperation is their best approach. The neural networks show less implicit cooperation, yet in most cases they are able to implicitly collude to some degree, as very few scenarios cause prices to fall to marginal cost over the entire selling period.

Thus, we can conclude that using simulation-based models for evaluating and developing machine learning algorithms are a valuable approach. However, we must not forget that we cannot fully rely on results from simulations, as what may seem reasonable in a constructed market, might not be realistic. Hence, the value of simulation can only be confirmed by empirical studies of real-world implementation. Still, before exploring real-world utilization of machine learning algorithms we find some issues that need to be addressed.

Since machine learning algorithms use experienced data to learn, their ability to predict future outcomes that do not adhere to previous behavior is limited. Consequently, neural networks and Q-learning alike all have an inability to deal with non-stationary markets. The more un-stationary the market, the more problems the algorithms will have at pricing their goods effectively. If the underlying probabilities of the market rapidly change, due to sudden alteration of buyer or competitor behaviors, the algorithms' previously learned experience can instantly be close to worthless. This a huge issue for the look-up Q-learning algorithm, due to the extensive training time required to develop a decent price policy.

Furthermore, the more complex the market, the more complex the algorithms need to be. For instance, if a market has large volatility in consumer valuations, the algorithms need to be able to represent this complexity properly. For neural networks, this implies engineering a larger network, and with a larger network intricate issues of training and overfitting arise. For a look-up Q-learning algorithm, a more complex market requires an even longer training period.

In conclusion, we find substantial evidence that sellers should pursue dynamic pricing of goods using machine learning. However, there are many unanswered questions still needed to be addressed, and consequently, we end this thesis by suggesting some exciting topics for future research in this field of science. First of all, the need for empirical scientific evidence of machine learning is of great importance for justifying these methods. Secondly, researchers should pursue the development of even more realistic market models incorporating traits as strategic multiagent cooperation, demand dependencies, heterogeneous goods and more advanced buyer decision-making processes. Third, exploring what algorithm might be the best response to a proposed multiagent market would be welcomed. By simulation, one could then gain a numerical understanding of what might be a sellers' best-response algorithm when faced with a variety of machine learning agents. Finally, we encourage the further development of a standardized model for simulating markets, enabling researchers to: (1) evaluate the effects of single and multiple learning agents utilizing various methods of machine learning have on market dynamics, (2) compare and benchmark the performance of one algorithm against others under equal conditions.

Bibliography

M. Adnan, M. Nagi, K. Kianmehr, R. Tahboub, M. Ridley, and J. Rokne. Promoting where, when and what? an analysis of web logs by integrating data mining and social network techniques to guide ecommerce business promotions. *Social Network Analysis and Mining*, 1(3):173–185, 2011.

H.-s. Ahn, M. Güsmüs, and P. Kaminsky. Pricing and manufacturing decisions when demand is a function of prices in multiple periods. *Operations Research*, 55(6):1039–1057, 2007.

S. P. Anderson, A. de Palma, and J.-F. Thisse. *Discrete Choice Theory of Product Differentiation.* The MIT Press, Cambridge, Mass, 1st edition, 1992.

W. L. Baker, E. Lin, M. V. Marn, and C. C. Zawada. Getting prices right on the web. *McKinsey Quarterly*, 2, 2001.

P. P. Belobaba. Survey paper - airline yield management an overview of seat inventory control. *Transportation Science*, 21(2):63–73, 1987.

E. A. Boyd and I. C. Bilegan. Revenue management and e-commerce. *Management Science*, 49(10):1363–1386, 2003.

D. Braziunas. Pomdp solution methods. *University of Toronto, Tech Rep.*, 2003.

C. H. Brooks, S. Fay, J. K. MacKie-Mason, J. O. Kephart, and E. H. Durfee. Automated strategy searches in an electronic goods market: Learning and complex price schedules. In *Electronic Commerce 1999 (EC-99)*, 1999.

C. H. Brooks, R. Das, J. O. Kephart, J. K. MacKie-Mason, R. S. Gazzale, and E. H. Durfee. Information bundling in a dynamic environment. *Society for Computational Economics*, 2001.

C. H. Brooks, R. S. Gazzale, R. Das, J. O. Kephart, J. K. MacKie-Mason, and E. H. Durfee. Model selection in an information economy: Choosing what to learn. *Computational Intelligence*, 18(4):566–582, 2002.

