Matthew F. Dixon
Igor Halperin
Paul Bilokon

Machine Learning in Finance

From Theory to Practice

MOREMEDIA

Springer

Machine Learning in Finance

Matthew F. Dixon • Igor Halperin • Paul Bilokon

Machine Learning in Finance

From Theory to Practice

Springer

Matthew F. Dixon
Department of Applied Mathematics
Illinois Institute of Technology
Chicago, IL, USA

Igor Halperin
Tandon School of Engineering
New York University
Brooklyn, NY, USA

Paul Bilokon
Department of Mathematics
Imperial College London
London, UK

Additional material to this book can be downloaded from http://mypages.iit.edu/~mdixon7/book/ML_Finance_Codes-Book.zip

ISBN 978-3-030-41067-4 ISBN 978-3-030-41068-1 (eBook)
https://doi.org/10.1007/978-3-030-41068-1

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Once you eliminate the impossible, whatever remains, no matter how improbable, must be the truth.

—Arthur Conan Doyle

Introduction

Machine learning in finance sits at the intersection of a number of emergent and established disciplines including pattern recognition, financial econometrics, statistical computing, probabilistic programming, and dynamic programming. With the trend towards increasing computational resources and larger datasets, machine learning has grown into a central computational engineering field, with an emphasis placed on plug-and-play algorithms made available through open-source machine learning toolkits. Algorithm focused areas of finance, such as algorithmic trading have been the primary adopters of this technology. But outside of engineering-based research groups and business activities, much of the field remains a mystery.

A key barrier to understanding machine learning for non-engineering students and practitioners is the absence of the well-established theories and concepts that financial time series analysis equips us with. These serve as the basis for the development of financial modeling intuition and scientific reasoning. Moreover, machine learning is heavily entrenched in engineering ontology, which makes developments in the field somewhat intellectually inaccessible for students, academics, and finance practitioners from the quantitative disciplines such as mathematics, statistics, physics, and economics. Consequently, there is a great deal of misconception and limited understanding of the capacity of this field. While machine learning techniques are often effective, they remain poorly understood and are often mathematically indefensible. How do we place key concepts in the field of machine learning in the context of more foundational theory in time series analysis, econometrics, and mathematical statistics? Under which simplifying conditions are advanced machine learning techniques such as deep neural networks mathematically equivalent to well-known statistical models such as linear regression? How should we reason about the perceived benefits of using advanced machine learning methods over more traditional econometrics methods, for different financial applications? What theory supports the application of machine learning to problems in financial modeling? How does reinforcement learning provide a model-free approach to the Black–Scholes–Merton model for derivative pricing? How does Q-learning generalize discrete-time stochastic control problems in finance?

This book is written for advanced graduate students and academics in financial econometrics, management science, and applied statistics, in addition to quants and data scientists in the field of quantitative finance. We present machine learning as a non-linear extension of various topics in quantitative economics such as financial econometrics and dynamic programming, with an emphasis on novel algorithmic representations of data, regularization, and techniques for controlling the bias-variance tradeoff leading to improved out-of-sample forecasting. The book is presented in three parts, each part covering theory and applications. The first part presents supervised learning for cross-sectional data from both a Bayesian and frequentist perspective. The more advanced material places a firm emphasis on neural networks, including deep learning, as well as Gaussian processes, with examples in investment management and derivatives. The second part covers supervised learning for time series data, arguably the most common data type used in finance with examples in trading, stochastic volatility, and fixed income modeling. Finally, the third part covers reinforcement learning and its applications in trading, investment, and wealth management. We provide Python code examples to support the readers' understanding of the methodologies and applications. As a bridge to research in this emergent field, we present the frontiers of machine learning in finance from a researcher's perspective, highlighting how many well-known concepts in statistical physics are likely to emerge as research topics for machine learning in finance.

Prerequisites

This book is targeted at graduate students in data science, mathematical finance, financial engineering, and operations research seeking a career in quantitative finance, data science, analytics, and fintech. Students are expected to have completed upper section undergraduate courses in linear algebra, multivariate calculus, advanced probability theory and stochastic processes, statistics for time series (econometrics), and gained some basic introduction to numerical optimization and computational mathematics. Students shall find the later chapters of this book, on reinforcement learning, more accessible with some background in investment science. Students should also have prior experience with Python programming and, ideally, taken a course in computational finance and introductory machine learning. The material in this book is more mathematical and less engineering focused than most courses on machine learning, and for this reason we recommend reviewing the recent book, *Linear Algebra and Learning from Data* by Gilbert Strang as background reading.

Advantages of the Book

Readers will find this book useful as a bridge from well-established foundational topics in financial econometrics to applications of machine learning in finance. Statistical machine learning is presented as a non-parametric extension of financial econometrics and quantitative finance, with an emphasis on novel algorithmic representations of data, regularization, and model averaging to improve out-of-sample forecasting. The key distinguishing feature from classical financial econometrics and dynamic programming is the absence of an assumption on the data generation process. This has important implications for modeling and performance assessment which are emphasized with examples throughout the book. Some of the main contributions of the book are as follows:

- The textbook market is saturated with excellent books on machine learning. However, few present the topic from the prospective of financial econometrics and cast fundamental concepts in machine learning into canonical modeling and decision frameworks already well established in finance such as financial time series analysis, investment science, and financial risk management. Only through the integration of these disciplines can we develop an intuition into how machine learning theory informs the practice of financial modeling.
- Machine learning is entrenched in engineering ontology, which makes developments in the field somewhat intellectually inaccessible for students, academics, and finance practitioners from quantitative disciplines such as mathematics, statistics, physics, and economics. Moreover, financial econometrics has not kept pace with this transformative field, and there is a need to reconcile various modeling concepts between these disciplines. This textbook is built around powerful mathematical ideas that shall serve as the basis for a graduate course for students with prior training in probability and advanced statistics, linear algebra, times series analysis, and Python programming.
- This book provides financial market motivated and compact theoretical treatment of financial modeling with machine learning for the benefit of regulators, wealth managers, federal research agencies, and professionals in other heavily regulated business functions in finance who seek a more theoretical exposition to allay concerns about the "black-box" nature of machine learning.
- Reinforcement learning is presented as a model-free framework for stochastic control problems in finance, covering portfolio optimization, derivative pricing, and wealth management applications without assuming a data generation process. We also provide a model-free approach to problems in market microstructure, such as optimal execution, with Q-learning. Furthermore, our book is the first to present on methods of inverse reinforcement learning.
- Multiple-choice questions, numerical examples, and more than 80 end-of-chapter exercises are used throughout the book to reinforce key technical concepts.

- This book provides Python codes demonstrating the application of machine learning to algorithmic trading and financial modeling in risk management and equity research. These codes make use of powerful open-source software toolkits such as Google's TensorFlow and Pandas, a data processing environment for Python.

Overview of the Book

Chapter 1

Chapter 1 provides the industry context for machine learning in finance, discussing the critical events that have shaped the finance industry's need for machine learning and the unique barriers to adoption. The finance industry has adopted machine learning to varying degrees of sophistication. How it has been adopted is heavily fragmented by the academic disciplines underpinning the applications. We view some key mathematical examples that demonstrate the nature of machine learning and how it is used in practice, with the focus on building intuition for more technical expositions in later chapters. In particular, we begin to address many finance practitioner's concerns that neural networks are a "black-box" by showing how they are related to existing well-established techniques such as linear regression, logistic regression, and autoregressive time series models. Such arguments are developed further in later chapters.

Chapter 2

Chapter 2 introduces probabilistic modeling and reviews foundational concepts in Bayesian econometrics such as Bayesian inference, model selection, online learning, and Bayesian model averaging. We develop more versatile representations of complex data with probabilistic graphical models such as mixture models.

Chapter 3

Chapter 3 introduces Bayesian regression and shows how it extends many of the concepts in the previous chapter. We develop kernel-based machine learning methods—specifically Gaussian process regression, an important class of Bayesian machine learning methods—and demonstrate their application to "surrogate" models of derivative prices. This chapter also provides a natural point from which to develop intuition for the role and functional form of regularization in a frequentist setting—the subject of subsequent chapters.

Chapter 4

Chapter 4 provides a more in-depth description of supervised learning, deep learning, and neural networks—presenting the foundational mathematical and statistical learning concepts and explaining how they relate to real-world examples in trading, risk management, and investment management. These applications present challenges for forecasting and model design and are presented as a reoccurring theme throughout the book. This chapter moves towards a more engineering style exposition of neural networks, applying concepts in the previous chapters to elucidate various model design choices.

Chapter 5

Chapter 5 presents a method for interpreting neural networks which imposes minimal restrictions on the neural network design. The chapter demonstrates techniques for interpreting a feedforward network, including how to rank the importance of the features. In particular, an example demonstrating how to apply interpretability analysis to deep learning models for factor modeling is also presented.

Chapter 6

Chapter 6 provides an overview of the most important modeling concepts in financial econometrics. Such methods form the conceptual basis and performance baseline for more advanced neural network architectures presented in the next chapter. In fact, each type of architecture is a generalization of many of the models presented here. This chapter is especially useful for students from an engineering or science background, with little exposure to econometrics and time series analysis.

Chapter 7

Chapter 7 presents a powerful class of probabilistic models for financial data. Many of these models overcome some of the severe stationarity limitations of the frequentist models in the previous chapters. The fitting procedure demonstrated is also different—the use of Kalman filtering algorithms for state-space models rather than maximum likelihood estimation or Bayesian inference. Simple examples of hidden Markov models and particle filters in finance and various algorithms are presented.

Chapter 8

Chapter 8 presents various neural network models for financial time series analysis, providing examples of how they relate to well-known techniques in financial econometrics. Recurrent neural networks (RNNs) are presented as non-linear time series models and generalize classical linear time series models such as $AR(p)$. They provide a powerful approach for prediction in financial time series and generalize to non-stationary data. The chapter also presents convolution neural networks for filtering time series data and exploiting different scales in the data. Finally, this chapter demonstrates how autoencoders are used to compress information and generalize principal component analysis.

Chapter 9

Chapter 9 introduces Markov decision processes and the classical methods of dynamic programming, before building familiarity with the ideas of reinforcement learning and other approximate methods for solving MDPs. After describing Bellman optimality and iterative value and policy updates before moving to Q-learning, the chapter quickly advances towards a more engineering style exposition of the topic, covering key computational concepts such as greediness, batch learning, and Q-learning. Through a number of mini-case studies, the chapter provides insight into how RL is applied to optimization problems in asset management and trading. These examples are each supported with Python notebooks.

Chapter 10

Chapter 10 considers real-world applications of reinforcement learning in finance, as well as further advances the theory presented in the previous chapter. We start with one of the most common problems of quantitative finance, which is the problem of optimal portfolio trading in discrete time. Many practical problems of trading or risk management amount to different forms of dynamic portfolio optimization, with different optimization criteria, portfolio composition, and constraints. The chapter introduces a reinforcement learning approach to option pricing that generalizes the classical Black–Scholes model to a data-driven approach using Q-learning. It then presents a probabilistic extension of Q-learning called G-learning and shows how it can be used for dynamic portfolio optimization. For certain specifications of reward functions, G-learning is semi-analytically tractable and amounts to a probabilistic version of linear quadratic regulators (LQRs). Detailed analyses of such cases are presented and we show their solutions with examples from problems of dynamic portfolio optimization and wealth management.

Chapter 11

Chapter 11 provides an overview of the most popular methods of inverse reinforcement learning (IRL) and imitation learning (IL). These methods solve the problem of optimal control in a data-driven way, similarly to reinforcement learning, however with the critical difference that now rewards are *not* observed. The problem is rather to learn the reward function from the observed behavior of an agent. As behavioral data without rewards are widely available, the problem of learning from such data is certainly very interesting. The chapter provides a moderate-level description of the most promising IRL methods, equips the reader with sufficient knowledge to understand and follow the current literature on IRL, and presents examples that use simple simulated environments to see how these methods perform when we know the “ground truth" rewards. We then present use cases for IRL in quantitative finance that include applications to trading strategy identification, sentiment-based trading, option pricing, inference of portfolio investors, and market modeling.

Chapter 12

Chapter 12 takes us forward to emerging research topics in quantitative finance and machine learning. Among many interesting emerging topics, we focus here on two broad themes. The first one deals with unification of supervised learning and reinforcement learning as two tasks of perception-action cycles of agents. We outline some recent research ideas in the literature including in particular information theory-based versions of reinforcement learning and discuss their relevance for financial applications. We explain why these ideas might have interesting practical implications for RL financial models, where feature selection could be done within the general task of optimization of a long-term objective, rather than outside of it, as is usually performed in “alpha-research.”

The second topic presented in this chapter deals with using methods of reinforcement learning to construct models of market dynamics. We also introduce some advanced physics-based approaches for computations for such RL-inspired market models.

Source Code

Many of the chapters are accompanied by Python notebooks to illustrate some of the main concepts and demonstrate application of machine learning methods. Each notebook is lightly annotated. Many of these notebooks use TensorFlow. We recommend loading these notebooks, together with any accompanying Python source files and data, in Google Colab. Please see the appendices of each chapter accompanied by notebooks, and the `README.md` in the subfolder of each chapter, for further instructions and details.

Scope

We recognize that the field of machine learning is developing rapidly and to keep abreast of the research in this field is a challenging pursuit. Machine learning is an umbrella term for a number of methodology classes, including supervised learning, unsupervised learning, and reinforcement learning. This book focuses on supervised learning and reinforcement learning because these are the areas with the most overlap with econometrics, predictive modeling, and optimal control in finance. Supervised machine learning can be categorized as generative and discriminative. Our focus is on discriminative learners which attempt to partition the input space, either directly, through affine transformations or through projections onto a manifold. Neural networks have been shown to provide a universal approximation to a wide class of functions. Moreover, they can be shown to reduce to other well-known statistical techniques and are adaptable to time series data.

Extending time series models, a number of chapters in this book are devoted to an introduction to reinforcement learning (RL) and inverse reinforcement learning (IRL) that deal with problems of optimal control of such time series and show how many classical financial problems such as portfolio optimization, option pricing, and wealth management can naturally be posed as problems for RL and IRL. We present simple RL methods that can be applied for these problems, as well as explain how neural networks can be used in these applications.

There are already several excellent textbooks covering other classical machine learning methods, and we instead choose to focus on how to cast machine learning into various financial modeling and decision frameworks. We emphasize that much of this material is not unique to neural networks, but comparisons of alternative supervised learning approaches, such as random forests, are beyond the scope of this book.

Multiple-Choice Questions

Multiple-choice questions are included after introducing a key concept. The correct answers to all questions are provided at the end of each chapter with selected, partial, explanations to some of the more challenging material.

Exercises

The exercises that appear at the end of every chapter form an important component of the book. Each exercise has been chosen to reinforce concepts explained in the text, to stimulate the application of machine learning in finance, and to gently bridge material in other chapters. It is graded according to difficulty ranging from (*), which denotes a simple exercise which might take a few minutes to complete, through to (***), which denotes a significantly more complex exercise. Unless specified otherwise, all equations referenced in each exercise correspond to those in the corresponding chapter.

Instructor Materials

The book is supplemented by a separate Instructor's Manual which provides worked solutions to the end of chapter questions. Full explanations for the solutions to the multiple-choice questions are also provided. The manual provides additional notes and example code solutions for some of the programming exercises in the later chapters.

