Machine Learning in Finance: From Theory to Practice Instructor's Manual

Matthew F. Dixon, Igor Halperin and Paul Bilokon

Matthew Dixon
Department of Applied Math, Illinois Institute of Technology e-mail: matthew.dixon@iit.edu

Igor Halperin
NYU Tandon School of Engineering and Fidelity Investments, e-mail: igor.halperin@nyu.edu, e-mail: igor.halperin@fmr.com

Paul Bilokon
Thalesians Ltd, London, e-mail: paul@thalesians.com

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?

Advantage of the Book

This book is written for advanced graduate students and academics in the mathematical sciences, in addition to quants and data scientists in the field of finance. Readers will find it useful as a bridge from these well-established foundational topics to applications of machine learning in finance. Machine learning is presented as a non-parametric extension of financial econometrics, 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 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.
- Multi-choice questions, numerical examples and approximately 80 end-of-chapter exercises are used throughout the book to reinforce the main 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. The codes have provided so that they can either be presented as laboratory session material or used as a programming assignment.

Recommended Course Syllabus

This book has been written as an introductory text book for a graduate course in machine learning in finance for students with strong mathematical preparation in probability, statistics, and time series analysis. The book therefore assumes, and does not provide, concepts in elementary probability and statistics. In particular, undergraduate preparation in probability theory should include discrete and continuous random variables, conditional probabilities and expectations, and Markov chains. Statistics preparation includes experiment design, statistical inference, regression and logistic regression models, and analysis of time series, with examples in ARMA models. Preparation in financial econometrics and Bayesian statistics in addition to some experience in the capital markets or in investment management is advantageous but not necessary.

Our experience in teaching upper section undergraduate and graduate programs in machine learning in finance and related courses in the departments of applied math and financial engineering have been that students with little programming skills, despite having strong math backgrounds, have difficulty with the programming assignments. It is therefore our recommendation that a course in Python programming be a prerequisite or that a Python bootcamp be run in conjunction with the beginning of the course. The course should equip students with a solid foundation in data structures, elementary algorithms and control flow in Python. Some supplementary material to support programming has been been provided in the Appendices of the book, with references to further supporting material.

Students with a background in computer science often have a distinct advantage in the programming assignments, but often need to be referred to other textbooks on probability and time series analysis first. Exercises at the end of Chapter 1 will be especially helpful in adapting to the mindset of a quant, with the focus on economic games and simple numerical puzzles. In general we encourage liberal use of these applied probability problems as they aid understanding of the key mathematical ideas and build intuition for how they translate into practice.

Overview of the Textbook

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.

This chapter is written to be self-contained and could form the basis of a bootcamp or short workshop on machine learning in finance. The end-of-chapter questions have also been designed to assess the students' background in probability and quantitative finance.

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. 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–Merton 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 (LQR). 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, 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 part 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.”

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 taking a few minutes to complete, through to (***), which denotes a significantly more complex exercise. Unless specified otherwise, all equations referred in each exercise correspond to those in the corresponding chapter.

Python Notebooks

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 textbook chapter accompanied by notebooks, and the `README.md` in the subfolder of each textbook chapter, for further instructions and details.

Instructor Materials

This Instructor's Manual provides worked solutions to almost all of the end-of-chapter questions. Additionally this Instructor's Manual is accompanied by a folder with notebook solutions to some of the programming assignments. Occasionally, some addition notes on the notebook solution are also provided.

Contents

Part I
Machine Learning with Cross-Sectional Data

Chapter 1
Introduction

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

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

a. Given that α is known to Player 2, state the strategy[1] that will give Player 2 an expected payoff, over multiple games, of \$1 without knowing p.
b. Suppose now that p is known to both players. In a given round, what is the optimal choice of α for Player 1?
c. Suppose Player 2 knows with complete certainty, that G will be 1 for a particular round, what will be the payoff for Player 2?
d. Suppose Player 2 has complete knowledge in rounds $\{1, ..., i\}$ 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

[1] 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.

dollar costs, so that, for example, if Player 2 pays Player 1 \$1.5 to enter the game, then the payoff will be $1.5V_1$.

Solution 1.1

a. If Player 2 chooses payoff V_1 with probability α and V_2 otherwise then the payoff will be

$$\frac{\alpha p}{\alpha} + \frac{(1-\alpha)(1-p)}{(1-\alpha)} = p + (1-p) = 1.$$

So Player 2 is expected to win or at least break even, regardless of their level of knowledge merely because Player 1 had to move first.

b. If Player 1 chooses $\alpha = p$, then the expected payoff to Player 2 is exactly \$1. Suppose Player 1 chooses $\alpha > p$, then Player 2 takes V_2 and the payoff is

$$\frac{1-p}{1-\alpha} > 1.$$

And similarly if Player 1 chooses $\alpha < p$, then Player 2 chooses V_1 and also expects to earn more than \$1.

N.B.: If Player 1 chooses $\alpha = p$, then Player 2 in expectation earns exactly \$1 regardless of the choices made by Player 2.

c. Since $G = 1$ is known to Player 2, they choose V_1 and earn $\$\frac{1}{\alpha}$.

d. After the first round, Player 2 will have received $\$\frac{1}{\alpha}$. Reinvesting this in the second round, the payoff will be $\$\frac{1}{\alpha^2}$. Therefore after i rounds, the payer to Player 2 will be:

$$\frac{1}{\prod_{k=1}^{i} \alpha_k}.$$

N.B.: This problem and the next are developing the intuition for the use of logarithms in entropy. One key part of this logic is that the "right" way to consider the likelihood of a dataset is by multiplying together the probability of each observation, not summing or averaging them. Another reason to prefer a product is that under a product the individual events form a σ-algebra, such that any subset of events is itself an event that is priced fairly by the game. For instance, a player can choose to bet on events $i = 1$ and $i = 2$ both happening, and the resulting event is priced fairly, at odds calculated from its actual probability of occurrence, $p_1 p_2$.

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 as the results of some experiment.

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

Table 1: *Sample model data.*

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.