L. Chan, Z. Shen, D. Simchi-Levi, and J. L. Swann. Coordination of pricing and inventory decisions: A survey and classification. In D. Simchi-Levi, S. Wu, and Z.-J. Shen, editors, *Handbook of Quantitative Supply Chain Analysis*, volume 74 of *International Series in Operations Research and Management Science*, pages 335–392. Springer US, 2004.

Y. Cheng. Dynamic pricing decision for perishable goods: A q-learning approach. In *4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08.*, pages 1–5, 2008.

Y. Cheng. Real time demand learning-based q-learning approach for dynamic pricing in e-retailing setting. *International Symposium on Information Engineering and Electronic Commerce*, pages 594–598, 2009.

V. Chinthalapati, N. Yadati, and R. Karumanchi. Learning dynamic prices in multiseller electronic retail markets with price sensitive customers, stochastic demands, and inventory replenishments. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 36(1):92–106, 2006.

B. D. Chung, J. Li, T. Yao, and T. L. Friesz. Demand learning and dynamic pricing under competition in a state-space framework. *Transactions on Engineering Management*, 59(2):240–249, 2012.

R. H. Coase. Durability and monoply. *Journal of Law and Economics*, 15 (1):143–149, 1972.

A. Collins and L. Thomas. Comparing reinforcement learning approaches for solving game theoretic models: a dynamic airline pricing game example. *Journal of the Operational Research Society*, 63:1165–1173, 2012.

A. Collins and L. Thomas. Learning competitive dynamic airline pricing under different customer models. *Journal of Revenue and Pricing Management*, 12(5):416–430, 2013.

Coursera. Machine learning. `https://www.coursera.org/learn/machine-learning` [Accessed: 2015-11-30], 2015.

A. V. den Boer. *Dynamic Pricing and Learning.* PhD thesis, VU University Amsterdam, 2013.

A. V. den Boer. Dynamic pricing and learning: Historical origins, current research and new directions. *Operations Research and Management Science*, 20:1–18, 2015.

J. M. Dimicco, A. Greenwald, and P. Maes. Dynamic pricing strategies under a finite time horizon. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, 2001.

J. M. Dimicco, P. Maes, and A. Greenwald. Learning curve: A simulation based approach to dynamic pricing. *Electronic Commerce Research*, 3: 245–276, 2003.

I. Dogan and A. R. Güner. A reinforcement learning approach to competitive ordering and pricing problem. *Expert Systems*, 32:39–48, 2015.

easyJet. Dynamic pricing explained. `http://www.easyjet.com/en/policy/dynamic-pricing` [Accessed: 2015-12-13], 2015.

W. Elmaghraby and P. Keskinocak. Dynamic pricing in the presence of inventory considerations: Research overview, current practices and future directions. *Management Science*, 49(10):1287–1309, 2003.

M. Fisher, S. Gallino, and J. Li. Competition-based dynamic pricing in online retailing: A methodology validated with field experiements. *Availiable at SSRN*, 2016.

G. Gallego and G. J. van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science*, 40(8): 999–1020, 1994.

E. Garbarino and O. F. Lee. Dynamic pricing in internet retail: Effects on consumer trust. *Psychology & marketing*, 20(6):495–513, 2003.

T. K. Ghose and T. T. Tran. Dynamic pricing in electronic commerce using neural network. In *E-Technologies: Innovation in an Open World*, Lecture Notes in Business Information Processing, pages 227–232. Springer Berlin Heidelberg, 2009.

A. R. Greenwald and J. O. Kephart. Shopbots and pricebots. In A. Moukas, F. Ygge, and C. Sierra, editors, *Agent Mediated Electronic Commerce II*, volume 1788 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin Heidelberg, 2000.

A. R. Greenwald, J. O. Kephart, and G. Tesauro. Strategic pricebot dynamics. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 58–67, New York, NY, USA, 1999.

W. Han. A dynamic pricing algorithm by bayesian q-learning. In *Second International Conference on Computer Modeling and Simulation, 2010. ICCMS '10.*, volume 2, pages 515–519, 2010.

W. Han, L. Liu, and H. Zheng. Dynamic pricing by multiagent reinforcement learning. In *International Symposium onElectronic Commerce and Security, 2008*, pages 226–229, 2008.

K. H. Haws and W. O. Bearden. Dynamic pricing and consumer fairness perceptions. *Journal of Consumer Research*, 33(3):304–311, 2006.

W. Jintian and Z. Lei. Application of reinforcement learning in dynamic pricing algorithms. In *IEEE International Conference on Automation and Logistics, 2009. ICAL '09.*, pages 419–423, 2009.