Acknowledgements

This book is dedicated to the late Mark Davis (Imperial College) who was an inspiration in the field of mathematical finance and engineering, and formative in our careers. Peter Carr, Chair of the Department of Financial Engineering at NYU Tandon, has been instrumental in supporting the growth of the field of machine learning in finance. Through providing speaker engagements and machine learning instructorship positions in the MS in Algorithmic Finance Program, the authors have been able to write research papers and identify the key areas required by a text book. Miquel Alonso (AIFI), Agostino Capponi (Columbia), Rama Cont (Oxford), Kay Giesecke (Stanford), Ali Hirsa (Columbia), Sebastian Jaimungal (University of Toronto), Gary Kazantsev (Bloomberg), Morton Lane (UIUC), Jörg Osterrieder (ZHAW) have established various academic and joint academic-industry workshops

and community meetings to proliferate the field and serve as input for this book. At the same time, there has been growing support for the development of a book in London, where several SIAM/LMS workshops and practitioner special interest groups, such as the Thalesians, have identified a number of compelling financial applications. The material has grown from courses and invited lectures at NYU, UIUC, Illinois Tech, Imperial College and the 2019 Bootcamp on Machine Learning in Finance at the Fields Institute, Toronto.

Along the way, we have been fortunate to receive the support of Tomasz Bielecki (Illinois Tech), Igor Cialenco (Illinois Tech), Ali Hirsa (Columbia University), and Brian Peterson (DV Trading). Special thanks to research collaborators and colleagues Kay Giesecke (Stanford University), Diego Klabjan (NWU), Nick Polson (Chicago Booth), and Harvey Stein (Bloomberg), all of whom have shaped our understanding of the emerging field of machine learning in finance and the many practical challenges. We are indebted to Sri Krishnamurthy (QuantUniversity), Saeed Amen (Cuemacro), Tyler Ward (Google), and Nicole Königstein for their valuable input on this book. We acknowledge the support of a number of Illinois Tech graduate students who have contributed to the source code examples and exercises: Xiwen Jing, Bo Wang, and Siliang Xong. Special thanks to Swaminathan Sethuraman for his support of the code development, to Volod Chernat and George Gvishiani who provided support and code development for the course taught at NYU and Coursera. Finally, we would like to thank the students and especially the organisers of the MSc Finance and Mathematics course at Imperial College, where many of the ideas presented in this book have been tested: Damiano Brigo, Antoine (Jack) Jacquier, Mikko Pakkanen, and Rula Murtada. We would also like to thank Blanka Horvath for many useful suggestions.

Chicago, IL, USA — Matthew F. Dixon
Brooklyn, NY, USA — Igor Halperin
London, UK — Paul Bilokon
December 2019

Contents

About the Authors

Matthew F. Dixon is an Assistant Professor of Applied Math at the Illinois Institute of Technology. His research in computational methods for finance is funded by Intel. Matthew began his career in structured credit trading at Lehman Brothers in London before pursuing academics and consulting for financial institutions in quantitative trading and risk modeling. He holds a Ph.D. in Applied Mathematics from Imperial College (2007) and has held postdoctoral and visiting professor appointments at Stanford University and UC Davis, respectively. He has published over 20 peer-reviewed publications on machine learning and financial modeling, has been cited in Bloomberg Markets and the Financial Times as an AI in fintech expert, and is a frequently invited speaker in Silicon Valley and on Wall Street. He has published R packages, served as a Google Summer of Code mentor, and is the co-founder of the Thalesians Ltd.

Igor Halperin is a Research Professor in Financial Engineering at NYU and an AI Research Associate at Fidelity Investments. He was previously an Executive Director of Quantitative Research at JPMorgan for nearly 15 years. Igor holds a Ph.D. in Theoretical Physics from Tel Aviv University (1994). Prior to joining the financial industry, he held postdoctoral positions in theoretical physics at the Technion and the University of British Columbia.

Paul Bilokon is CEO and Founder of Thalesians Ltd. and an expert in electronic and algorithmic trading across multiple asset classes, having helped build such businesses at Deutsche Bank and Citigroup. Before focusing on electronic trading, Paul worked on derivatives and has served in quantitative roles at Nomura, Lehman Brothers, and Morgan Stanley. Paul has been educated at Christ Church College, Oxford, and Imperial College. Apart from mathematical and computational finance, his academic interests include machine learning and mathematical logic.

Part I
Machine Learning with Cross-Sectional Data

Chapter 1
Introduction

This chapter introduces the industry context for machine learning in finance, discussing the critical events that have shaped the finance industry's need for machine learning and the unique barriers to adoption. The finance industry has adopted machine learning to varying degrees of sophistication. How it has been adopted is heavily fragmented by the academic disciplines underpinning the applications. We view some key mathematical examples that demonstrate the nature of machine learning and how it is used in practice, with the focus on building intuition for more technical expositions in later chapters. In particular, we begin to address many finance practitioner's concerns that neural networks are a "black-box" by showing how they are related to existing well-established techniques such as linear regression, logistic regression, and autoregressive time series models. Such arguments are developed further in later chapters. This chapter also introduces reinforcement learning for finance and is followed by more in-depth case studies highlighting the design concepts and practical challenges of applying machine learning in practice.

1 Background

In 1955, John McCarthy, then a young Assistant Professor of Mathematics, at Dartmouth College in Hanover, New Hampshire, submitted a proposal with Marvin Minsky, Nathaniel Rochester, and Claude Shannon for the Dartmouth Summer Research Project on Artificial Intelligence (McCarthy et al. 1955). These organizers were joined in the summer of 1956 by Trenchard More, Oliver Selfridge, Herbert Simon, Ray Solomonoff, among others. The stated goal was ambitious:

"The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to

M. F. Dixon et al., *Machine Learning in Finance*,

https://doi.org/10.1007/978-3-030-41068-1_1

find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves." Thus the field of artificial intelligence, or AI, was born.

Since this time, AI has perpetually strived to outperform humans on various judgment tasks (Pinar Saygin et al. 2000). The most fundamental metric for this success is the Turing test—a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human (Turing 1995). In recent years, a pattern of success in AI has emerged—one in which machines outperform in the presence of a large number of decision variables, usually with the best solution being found through evaluating an exponential number of candidates in a constrained high-dimensional space. Deep learning models, in particular, have proven remarkably successful in a wide field of applications (DeepMind 2016; Kubota 2017; Esteva et al. 2017) including image processing (Simonyan and Zisserman 2014), learning in games (DeepMind 2017), neuroscience (Poggio 2016), energy conservation (DeepMind 2016), skin cancer diagnostics (Kubota 2017; Esteva et al. 2017).

One popular account of this reasoning points to humans' perceived inability to process large amounts of information and make decisions beyond a few key variables. But this view, even if fractionally representative of the field, does no justice to AI or human learning. Humans are not being replaced any time soon. The median estimate for human intelligence in terms of gigaflops is about 10^4 times more than the machine that ran alpha-go. Of course, this figure is caveated on the important question of whether the human mind is even a Turing machine.

1.1 Big Data—Big Compute in Finance

The growth of machine-readable data to record and communicate activities throughout the financial system combined with persistent growth in computing power and storage capacity has significant implications for every corner of financial modeling. Since the financial crises of 2007–2008, regulatory supervisors have reoriented towards "data-driven" regulation, a prominent example of which is the collection and analysis of detailed contractual terms for the bank loan and trading book stress-testing programs in the USA and Europe, instigated by the crisis (Flood et al. 2016).

"Alternative data"—which refers to data and information outside of the usual scope of securities pricing, company fundamentals, or macroeconomic indicators—is playing an increasingly important role for asset managers, traders, and decision makers. Social media is now ranked as one of the top categories of alternative data currently used by hedge funds. Trading firms are hiring experts in machine learning with the ability to apply natural language processing (NLP) to financial news and other unstructured documents such as earnings announcement reports and SEC 10K reports. Data vendors such as Bloomberg, Thomson Reuters, and RavenPack are providing processed news sentiment data tailored for systematic trading models.

In de Prado (2019), some of the properties of these new, alternative datasets are explored: (a) many of these datasets are unstructured, non-numerical, and/or non-categorical, like news articles, voice recordings, or satellite images; (b) they tend to be high-dimensional (e.g., credit card transactions) and the number of variables may greatly exceed the number of observations; (c) such datasets are often sparse, containing NaNs (not-a-numbers); (d) they may implicitly contain information about networks of agents.

Furthermore, de Prado (2019) explains why classical econometric methods fail on such datasets. These methods are often based on linear algebra, which fail when the number of variables exceeds the number of observations. Geometric objects, such as covariance matrices, fail to recognize the topological relationships that characterize networks. On the other hand, machine learning techniques offer the numerical power and functional flexibility needed to identify complex patterns in a high-dimensional space offering a significant improvement over econometric methods.

The "black-box" view of ML is dismissed in de Prado (2019) as a misconception. Recent advances in ML make it applicable to the evaluation of plausibility of scientific theories; determination of the relative informational variables (usually referred to as features in ML) for explanatory and/or predictive purposes; causal inference; and visualization of large, high-dimensional, complex datasets.

Advances in ML remedy the shortcomings of econometric methods in goal setting, outlier detection, feature extraction, regression, and classification when it comes to modern, complex alternative datasets. For example, in the presence of p features there may be up to $2^p - p - 1$ multiplicative interaction effects. For two features there is only one such interaction effect, x_1x_2. For three features, there are $x_1x_2, x_1x_3, x_2x_3, x_1x_2x_3$. For as few as ten features, there are 1,013 multiplicative interaction effects. Unlike ML algorithms, econometric models do not "learn" the structure of the data. The model specification may easily miss some of the interaction effects. The consequences of missing an interaction effect, e.g. fitting $y_t = x_{1,t} + x_{2,t} + \epsilon_t$ instead of $y_t = x_{1,t} + x_{2,t} + x_{1,t}x_{2,t} + \epsilon_t$, can be dramatic. A machine learning algorithm, such as a decision tree, will recursively partition a dataset with complex patterns into subsets with simple patterns, which can then be fit independently with simple linear specifications. Unlike the classical linear regression, this algorithm "learns" about the existence of the $x_{1,t}x_{2,t}$ effect, yielding much better out-of-sample results.

There is a draw towards more empirically driven modeling in asset pricing research—using ever richer sets of firm characteristics and "factors" to describe and understand differences in expected returns across assets and model the dynamics of the aggregate market equity risk premium (Gu et al. 2018). For example, Harvey et al. (2016) study 316 "factors," which include firm characteristics and common factors, for describing stock return behavior. Measurement of an asset's risk premium is fundamentally a problem of prediction—the risk premium is the conditional expectation of a future realized excess return. Methodologies that can reliably attribute excess returns to tradable anomalies are highly prized. Machine learning provides a non-linear empirical approach for modeling realized security returns from firm characteristics. Dixon and Polson (2019) review the formulation of asset pricing models for measuring asset risk premia and cast neural networks in canonical asset pricing frameworks.

1.2 Fintech

The rise of data and machine learning has led to a "fintech" industry, covering digital innovations and technology-enabled business model innovations in the financial sector (Philippon 2016). Examples of innovations that are central to fintech today include cryptocurrencies and the blockchain, new digital advisory and trading systems, peer-to-peer lending, equity crowdfunding, and mobile payment systems. Behavioral prediction is often a critical aspect of product design and risk management needed for consumer-facing business models; consumers or economic agents are presented with well-defined choices but have unknown economic needs and limitations, and in many cases do not behave in a strictly economically rational fashion. Therefore it is necessary to treat parts of the system as a black-box that operates under rules that cannot be known in advance.

1.2.1 Robo-Advisors

Robo-advisors are financial advisors that provide financial advice or portfolio management services with minimal human intervention. The focus has been on portfolio management rather than on estate and retirement planning, although there are exceptions, such as Blooom. Some limit investors to the ETFs selected by the service, others are more flexible. Examples include Betterment, Wealthfront, Wise-Banyan, FutureAdvisor (working with Fidelity and TD Ameritrade), Blooom, Motif Investing, and Personal Capital. The degree of sophistication and the utilization of machine learning are on the rise among robo-advisors.

1.2.2 Fraud Detection

In 2011 fraud cost the financial industry approximately $80 billion annually (Consumer Reports, June 2011). According to PwC's Global Economic Crime Survey 2016, 46% of respondents in the Financial Services industry reported being victims of economic crime in the last 24 months—a small increase from 45% reported in 2014. 16% of those that reported experiencing economic crime had suffered more than 100 incidents, with 6% suffering more than 1,000. According to the survey, the top 5 types of economic crime are asset misappropriation (60%, down from 67% in 2014), cybercrime (49%, up from 39% in 2014), bribery and corruption (18%, down from 20% in 2014), money laundering (24%, as in 2014), and accounting fraud (18%, down from 21% in 2014). Detecting economic crimes is one of the oldest successful applications of machine learning in the financial services industry. See Gottlieb et al. (2006) for a straightforward overview of some of the classical methods: logistic regression, naïve Bayes, and support vector machines. The rise of electronic trading has led to new kinds of financial fraud and market manipulation. Some exchanges are investigating the use of deep learning to counter spoofing.

1.2.3 Cryptocurrencies

Blockchain technology, first implemented by Satoshi Nakamoto in 2009 as a core component of Bitcoin, is a distributed public ledger recording transactions. Its usage allows secure peer-to-peer communication by linking blocks containing hash pointers to a previous block, a timestamp, and transaction data. Bitcoin is a decentralized digital currency (cryptocurrency) which leverages the blockchain to store transactions in a distributed manner in order to mitigate against flaws in the financial industry.

In contrast to existing financial networks, blockchain based cryptocurrencies expose the entire transaction graph to the public. This openness allows, for example, the most significant agents to be immediately located (pseudonymously) on the network. By processing all financial interactions, we can model the network with a high-fidelity graph, as illustrated in Fig. 1.1 so that it is possible to characterize how the flow of information in the network evolves over time. This novel data representation permits a new form of financial econometrics—with the emphasis on the topological network structures in the microstructure rather than solely the covariance of historical time series of prices. The role of users, entities, and their interactions in formation and dynamics of cryptocurrency risk investment, financial predictive analytics and, more generally, in re-shaping the modern financial world is a novel area of research (Dyhrberg 2016; Gomber et al. 2017; Sovbetov 2018).

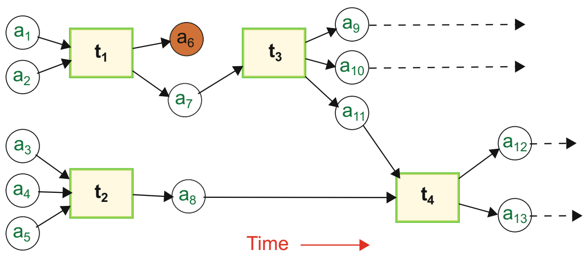

Fig. 1.1 A transaction–address graph representation of the Bitcoin network. Addresses are represented by circles, transactions with rectangles, and edges indicate a transfer of coins. Blocks order transactions in time, whereas each transaction with its input and output nodes represents an immutable decision that is encoded as a subgraph on the Bitcoin network. Source: Akcora et al. (2018)

2 Machine Learning and Prediction

With each passing year, finance becomes increasingly reliant on computational methods. At the same time, the growth of machine-readable data to monitor, record, and communicate activities throughout the financial system has significant implications for how we approach the topic of modeling. One of the reasons that AI and the set of computer algorithms for learning, referred to as "machine learning," have been successful is a result of a number of factors beyond computer hardware and software advances. Machines are able to model complex and high-dimensional data generation processes, sweep through millions of model configurations, and then robustly evaluate and correct the models in response to new information (Dhar 2013). By continuously updating and hosting a number of competing models, they prevent any one model leading us into a data gathering silo effective only for that market view. Structurally, the adoption of ML has even shifted our behavior—the way we reason, experiment, and shape our perspectives from data using ML has led to empirically driven trading and investment decision processes.