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

$$g((1, 0)) = (0, 0, 1), \text{ Predicts C with 100\% certainty.} \tag{2}$$

$$h((1, 0)) = (0, 0.5, 0.5), \text{ Predicts B or C with 50\% certainty.} \tag{3}$$

a. Show that each model has the same total absolute error, over the samples where $\boldsymbol{x} = (1, 0)$.
b. Show that all three models assign the same average probability to the values from Table 1 when $\boldsymbol{x} = (1, 0)$.
c. 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 $\alpha = 1 - p$.
d. Show also that if the market game in Exercise 1 is now played with model h, the losses to Player 1 are bounded.

Solution 1.2

a. Model f has an absolute error of $1 + 0$, g has an error of $0 + 1$, and h has an error of $0.5 + 0.5$, so they are all the same.
b. The calculation the error is the same as above, $1 + 0 = 0 + 1 = 0.5 + 0.5$.
c. In the fourth row, Player 1 pays out $\frac{1}{0}$, if $\alpha = 1 - p$, and p denotes the model confidence of event B or C, which is unbounded.
d. Player 1 pays out $\frac{1}{0.5} + \frac{1}{0.5} = 4$, which is bounded.

Additional problems of this sort can be generated by simply requiring that the data set in question have at least two rows with identical values of $\boldsymbol{x}$, but different values of G. This ensures that no model could predict all of the events (since a model must be a function of $\boldsymbol{x}$), and thus it is necessary to consider the probabilities assigned to mispredicted events.

Provided that some of the models assign absolute certainty ($p = 1.0$) to some incorrectly predicted outcomes, the unbounded losses in Part (3) will occur for those models.

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.

a. How many similar models produce Table 1.1 with likelihood 1.0?
b. Produce at least 4 dissimilar models that produce Table 1.1 with likelihood 0.9.
c. How many dissimilar models can produce Table 1.1 with likelihood exactly 0.95?

Solution 1.3

a. There is only one model that can produce Table 1.1 with likelihood 1.0, it is

$$g(\boldsymbol{x}) = \begin{cases} \{1, 0, 0\} & \text{if } \boldsymbol{x} = (0, 1) \\ \{0, 1, 0\} & \text{if } \boldsymbol{x} = (1, 1) \\ \{0, 0, 1\} & \text{if } \boldsymbol{x} = (1, 0) \\ \{0, 0, 1\} & \text{if } \boldsymbol{x} = (0, 0) \end{cases} \tag{4}$$

There are no dissimilar models that can produce the output in Table 1.1 with likelihood 1.0.

b. This can be done many ways, but most easily by perfectly predicting some rows and not others. One such set of models is:

$$g_1(\boldsymbol{x}) = \begin{cases} \{0.9, 0.1, 0\} & \text{if } \boldsymbol{x} = (0,1) \\ \{0,1,0\} & \text{if } \boldsymbol{x} = (1,1) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (1,0) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (0,0) \end{cases} \tag{5}$$

$$g_2(\boldsymbol{x}) = \begin{cases} \{1,0,0\} & \text{if } \boldsymbol{x} = (0,1) \\ \{0.1,0.9,0\} & \text{if } \boldsymbol{x} = (1,1) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (1,0) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (0,0) \end{cases} \tag{6}$$

$$g_3(\boldsymbol{x}) = \begin{cases} \{1,0,0\} & \text{if } \boldsymbol{x} = (0,1) \\ \{0,1,0\} & \text{if } \boldsymbol{x} = (1,1) \\ \{0,0.1,0.9\} & \text{if } \boldsymbol{x} = (1,0) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (0,0) \end{cases} \tag{7}$$

$$g_3(\boldsymbol{x}) = \begin{cases} \{1,0,0\} & \text{if } \boldsymbol{x} = (0,1) \\ \{0,1,0\} & \text{if } \boldsymbol{x} = (1,1) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (1,0) \\ \{0,0.1,0.9\} & \text{if } \boldsymbol{x} = (0,0) \end{cases} \tag{8}$$

When generating models of this form, it is not necessary to use 0.9, 0.1, and 0.0. If these three values are labeled α, β, and γ, then it is enough that $\alpha^2 < 0.95$, $\alpha >= 0.9$, and $\alpha + \beta + \gamma = 1.0$.

c. It's clear from the setup that any models that give the table with likelihood 0.95 are right on the boundary of being dissimilar, so they must not in any other circumstance agree. There are an infinite number of models. For example, here are two

$$g_1(\boldsymbol{x}) = \begin{cases} \{0.95, 0.05, 0\} & \text{if } \boldsymbol{x} = (0,1) \\ \{0,1,0\} & \text{if } \boldsymbol{x} = (1,1) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (1,0) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (0,0) \end{cases} \tag{9}$$

$$g_2(\boldsymbol{x}) = \begin{cases} \{0.95, 0, 0.05\} & \text{if } \boldsymbol{x} = (0,1) \\ \{0,1,0\} & \text{if } \boldsymbol{x} = (1,1) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (1,0) \\ \{0,0,1\} & \text{if } \boldsymbol{x} = (0,0) \end{cases} \tag{10}$$

but they could have easily have been generated with $0.95 + a + (0.05 - a), a \in [0, 0.05]$.

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*

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

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.

Solution 1.4

Given the probability π_i that the class label is correctly assigned, the error function can be written as

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

Clearly when $\pi_i = 1$ we recover the cross-entropy given by the 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 Equation 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.

Solution 1.5

Setting the derivative of Eq. 1.16 w.r.t. u_t to zero gives

$$\frac{\partial}{\partial u_t}\mathbb{E}\left[\sum_{t=0}^{T} u_t\phi_t - \lambda u_t^2 \mathbb{V}\left[\phi_t | S_t\right] \middle| S_t = s\right] = 0.$$

This gives

$$\mathbb{E}\left[\phi_t | S_t\right] - 2\lambda u_t \mathbb{V}\left[\phi_t | S_t\right] = 0$$

and rearranging gives the optimal action:

$$u_t = \frac{\mathbb{E}\left[\phi_t | S_t\right]}{2\lambda \mathbb{V}\left[\phi_t | S_t\right]}.$$

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.