J. O. Kephart, J. E. Hanson, and J. Sairamesh. Price and niche wars in a free-market economy of software agents. *Artificial Life*, 4:1–23, 1998.

J. O. Kephart, C. H. Brooks, and R. Das. Pricing information bundles in a dynamic environment. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, 2001.

D. Kong. One dynamic pricing strategy in agent economy using neural network based on online learning. In *IEEE/WIC/ACM International Conference on Web Intelligence, 2004. WI 2004. Proceedings.*, pages 98–102, 2004.

V. Könönen. Dynamic pricing based on asymmetric multiagent reinforcement learning. *International Journal of Intelligent Systems*, 21:73–98, 2006.

E. Kutschinski, T. Uthmann, and D. Polani. Learning competitive pricing strategies by multi-agent reinforcement learning. *Journal of Economic Dynamic and Control*, 27:2207–2218, 2003.

Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–48. Springer, 2012.

Y. Levin, J. McGill, and M. Nediak. Dynamic pricing in the presence of strategic consumers and oligopolistic competition. *Management Science*, 55(1):32–46, 2009.

Y. Levin, J. McGill, and M. Nediak. Optimal dynamic pricing of perishable items by a monopolist facing strategic consumers. *Production and Operations Management*, 19(1):40–60, 2010.

T. Levina, Y. Levin, J. McGill, and N. Mikhail. Dynamic pricing with online learnign and strategic consumers: An application of the aggregating algorithm. *Operations Research*, 57(2):327–341, 2009.

C. Li, H. Wang, and Y. Zhang. Dynamic pricing decision in a duopolistic retailing market. In *Proceedings of the 6th World Congress on Intelligent Control and Automation*, pages 6993–6997, 2006.

J. Li, N. Granados, and S. Netessine. Are consumers strategic? structural estimation from the air-travel industry. *Management Science*, 60(9):2114–2137, 2014.

K. Littlewood. Forecasting and control of passenger bookings. *AGIFORS Symposium Proc.*, 12, 1972.

M. Littman and I. Charles. Q-learning convergence. `https://www.udacity.com/course/viewer#!/c-ud262/l-643978935/m-634899063` [Accessed: 2015-11-29], 2015.

G. Liu and H. Wang. An online sequential feed-forward network model for demand curve prediciton. *Journal of Information and Computational Science*, 10(10):3063–3069, 2013.

B.-L. Lu, H. Kita, and Y. Nishikawa. Inverting feedforward neural networks using linear and nonlinear programming. *IEEE Transactions on Neural Networks*, 10(6):1271–1290, 1999.

J. I. McGill and G. J. van Ryzin. Revenue management: Research overview and prospects. *Transportation Science*, 33(2):233–256, 1999.

M. McLean. Insights on amazon's dynamic pricing. `https://www.youtube.com/watch?v=0a3eBQB1uUA` [Accessed: 2015-12-10], 2014.

K. B. Monroe. *The Pricing Strategy Audit.* Pearson Education, Limited, 1st edition, 1999.

P. B. Mullen, C. K. Monson, K. D. Seppi, and S. C. Warnick. Particle swarm optimization in dynamic pricing. In *Congress on Evolutionary Computation*, pages 1232–1239, 2006.

Y. Narahari, C. V. L. Raju, K. Ravikumar, and S. Shah. Dynamic pricing models for electronic business. *Sadhana*, 30:231–256, 2005.

S. Netessine and R. Shumsky. Introduction to the theory and practice of yield management. *INFORMS Transactions on Education*, 3(1):34–44, 2002.

A. Odlyzko. Privacy, economics, and price discrimination on the internet. *ICEC2003: Fifth International Conference on Electronic Commerce*, pages 355–366, 2003.

B. O'Meara. Jet.com redefines online shopping with transparent, dynamic pricing. `http://blogs.microsoft.com/business-matters/2015/08/03/jet-com-redefines-online-shopping-with-transparent-dynamic-pricing/` [Accessed: 2015-12-13], 2015.

N. R. S. Raghavan. Data mining in e-commerce: A survey. *Sadhana*, 30: 275–289, 2005.

C. Raju, Y. Narahari, and K. Ravikumar. Learning dynamic prices in electronic retail markets with customer segmentation. *Annals of Operations Research*, 143(1):59–75, 2006.

S. Ramezani, P. A. N. Bosman, and H. L. Poutré. Adaptive strategies for dynamic pricing agents. In *International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM*, volume 2, pages 323–328, 2011.