Machine learning is a broad area, covering various classes of algorithms for pattern recognition and decision-making. In **supervised learning**, we are given labeled data, i.e. pairs $(x_1, y_1), \ldots, (x_n, y_n)$, $x_1, \ldots, x_n \in X$, $y_1, \ldots, y_n \in Y$, and the goal is to learn the relationship between X and Y. Each observation x_i is referred to as a **feature vector** and y_i is the **label** or **response**. In **unsupervised learning**, we are given unlabeled data, $x_1, x_2, \ldots, x_n$ and our goal is to retrieve exploratory information about the data, perhaps grouping similar observations or capturing some hidden patterns. Unsupervised learning includes **cluster analysis** algorithms such as hierarchical clustering, k-means clustering, self-organizing maps, Gaussian mixture, and hidden Markov models and is commonly referred to as data mining. In both instances, the data could be financial time series, news documents, SEC documents, and textual information on important events. The third type of machine learning paradigm is **reinforcement learning** and is an algorithmic approach for enforcing Bellman optimality of a Markov Decision Process—defining a set of states and actions in response to a changing regime so as to maximize some notion of cumulative reward. In contrast to supervised learning, which just considers a single action at each point in time, reinforcement learning is concerned with the optimal sequence of actions. It is therefore a form of dynamic programming that is used for decisions leading to optimal trade execution, portfolio allocation, and liquidation over a given horizon.

Supervised learning addresses a fundamental prediction problem: Construct a non-linear predictor, $\hat{Y}(X)$, of an output, Y, given a high-dimensional input matrix $X = (X_1, \ldots, X_P)$ of P variables. Machine learning can be simply viewed as the study and construction of an input–output map of the form

$$Y = F(X) \text{ where } X = (X_1, \ldots, X_P).$$

$F(X)$ is sometimes referred to as the "data-feature" map. The output variable, Y, can be continuous, discrete, or mixed. For example, in a classification problem,

$F : X \to G$, where $G \in \mathcal{K} := \{0, \ldots, K-1\}$, K is the number of categories and $\hat{G}$ is the predictor.

Supervised machine learning uses a parameterized[1] model $g(X|\boldsymbol{\theta})$ over independent variables X, to predict the continuous or categorical output Y or G. The model is parameterized by one or more free parameters θ which are fitted to data. Prediction of categorical variables is referred to as *classification* and is common in pattern recognition. The most common approach to predicting categorical variables is to encode the response G as one or more binary values, then treat the model prediction as continuous.

? Multiple Choice Question 1

Select all the following correct statements:

1. Supervised learning involves learning the relationship between input and output variables.
2. Supervised learning requires a human supervisor to prepare labeled training data.
3. Unsupervised learning does not require a human supervisor and is therefore superior to supervised learning.
4. Reinforcement learning can be viewed as a generalization of supervised learning to Markov Decision Processes.

There are two different classes of supervised learning models, *discriminative* and *generative*. A discriminative model learns the decision boundary between the classes and implicitly learns the distribution of the output conditional on the input. A generative model explicitly learns the joint distribution of the input and output. An example of the former is a neural network or a decision tree and a restricted Boltzmann machine (RBM) is an example of the latter. Learning the joint distribution has the advantage that by the Bayes' rule, it can also give the conditional distribution of the output given the input, but also be used for other purposes such as selecting features based on the joint probability. Generative models are typically more difficult to build.

This book will mostly focus on discriminative models only, but the distinction should be made clear. A discriminative model predicts the probability of an output given an input. For example, if we are predicting the probability of a label $G = k, k \in \mathcal{K}$, then $g(\boldsymbol{x}|\boldsymbol{\theta})$ is a map $g : \mathbb{R}^p \to [0, 1]^K$ and the outputs represent a discrete probability distribution over G referred to as a "one-hot" encoding—a K-vector of zeros with 1 at the kth position:

$$\hat{G}_k := \mathbb{P}(G = k \mid X = \boldsymbol{x}, \theta) = g_k(\boldsymbol{x}|\boldsymbol{\theta}) \tag{1.1}$$

[1]The model is referred to as *non-parametric* if the parameter space is infinite dimensional and *parametric* if the parameter space is finite dimensional.

and hence we have that

$$\sum_{k \in \mathcal{K}} g_k(\boldsymbol{x}|\boldsymbol{\theta}) = 1. \tag{1.2}$$

In particular, when G is dichotomous ($K = 2$), the second component of the model output is the conditional expected value of G

$$\hat{G} := \hat{G}_1 = g_1(\boldsymbol{x}|\boldsymbol{\theta}) = 0 \cdot \mathbb{P}(G = 0 \mid X = \boldsymbol{x}, \theta) + 1 \cdot \mathbb{P}(G = 1 \mid X = \boldsymbol{x}, \theta) = \mathbb{E}[G \mid X = \boldsymbol{x}, \theta]. \tag{1.3}$$

The conditional variance of G is given by

$$\sigma^2 := \mathbb{E}[(G - \hat{G})^2 \mid X = \boldsymbol{x}, \theta] = g_1(\boldsymbol{x}|\boldsymbol{\theta}) - (g_1(\boldsymbol{x}|\boldsymbol{\theta}))^2, \tag{1.4}$$

which is an inverted parabola with a maximum at $g_1(\boldsymbol{x}|\boldsymbol{\theta}) = 0.5$. The following example illustrates a simple discriminative model which, here, is just based on a set of fixed rules for partitioning the input space.

Example 1.1 Model Selection

Suppose $G \in \{A, B, C\}$ and the input $X \in \{0, 1\}^2$ are binary 2-vectors given in Table 1.1.

Table 1.1 Sample model data

G	$\boldsymbol{x}$
A	(0, 1)
B	(1, 1)
C	(1, 0)
C	(0, 0)

To match the input and output in this case, one could define a parameter-free step function $g(\boldsymbol{x})$ over $\{0, 1\}^2$ so that

$$g(\boldsymbol{x}) = \begin{cases} \{1, 0, 0\} & \text{if } \boldsymbol{x} = (0, 1) \\ \{0, 1, 0\} & \text{if } \boldsymbol{x} = (1, 1) \\ \{0, 0, 1\} & \text{if } \boldsymbol{x} = (1, 0) \\ \{0, 0, 1\} & \text{if } \boldsymbol{x} = (0, 0). \end{cases} \tag{1.5}$$

The discriminative model $g(\boldsymbol{x})$, defined in Eq. 1.5, specifies a set of fixed rules which predict the outcome of this experiment with 100% accuracy. Intuitively, it seems clear that such a model is flawed if the actual relation between inputs and outputs is non-deterministic. Clearly, a skilled analyst would typically not build such a model. Yet, hard-wired rules such as this are ubiquitous in the finance industry such as rule-based technical analysis and heuristics used for scoring such as credit ratings.

If the model is allowed to be general, there is no reason why this particular function should be excluded. Therefore automated systems analyzing datasets such as this may be prone to construct functions like those given in Eq. 1.5 unless measures are taken to prevent it. It is therefore incumbent on the model designer to understand what makes the rules in Eq. 1.5 objectionable, with the goal of using a theoretically sound process to generalize the input–output map to other data.

Example 1.2 Model Selection (Continued)

Consider an alternate model for Table 1.1

$$h(\boldsymbol{x}) = \begin{cases} \{0.9, 0.05, 0.05\} & \text{if } \boldsymbol{x} = (0, 1) \\ \{0.05, 0.9, 0.05\} & \text{if } \boldsymbol{x} = (1, 1) \\ \{0.05, 0.05, 0.9\} & \text{if } \boldsymbol{x} = (1, 0) \\ \{0.05, 0.05, 0.9\} & \text{if } \boldsymbol{x} = (0, 0). \end{cases}$$

If this model were sampled, it would produce the data in Table 1.1 with probability $(0.9)^4 = 0.6561$. We can hardly exclude this model from consideration on the basis of the results in Table 1.1, so which one do we choose?

Informally, the heart of the model selection problem is that model g has excessively high confidence about the data, when that confidence is often not warranted. Many other functions, such as h, could have easily generated Table 1.1. Though there is only one model that can produce Table 1.1 with probability 1.0, there is a whole family of models that can produce the table with probability at least 0.66. Many of these plausible models do not assign overwhelming confidence to the results. To determine which model is best on average, we need to introduce another key concept.

2.1 Entropy

Model selection in machine learning is based on a quantity known as **entropy**. Entropy represents the amount of information associated with each event. To illustrate the concept of entropy, let us consider a non-fair coin toss. There are two outcomes, $\Omega = \{H, T\}$. Let Y be a Bernoulli random variable representing the coin flip with density $f(Y = 1) = \mathbb{P}(H) = p$ and $f(Y = 0) = \mathbb{P}(T) = 1 - p$. The (binary) entropy of Y under f is

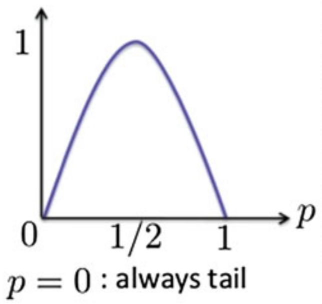

Fig. 1.2 (Left) This figure shows the binary entropy of a biased coin. If the coin is fully biased, then each flip provides no new information as the outcome is already known and hence the entropy is zero. (Right) The concept of entropy was introduced by Claude Shannon[2] in 1948[3] and was originally intended to represent an upper limit on the average length of a lossless compression encoding. Shannon's entropy is foundational to the mathematical discipline of information theory

$$\mathcal{H}(f) = -p\log_2 p - (1-p)log_2(1-p) \leq 1\text{bit}. \tag{1.6}$$

The reason why base 2 is chosen is so that the upper bound represents the number of bits needed to represent the outcome of the random variable, i.e. $\{0, 1\}$ and hence 1 bit.

The binary entropy for a biased coin is shown in Fig. 1.2. If the coin is fully biased, then each flip provides no new information as the outcome is already known. The maximum amount of information that can be revealed by a coin flip is when the coin is unbiased.

Let us now reintroduce our parameterized mass in the setting of the biased coin. Let us consider an i.i.d. discrete random variable $Y : \Omega \to \mathcal{Y} \subset \mathbb{R}$ and let

$$g(y|\theta) = \mathbb{P}(\omega \in \Omega; Y(\omega) = y)$$

denote a parameterized probability mass function for Y.

We can measure how different $g(y|\theta)$ is from the true density $f(y)$ using the cross-entropy

$$\mathcal{H}(f, g) := -\mathbb{E}_f\left[\log_2 g\right] = \sum_{y\in\mathcal{Y}} f(y)\log_2 g(y|\theta) \geq \mathcal{H}(f), \tag{1.7}$$

[2]Photo: Jacobs, Konrad [CC BY-SA 2.0 de (https://creativecommons.org/licenses/by-sa/2.0/de/deed.en)].

[3]C. Shannon, A Mathematical Theory of Communication, The Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, July, October, 1948.

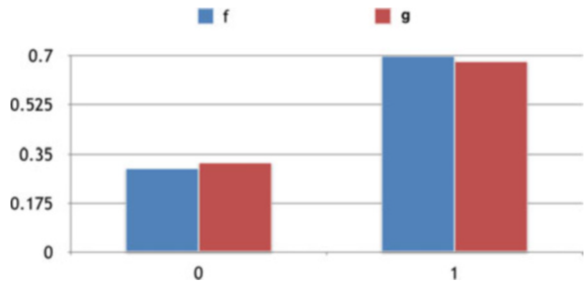

Fig. 1.3 A comparison of the true distribution, f, of a biased coin with a parameterized model g of the coin

so that $\mathcal{H}(f, f) = \mathcal{H}(f)$, where $\mathcal{H}(f)$ is the entropy of f:

$$\mathcal{H}(f) := -\mathbb{E}_f[log_2 f] = -\sum_{y \in \mathcal{Y}} f(y) log_2 f(y). \tag{1.8}$$

If $g(y|\theta)$ is a model of the non-fair coin with $g(Y = 1|\theta) = p_\theta$, $g(Y = 0|\theta) = 1 - p_\theta$. The cross-entropy is

$$\mathcal{H}(f, g) = -p \log_2 p_\theta - (1 - p) log_2 (1 - p_\theta) \geq -p \log_2 p - (1 - p) log_2 (1 - p). \tag{1.9}$$

Let us suppose that $p = 0.7$ and $p_\theta = 0.68$, as illustrated in Fig. 1.3, then the cross-entropy is

$$\mathcal{H}(f, g) = -0.3 \log_2(0.32) - 0.7 \log_2(0.68) = 0.8826322.$$

Returning to our experiment in Table 1.1, let us consider the cross-entropy of these models which, as you will recall, depends on inputs too. Model g completely characterizes the data in Table 1.1 and we interpret it here as the truth. Model h, however, only summarizes some salient aspects of the data, and there is a large family of tables that would be consistent with model h. In the presence of noise or strong evidence indicating that Table 1.1 was the only possible outcome, we should interpret models like h as a more plausible explanation of the actual underlying phenomenon.

Evaluating the cross-entropy between model h and model g, we get $-\log_2(0.9)$ for each observation in the table, which gives the negative log-likelihood when summed over all samples. The cross-entropy is at its minimum when $h = g$, we get $-\log_2(1.0) = 0$. If g were a parameterized model, then clearly minimizing cross-entropy or equivalently maximizing log-likelihood gives the maximum likelihood estimate of the parameter. We shall revisit the topic of parameter estimation in Chap. 2.

? Multiple Choice Question 2

Select all of the following statements that are correct:

1. Neural network classifiers are a discriminative model which output probabilistic weightings for each category, given an input feature vector.
2. If the data is independent and identically distributed (i.i.d.), then the output of a dichotomous classifier is a conditional probability of a Bernoulli random variable.
3. A θ-parameterized discriminative model for a biased coin dependent on the environment X can be written as $\{g_i(X|\theta)\}_{i=0}^1$.
4. A model of two biased coins, both dependent on the environment X, can be equivalently modeled with either the pair $\{g_i^{(1)}(X|\theta)\}_{i=0}^1$ and $\{g_i^{(2)}(X|\theta)\}_{i=0}^1$, or the multi-classifier $\{g_i(X|\theta)\}_{i=0}^3$.

2.2 *Neural Networks*

Neural networks represent the non-linear map $F(X)$ over a high-dimensional input space using hierarchical layers of abstractions. An example of a neural network is a feedforward network—a sequence of L layers[4] formed via composition:

> Deep Feedforward Networks

A deep feedforward network is a function of the form

$$\hat{Y}(X) := F_{W,b}(X) = \left(f^{(L)}_{W^{(L)},b^{(L)}} \dots \circ f^{(1)}_{W^{(1)},b^{(1)}}\right)(X),$$

where

- $f^{(l)}_{W^{(l)},b^{(l)}}(X) := \sigma^{(l)}(W^{(l)}X + b^{(l)})$ is a semi-affine function, where $\sigma^{(l)}$ is a univariate and continuous non-linear activation function such as $max(\cdot, 0)$ or $tanh(\cdot)$.
- $W = (W^{(1)}, \dots, W^{(L)})$ and $b = (b^{(1)}, \dots, b^{(L)})$ are weight matrices and offsets (a.k.a. biases), respectively.

[4]Note that we do not treat the input as a layer. So there are $L - 1$ hidden layers and an output layer.