Solution 1.6

Plugging the basis function representation for u_t into Eq. 1.16, setting the derivative of the resulting expressing with respect to $\theta_{k'} = \theta_{k'}(t)$, we obtain a system of linear equations for each $k' = 1, \ldots, K$:

$$2\lambda \sum_{k=1}^{K} \theta_k \mathbb{E}\left[\Psi_k(S) \Psi_{k'}(S) \mathrm{Var}\left[\phi | S\right]\right] = \mathbb{E}\left[\Psi_{k'}(S)\right].$$

This system of linear equations can be conveniently solved by defining a pair of a matrix A and vector B whose elements are defined as follows:

$$A_{kk'} = \mathbb{E}\left[\Psi_k(S) \Psi_{k'}(S) \mathrm{Var}\left[\phi | S\right]\right], \quad B_{k'} = \mathbb{E}\left[\Psi_{k'}(S)\right].$$

The solution of the linear system above is then given by the following simple matrix-valued formula for the vector θ of parameters θ_k:

$$\theta = \frac{1}{2\lambda} \mathbf{A}^{-1} \mathbf{B}.$$

Note that this relation involves matrix inversion, and in practice might have to be regularized by adding a unity matrix to matrix **A** with a tiny regularization parameter before the matrix inversion step.

Chapter 2
Probabilistic Modeling

Exercise 2.1: Applied Bayes' Theorem

An accountant is 95 percent effective in detecting fraudulent accounting when it is, in fact, present. However, the audit also yields a "false positive" result for one percent of the non-fraudulent companies audited. If 0.1 percent of the companies are actually fraudulent, what is the probability that a company is fraudulent given that the audit revealed fraudulent accounting?

Solution 2.1

Let $\mathcal{F}$ denote the presence of fraud, $\mathcal{F}$ denote its absence, and + denote a positive audit result (i.e. the audit revealed fraudulent accounting). Then $\mathbb{P}(+|\mathcal{F}) = .95$, $\mathbb{P}(+|\mathcal{F}^c) = .01$, and $\mathbb{P}(\mathcal{F}) = .001$. Then according to Bayes' theorem

$$\mathbb{P}(\mathcal{F}|+) = \frac{\mathbb{P}(\mathcal{F}) \cdot \mathbb{P}(+|\mathcal{F})}{\mathbb{P}(+)} = \frac{.001(.95)}{.001(.95) + .999(.01)} = .0868.$$

Exercise 2.2*: FX and Equity

A currency strategist has estimated that JPY will strengthen against USD with probability 60% if S&P 500 continues to rise. JPY will strengthen against USD with probability 95% if S&P 500 falls or stays flat. We are in an upward trending market at the moment, and we believe that the probability that S&P 500 will rise is 70%. We then learn that JPY has actually strengthened against USD. Taking this new information into account, what is the probability that S&P 500 will rise? Hint: Recall Bayes' rule: $P(A \mid B) = \frac{P(B \mid A)}{P(B)} P(A)$.

Solution 2.2

Denote the events as

- H = “S&P continues to rise”,
- E = “JPY strengthens against USD”.

Then

$$\mathbb{P}[E \mid H] = 0.6, \quad \mathbb{P}\left[E \mid \bar{H}\right] = 0.95, \quad \mathbb{P}[H] = 0.7.$$

By Bayes’s rule,

$$\mathbb{P}[H \mid E] = \frac{\mathbb{P}[E \mid H]\,\mathbb{P}[H]}{\mathbb{P}[E]},$$

and we can use the Law of Total Probability to rewrite the denominator to get

$$\mathbb{P}[H \mid E] = \frac{\mathbb{P}[E \mid H]\,\mathbb{P}[H]}{\mathbb{P}[E \mid H]\,\mathbb{P}[H] + \mathbb{P}\left[E \mid \bar{H}\right]\mathbb{P}\left[\bar{H}\right]},$$

so we can now substitute in the given probabilities and obtain the answer:

$$\mathbb{P}[H \mid E] = \frac{0.6 \times 0.7}{0.6 \times 0.7 + 0.95 \times (1 - 0.7)} = 0.595 \approx 60\%,$$

and so $\mathbb{P}[H \mid E] < \mathbb{P}[H]$, the posterior is less than the prior. This decrease in probability is due to the fact that the evidence supports $\bar{H}$ more strongly than H: $\mathbb{P}\left[E \mid \bar{H}\right] > \mathbb{P}[E \mid H]$.

Exercise 2.3**: Bayesian inference in trading

Suppose there are n conditionally independent, but not identical, Bernoulli trials $G_1, \ldots, G_n$ generated from the map $P(G_i = 1 \mid X = x_i) = g_1(x_i \mid \theta)$ with $\theta \in [0, 1]$. Show that the likelihood of $G \mid X$ is given by

$$p(G \mid X, \theta) = \prod_{i=1}^{n} (g_1(x_i \mid \theta))^{G_i} \cdot (g_0(x_i \mid \theta))^{1-G_i} \tag{11}$$

and the log-likelihood of $G \mid X$ is given by

$$ln\ p(G \mid X, \theta) = \sum_{i=1}^{n} G_i ln(g_1(x_i \mid \theta)) + (1 - G_i) ln(g_0(x_i \mid \theta)). \tag{12}$$

Using Bayes’ rule, write the condition probability density function of θ (the "posterior") given the data (X, G) in terms of the above likelihood function.

From the previous example, suppose that $G = 1$ corresponds to JPY strengthening against the dollar and X are the S&P 500 daily returns and now

$$g_1(x \mid \theta) = \theta \mathbb{1}_{x>0} + (\theta + 35)1_{x\leq 0}. \tag{13}$$

Starting with a neutral view on the parameter θ (i.e. $\theta \in [0,1]$), learn the distribution of the parameter θ given that JPY strengthens against the dollar for two of the three days and S&P 500 is observed to rise for 3 consecutive days. Hint: You can use the Beta density function with a scaling constant $\Gamma(\alpha, \beta)$

$$p(\theta|\alpha,\beta) = \frac{(\alpha+\beta-1)!}{(\alpha-1)!(\beta-1)!}\theta^{\alpha-1}(1-\theta)^{\beta-1} = \Gamma(\alpha,\beta)\theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{14}$$

to evaluate the integral in the marginal density function.
If θ represents the currency analyst's opinion of JPY strengthening against the dollar, what is the probability that the model overestimates the analyst's estimate?