R. Rana and F. S. Oliveira. Dynamic pricing policies for interdependend perishable products or services using reinforcement learning. *Expert Systems with Applications*, 42:426–436, 2015.

W. Reinartz. Customizing prices in online markets. *Symphonya. Emerging Issues in Management*, 1:55–65, 2002.

P. Rusmevichientong, J. A. Salisbury, L. T. Truss, B. Van Roy, and P. W. Glynn. Opportunities and challenges in using online preference data for vehicle pricing: A case study at general motors. *Journal of Revenue and Pricing Management*, 5(1):45–61, 2006.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, New Jersey, 3rd edition edition, 2010.

J. Sairamesh and J. O. Kephart. Price dynamic of vertically differentiated information markets. In *Proceedings of the First International Conference on Information and Computer Economies*, ICE '98, pages 28–36, New York, NY, USA, 1998. ACM.

M. Schwind. *Dynamic Pricing and Automated Resource Allocation for Complex Information Services.* Springer Berlin Heidelberg, 2007.

A. Sen. A comparison of fixed and dynamic pricing policies in revenue management. *Omega*, 41:586–597, 2013.

A. Shaked and J. Sutton. Product differentiation and industrial structure. *The Journal of Industrial Economics*, 36(2):131–146, 1987.

S. Shakya, F. Oliveira, and G. Owusu. Analysing the effect of demand uncertainty in dynamic pricing with eas. In *Research and Development in Intelligent Systems XXV*, pages 77–90. Springer, 2009.

S. Shakya, M. Kern, G. Owusu, and C. M. Chin. Neural network demand models and evolutionary optimisers for dynamic pricing. *Knowledge-Based Systems*, 29:44–53, 2012.

Z.-J. M. Shen and X. Su. Customer behavior modeling in revenue management and auctions: A review and new research opportunities. *Production and Operations Management*, 16(6):713–728, 2007.

P. Simon. *Too Big to Ignore: The Business Case for Big Data.* Wiley, 2013.

B. C. Smith, J. F. Kleimkuhler, and R. M. Darrow. Yield management at american airlines. *Interfaces*, 22(1):8–31, 1992.

M. Sridharan and G. Tesauro. Multi-agent q-learning and regression trees for automated pricing decisions. In *Game Theory and Decision Theory in Agent-Based Systems*, pages 217–234. Springer US, 2002.

C. Steiner. *Automate This: How Algorithms Came to Rule our World.* Penguin Group, 2012.

X. Su. Intertemporal pricing with strategic customer behavior. *Management Science*, 53(5):726–741, 2007.

J. Surowiecki. In praise of efficient price gouging. `http://www.technologyreview.com/review/529961/in-praise-of-efficient-price-gouging/` [Accessed: 2015-12-12], 2014.

K. T. Talluri and G. J. van Ryzin. *The Theory and Practice of Revenue Management*, volume 68 of *International Series in Operations Research and Management Science*. Springer US, 2004.

Z. Tang, M. D. Smith, and A. Montgomery. The impact of shopbot use on prices and price disperison: Evidence from online book retailing. *International Journal of Industrial Organization*, 28:579–590, 2010.

G. Tesauro and J. O. Kephart. Pricing in agent economies using multi-agent q-learning. *Autonomous Agents and Multi-Agent Systems*, 5:289–304, 2002.

J. Tirole. *The Theory of Industrial Organization.* MIT Press, Cambridge, Mass, 1988.

Uber. The company. `https://www.uber.com/about` [Accessed: 2015-12-14], 2015.

M.-J. Vázquez-Gallo, M. Estévez, and S. Egido. Active learning and dynamic pricing. *American Journal of Operations Research*, 4:90–100, 2014.

D. Vengerov. A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments. *Future Generation Computer Systems*, 24(7):687–693, 2008.

C. J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, Cambridge, 1989.

C. J. C. H. Watkins. Q-learning. *Machine Learning*, 8:279–292, 1992.

L. R. Weatherford and S. E. Bodily. A taxonomy and research overview of perishable-assets revenue management: Yield management, overbooking, and pricing. *Operations Research*, 40(5):831–844, 1992.

WiseCommerce. Wisedynamic. `http://www.wiseruk.co.uk/dynamic` [Accessed: 2015-12-14], 2015.

M. Wooldridge. *An Introduction to MultiAgent Systems.* John Wiley and Sons Ltd, West Sussex, United Kingdom, 2nd edition, 2009.

C. H. Xia and P. Dube. Dynamic pricing in e-services under demand uncertainty. *Production and Operations Management*, 16(6):701–712, 2007.