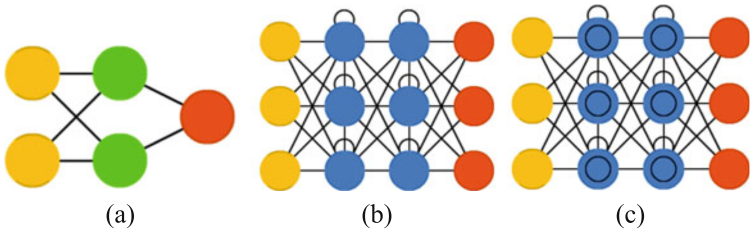

Fig. 1.4 Examples of neural networks architectures discussed in this book. Source: Van Veen, F. & Leijnen, S. (2019), "The Neural Network Zoo," Retrieved from https://www.asimovinstitute.org/neural-network-zoo. The input nodes are shown in yellow and represent the input variables, the green nodes are the hidden neurons and present hidden latent variables, the red nodes are the outputs or responses. Blue nodes denote hidden nodes with recurrence or memory. (**a**) Feedforward. (**b**) Recurrent. (**c**) Long short-term memory

An earlier example of a feedforward network architecture is given in Fig. 1.4a. The input nodes are shown in yellow and represent the input variables, the green nodes are the hidden neurons and present hidden latent variables, the red nodes are the outputs or responses. The activation functions are essential for the network to approximate non-linear functions. For example, if there is one hidden layer and $\sigma^{(1)}$ is the identify function, then

$$\hat{Y}(X) = W^{(2)}(W^{(1)}X + b^{(1)}) + b^{(2)} = W^{(2)}W^{(1)}X + W^{(2)}b^{(1)} + b^{(2)} = W'X + b' \tag{1.10}$$

is just linear regression, i.e. an affine transformation.[5] Clearly, if there are no hidden layers, the architecture recovers standard linear regression

$$Y = WX + b$$

and logistic regression $\phi(WX + b)$, where ϕ is a sigmoid or softmax function, when the response is continuous or categorical, respectively. Some of the terminology used here and the details of this model will be described in Chap. 4.

The theoretical roots of feedforward neural networks are given by the Kolmogorov–Arnold representation theorem (Arnold 1957; Kolmogorov 1957) of multivariate functions. Remarkably, Hornik et al. (1989) showed how neural networks, with one hidden layer, are universal approximators to non-linear functions.

Clearly there are a number of issues in any architecture design and inference of the model parameters (W, b). How many layers? How many neurons N_l in each hidden layer? How to perform "variable selection"? How to avoid over-fitting? The details and considerations given to these important questions will be addressed in Chap. 4.

[5] While the functional form of the map is the same as linear regression, neural networks do not assume a data generation process and hence inference is not identical to ordinary least squares regression.

3 Statistical Modeling vs. Machine Learning

Supervised machine learning is often an algorithmic form of statistical model estimation in which the data generation process is treated as an unknown (Breiman 2001). Model selection and inference is automated, with an emphasis on processing large amounts of data to develop robust models. It can be viewed as a highly efficient data compression technique designed to provide predictors in complex settings where relations between input and output variables are non-linear and input space is often high-dimensional. Machine learners balance filtering data with the goal of making accurate and robust decisions, often discrete and as a categorical function of input data.

This fundamentally differs from maximum likelihood estimators used in standard statistical models, which assume that the data was generated by the model and typically have difficulty with over-fitting, especially when applied to high-dimensional datasets. Given the complexity of modern datasets, whether they are limit order books or high-dimensional financial time series, it is increasingly questionable whether we can posit inference on the basis of a known data generation process. It is a reasonable assertion, even if an economic interpretation of the data generation process can be given, that the exact form cannot be known all the time.

The paradigm that machine learning provides for data analysis therefore is very different from the traditional statistical modeling and testing framework. Traditional fit metrics, such as R^2, t-values, p-values, and the notion of *statistical significance*, are replaced by out-of-sample forecasting and understanding the bias–variance tradeoff. Machine learning is data-driven and focuses on finding structure in large datasets. The main tools for variable or predictor selection are *regularization* and *dropout* which are discussed in detail in Chap. 4.

Table 1.2 contrasts maximum likelihood estimation-based inference with supervised machine learning. The comparison is somewhat exaggerated for ease of explanation. Rather the two approaches should be viewed as opposite ends of a continuum of methods. Linear regression techniques such as LASSO and ridge regression, or hybrids such as Elastic Net, fall somewhere in the middle, providing some combination of the explanatory power of maximum likelihood estimation while retaining out-of-sample predictive performance on high-dimensional datasets.

3.1 Modeling Paradigms

Machine learning and statistical methods can be further characterized by whether they are parametric or non-parametric. *Parametric models* assume some finite set of parameters and attempt to model the response as a function of the input variables and the parameters. Due to the finiteness of the parameter space, they have limited flexibility and cannot capture complex patterns in big data. As a general rule, examples of parametric models include ordinary least squares linear

Table 1.2 This table contrasts maximum likelihood estimation-based inference with supervised machine learning. The comparison is somewhat exaggerated for ease of explanation; however, the two should be viewed as opposite ends of a continuum of methods. Regularized linear regression techniques such as LASSO and ridge regression, or hybrids such as Elastic Net, provide some combination of the explanatory power of maximum likelihood estimation while retaining out-of-sample predictive performance on high-dimensional datasets

Property	Statistical inference	Supervised machine learning
Goal	Causal models with explanatory power	Prediction performance, often with limited explanatory power
Data	The data is generated by a model	The data generation process is unknown
Framework	Probabilistic	Algorithmic and Probabilistic
Expressibility	Typically linear	Non-linear
Model selection	Based on information criteria	Numerical optimization
Scalability	Limited to lower-dimensional data	Scales to high-dimensional input data
Robustness	Prone to over-fitting	Designed for out-of-sample performance
Diagnostics	Extensive	Limited

regression, polynomial regression, mixture models, neural networks, and hidden Markov models.

Non-parametric models treat the parameter space as infinite dimensional—this is equivalent to introducing a hidden or latent function. The model structure is, for the most part, not specified a priori and they can grow in complexity with more data. Examples of non-parametric models include kernel methods such as support vector machines and Gaussian processes, the latter will be the focus of Chap. 3.

Note that there is a gray area in whether neural networks are parametric or non-parametric and it strictly depends on how they are fitted. For example, it is possible to treat the parameter space in a neural network as infinite dimensional and hence characterize neural networks as non-parametric (see, for example, Philipp and Carbonell (2017)). However, this is an exception rather than the norm.

While on the topic of modeling paradigms, it is helpful to further distinguish between *probabilistic models*, the subject of the next two chapters, and deterministic models, the subject of Chaps. 4, 5, and 8. The former treats the parameters as random and the latter assumes that the parameters are given.

Within probabilistic modeling, a particular niche is occupied by the so-called *state-space models*. In these models one assumes the existence of a certain unobserved, latent, process, whose evolution drives a certain observable process. The evolution of the latent process and the dependence of the observable process on the latent process may be given in stochastic, probabilistic terms, which places the state-space models within the realm of probabilistic modeling.

Note, somewhat counter to the terminology, that a deterministic model may produce a probabilistic output, for example, a logistic regression gives the probability that the response is positive given the input variables. The choice of whether to use a probabilistic or deterministic model is discussed further in the next chapter and falls under the more general and divisive topic of "Bayesian versus frequentist modeling."

3.2 Financial Econometrics and Machine Learning

Machine learning generalizes parametric methods in financial econometrics. A taxonomy of machine learning in econometrics in shown in Fig. 1.5 together with the section references to the material in the first two parts of this book.

When the data is a time series, neural networks can be configured with recurrence to build memory into the model. By relaxing the modeling assumptions needed for econometrics techniques, such as ARIMA (Box et al. 1994) and GARCH models (Bollerslev 1986), recurrent neural networks provide a semi-parametric or even non-parametric extension of classical time series methods. That use, however, comes with much caution. Whereas financial econometrics is built on rigorous experimental design methods such as the estimation framework of Box and Jenkins (1976), recurrent neural networks have grown from the computational engineering literature and many engineering studies overlook essential diagnostics such as Dickey–Fuller tests for verifying stationarity of the time series, a critical aspect of financial time series modeling. We take an integrative approach, showing how to cast recurrent neural networks into financial econometrics frameworks such as Box-Jenkins.

More formally, if the input–output pairs $\mathcal{D} = \{X_t, Y_t\}_{t=1}^N$ are autocorrelated observations of X and Y at times $t = 1, \ldots, N$, then the fundamental prediction problem can be expressed as a sequence prediction problem: construct a non-linear

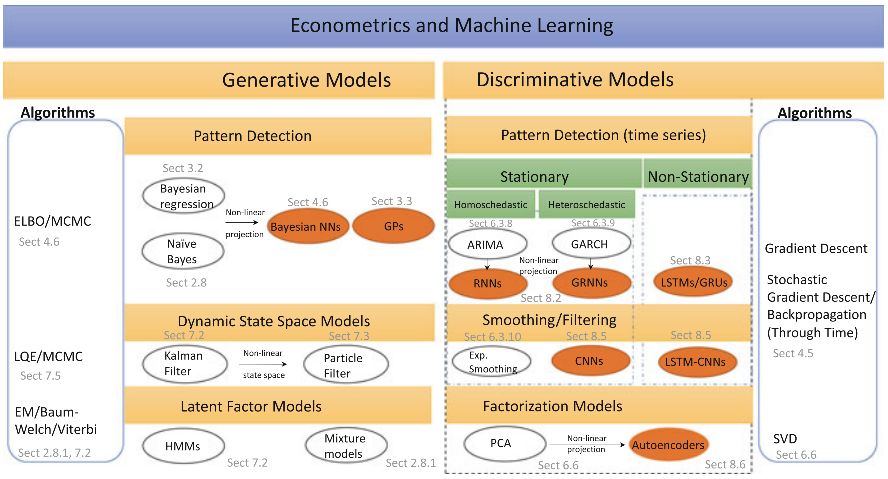

Fig. 1.5 Overview of how machine learning generalizes parametric econometrics, together with the section references to the material in the first two parts of this book

times series predictor, $\hat{Y}(\mathcal{X}_t)$, of an output, Y, using a high-dimensional input matrix of T length sub-sequences $\mathcal{X}_t$:

$$\hat{Y}_t = F(\mathcal{X}_t) \text{ where } \mathcal{X}_t := seq_{T,0}(X_t) := (X_{t-T+1}, \dots, X_t), \tag{1.11}$$

where X_{t-j} is a jth lagged observation of X_t, $X_{t-j} = L^j[X_j]$, for $0 = 1, \dots, T-1$. Sequence learning, then, is just a composition of a non-linear map and a vectorization of the lagged input variables. If the data is i.i.d., then no sequence is needed (i.e., $T = 1$), and we recover the standard cross-sectional prediction problem which can be approximated with a feedforward neural network model.

Recurrent neural networks (RNNs), shown in Fig. 1.4b, are time series methods or sequence learners which have achieved much success in applications such as natural language understanding, language generation, video processing, and many other tasks (Graves 2012). There are many types of RNNs—we will just concentrate on simple RNN models for brevity of notation. Like multivariate structural autoregressive models, RNNs apply an autoregressive function $f^{(1)}_{W^{(1)},b^{(1)}}(\mathcal{X}_t)$ to each input sequence $\mathcal{X}_t$, where T denotes the look back period at each time step—the maximum number of lags. However, rather than directly imposing a linear autocovariance structure, a RNN provides a flexible functional form to directly model the predictor, $\hat{Y}$.

A simple RNN can be understood as an unfolding of a single hidden layer neural network (a.k.a. Elman network (Elman 1991)) over all time steps in the sequence, $j = 0, \dots, T$. For each time step, j, this function $f^{(1)}_{W^{(1)},b^{(1)}}(\mathcal{X}_{t,j})$ generates a hidden state Z_{t-j} from the current input X_{t-j} and the previous hidden state Z_{t-j-1} and $\mathcal{X}_{t,j} = seq_{T,j}(X_t) \subset \mathcal{X}_t$ which appears in general form as:

$$\text{response:} \quad \hat{Y}_t = f^{(2)}_{W^{(2)},b^{(2)}}(Z_t) := \sigma^{(2)}(W^{(2)}Z_t + b^{(2)}),$$

$$\begin{aligned}\text{hidden states: } Z_{t-j} &= f^{(1)}_{W^{(1)},b^{(1)}}(\mathcal{X}_{t,j}) \\ &:= \sigma^{(1)}(W_z^{(1)}Z_{t-j-1} + W_x^{(1)}X_{t-j} + b^{(1)}), \ j \in \{T, \dots, 0\},\end{aligned}$$

where $\sigma^{(1)}$ is an activation function such as $\tanh(x)$ and $\sigma^{(2)}$ is either a softmax function, sigmoid function, or identity function depending on whether the response is categorical, binary, or continuous, respectively. The connections between the extremal inputs X_t and the H hidden units are weighted by the time invariant matrix $W_x^{(1)} \in \mathbb{R}^{H \times P}$. The recurrent connections between the H hidden units are weighted by the time invariant matrix $W_z^{(1)} \in \mathbb{R}^{H \times H}$. Without such a matrix, the architecture is simply a single layered feedforward network without memory—each independent observation X_t is mapped to an output $\hat{Y}_t$ using the same hidden layer.

It is important to note that a plain RNN, de-facto, is not a deep network. The recurrent layer has the deceptive appearance of being a deep network when "unfolded," i.e. viewed as being repeatedly applied to each new input, X_{t-j}, so that $Z_{t-j} = \sigma^{(1)}(W_z^{(1)}Z_{t-j-1} + W_x^{(1)}X_{t-j})$. However the same recurrent weights remain fixed over all repetitions—there is only one recurrent layer with weights $W_z^{(1)}$.

The amount of memory in the model is equal to the sequence length T. This means that the maximum lagged input that affects the output, $\hat{Y}_t$, is X_{t-T}. We shall see later in Chap. 8 that RNNs are simply non-linear autoregressive models with exogenous variables (NARX). In the special case of the univariate time series prediction $\hat{X}_t = F(\mathcal{X}_{t-1})$, using $T = p$ previous observations $\{X_{t-i}\}_{i=1}^p$, only one neuron in the recurrent layer with weight ϕ and no activation function, a RNN is an AR(p) model with zero drift and geometric weights:

$$\hat{X}_t = (\phi_1 L + \phi_2 L^2 + \cdots + \phi_p L^p)[X_t],\ \phi_i := \phi^i,$$

with $|\phi| < 1$ to ensure that the model is stationary. The order p can be found through autocorrelation tests of the residual if we make the additional assumption that the error $X_t - \hat{X}_t$ is Gaussian. Example tests include the Ljung–Box and Lagrange multiplier tests. However, the over-reliance on parametric diagnostic tests should be used with caution since the conditions for satisfying the tests may not be satisfied on complex time series data. Because the weights are time independent, plain RNNs are static time series models and not suited to non-covariance stationary time series data.

Additional layers can be added to create deep RNNs by stacking them on top of each other, using the hidden state of the RNN as the input to the next layer. However, RNNs have difficulty in learning long-term dynamics, due in part to the vanishing and exploding gradients that can result from propagating the gradients down through the many unfolded layers of the network. Moreover, RNNs like most methods in supervised machine learning are inherently designed for stationary data. Oftentimes, financial time series data is non-stationary.