Solution 2.3

If the trials are conditionally independent then

$$p(G \mid X, \theta) = \prod_{i=1}^{n} p(G = G_i \mid X = x_i, \theta)$$

and since the conditional probability of $G_i = 1$ given $X = x_i$ is $p_i = g_1(x_i \mid \theta))$ it follows that

$$p(G \mid X, \theta) = \prod_{i=1}^{n} p(G = G_i \mid X = x_i, \theta) = p_i^{G_i}(1-p_i)^{1-G_i}$$

and taking logs gives

$$ln\ p(G \mid X, \theta) = \sum_{i=1}^{n} ln\ p(G = G_i \mid X = x_i, \theta) = \sum_{i=1}^{n} G_i ln\ p_i + (1-G_i) ln\ (1-p_i).$$

From Bayes' rule, the conditional density function of θ (the "posterior") is given by

$$p(\theta \mid G, X) = \frac{p(G \mid X, \theta)}{p(G \mid X)} p(\theta).$$

Since $p(\theta = 0.6) = 1$ and zero for all estimates (the currency strategist's "prior")

From the previous example, suppose that $G = 1$ corresponds to JPY strengthening against the dollar and X are the S&P 500 daily returns and now

$$g_1(x \mid \theta) = \theta \mathbf{1}_{x>0} + (\theta + 35)\mathbf{1}_{x\leq 0}.$$

Starting with a uniform prior $p(\theta) = 1,\ \forall \theta \in [0,1]$, learn the distribution of the parameter θ given that JPY strengthens against the dollar for two of the three days and S&P 500 is observed to rise for 3 consecutive days. So the likelihood function is given by $\theta^2(1-\theta)$ and from Bayes' law:

$$p(\theta \mid G, X) = \frac{p(G \mid X, \theta)}{p(G \mid X)} p(\theta) = \frac{\theta^2(1-\theta)}{\int_0^1 \theta^2(1-\theta)d\theta} = \Gamma(3,2)\theta^2(1-\theta).$$

where we have used the Beta density function with a scaling constant $\Gamma(\alpha, \beta)$

$$p(\theta|\alpha,\beta) = \frac{(\alpha+\beta-1)!}{(\alpha-1)!(\beta-1)!}\theta^{\alpha-1}(1-\theta)^{\beta-1} = \Gamma(\alpha,\beta)\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

to evaluate the integral in the marginal density function. The probability that the model overestimates the analyst's estimate is:

$$\mathbb{P}[\theta > 0.6 \mid G, X] = \Gamma(3,2)\int_{0.6}^{1} \theta^2(1-\theta)d\theta = \Gamma(3,2)(1-\int_{0}^{0} .6\theta^2(1-\theta)d\theta = 12\left[\frac{\theta^3}{3} - \frac{\theta^4}{4}\right]_0^{0.6} = 0.4752.$$

Exercise 2.4*: Bayesian inference in trading

Suppose that you observe the following daily sequence of directional changes in the JPY/USD exchange rate (U (up), D(down or stays flat)):

```
U, D, U, U, D
```

and the corresponding daily sequence of S&P 500 returns is

```
-0.05, 0.01, -0.01, -0.02, 0.03
```

You propose the following probability model to explain the behavior of JPY against USD given the directional changes in S&P 500 returns: Let G denote a Bernoulli R.V., where $G = 1$ corresponds to JPY strengthening against the dollar and r are the S&P 500 daily returns. All observations of G are conditionally independent (but *not* identical) so that the likelihood is

$$p(G \mid r, \theta) = \prod_{i=1}^{n} p(G = G_i \mid r = r_i, \theta)$$

where

$$p(G_i = 1 \mid r = r_i, \theta) = \begin{cases} \theta_u, & r_i > 0 \\ \theta_d, & r_i \leq 0 \end{cases}$$

Compute the full expression for the likelihood that the data was generated by this model.

Solution 2.4

The data $D = (G, r)$ consists of all contemporaneous observations of G and r. The trials are conditionally independent then the joint likelihood of G given r and θ can be written as the product of marginal conditional density functions of G:

$$p(G \mid r, \theta) = \prod_{i=1}^{n} p(G = G_i \mid r = r_i, \theta)$$

and since the conditional probability of $G_i = 1$ given $r = r_i$ is $\theta_i \in \{\theta_u, \theta_d\}$, the data yields

$$\begin{aligned} p(G \mid r, \theta) = \prod_{i=1}^{n} p(G = G_i \mid r = r_i, \theta) &= \theta_i^{G_i}(1-\theta_i)^{1-G_i} \\ &= \theta_d \cdot (1-\theta_u) \cdot \theta_d \cdot \theta_d \cdot (1-\theta_u) \\ &= \theta_d^3 (1-\theta_u)^2. \end{aligned}$$

Exercise 2.5: Model comparison

Suppose you observe the following daily sequence of direction changes in the stock market (U (up), D(down)):

```
U, D, U, U, D, D, D, D, U, U, U, U, U, U, U, D, U, D, U, D,
U, D, D, D, D, U, U, D, U, D, U, U, U, D, U, D, D, D, U, U,
D, D, D, U, D, U, D, U, D, D
```

You compare two models for explaining its behavior. The first model, $\mathcal{M}_1$, assumes that the probability of an upward movement is fixed to 0.5 and the data is i.i.d.

The second model, $\mathcal{M}_2$, also assumes the data is i.i.d. but that the probability of an upward movement is set to an unknown $\theta \in \Theta = (0, 1)$ with a uniform prior on $\theta : p(\theta|\mathcal{M}_2) = 1$. For simplicity, we additionally choose a uniform model prior $p(\mathcal{M}_1) = p(\mathcal{M}_2)$.

Compute the model evidence for each model.

Compute the Bayes' factor and indicate which model should we prefer in light of this data?

Solution 2.5

There are $n = 50$ observations of which 25 are D and the remaining 25 are U. Let X denote the number of Us. The first model, $\mathcal{M}_1$ assumes that the probability of an upward movement is fixed to 0.5 and the data is i.i.d. The second model, $\mathcal{M}_2$, assumes the probability of an upward movement is set to an unknown $\theta \in \Theta = (0, 1)$ with a uniform prior density on $\theta : p(\theta|\mathcal{M}_2) = 1$. For simplicity, we additionally choose a uniform model prior $p(\mathcal{M}_1) = p(\mathcal{M}_2)$.

Which model is most likely to have generated the given data? We compute the model evidence for each model. The model evidence for $\mathcal{M}_2$ is the probability mass function

$$p(X = 25|\mathcal{M}_2) = \binom{50}{25}\frac{1}{2^{50}} = 0.11227.$$

The model evidence of $\mathcal{M}_2$ requires integrating over θ:

$$\begin{aligned} p(X = 25|\mathcal{M}_2) &= \int_0^1 p(X = 25|\theta, \mathcal{M}_2)p(\theta|\mathcal{M}_2)d\theta \\ &= \int_0^1 \binom{50}{25}\theta^{25}(1-\theta)^{25}d\theta \\ &= \frac{\binom{50}{25}}{\Gamma(26, 26)} = 0.01960784. \end{aligned}$$

Note that we have used the definition of the Beta density function with a scaling constant $\Gamma(\alpha, \beta)$

$$p(\theta|\alpha, \beta) = \frac{(\alpha+\beta-1)!}{(\alpha-1)!(\beta-1)!}\theta^{\alpha-1}(1-\theta)^{\beta-1} = \Gamma(\alpha, \beta)\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

to evaluate the integral in the marginal density function above, where $\alpha = 26$ and $\beta = 26$. $\Gamma(26, 26) = 6.4469 \times 10^{15}$ and $\binom{50}{25} = 1.264 \times 10^{14}$.

The Bayes' factor in favor of $\mathcal{M}_1$ is 5.7252 and thus $|lnBF| = 1.744877$ and, from Jeffrey's table, we see that there is weak evidence in favor of $\mathcal{M}_1$ since the log of the factor is between 1 and 2.5.

Exercise 2.6: Bayesian prediction and updating

Using Bayesian prediction, predict the probability of an upward movement given the best model and data in Exercise 2.5.

Suppose now that you observe the following new daily sequence of direction changes in the stock market (U (up), D(down)):

```
D, U, D, D, D, D, U, D, U, D, U, D, D, D, U, U, D, U, D, D, D,
U, U, D, D, D, U, D, U, D, U, D, D, D, U, D, U, D, U, D, D, D,
D, U, U, D, U, D, U, U
```

Using the best model from Exercise 2.5, compute the new posterior distribution function based on the new data and the data in the previous question and predict the probability of an upward price movement given all data. State all modeling assumptions clearly.

Solution 2.6

Part I If we choose $\mathcal{M}_2$ (even though there is weak evidence for the "best" model being $\mathcal{M}_1$), we can write that the density of the new predicted value y' (assuming it is just one new data point) given the previous data y is the expected value of the likelihood of the new data under the posterior density $p(\theta|y,\mathcal{M}_2)$:

$$p(y'|y,\mathcal{M}_2) = \mathbb{E}_{\theta|y}[p(y'|y,\theta,\mathcal{M}_2)] = \int_0^1 p(y'|y,\theta,\mathcal{M}_2)p(\theta|y,\mathcal{M}_2)d\theta.$$

The model evidence for $\mathcal{M}_2$, written in terms of y, in the previous question is

$$p(y|\mathcal{M}_2) = \int_0^1 p(y|\theta,\mathcal{M}_2)p(\theta|\mathcal{M}_2)d\theta = \frac{1}{\Gamma(26,26)} = 1.5511 \times 10^{-16}$$

since the likelihood function is

$$p(y|\theta,\mathcal{M}_2) = \prod_{i=1}^{n} \theta^{y_i}(1-\theta)^{1-y_i} = \theta^{\sum_{i=1}^n y_i}(1-\theta)^{\sum_{i=1}^n(1-y_i)} = \theta^x(1-\theta)^{n-x} = \theta^{25}(1-\theta)^{25}$$

where we have mapped Bernoulli trials y_i to 1 and zeros instead of U's and D's. Note that we dropped the $\binom{n}{x}$ term since this is a likelihood function over all y.

By Bayes' law, the posterior density is

$$p(\theta|y,\mathcal{M}_2) = \frac{p(y|\theta,\mathcal{M}_2)}{p(y|\mathcal{M}_2)}p(\theta|\mathcal{M}_2) = \frac{\theta^{25}(1-\theta)^{25}}{\Gamma(26,26)^{-1}} \cdot 1,$$

and the prediction density is

$$p(y'|y,\mathcal{M}_2) = \mathbb{E}_{\theta|y}[p(y'|y,\theta,\mathcal{M}_2)] = \Gamma(26,26)\int_0^1 \theta^{y'}(1-\theta)^{1-y'}\theta^{25}(1-\theta)^{25}d\theta$$

where we have used the likelihood function $p(y'|y,\theta,\mathcal{M}_2) = \theta^{y'}(1-\theta)^{1-y'}$. So the prediction of $y' = 1$ is the probability mass function for the upward movement

$$p(y' = 1|y,\mathcal{M}_2) = \Gamma(26,26)\int_0^1 \theta^{26}(1-\theta)^{25}d\theta = \frac{\Gamma(26,26)}{\Gamma(27,26)} = 0.5$$

and downward movement

$$p(y' = 0|y,\mathcal{M}_2) = \Gamma(26,26)\int_0^1 \theta^{25}(1-\theta)^{26}d\theta = \frac{\Gamma(26,26)}{\Gamma(26,27)} = 0.5.$$

and they sum to 1.

Part II Now let's learn again use the new data y' given in the question which consists of 50 new data points, of which 30 are D's and 20 are U's. We use the Bayesian updating ('online learning') formula

$$p(\theta|y', y, \mathcal{M}_2) = \frac{p(y'|\theta, \mathcal{M}_2)p(\theta|y, \mathcal{M}_2)}{\int_{\theta\in\Theta} p(y'|\theta, \mathcal{M}_2)p(\theta|y, \mathcal{M}_2)d\theta}. \tag{15}$$

The new model evidence is

$$p(y'|y, \mathcal{M}_2) = \int_0^1 p(y'|\theta, \mathcal{M}_2)p(\theta|y, \mathcal{M}_2)d\theta$$

and using the likelihood function evaluated on the new data

$$p(y'|\theta, \mathcal{M}_2) = \theta^{20}(1-\theta)^{30}$$

and recalling the posterior over y

$$p(\theta|y, \mathcal{M}_2) = \theta^{25}(1-\theta)^{25}\Gamma(26, 26)$$

we have

$$p(y'|y, \mathcal{M}_2) = \int_0^1 \theta^{20}(1-\theta)^{30}\theta^{25}(1-\theta)^{25}\Gamma(26, 26)d\theta = \frac{\Gamma(26, 26)}{\Gamma(46, 56)}.$$

so substituting the new model evidence and the new likelihood into Equation 15 for the new posterior density is

$$p(\theta|y', y, \mathcal{M}_2) = \theta^{45}(1-\theta)^{55}\Gamma(46, 56).$$

Part III Finally let's compute the probability again now assuming that y^* is the next observation and we will merge y', y. So now (y, y') contains 102 observations of the stock market variable Y.