In Chap. 8, we shall introduce gated recurrent units (GRUs) and long short term memory (LSTM) networks, the latter is shown in Fig. 1.4c as a particular form of recurrent network which provide a solution to this problem by incorporating memory units. In the language of time series modeling, we shall construct dynamic RNNs which are suitable for non-stationary data. More precisely, we shall see that these architecture shall learn when to forget previous hidden states and when to update hidden states given new information.

This ability to model hidden states is of central importance in financial time series modeling and applications in trading. Mixture models and hidden Markov models have historically been the primary probabilistic methods used in quantitative finance and econometrics to model regimes and are reviewed in Chap. 2 and Chap. 7 respectively. Readers are encouraged to review Chap. 2, before reading Chap. 7.

? Multiple Choice Question 3

Select all the following correct statements:

1. A linear recurrent neural network with a memory of p lags is an autoregressive model $AR(p)$ with non-parametric error.

2. Recurrent neural networks, as time series models, are guaranteed to be stationary, for any choice of weights.
3. The amount of memory in a shallow recurrent network corresponds to the number of times a single perceptron layer is unfolded.
4. The amount of memory in a deep recurrent network corresponds to the number of perceptron layers.

3.3 Over-fitting

Undoubtedly the pivotal concern with machine learning, and especially deep learning, is the propensity for over-fitting given the number of parameters in the model. This is why skill is needed to fit deep neural networks.

In frequentist statistics, over-fitting is addressed by penalizing the likelihood function with a penalty term. A common approach is to select models based on Akaike's information criteria (Akaike 1973), which assumes that the model error is Gaussian. The penalty term is in fact a sample bias correction term to the Kullback–Leibler divergence (the relative Entropy) and is applied post-hoc to the unpenalized maximized loss likelihood.

Machine learning methods such as least absolute shrinkage and selection operator (LASSO) and ridge regression more conveniently directly optimize a loss function with a penalty term. Moreover the approach is not restricted to modeling error distributional assumptions. LASSO or L_1 regularization favors sparser parameterizations, whereas ridge regression or L_2 reduces the magnitude of the parameters. Regularization is arguably the most important aspect of why machine learning methods have been so successful in finance and other distributions. Conversely, its absence is why neural networks fell out-of-favor in the finance industry in the 1990s.

Regularization and information criteria are closely related, a key observation which enables us to express model selection in terms of information entropy and hence root our discourse in the works of Shannon (1948), Wiener (1964), Kullback and Leibler (1951). How to choose weights, the concept of regularization for model selection, and cross-validation is discussed in Chap. 4.

It turns out that the choice of priors in Bayesian modeling provides a probabilistic analog to LASSO and ridge regression. L_2 regularization is equivalent to a Gaussian prior and L_1 is an equivalent to a Laplacian prior. Another important feature of Bayesian models is that they have a natural mechanism for prevention of over-fitting built-in. Introductory Bayesian modeling is covered extensively in Chap. 2.

4 Reinforcement Learning

Recall that supervised learning is essentially a paradigm for inferring the parameters of a map between input data and an output through minimizing an error over training samples. Performance generalization is achieved through estimating regularization parameters on cross-validation data. Once the weights of a network are learned, they are not updated in response to new data. For this reason, supervised learning can be considered as an "offline" form of learning, i.e. the model is fitted offline. Note that we avoid referring to the model as static since it is possible, under certain types of architectures, to create a dynamical model in which the map between input and output changes over time. For example, as we shall see in Chap. 8, a LSTM maintains a set of hidden state variables which result in a different form of the map over time.

In such learning, a "teacher" provides an exact right output for each data point in a training set. This can be viewed as "feedback" from the teacher, which for supervised learning amounts to informing the agent with the correct label each time the agent classifies a new data point in the training dataset. Note that this is opposite to unsupervised learning, where there is no teacher to provide correct answers to a ML algorithm, which can be viewed as a setting with no teacher, and, respectively, no feedback from a teacher.

An alternative learning paradigm, referred to as "reinforcement learning," exists which models a sequence of decisions over state space. The key difference of this setting from supervised learning is feedback from the teacher is somewhat in between of the two extremes of unsupervised learning (no feedback at all) and supervised learning that can be viewed as feedback by providing the right labels. Instead, such partial feedback is provided by "rewards" which encourage a desired behavior, but without explicitly instructing the agent what exactly it should do, as in supervised learning.

The simplest way to reason about reinforcement learning is to consider machine learning tasks as a problem of an agent interacting with an environment, as illustrated in Fig. 1.6.

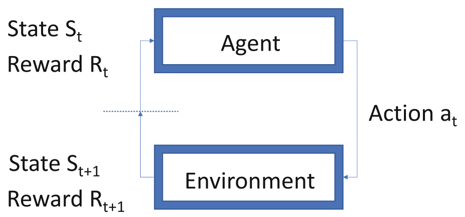

Fig. 1.6 This figure shows a reinforcement learning agent which performs actions at times $t_0, \ldots, t_n$. The agent perceives the environment through the state variable S_t. In order to perform better on its task, feedback on an action a_t is provided to the agent at the next time step in the form of a reward R_t

The agent learns about the environment in order to perform better on its task, which can be formulated as the problem of performing an optimal **action**. If an action performed by an agent is always the same and does not impact the environment, in this case we simply have a perception task, because learning about the environment helps to improve performance on this task. For example, you might have a model for prediction of mortgage defaults where the action is to compute the default probability for a given mortgage. The agent, in this case, is just a predictive model that produces a number and there is measurement of how the model impacts the environment. For example, if a model at a large mortgage broker predicted that all borrowers will default, it is very likely that this would have an impact on the mortgage market, and consequently future predictions. However, this feedback is ignored as the agent just performs perception tasks, ideally suited for supervised learning. Another example is in trading. Once an action is taken by the strategy there is feedback from the market which is referred to as "market impact."

Such a learner is configured to maximize a long-run utility function under some assumptions about the environment. One simple assumption is to treat the environment as being fully observable and evolving as a first-order Markov process. A Markov Decision Process (MDP) is then the simplest modeling framework that allows us to formalize the problem of reinforcement learning. A task solved by MDPs is the problem of **optimal control**, which is the problem of choosing action variables over some period of time, in order to maximize some objective function that depends both on the future states and action taken. In a discrete-time setting, the state of the environment $S_t \in \mathcal{S}$ is used by the learner (a.k.a. agent) to decide which action $a_t \in \mathcal{A}(S_t)$ to take at each time step. This decision is made dynamic by updating the probabilities of selecting each action conditioned on S_t. These conditional probabilities $\pi_t(a|s)$ are referred to as the agent's **policy**. The mechanism for updating the policy as a result of its learning is as follows: one time step later and as a consequence of its action, the learner receives a reward defined by a **reward function**, an immediate reward given the current state S_t and action taken a_t.

As a result of the dynamic environment and the action of the agent, we transition to a new state S_{t+1}. A reinforcement learning method specifies how to change the policy so as to maximize the total amount of reward received over the long-run. The constructs for reinforcement learning will be formalized in Chap. 9 but we shall informally discuss some of the challenges of reinforcement learning in finance here.

Most of the impressive progress reported recently with reinforcement learning by researchers and companies such as Google's DeepMind or OpenAI, such as playing video games, walking robots, self-driving cars, etc., assumes complete observability, using Markovian dynamics. The much more challenging problem, which is a better setting for finance, is how to formulate reinforcement learning for partially observable environments, where one or more variables are hidden.

Another, more modest, challenge exists in how to choose the optimal policy when no environment is fully observable but the dynamic process for how the states evolve over time is unknown. It may be possible, for simple problems, to reason about how the states evolve, perhaps adding constraints on the state-action space. However, the problem is especially acute in high-dimensional discrete state spaces, arising from, say, discretizing continuous state spaces. Here, it is typically intractable to enumerate all combinations of states and actions and it is hence not possible to exactly solve the optimal control problem. Chapter 9 will present approaches for approximating the optimal control problem. In particular, we will turn to neural networks to approximate an action function known as a "Q-function." Such an approach is referred to as "Q-Learning" and more recently, with the use of deep learning to approximate the Q-function, is referred to as "Deep Q-Learning."

To fix ideas, we consider a number of examples to illustrate different aspects of the problem formulation and challenge in applying reinforcement learning. We start with arguably the most famous toy problem used to study stochastic optimal control theory, the "multi-armed bandit problem." This problem is especially helpful in developing our intuition of how an agent must balance the competing goals of exploring different actions versus exploitation of known outcomes.

Example 1.3 Multi-armed Bandit Problem

Suppose there is a fixed and finite set of n actions, a.k.a. arms, denoted $\mathcal{A}$. Learning proceeds in rounds, indexed by $t = 1, \ldots, T$. The number of rounds T, a.k.a. the time horizon, is fixed and known in advance. In each round, the agent picks an arm a_t and observes the reward $R_t(a_t)$ for the chosen arm only. For avoidance of doubt, the agent does not observe rewards for other actions that could have been selected. If the goal is to maximize total reward over all rounds, how should the agent choose an arm?

Suppose the rewards R_t are independent and identical random variables with distribution $\boldsymbol{\nu} \in [0, 1]^n$ and mean $\boldsymbol{\mu}$. The best action is then the distribution with the maximum mean μ^*.

The difference between the player's accumulated reward and the maximum the player (a.k.a. the "cumulative regret") could have obtained had she known all the parameters is

$$\bar{R}_T = T\mu^* - \mathbb{E} \sum_{t \in [T]} R_t.$$

Intuitively, an agent should pick arms that performed well in the past, yet the agent needs to ensure that no good option has been missed.

The theoretical origins of reinforcement learning are in stochastic dynamic programming. In this setting, an agent must make a sequence of decisions under uncertainty about the reward. If we can characterize this uncertainty with probability distributions, then the problem is typically much easier to solve. We shall assume that the reader has some familiarity with dynamic programming—the extension to stochastic dynamic programming is a relatively minor conceptual development. Note that Chap. 9 will review pertinent aspects of dynamic programming, including

Bellman optimality. The following optimal payoff example will likely just serve as a simple review exercise in dynamic programming, albeit with uncertainty introduced into the problem. As we follow the mechanics of solving the problem, the example exposes the inherent difficulty of relaxing our assumptions about the distribution of the uncertainty.

Example 1.4 Uncertain Payoffs

A strategy seeks to allocate \$600 across 3 markets and is equally profitable once the position is held, returning 1% of the size of the position over a short trading horizon $[t, t+1]$. However, the markets vary in liquidity and there is a lower probability that the larger orders will be filled over the horizon. The amount allocated to each market must be either $K = \{100, 200, 300\}$.

Strategy	Allocation	Fill probability
M_1	100	0.8
	200	0.7
	300	0.6
M_2	100	0.75
	200	0.7
	300	0.65
M_3	100	0.75
	200	0.75
	300	0.6

Strategy	Allocation	Return
M_1	100	0.8
	200	1.4
	300	1.8
M_2	100	0.75
	200	1.4
	300	1.95
M_3	100	0.75
	200	1.5
	300	1.8

The optimal allocation problem under uncertainty is a stochastic dynamic programming problem. We can define value functions $v_i(x)$ for total allocation amount x for each stage of the problem, corresponding to the markets. We then find the optimal allocation using the backward recursive formulae:

$$v_3(x) = R_3(x), \forall x \in K,$$

$$v_2(x) = \max_{k \in K}\{R_2(k) + v_3(x-k)\}, \forall x \in K + 200,$$

$$v_1(x) = \max_{k \in K}\{R_1(k) + v_2(x-k)\}, x = 600,$$

The left-hand side of the table below tabulates the values of $R_2 + v_3$ corresponding to the second stage of the backward induction for each pair (M_2, M_3).

(continued)

Example 1.4 (continued)

$R_2 + v_3$	M_2		
M_3	100	200	300
100	1.5	2.15	2.7
200	2.25	2.9	3.45
300	2.55	3.2	3.75

M_1	(M_2^*, M_3^*)	v_2	R_1	$R_1 + v_2$
100	(300, 200)	3.45	0.8	4.25
200	(200, 200)	2.9	1.4	4.3
300	(100, 200)	2.25	1.8	4.05

The right-hand side of the above table tabulates the values of $R_1 + v_2$ corresponding to the third and final stage of the backward induction for each tuple (M_1, M_2^*, M_3^*).

In the above example, we can see that the allocation {200, 200, 200} maximizes the reward to give $v_1(600) = 4.3$. While this exercise is a straightforward application of a Bellman optimality recurrence relation, it provides a glimpse of the types of stochastic dynamic programming problems that can be solved with reinforcement learning. In particular, if the fill probabilities are unknown but must be learned over time by observing the outcome over each period $[t_i, t_{i+1})$, then the problem above cannot be solved by just using backward recursion. Instead we will move to the framework of reinforcement learning and attempt to learn the best actions given the data. Clearly, in practice, the example is much too simple to be representative of real-world problems in finance—the profits will be unknown and the state space is significantly larger, compounding the need for reinforcement learning. However, it is often very useful to benchmark reinforcement learning on simple stochastic dynamic programming problems with closed-form solutions.

In the previous example, we assumed that the problem was static—the variables in the problem did not change over time. This is the so-called static allocation problem and is somewhat idealized. Our next example will provide a glimpse of the types of problems that typically arise in optimal portfolio investment where random variables are dynamic. The example is also seated in more classical finance theory, that of a "Markowitz portfolio" in which the investor seeks to maximize a risk-adjusted long-term return and the wealth process is self-financing.[6]

Example 1.5 Optimal Investment in an Index Portfolio

Let S_t be a time-t price of a risky asset such as a sector exchange-traded fund (ETF). We assume that our setting is discrete time, and we denote different time steps by integer valued-indices $t = 0, \dots, T$, so there are $T + 1$ values on our discrete-time grid. The discrete-time random evolution of the risky asset S_t is

(continued)

[6]A wealth process is self-financing if, at each time step, any purchase of an additional quantity of the risky asset is funded from the bank account. Vice versa, any proceeds from a sale of some quantity of the asset go to the bank account.

Example 1.5 (continued)

$$S_{t+1} = S_t\,(1 + \phi_t)\,, \tag{1.12}$$

where ϕ_t is a random variable whose probability distribution may depend on the current asset value S_t. To ensure non-negativity of prices, we assume that ϕ_t is bounded from below $\phi_t \geq -1$.

Consider a wealth process W_t of an investor who starts with an initial wealth $W_0 = 1$ at time $t = 0$ and, at each period $t = 0, \ldots, T - 1$ allocates a fraction $u_t = u_t(S_t)$ of the total portfolio value to the risky asset, and the remaining fraction $1 - u_t$ is invested in a risk-free bank account that pays a risk-free interest rate $r_f = 0$. We will refer to a set of decision variables for all time steps as a *policy* $\pi := \{u_t\}_{t=0}^{T-1}$. The wealth process is self-financing and so the wealth at time $t + 1$ is given by

$$W_{t+1} = (1 - u_t)\,W_t + u_t W_t\,(1 + \phi_t)\,. \tag{1.13}$$

This produces the one-step return

$$r_t = \frac{W_{t+1} - W_t}{W_t} = u_t\phi_t. \tag{1.14}$$

Note this is a random function of the asset price S_t. We define one-step rewards R_t for $t = 0, \ldots, T - 1$ as risk-adjusted returns

$$R_t = r_t - \lambda \mathrm{Var}\,[r_t|S_t] = u_t\phi_t - \lambda u_t^2 \mathrm{Var}\,[\phi_t|S_t]\,, \tag{1.15}$$

where λ is a risk-aversion parameter.[a] We now consider the problem of maximization of the following concave function of the control variable u_t:

$$V^{\pi}(s) = \max_{u_t} \mathbb{E}\left[\sum_{t=0}^{T} R_t \,\middle|\, S_t = s\right] = \max_{u_t} \mathbb{E}\left[\sum_{t=0}^{T} u_t\phi_t - \lambda u_t^2 \mathrm{Var}\,[\phi_t|S_t] \,\middle|\, S_t = s\right]. \tag{1.16}$$

Equation 1.16 defines an optimal investment problem for $T - 1$ steps faced by an investor whose objective is to optimize risk-adjusted returns over each period. This optimization problem is equivalent to maximizing the long-run

(continued)

[a]Note, for avoidance of doubt, that the risk-aversion parameter must be scaled by a factor of $\frac{1}{2}$ to ensure consistency with the finance literature.