The prediction density is now

$$p(y^*|y, y', \mathcal{M}_2) = \mathbb{E}_{\theta|y,y'}[p(y^*|y, y', \theta, \mathcal{M}_2)] = \Gamma(46, 56)\int_0^1 \theta^{y^*}(1-\theta)^{1-y^*}\theta^{45}(1-\theta)^{55}d\theta$$

and evaluating the density at $y^* = 1$:

$$p(y^* = 1|y, y', \mathcal{M}_2) = \frac{\Gamma(46, 56)}{\Gamma(47, 56)} = 0.4509804.$$

The probability that the next value y^* will be a one has decreased after observing all data. This is because there are relatively more D's in the new data than in the old data.

Using Model 1 Under Model $\mathcal{M}_1$

$$p(y|\mathcal{M}_1) = \int_0^1 p(y|\theta, \mathcal{M}_1)p(\theta|\mathcal{M}_1)d\theta = \frac{1}{2^{50}}$$

where $p(\theta|\mathcal{M}_1) = 1$ when $\theta = 1/2$ and $p(\theta|\mathcal{M}_1) = 0$ otherwise (i.e., the prior is a Dirac delta function) and the likelihood function is

$$p(y|\theta = 1/2, \mathcal{M}_1) = \frac{1}{2^{50}}.$$

By Bayes' law, the posterior density can only be 1

$$p(\theta|y, \mathcal{M}_1) = \frac{p(y|\theta, \mathcal{M}_1)}{p(y|\mathcal{M}_1)} p(\theta|\mathcal{M}_1) = 1, \theta = 1/2$$

and zero for all other values of θ. The prediction density, for either up or downward movements is,

$$p(y'|y, \mathcal{M}_1) = \mathbb{E}_{\theta|y}[p(y'|y, \theta = 1/2, \mathcal{M}_1)] = \frac{1}{2} \cdot 1.$$

Note that the prediction will not change under new data.

N.B.: An alternative formulation writes the old posterior in the previous question conditioned on $X = x$ and the new posterior conditioned on $X' = x'$ and $X = x$. The difference will be a factor up to the n choose x operator, which will cancel through.

Exercise 2.7: Logistic regression is Naive Bayes

Suppose that G and $X \in \{0, 1\}^p$ are Bernoulli random variables and the X_is are mutually independent given G—that is, $\mathbb{P}[X \mid G] = \prod_{i=1}^{p} \mathbb{P}[X_i \mid G]$. Given a naive Bayes' classifier $\mathbb{P}[G \mid X]$, show that the following logistic regression model produces equivalent output if the weights are

$$\begin{aligned} w_0 &= \log \frac{\mathbb{P}[G]}{\mathbb{P}[G^c]} + \sum_{i=1}^{p} \log \frac{\mathbb{P}[X_i = 0 \in G]}{\mathbb{P}[X_i = 0 \in G^c]} \\ w_i &= \log \frac{\mathbb{P}[X_i = 1 \in G]}{\mathbb{P}[X_i = 1 \in G^c]} \cdot \frac{\mathbb{P}[X_i = 0 \in G^c]}{\mathbb{P}[X_i = 0 \in G]}, \quad i = 1, \ldots, p. \end{aligned}$$

Solution 2.7

The Naive Bayes' classifier is given by:

$$\begin{aligned}\mathbb{P}[G \mid X] &= \frac{\mathbb{P}[X \mid G]\,\mathbb{P}[G]}{\mathbb{P}[X]} \\ &= \frac{\mathbb{P}[X \mid G]\,\mathbb{P}[G]}{\mathbb{P}[X \mid G]\,\mathbb{P}[G] + \mathbb{P}[X \mid G^c]\,\mathbb{P}[G^c]} \\ &= \frac{1}{1 + \frac{\mathbb{P}[X \mid G^c]\mathbb{P}[G^c]}{\mathbb{P}[X \mid G]\,\mathbb{P}[G]}} \\ &= \frac{1}{1 + \exp\left[\ln \frac{\mathbb{P}[X \mid G^c]\mathbb{P}[G^c]}{\mathbb{P}[X \mid G]\mathbb{P}[G]}\right]} \\ &= \frac{1}{1 + \exp\left[-\ln \frac{\mathbb{P}[X \mid G]\mathbb{P}[G]}{\mathbb{P}[X \mid G^c]\mathbb{P}[G^c]}\right]}\end{aligned} \tag{16}$$

We want

$$\mathbb{P}[G \mid X] = \frac{1}{1 + \exp\left[-\sum_{i=0}^{p} w_i X_i\right]} \tag{17}$$

By combining Eq. 16 and Eq. 17 we get:

$$-\log \frac{\mathbb{P}[X \mid G]\,\mathbb{P}[G]}{\mathbb{P}[X \mid G^c]\,\mathbb{P}[G^c]} = -\sum_{i=0}^{p} w_i X_i.$$

Using Naive Bayes' assumption and some simplifications we have:

$$\sum_{i=0}^{n} w_i x_i = \log \frac{\mathbb{P}[G]}{\mathbb{P}[G^c]} + \sum_{i=1}^{n} \log \frac{\mathbb{P}[x_i \mid G]}{\mathbb{P}[x_i \mid G^c]}. \tag{18}$$

Now we want the RHS of Eq. 18 to be a linear function of X_i's, so we perform the following substitutions ($i > 0$):

$$\begin{aligned}\log \frac{\mathbb{P}[X_i \mid G]}{\mathbb{P}[X_i \mid G^c]} &= X_i \log \frac{\mathbb{P}[X_i = 1 \mid G]}{\mathbb{P}[X_i = 1 \mid G^c]} \\ &\quad + (1 - X_i) \log \frac{\mathbb{P}[X_i = 0 \mid G]}{\mathbb{P}[X_i = 0 \mid G^c]}.\end{aligned} \tag{19}$$

By combining Eq. 18 and Eq. 19 we can show equivalence. To solve for w_0 we plug in $X_i = 0$ (for $i > 0$ because X_0 is a dummy feature for the bias term so its always 1). To compute w_i we take the coefficient of X_i in Eq. 19.

Exercise 2.8**: Restricted Boltzmann Machines

Consider a probabilistic model with two types of binary variables: visible binary stochastic units $\mathbf{v} \in \{0, 1\}^D$ and hidden binary stochastic units $\mathbf{h} \in \{0, 1\}^F$, where D and F are the number of visible and hidden units respectively. The joint probability density to observe their values is given by the exponential distribution

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp\left(-E(\mathbf{v},\mathbf{h})\right), \quad Z = \sum_{\mathbf{v},\mathbf{h}} \exp\left(-E(\mathbf{v},\mathbf{h})\right)$$

and where the energy $E(\mathbf{v},\mathbf{h})$ of the state $\{\mathbf{v},\mathbf{h}\}$ is

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{v}^T W\mathbf{h} - \mathbf{b}^T\mathbf{v} - \mathbf{a}^T\mathbf{h} = -\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij}v_ih_j - \sum_{i=1}^{D} b_iv_i - \sum_{j=1}^{F} a_jh_j,$$

with model parameters $\mathbf{a}$, $\mathbf{b}, W$. This probabilistic model is called the restricted Boltzmann machine. Show that conditional probabilities for visible and hidden nodes are given by the sigmoid function $\sigma(x) = 1/(1+e^{-x})$:

$$\mathbb{P}\left[v_i = 1 \mid \mathbf{h}\right] = \sigma\left(\sum_j W_{ij}h_j + b_i\right), \quad \mathbb{P}\left[h_i = 1 \mid \mathbf{v}\right] = \sigma\left(\sum_j W_{ij}v_j + a_i\right).$$

Solution 2.8

Consider the first expression $\mathbb{P}\left[v_i = 1|\mathbf{h}\right]$. Let $\mathbf{v}_{-i}$ stands for the vector of visible units with the ith components removed. This conditional probability can be computed as follows:

$$\mathbb{P}\left[v_i = 1|\mathbf{h}\right] = \sum_{\mathbf{v}_{-i}} \frac{\mathbb{P}\left[v_i = 1, \mathbf{v}_{-i}, \mathbf{h}\right]}{p(\mathbf{h})} \tag{20}$$

Probabilities entering this expression can be written as follows:

$$p(\mathbf{h}) = \sum_{\mathbf{v}} p(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp\left(\mathbf{a}^T\mathbf{h}\right)\sum_{\mathbf{v}_{-i},v_i}\exp\left(\mathbf{v}^T W\mathbf{h} + \mathbf{b}^T\mathbf{v}\right)$$

$$\mathbb{P}\left[v_i = 1, \mathbf{v}_{-i}, \mathbf{h}\right] = \frac{1}{Z}\exp\left(\mathbf{a}^T\mathbf{h}\right)\exp\left(\sum_{i'\neq i}^{D}\sum_{j=1}^{F} W_{i'j}v_{i'}h_j + \sum_{i'\neq i}^{D} b_{i'}v_{i'} + \sum_j W_{ij}h_j + b_i\right)$$

Substituting these expressions into (20) and simplifying, we obtain

$$\mathbb{P}\left[v_i = 1|\mathbf{h}\right] = \frac{\exp\left(\sum_j W_{ij}h_j + b_i\right)}{\exp\left(\sum_j W_{ij}h_j + b_i\right) + 1} = \sigma\left(\sum_j W_{ij}h_j + b_i\right)$$

The second relation is obtained in a similar way.

Chapter 3
Bayesian Regression & Gaussian Processes

Exercise 3.1: Posterior Distribution of Bayesian Linear Regression

Consider the Bayesian linear regression model

$$y_i = \boldsymbol{\theta}^T X + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_n^2), \boldsymbol{\theta} \sim \mathcal{N}(\mu, \Sigma).$$

Show that the posterior over data $\mathcal{D}$ is given by the distribution

$$\boldsymbol{\theta}|\mathcal{D} \sim \mathcal{N}(\mu', \Sigma'),$$

with moments:

$$\begin{aligned}\mu' &= \Sigma' a = (\Sigma^{-1} + \frac{1}{\sigma_n^2} X X^T)^{-1} (\Sigma^{-1}\mu + \frac{1}{\sigma_n^2}\mathbf{y}^T X)\\ \Sigma' &= A^{-1} = (\Sigma^{-1} + \frac{1}{\sigma_n^2} X X^T)^{-1}.\end{aligned}$$

Solution 3.1

See Eq. 3.12 to Eq. 3.19 in the chapter for the derivation for the posterior distribution moments.

Exercise 3.2: Normal conjugate distributions

Suppose that the prior is $p(\theta) = \phi(\theta;\ \mu_0, \sigma_0^2)$ and the likelihood is given by

$$p(x_{1:n} \mid \theta) = \prod_{i=1}^{n} \phi(x_i;\ \theta, \sigma^2),$$

where σ^2 is assumed to be known. Show that the posterior is also normal, $p(\theta \mid x_{1:n}) = \phi(\theta;\ \mu_{\text{post}}, \sigma^2_{\text{post}})$, where

$$\mu_{\text{post}} = \frac{\sigma_0^2}{\frac{\sigma^2}{n} + \sigma_0^2}\bar{x} + \frac{\sigma^2}{\frac{\sigma^2}{n} + \sigma_0^2}\mu_0,$$

$$\sigma^2_{\text{post}} = \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}},$$

where $\bar{x} := \frac{1}{n}\sum_{i=1}^n x_i$.