Example 1.5 (continued)

returns over the period $[0, T]$. For each $t = T-1, T-2, \ldots, 0$, the optimality condition for action u_t is now obtained by maximization of $V^\pi(s)$ with respect to u_t:

$$u_t^* = \frac{\mathbb{E}[\phi_t | S_t]}{2\lambda \text{Var}[\phi_t | S_t]}, \tag{1.17}$$

where we allow for short selling in the ETF (i.e., $u_t < 0$) and borrowing of cash $u_t > 1$.

This is an example of a stochastic optimal control problem for a portfolio that maximizes its cumulative risk-adjusted return by repeatedly rebalancing between cash and a risky asset. Such problems can be solved using means of dynamic programming or reinforcement learning. In our problem, the dynamic programming solution is given by an analytical expression (1.17). Chapter 9 will present more complex settings including reinforcement learning approaches to optimal control problems, as well as demonstrate how expressions like the optimal action of Eq. 1.17 can be computed in practice.

? Multiple Choice Question 4

Select all the following correct statements:

1. The name “Markov processes” first historically appeared as a result of a misspelled name “Mark-Off processes” that was previously used for random processes that describe learning in certain types of video games, but has become a standard terminology since then.
2. The goal of (risk-neutral) reinforcement learning is to maximize the expected total reward by choosing an optimal policy.
3. The goal of (risk-neutral) reinforcement learning is to neutralize risk, i.e. make the variance of the total reward equal zero.
4. The goal of risk-sensitive reinforcement learning is to teach a RL agent to pick action policies that are most prone to risk of failure. Risk-sensitive RL is used, e.g. by venture capitalists and other sponsors of RL research, as a tool to assess the feasibility of new RL projects.

5 Examples of Supervised Machine Learning in Practice

The practice of machine learning in finance has grown somewhat commensurately with both theoretical and computational developments in machine learning. Early adopters have been the quantitative hedge funds, including Bridgewater Associates, Renaissance Technologies, WorldQuant, D.E. Shaw, and Two Sigma who have embraced novel machine learning techniques although there are mixed degrees of adoption and a healthy skepticism exists that machine learning is a panacea for quantitative trading. In 2015, Bridgewater Associates announced a new artificial intelligence unit, having hired people from IBM Watson with expertise in deep learning. Anthony Ledford, chief scientist at MAN AHL: "It's at an early stage. We have set aside a pot of money for test trading. With deep learning, if all goes well, it will go into test trading, as other machine learning approaches have." Winton Capital Management's CEO David Harding: "People started saying, 'There's an amazing new computing technique that's going to blow away everything that's gone before.' There was also a fashion for genetic algorithms. Well, I can tell you none of those companies exist today—not a sausage of them."

Some qualifications are needed to more accurately assess the extent of adoption. For instance, there is a false line of reasoning that ordinary least squares regression and logistic regression, as well as Bayesian methods, are machine learning techniques. Only if the modeling approach is algorithmic, without positing a data generation process, can the approach be correctly categorized as machine learning. So regularized regression without use of parametric assumptions on the error distribution is an example of machine learning. Unregularized regression with, say, Gaussian error is not a machine learning technique. The functional form of the input–output map is the same in both cases, which is why we emphasize that the functional form of the map is not a sufficient condition for distinguishing ML from statistical methods.

With that caveat, we shall view some examples that not only illustrate some of the important practical applications of machine learning prediction in algorithmic trading, high-frequency market making, and mortgage modeling but also provide a brief introduction to applications that will be covered in more detail in later chapters.

5.1 Algorithmic Trading

Algorithmic trading is a natural playground for machine learning. The idea behind algorithmic trading is that trading decisions should be based on data, not intuition. Therefore, it should be viable to automate this decision-making process using an algorithm, either specified or learned. The advantages of algorithmic trading include complex market pattern recognition, reduced human produced error, ability to test on historic data, etc. In recent times, as more and more information is being digitized, the feasibility and capacity of algorithmic trading has been expanding drastically. The number of hedge funds, for example, that apply machine learning for algorithmic trading is steadily increasing.

Here we provide a simple example of how machine learning techniques can be used to improve traditional algorithmic trading methods, but also provide new trading strategy suggestions. The example here is not intended to be the "best" approach, but rather indicative of more out-of-the-box strategies that machine learning facilitates, with the emphasis on minimizing out-of-sample error by pattern matching through efficient compression across high-dimensional datasets.

Momentum strategies are one of the most well-known algo-trading strategies; In general, strategies that predict prices from historic price data are categorized as momentum strategies. Traditionally momentum strategies are based on certain regression-based econometric models, such as ARIMA or VAR (see Chap. 6). A drawback of these models is that they impose strong linearity which is not consistently plausible for time series of prices. Another caveat is that these models are parametric and thus have strong bias which often causes underfitting. Many machine learning algorithms are both non-linear and semi/non-parametric, and therefore prove complementary to existing econometric models.

In this example we build a simple momentum portfolio strategy with a feedforward neural network. We focus on the S&P 500 stock universe, and assume we have daily close prices for all stocks over a ten-year period.[7]

Problem Formulation

The most complex practical aspect of machine learning is how to choose the input ("features") and output. The type of desired output will determine whether a regressor or classifier is needed, but the general rule is that it must be actionable (i.e., tradable). Suppose our goal is to invest in an equally weighted, long only, stock portfolio only if it beats the S&P 500 index benchmark (which is a reasonable objective for a portfolio manager). We can therefore label the portfolio at every observation t based on the mean directional excess return of the portfolio:

$$G_t = \begin{cases} 1 & \frac{1}{N}\sum_i r^i_{t+h,t} - \tilde{r}_{t+h,t} \geq \epsilon, \\ 0 & \frac{1}{N}\sum_i r^i_{t+h,t} - \tilde{r}_{t+h,t} < 0, \end{cases} \tag{1.18}$$

where $r^i_{t+h,t}$ is the return of stock i between times t and $t+h$, $\tilde{r}_{t+h,t}$ is the return of the S&P 500 index in the same period, and ϵ is some target next period excess portfolio return. Without loss of generality, we could invest in the universe ($N = 500$), although this is likely to have adverse practical implications such as excessive transaction costs. We could easily just have restricted the number of stocks to a subset, such as the top decile of performing stocks in the last period. Framed this way, the machine learner is thus informing us when our stock selection strategy will outperform the market. It is largely agnostic to how the stocks are selected, provided the procedure is systematic and based solely on the historic data provided to the classifier. It is further worth noting that the map between the decision to hold the customized portfolio has a non-linear relationship with the past returns of the universe.

To make the problem more concrete, let us set $h = 5$ days. The algorithmic strategy here is therefore automating the decision to invest in the customized

[7]The question of how much data is needed to train a neural network is a central one, with the immediate concern being insufficient data to avoid over-fitting. The amount of data needed is complex to assess; however, it is partly dependent on the number of edges in the network and can be assessed through bias–variance analysis, as described in Chap. 4.

Table 1.3 Training samples for a classification problem

Date	X_1	X_2	…	X_{500}	G
2007-01-03	0.051	−0.035		0.072	0
2017-01-04	−0.092	0.125		−0.032	0
2017-01-05	0.021	0.063		−0.058	1
…					
2017-12-29	0.093	−0.023		0.045	1
2017-12-30	0.020	0.019		0.022	1
2017-12-31	−0.109	0.025		−0.092	1

portfolio or the S&P 500 index every week based on the previous 5-day realized returns of all stocks. To apply machine learning to this decision, the problem translates into finding the weights in the network between past returns and the decision to invest in the equally weighted portfolio. For avoidance of doubt, we emphasize that the interpretation of the optimal weights differs substantially from Markowitz's mean–variance portfolios, which simply finds the portfolio weights to optimize expected returns for a given risk tolerance. Here, we either invest equal amounts in all stocks of the portfolio or invest the same amount in the S&P 500 index and the weights in the network signify the relevance of past stock returns in the expected excess portfolio return outperforming the market.

Data

Feature engineering is always important in building models and requires careful consideration. Since the original price data does not meet several machine learning requirements, such as stationarity and i.i.d. distributional properties, one needs to engineer input features to prevent potential "garbage-in-garbage-out" phenomena. In this example, we take a simple approach by using only the 5-day realized returns of all S&P 500 stocks.[8] Returns are scale-free and no further standardization is needed. So for each time t, the input features are

$$\mathbf{X}_t = \left[r^1_{t,t-5}, \ldots, r^{500}_{t,t-5} \right]. \tag{1.19}$$

Now we can aggregate the features and labels into a panel indexed by date. Each column is an entry in Eq. 1.19, except for the last column which is the assigned label from Eq. 1.18, based on the realized excess stock returns of the portfolio. An example of the labeled input data (X, G) is shown in Table 1.3.

The process by which we train the classifier and evaluate its performance will be described in Chap. 4, but this example illustrates how algo-trading strategies can be crafted around supervised machine learning. Our model problem could be tailored

[8]Note that the composition of the S&P 500 changes over time and so we should interpret a feature as a fixed symbol.

for specific risk-reward and performance reporting metrics such as, for example, Sharpe or information ratios meeting or exceeding a threshold.

ϵ is typically chosen to be a small value so that the labels are not too imbalanced. As the value ϵ is increased, the problem becomes an "outlier prediction problem"—a highly imbalanced classification problem which requires more advanced sampling and interpolation techniques beyond an off-the-shelf classifier.

In the next example, we shall turn to another important aspect of machine learning in algorithmic trading, namely execution. How the trades are placed is a significant aspect of algorithmic trading strategy performance, not only to minimize price impact of market taking strategies but also for market making. Here we shall look to transactional data to perfect the execution, an engineering challenge by itself just to process market feeds of tick-by-tick exchange transactions. The example considers a market making application but could be adapted for price impact and other execution considerations in algorithmic trading by moving to a reinforcement learning framework.

A common mistake is to assume that building a predictive model will result in a profitable trading strategy. Clearly, the consideration given to reliably evaluating machine learning in the context of trading strategy performance is a critical component of its assessment.

5.2 High-Frequency Trade Execution

Modern financial exchanges facilitate the electronic trading of instruments through an instantaneous double auction. At each point in time, the market demand and the supply can be represented by an electronic limit order book, a cross-section of orders to execute at various price levels away from the market price as illustrated in Table 1.4.

Electronic market makers will quote on both sides of the market in an attempt to capture the bid–ask spread. Sometimes a large market order, or a succession of smaller markets orders, will consume an entire price level. This is why the market price fluctuates in liquid markets—an effect often referred to by practitioners as a "price-flip." A market maker can take a loss if only one side of the order is filled as a result of an adverse price movement.

Figure 1.7 (left) illustrates a typical mechanism resulting in an adverse price movement. A snapshot of the limit order book at time t, before the arrival of a market order, and after at time $t+1$ is shown in the left and right panels, respectively. The resting orders placed by the market marker are denoted with the "+" symbol—red denotes a bid and blue denotes an ask quote. A buy market order subsequently arrives and matches the entire resting quantity of best ask quotes. Then at event time $t + 1$ the limit order book is updated—the market maker's ask has been filled (blue minus symbol) and the bid now rests away from the inside market. The market marker may systematically be forced to cancel the bid and buy back at a higher price, thus taking a loss.

Table 1.4 This table shows a snapshot of the limit order book of S&P 500 e-mini futures (ES). The top half ("sell-side") shows the ask volumes and prices and the lower half ("buy side") shows the bid volumes and prices. The quote levels are ranked by the most competitive at the center (the "inside market"), outward to the least competitive prices at the top and bottom of the limit order book. Note that only five bid or ask levels are shown in this example, but the actual book is much deeper

Bid	Price	Ask
	2170.25	1284
	2170.00	1642
	2169.75	1401
	2169.50	1266
	2169.25	290
477	2169.00	
1038	2168.75	
950	2168.50	
1349	2168.25	
1559	2168.00	

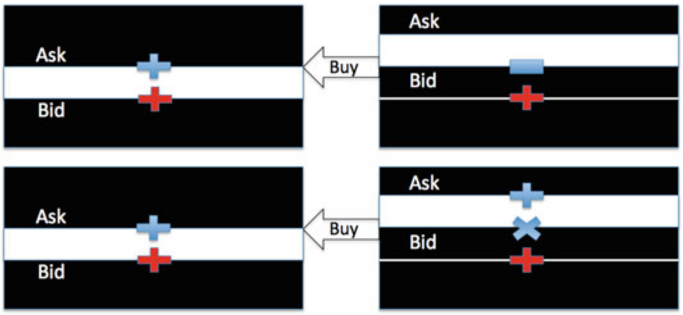

Fig. 1.7 (Top) A snapshot of the limit order book is taken at time t. Limit orders placed by the market marker are denoted with the "+" symbol—red denotes a bid and blue denotes an ask. A buy market order subsequently arrives and matches the entire resting quantity of best ask quotes. Then at event time $t + 1$ the limit order book is updated. The market maker's ask has been filled (blue minus symbol) and the bid rests away from the inside market. (Bottom) A pre-emptive strategy for avoiding adverse price selection is illustrated. The ask is requoted at a higher ask price. In this case, the bid is not replaced and the market maker may capture a tick more than the spread if both orders are filled

Machine learning can be used to predict these price movements (Kearns and Nevmyvaka 2013; Kercheval and Zhang 2015; Sirignano 2016; Dixon et al. 2018; Dixon 2018b,a) and thus to potentially avoid adverse selection. Following Cont and de Larrard (2013) we can treat queue sizes at each price level as input variables. We can additionally include properties of market orders, albeit in a form which our machines deem most relevant to predicting the direction of price movements (a.k.a. feature engineering). In contrast to stochastic modeling, we do not impose conditional distributional assumptions on the independent variables (a.k.a. features) nor assume that price movements are Markovian. Chapter 8 presents a RNN for mid-price prediction from the limit order book history which is the starting point for the more in-depth study of Dixon (2018b) which includes market orders and demonstrates the superiority of RNNs compared to other time series methods such as Kalman filters.

We reiterate that the ability to accurately predict does not imply profitability of the strategy. Complex issues concerning queue position, exchange matching rules, latency, position constraints, and price impact are central considerations for practitioners. The design of profitable strategies goes beyond the scope of this book but the reader is referred to de Prado (2018) for the pitfalls of backtesting and designing algorithms for trading. Dixon (2018a) presents a framework for evaluating the performance of supervised machine learning algorithms which accounts for latency, position constraints, and queue position. However, supervised learning is ultimately not the best machine learning approach as it cannot capture the effect of market impact and is too inflexible to incorporate more complex strategies. Chapter 9 presents examples of reinforcement learning which demonstrate how to capture market impact and also how to flexibly formulate market making strategies.

5.3 Mortgage Modeling

Beyond the data rich environment of algorithmic trading, does machine learning have a place in finance? One perspective is that there simply is not sufficient data for some "low-frequency" application areas in finance, especially where traditional models have failed catastrophically. The purpose of this section is to serve as a sobering reminder that long-term forecasting goes far beyond merely selecting the best choice of machine learning algorithm and why there is no substitute for strong domain knowledge and an understanding of the limitations of data.

In the USA, a mortgage is a loan collateralized by real-estate. Mortgages are used to securitize financial instruments such as mortgage backed securities and collateralized mortgage obligations. The analysis of such securities is complex and has changed significantly over the last decade in response to the 2007–2008 financial crises (Stein 2012).

Unless otherwise specified, a mortgage will be taken to mean a "residential mortgage," which is a loan with payments due monthly that is collateralized by a single family home. Commercial mortgages do exist, covering office towers, rental apartment buildings, and industrial facilities, but they are different enough to be considered separate classes of financial instruments. Borrowing money to buy a house is one of the most common, and largest balance, loans that an individual borrower is ever likely to commit to. Within the USA alone, mortgages comprise a staggering \$15 trillion dollars in debt. This is approximately the same balance as the total federal debt outstanding (Fig. 1.8).

Within the USA, mortgages may be repaid (typically without penalty) at will by the borrower. Usually, borrowers use this feature to refinance their loans in favorable interest rate regimes, or to liquidate the loan when selling the underlying house. This

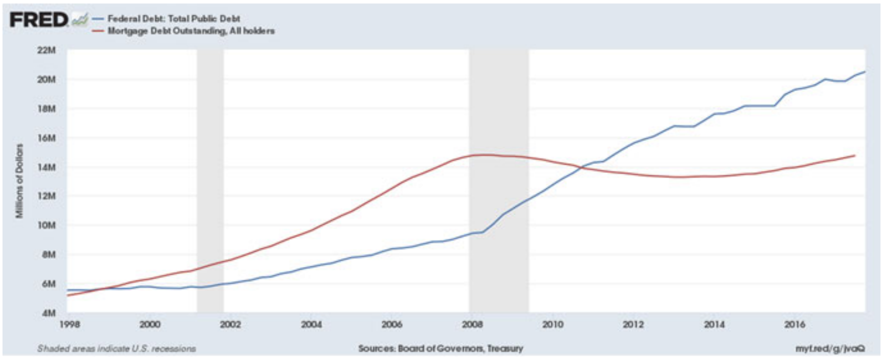

Fig. 1.8 Total mortgage debt in the USA compared to total federal debt, millions of dollars, unadjusted for inflation. Source: https://fred.stlouisfed.org/series/MDOAH, https://fred.stlouisfed.org/series/GFDEBTN

Table 1.5 At any time, the states of any US style residential mortgage is in one of the several possible states

Symbol	Name	Definition
P	Paid	All balances paid, loan is dissolved
C	Current	All payments due have been paid
3	30-days delinquent	Mortgage is delinquent by one payment
6	60-days delinquent	Delinquent by 2 payments
9	90+ delinquent	Delinquent by 3 or more payments
F	Foreclosure	Foreclosure has been initiated by the lender
R	Real-Estate-Owned (REO)	The lender has possession of the property
D	Default liquidation	Loan is involuntarily liquidated for nonpayment

has the effect of moving a great deal of financial risk off of individual borrowers, and into the financial system. It also drives a lively and well developed industry around modeling the behavior of these loans.

The mortgage model description here will generally follow the comprehensive work in Sirignano et al. (2016), with only a few minor deviations.

Any US style residential mortgage, in each month, can be in one of the several possible states listed in Table 1.5.

Consider this set of K available states to be $\mathbb{K} = \{P, C, 3, 6, 9, F, R, D\}$. Following the problem formulation in Sirignano et al. (2016), we will refer to the status of loan n at time t as $U_t^n \in \mathbb{K}$, and this will be represented as a probability vector using a standard one-hot encoding.

If $X = (X_1, \ldots, X_P)$ is the input matrix of P explanatory variables, then we define a probability transition density function $g : \mathbb{R}^P \to [0, 1]^{K \times K}$ parameterized by θ so that

$$\mathbb{P}(U_{t+1}^n = i \mid U_t^n = j, X_t^n) = g_{i,j}(X_t^n \mid \theta), \forall i, j \in \mathbb{K}. \tag{1.20}$$

Note that $g(X_t^n \mid \theta)$ is a time in-homogeneous $K \times K$ Markov transition matrix. Also, not all transitions are even conceptually possible—there are non-commutative states. For instance, a transition from C to 6 is not possible since a borrower cannot miss two payments in a single month. Here we will write $p_{(i,j)} := g_{i,j}(X_t^n \mid \theta)$ for ease of notation and because of the non-commutative state transitions where $p_{(i,j)} = 0$, the Markov matrix takes the form:

$$g(X_t^n \mid \theta) = \begin{bmatrix} 1 & p_{(c,p)} & p_{(3,p)} & 0 & 0 & 0 & 0 & 0 \\ 0 & p_{(c,c)} & p_{(3,c)} & p_{(6,c)} & p_{(9,c)} & p_{(f,c)} & 0 & 0 \\ 0 & p_{(c,3)} & p_{(3,3)} & p_{(6,3)} & p_{(9,3)} & p_{(f,3)} & 0 & 0 \\ 0 & 0 & p_{(3,6)} & p_{(6,6)} & p_{(9,6)} & p_{(f,6)} & 0 & 0 \\ 0 & 0 & 0 & p_{(6,9)} & p_{(9,9)} & p_{(f,9)} & 0 & 0 \\ 0 & 0 & 0 & p_{(6,f)} & p_{(9,f)} & p_{(f,f)} & 0 & 0 \\ 0 & 0 & 0 & p_{(6,r)} & p_{(9,r)} & p_{(f,r)} & p_{(r,r)} & 0 \\ 0 & 0 & 0 & p_{(6,d)} & p_{(9,d)} & p_{(f,d)} & p_{(r,d)} & 1 \end{bmatrix}.$$

Our classifier $g_{i,j}(X_t^n \mid \theta)$ can thus be constructed so that only the probability of transition between the commutative states are outputs and we can apply softmax functions on a subset of the outputs to ensure that $\sum_{j \in \mathbb{K}} g_{i,j}(X_t^n \mid \theta) = 1$ and hence the transition probabilities in each row sum to one.

For the purposes of financial modeling, it is important to realize that both states P and D are loan liquidation terminal states. However, state P is considered to be voluntary loan liquidation (e.g., prepayment due to refinance), whereas state D is considered to be involuntary liquidation (e.g., liquidation via foreclosure and auction). These states are not distinguishable in the mortgage data itself, but rather the driving force behind liquidation must be inferred from the events leading up to the liquidation.

One contributor to mortgage model misprediction in the run up to the 2008 financial crisis was that some (but not all) modeling groups considered loans liquidating from deep delinquency (e.g., status 9) to be the transition $9 \rightarrow P$ if no losses were incurred. However, behaviorally, these were typically defaults due to financial hardship, and they would have had losses in a more difficult house price regime. They were really $9 \rightarrow D$ transitions that just happened to be lossless due to strong house price gains over the life of the mortgage. Considering them to be voluntary prepayments (status P) resulted in systematic over-prediction of prepayments in the aftermath of major house price drops. The matrix above therefore explicitly excludes this possibility and forces delinquent loan liquidation to be always considered involuntary.

The reverse of this problem does not typically exist. In most states it is illegal to force a borrower into liquidation until at least 2 payments have been missed. Therefore, liquidation from C or 3 is always voluntary, and hence $C \rightarrow P$ and $3 \rightarrow P$. Except in cases of fraud or severe malfeasance, it is almost never economically advantageous for a lender to force liquidation from status 6, but it is not illegal. Therefore the transition $3 \rightarrow D$ is typically a data error, but $6 \rightarrow D$ is merely very rare.

Example 1.6 Parameterized Probability Transitions

If loan n is current in time period t, then

$$\mathbb{P}(U_t^n) = (0, 1, 0, 0, 0, 0, 0, 0)^T. \tag{1.21}$$

If we have $p_{(c,p)} = 0.05$, $p_{(c,c)} = 0.9$, and $p_{(c,3)} = 0.05$, then

$$\mathbb{P}(U_{t+1}^n \mid X_t^n) = g(X_t^n \mid \theta) \cdot \mathbb{P}(U_t^n) = (0.05, 0.9, 0.05, 0, 0, 0, 0, 0)^T. \tag{1.22}$$

Common mortgage models sometimes use additional states, often ones that are (without additional information) indistinguishable from the states listed above. Table 1.6 describes a few of these.

The reason for including these is the same as the reason for breaking out states like REO, status R. It is known on theoretical grounds that some model regressors from X_t^n should not be relevant for R. For instance, since the property is now owned by the lender, and the loan itself no longer exists, the interest rate (and rate incentive) of the original loan should no longer have a bearing on the outcome. To avoid over-fitting due to highly colinear variables, these known-useless variables are then excluded from transitions models starting in status R.

This is the same reason status T is sometimes broken out, especially for logistic regressions. Without an extra status listed in this way, strong rate disincentives could drive prepayments in the model to (almost) zero, but we know that people die and divorce in all rate regimes, so at least some minimal level of premature loan liquidations must still occur based on demographic factors, not financial ones.

5.3.1 Model Stability

Unlike many other models, mortgage models are designed to accurately predict events a decade or more in the future. Generally, this requires that they be built on regressors that themselves can be accurately predicted, or at least hedged. Therefore, it is common to see regressors like FICO at origination, loan age in months, rate incentive, and loan-to-value (LTV) ratio. Often LTV would be called MTMLTV if it is marked-to-market against projected or realized housing price moves. Of these regressors, original FICO is static over the life of the loan, age is deterministic,

Table 1.6 A brief description of mortgage states

Symbol	Name	Overlaps with	Definition
T	Turnover	P	Loan prepaid due to non-financial life event
U	Curtailment	C	Borrower overpaid to reduce loan principal

Table 1.7 Loan originations by year (Freddie Mac, FRM30)

Year	Loans originated
1999	976,159
2000	733,567
2001	1,542,025
2002	1,403,515
2003	2,063,488
2004	1,133,015
2005	1,618,748
2006	1,300,559
2007	1,238,814
2008	1,237,823
2009	1,879,477
2010	1,250,484
2011	1,008,731
2012	1,249,486
2013	1,375,423
2014	942,208

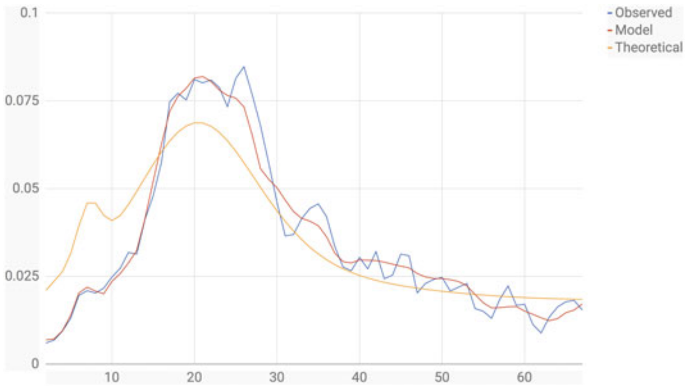

Fig. 1.9 Sample mortgage model predicting $C \rightarrow 3$ and fit on loans originated in 2001 and observed until 2006, by loan age (in months). The prepayment probabilities are shown on the y-axis

rates can be hedged, and MTMLTV is rapidly driven down by loan amortization and inflation thus eliminating the need to predict it accurately far into the future.

Consider the Freddie Mac loan level dataset of 30 year fixed rate mortgages originated through 2014. This includes each monthly observation from each loan present in the dataset. Table 1.7 shows the loan count by year for this dataset.

When a model is fit on 1 million observations from loans originated in 2001 and observed until the end of 2006, its $C \rightarrow P$ probability charted against age is shown in Fig. 1.9.

In Fig. 1.9 the curve observed is the actual prepayment probability of the observations with the given age in the test dataset, “Model” is the model prediction,

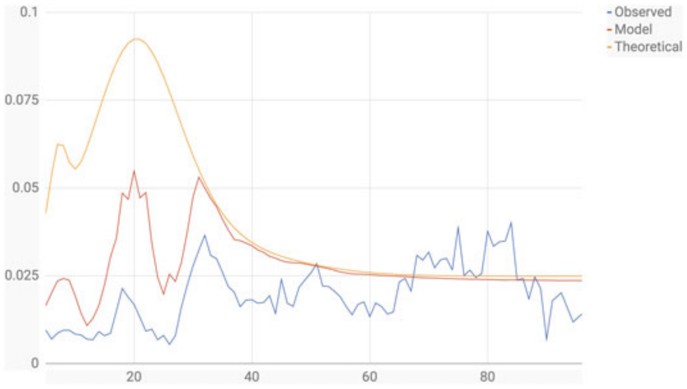

Fig. 1.10 Sample mortgage model predicting $C \rightarrow 3$ and fit on loans originated in 2006 and observed until 2015, by loan age (in months). The prepayment probabilities are shown on the y-axis

and "Theoretical" is the response to age by a theoretical loan with all other regressors from X_t^n held constant. Two observations are worth noting:

1. The marginal response to age closely matches the model predictions; and
2. The model predictions match actual behavior almost perfectly.

This is a regime where prepayment behavior is largely driven by age. When that same model is run on observations from loans originated in 2006 (the peak of housing prices before the crisis), and observed until 2015, Fig. 1.10 is produced.

Three observations are warranted from this figure:

1. The observed distribution is significantly different from Fig. 1.9;
2. The model predicted a decline of 25%, but the actual decline was approximately 56%; and
3. Prepayment probabilities are largely indifferent to age.

The regime shown here is clearly not driven by age. In order to provide even this level of accuracy, the model had to extrapolate far from any of the available data and "imagine" a regime where loan age is almost irrelevant to prepayment. This model meets with mixed success. This particular one was fit on only 8 regressors, a more complicated model might have done better, but the actual driver of this inaccuracy was a general tightening of lending standards. Moreover, there was no good data series available before the crisis to represent lending standards.

This model was reasonably accurate even though almost 15 years separated the start of the fitting data from the end of the projection period, and a lot happened in that time. Mortgage models in particular place a high premium on model stability, and the ability to provide as much accuracy as possible even though the underlying distribution may have changed dramatically from the one that generated the fitting data. Notice also that cross-validation would not help here, as we cannot draw testing data from the distribution we care about, since that distribution comes from the future.

Most importantly, this model shows that the low-dimensional projections of this (moderately) high-dimensional problem are extremely deceptive. No modeler would have chosen a shape like the model prediction from Fig. 1.9 as function of age. That prediction arises due to the interaction of several variables, interactions that are not interpretable from one-dimensional plots such as this. As we will see in subsequent chapters, such complexities in data are well suited to machine learning, but not without a cost. That cost is understanding the "bias–variance tradeoff" and understanding machine learning with sufficient rigor for its decisions to be defensible.

6 Summary

In this chapter, we have identified some of the key elements of supervised machine learning. Supervised machine learning

1. is an algorithmic approach to statistical inference which, crucially, does not depend on a data generation process;
2. estimates a parameterized map between inputs and outputs, with the functional form defined by the methodology such as a neural network or a random forest;
3. automates model selection, using regularization and ensemble averaging techniques to iterate through possible models and arrive at a model with the best out-of-sample performance; and
4. is often well suited to large sample sizes of high-dimensional non-linear covariates.

The emphasis on out-of-sample performance, automated model selection, and absence of a pre-determined parametric data generation process is really the key to machine learning being a more robust approach than many parametric, financial econometrics techniques in use today. The key to adoption of machine learning in finance is the ability to run machine learners alongside their parametric counterparts, observing over time the differences and limitations of parametric modeling based on in-sample fitting metrics. Statistical tests must be used to characterize the data and guide the choice of algorithm, such as, for example, tests for stationary. See Dixon and Halperin (2019) for a checklist and brief but rounded discussion of some of the challenges in adopting machine learning in the finance industry.

Capacity to readily exploit a wide form of data is their other advantage, but only if that data is sufficiently high quality and adds a new source of information. We close this chapter with a reminder of the failings of forecasting models during the financial crisis of 2008 and emphasize the importance of avoiding siloed data extraction. The application of machine learning requires strong scientific reasoning skills and is not a panacea for commoditized and automated decision-making.

7 Exercises

Exercise 1.1: Market Game**

Suppose that two players enter into a market game. The rules of the game are as follows: Player 1 is the market maker, and Player 2 is the market taker. In each round, Player 1 is provided with information $\boldsymbol{x}$, and must choose and declare a value $\alpha \in (0, 1)$ that determines how much it will pay out if a binary event G occurs in the round. $G \sim Bernoulli(p)$, where $p = g(\boldsymbol{x}|\boldsymbol{\theta})$ for some unknown parameter $\boldsymbol{\theta}$.

Player 2 then enters the game with a \$1 payment and chooses one of the following payoffs:

$$V_1(G, p) = \begin{cases} \frac{1}{\alpha} & \text{with probability } p \\ 0 & \text{with probability } (1-p) \end{cases}$$

or

$$V_2(G, p) = \begin{cases} 0 & \text{with probability } p \\ \frac{1}{(1-\alpha)} & \text{with probability } (1-p) \end{cases}$$

1. Given that α is known to Player 2, state the strategy[9] that will give Player 2 an expected payoff, over multiple games, of \$1 without knowing p.
2. Suppose now that p is known to both players. In a given round, what is the optimal choice of α for Player 1?
3. Suppose Player 2 knows with complete certainty, that G will be 1 for a particular round, what will be the payoff for Player 2?
4. Suppose Player 2 has complete knowledge in rounds $\{1, \ldots, i\}$ and can reinvest payoffs from earlier rounds into later rounds. Further suppose without loss of generality that $G = 1$ for each of these rounds. What will be the payoff for Player 2 after i rounds? You may assume that the each game can be played with fractional dollar costs, so that, for example, if Player 2 pays Player 1 \$1.5 to enter the game, then the payoff will be $1.5V_1$.

Exercise 1.2: Model Comparison**

Recall Example 1.2. Suppose additional information was added such that it is no longer possible to predict the outcome with 100% probability. Consider Table 1.8 as the results of some experiment.

Now if we are presented with $\boldsymbol{x} = (1, 0)$, the result could be B or C. Consider three different models applied to this value of $\boldsymbol{x}$ which encode the value A, B, or C.

[9]The strategy refers the choice of weight if Player 2 is to choose a payoff $V = wV_1 + (1-w)V_2$, i.e. a weighted combination of payoffs V_1 and V_2.

Table 1.8 Sample model data

G	$\boldsymbol{x}$
A	(0, 1)
B	(1, 1)
B	(1, 0)
C	(1, 0)
C	(0, 0)

$$f((1,0)) = (0,1,0),\ \text{Predicts B with 100\% certainty.} \tag{1.23}$$

$$g((1,0)) = (0,0,1),\ \text{Predicts C with 100\% certainty.} \tag{1.24}$$

$$h((1,0)) = (0,0.5,0.5),\ \text{Predicts B or C with 50\% certainty.} \tag{1.25}$$

1. Show that each model has the same total absolute error, over the samples where $\boldsymbol{x} = (1, 0)$.
2. Show that all three models assign the same average probability to the values from Table 1.8 when $\boldsymbol{x} = (1, 0)$.
3. Suppose that the market game in Exercise 1 is now played with models f or g. B or C each triggers two separate payoffs, V_1 and V_2, respectively. Show that the losses to Player 1 are unbounded when $\boldsymbol{x} = (1, 0)$ and $\alpha = 1 - p$.
4. Show also that if the market game in Exercise 1 is now played with model h, the losses to Player 1 are bounded.

Exercise 1.3: Model Comparison**

Example 1.1 and the associated discussion alluded to the notion that some types of models are more common than others. This exercise will explore that concept briefly.

Recall Table 1.1 from Example 1.1:

G	$\boldsymbol{x}$
A	(0, 1)
B	(1, 1)
C	(1, 0)
C	(0, 0)

For this exercise, consider two models "similar" if they produce the same projections for G when applied to the values of $\boldsymbol{x}$ from Table 1.1 with probability strictly greater than 0.95.

In the following subsections, the goal will be to produce sets of mutually dissimilar models that all produce Table 1.1 with a given likelihood.

1. How many similar models produce Table 1.1 with likelihood 1.0?
2. Produce at least 4 dissimilar models that produce Table 1.1 with likelihood 0.9.
3. How many dissimilar models can produce Table 1.1 with likelihood exactly 0.95?

Exercise 1.4*: Likelihood Estimation
When the data is i.i.d., the negative of log-likelihood function (the "error function") for a binary classifier is the *cross-entropy*

$$E(\boldsymbol{\theta}) = -\sum_{i=1}^{n} G_i ln\ (g_1(\mathbf{x}_i \mid \boldsymbol{\theta})) + (1 - G_i) ln\ (g_0(\mathbf{x}_i \mid \boldsymbol{\theta})).$$

Suppose now that there is a probability π_i that the class label on a training data point $\mathbf{x}_i$ has been correctly set. Write down the error function corresponding to the negative log-likelihood. Verify that the error function in the above equation is obtained when $\pi_i = 1$. Note that this error function renders the model robust to incorrectly labeled data, in contrast to the usual least squares error function.

Exercise 1.5: Optimal Action**
Derive Eq. 1.17 by setting the derivative of Eq. 1.16 with respect to the time-t action u_t to zero. Note that Eq. 1.17 gives a non-parametric expression for the optimal action u_t in terms of a ratio of two conditional expectations. To be useful in practice, the approach might need some further modification as you will use in the next exercise.

Exercise 1.6*: Basis Functions**
Instead of non-parametric specifications of an optimal action in Eq. 1.17, we can develop a *parametric* model of optimal action. To this end, assume we have a set of basic functions $\psi_k(S)$ with $k = 1, \ldots, K$. Here K is the total number of basis functions—the same as the dimension of your model space.

We now define the optimal action $u_t = u_t(S_t)$ in terms of coefficients θ_k of expansion over basis functions Ψ_k (for example, we could use spline basis functions, Fourier bases, etc.) :

$$u_t = u_t(S_t) = \sum_{k=1}^{K} \theta_k(t) \Psi_k(S_t).$$

Compute the optimal coefficients $\theta_k(t)$ by substituting the above equation for u_t into Eq. 1.16 and maximizing it with respect to a set of weights $\theta_k(t)$ for a t-th time step.

Appendix

Answers to Multiple Choice Questions

Question 1
Answer: 1, 2.

Answer 3 is incorrect. While it is true that unsupervised learning does not require a human supervisor to train the model, it is false to presume that the approach is superior.

Answer 4 is incorrect. Reinforcement learning cannot be viewed as a generalization of supervised learning to Markov Decision Processes. The reason is that reinforcement learning uses rewards to reinforce decisions, rather than labels to define the correct decision. For this reason, reinforcement learning uses a weaker form of supervision.

Question 2
Answer: 1,2,3.

Answer 4 is incorrect. Two separate binary models $\{g_i^{(1)}(X|\theta)\}_{i=0}^{1}$ and $\{g_i^{(2)}(X|\theta)\}_{i=0}^{1}$ will, in general, not produce the same output as a single, multi-class, model $\{g_i(X|\theta)\}_{i=0}^{3}$. Consider, as a counter example, the logistic models $g_0^{(1)} = g_0(X|\theta_1) = \frac{\exp\{-X^T\theta_1\}}{1+\exp\{-X^T\theta_1\}}$ and $g_0^{(2)} = g_0(X|\theta_2) = \frac{\exp\{-X^T\theta_2\}}{1+\exp\{-X^T\theta_2\}}$, compared with the multi-class model

$$g_i(X|\boldsymbol{\theta}') = \text{softmax}(\exp\{X^T\boldsymbol{\theta}'\}) = \frac{\exp\{(X^T\boldsymbol{\theta}')_i\}}{\sum_{k=0}^{K}\exp\{(X^T\boldsymbol{\theta}')_k\}}. \tag{1.26}$$

If we set $\theta_1 = \boldsymbol{\theta}'_0 - \boldsymbol{\theta}'_1$ and $\boldsymbol{\theta}'_2 = \boldsymbol{\theta}'_3 = 0$, then the multi-class model is equivalent to Model 1. Similarly if we set $\theta_2 = \boldsymbol{\theta}'_2 - \boldsymbol{\theta}'_3$ and $\boldsymbol{\theta}'_0 = \boldsymbol{\theta}'_1 = 0$, then the multi-class model is equivalent to Model 2. However, we cannot simultaneously match the outputs of Model 1 and Model 2 with the multi-class model.

Question 3
Answer: 1,2,3.

Answer 4 is incorrect. The layers in a deep recurrent network provide more expressibility between each lagged input and the hidden state variable, but are unrelated to the amount of memory in the network. The hidden layers in any multilayered perceptron are not the hidden state variables in our time series model. It is the degree of unfolding, i.e. number of hidden state vectors which determines the amount of memory in any recurrent network.

Question 4
Answer: 2.

References

Akaike, H. (1973). *Information theory and an extension of the maximum likelihood principle* (pp. 267–281).

Akcora, C. G., Dixon, M. F., Gel, Y. R., & Kantarcioglu, M. (2018). Bitcoin risk modeling with blockchain graphs. *Economics Letters, 173*(C), 138–142.

Arnold, V. I. (1957). *On functions of three variables* (Vol. 114, pp. 679–681).

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics, 31*, 307–327.

Box, G. E. P., & Jenkins, G. M. (1976). *Time series analysis, forecasting, and control.* San Francisco: Holden-Day.

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1994). *Time series analysis, forecasting, and control* (third ed.). Englewood Cliffs, NJ: Prentice-Hall.

Breiman, L. (2001). Statistical modeling: the two cultures (with comments and a rejoinder by the author). *Statistical Science, 16*(3), 199–231.

Cont, R., & de Larrard, A. (2013). Price dynamics in a Markovian limit order market. *SIAM Journal on Financial Mathematics, 4*(1), 1–25.

de Prado, M. (2018). *Advances in financial machine learning*. Wiley.

de Prado, M. L. (2019). Beyond econometrics: A roadmap towards financial machine learning. *SSRN*. Available at SSRN: https://ssrn.com/abstract=3365282 or http://dx.doi.org/10.2139/ssrn.3365282.

DeepMind (2016). DeepMind AI reduces Google data centre cooling bill by 40%. https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/.

DeepMind (2017). The story of AlphaGo so far. https://deepmind.com/research/alphago/.

Dhar, V. (2013, December). Data science and prediction. *Commun. ACM, 56*(12), 64–73.

Dixon, M. (2018a). A high frequency trade execution model for supervised learning. *High Frequency, 1*(1), 32–52.

Dixon, M. (2018b). Sequence classification of the limit order book using recurrent neural networks. *Journal of Computational Science, 24*, 277–286.

Dixon, M., & Halperin, I. (2019). *The four horsemen of machine learning in finance*.

Dixon, M., Polson, N., & Sokolov, V. (2018). Deep learning for spatio-temporal modeling: Dynamic traffic flows and high frequency trading. *ASMB*.

Dixon, M. F., & Polson, N. G. (2019, Mar). Deep fundamental factor models. *arXiv e-prints*, arXiv:1903.07677.

Dyhrberg, A. (2016). Bitcoin, gold and the dollar – a GARCH volatility analysis. *Finance Research Letters*.

Elman, J. L. (1991, Sep). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning, 7*(2), 195–225.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., et al. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature, 542*(7639), 115–118.

Flood, M., Jagadish, H. V., & Raschid, L. (2016). Big data challenges and opportunities in financial stability monitoring. *Financial Stability Review,* (20), 129–142.

Gomber, P., Koch, J.-A., & Siering, M. (2017). Digital finance and fintech: current research and future research directions. *Journal of Business Economics, 7*(5), 537–580.

Gottlieb, O., Salisbury, C., Shek, H., & Vaidyanathan, V. (2006). Detecting corporate fraud: An application of machine learning. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.142.7470.

Graves, A. (2012). *Supervised sequence labelling with recurrent neural networks*. Studies in Computational intelligence. Heidelberg, New York: Springer.

Gu, S., Kelly, B. T., & Xiu, D. (2018). *Empirical asset pricing via machine learning*. Chicago Booth Research Paper 18–04.

Harvey, C. R., Liu, Y., & Zhu, H. (2016). ... and the cross-section of expected returns. *The Review of Financial Studies, 29*(1), 5–68.

Hornik, K., Stinchcombe, M., & White, H. (1989, July). Multilayer feedforward networks are universal approximators. *Neural Netw., 2*(5), 359–366.

Kearns, M., & Nevmyvaka, Y. (2013). Machine learning for market microstructure and high frequency trading. *High Frequency Trading - New Realities for Traders*.

Kercheval, A., & Zhang, Y. (2015). Modeling high-frequency limit order book dynamics with support vector machines. *Journal of Quantitative Finance, 15*(8), 1315–1329.

Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR, 114*, 953–956.

Kubota, T. (2017, January). Artificial intelligence used to identify skin cancer.

Kullback, S., & Leibler, R. A. (1951, 03). On information and sufficiency. *Ann. Math. Statist., 22*(1), 79–86.

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (1955, August). A proposal for the Dartmouth summer research project on artificial intelligence. http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html.

Philipp, G., & Carbonell, J. G. (2017, Dec). Nonparametric neural networks. *arXiv e-prints*, arXiv:1712.05440.

Philippon, T. (2016). *The fintech opportunity*. CEPR Discussion Papers 11409, C.E.P.R. Discussion Papers.

Pinar Saygin, A., Cicekli, I., & Akman, V. (2000, November). Turing test: 50 years later. *Minds Mach., 10*(4), 463–518.

Poggio, T. (2016). Deep learning: mathematics and neuroscience. *A Sponsored Supplement to Science Brain-Inspired intelligent robotics: The intersection of robotics and neuroscience*, 9–12.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

Sirignano, J., Sadhwani, A., & Giesecke, K. (2016, July). Deep learning for mortgage risk. *ArXiv e-prints*.

Sirignano, J. A. (2016). Deep learning for limit order books. *arXiv preprint arXiv:1601.01987*.

Sovbetov, Y. (2018). Factors influencing cryptocurrency prices: Evidence from Bitcoin, Ethereum, Dash, Litcoin, and Monero. *Journal of Economics and Financial Analysis, 2*(2), 1–27.

Stein, H. (2012). Counterparty risk, CVA, and Basel III.

Turing, A. M. (1995). *Computers & thought*. Chapter Computing Machinery and Intelligence (pp. 11–35). Cambridge, MA, USA: MIT Press.

Wiener, N. (1964). *Extrapolation, interpolation, and smoothing of stationary time series*. The MIT Press.