Solution 3.2

Starting with $n = 1$, we use the Schur identity for the bi-Gaussian joint distribution

$$\mathbb{E}[\theta|x] = \mathbb{E}[\theta] + \frac{Cov(\theta, x)}{\sigma_x^2}(x - \mathbb{E}[x])$$
$$\mathbb{V}[\theta|x] = \mathbb{V}[\theta] + \frac{Cov^2(\theta, x)}{\sigma_x^2}.$$

Using that

$$x = \theta + \sigma\epsilon,\ \epsilon \sim \mathcal{N}(0,1)$$
$$\theta = \mu_0 + \sigma_0\delta,\ \delta \sim \mathcal{N}(0,1)$$

we have

$$\mathbb{E}[x] = \theta,$$
$$\mathbb{V}[x] = \mathbb{V}[\theta] + \sigma^2 = \sigma_0^2 + \sigma^2$$
$$Cov(x, \theta) = \mathbb{E}[(x - \mu_0)(\theta - \mu_0)] = \sigma_0^2$$

and plugging into the bi-Gaussian conditional moments and after some rearranging gives

$$\mu_{\text{post}} = \mathbb{E}[\theta|x] = \frac{\sigma_0^2}{\sigma^2 + \sigma_0^2}x + \frac{\sigma^2}{\sigma^2 + \sigma_0^2}\mu_0$$
$$\sigma^2_{\text{post}} = \mathbb{V}[\theta|x] = \frac{\sigma^2\sigma_0^2}{\sigma^2 + \sigma_0^2} = (\frac{1}{\sigma_0^2} + \frac{1}{\sigma^2})^{-1}.$$

Now for $n > 1$, we first show that

$$\begin{aligned}
p(x_{1:n}|\theta) &\propto \exp\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\theta)^2\} \\
&\propto \exp\{-\frac{1}{2\sigma^2}(\sum_{i=1}^{n}x_i^2 - 2\theta\sum_{i=1}^{n}x_i + n\theta^2)\} \\
&\propto \exp\{-\frac{n}{2\sigma^2}(\bar{x}-\theta)^2\} \\
&\propto p(\bar{x}|\theta),
\end{aligned}$$

where we keep only terms in θ. Given that $\bar{x}|\mu \sim \mathcal{N}(\theta, \sigma^2/n)$, we can substitute $\bar{x}$ into the previous result for the conditional moments to give the required result

$$\begin{aligned}
\mu_{\text{post}} &= \frac{\sigma_0^2}{\frac{\sigma^2}{n}+\sigma_0^2}\bar{x} + \frac{\sigma^2}{\frac{\sigma^2}{n}+\sigma_0^2}\mu_0 \\
\sigma_{\text{post}}^2 &= (\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2})^{-1}.
\end{aligned}$$

Exercise 3.3: Prediction with GPs

Show that the predictive distribution for a Gaussian Process, with model output over a test point, $\mathbf{f}_*$, and assumed Gaussian noise with variance σ_n^2, is given by

$$\mathbf{f}_* \mid \mathcal{D}, \mathbf{x}_* \sim \mathcal{N}(\mathbb{E}[\mathbf{f}_*|\mathcal{D}, \mathbf{x}_*], \text{var}[\mathbf{f}_*|\mathcal{D}, \mathbf{x}_*]),$$

where the moments of the posterior over X_* are

$$\begin{aligned}
\mathbb{E}[\mathbf{f}_*|\mathcal{D}, X_*] &= \mu_{X_*} + K_{X_*,X}[K_{X,X}+\sigma_n^2 I]^{-1}Y, \\
\text{var}[\mathbf{f}_*|\mathcal{D}, X_*] &= K_{X_*,X_*} - K_{X_*,X}[K_{X,X}+\sigma_n^2 I]^{-1}K_{X,X_*}.
\end{aligned}$$

Solution 3.3

Start with the joint density between $\mathbf{y}$ and f_* given in Eq. 3.35 expressed in terms of the partitioned covariance matrix.

$$\begin{bmatrix}\mathbf{y}\\ f_*\end{bmatrix} = \mathcal{N}\left(\begin{bmatrix}\mu_y\\ \mu_{f_*}\end{bmatrix}, \begin{bmatrix}\Sigma_{yy} & \Sigma_{yf_*}\\ \Sigma_{f_*y} & \Sigma_{f_*f_*}\end{bmatrix}\right).$$

Then use the Schur Identity to derive Eq. 3.36. Finally by writing $y = f(x)+\epsilon$, with an unknown $f(x)$, the covariance is given by

$$\Sigma_{yy} = K_{X,X} + \sigma_n^2 I,$$

where $K_{X,X}$ is known covariance of $f(X)$ over the training input X. Writing the other covariance terms gives the required form of the predictive distribution moments with K specified by some user defined kernel.

1 Programming Related Questions*

Exercise 3.4: Derivative Modeling with GPs

Using the notebook `Example-1-GP-BS-Pricing.ipynb`, investigate the effectiveness of a Gaussian process with RBF kernels for learning the shape of a European derivative (call) pricing function $V_t = f_t(S_t)$ where S_t is the underlying stock's spot price. The risk free rate is $r = 0.001$, the strike of the call is $K_C = 130$, the volatility of the underlying is $\sigma = 0.1$ and the time to maturity $\tau = 1.0$.

Your answer should plot the variance of the predictive distribution against the stock price, $S_t = s$, over a data set consisting of $n \in \{10, 50, 100, 200\}$ gridded values of the stock price $s \in \Omega^h := \{i\Delta s \mid i \in \{0, \ldots, n-1\}, \Delta s = 200/(n-1)\} \subseteq [0, 200]$ and the corresponding gridded derivative prices $V(s)$. Each observation of the dataset, $(s_i, v_i = f_t(s_i))$ is a gridded (stock, call price) pair at time t.

Solution 3.4

See notebook `GP_3_4.ipynb` for the implemented solution.

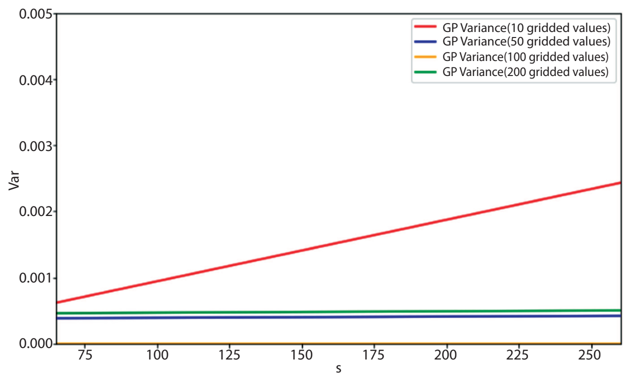

Fig. 1: *The predictive variance of the GP against the stock price for various training set sizes. The predictive variance is observed to monotonically decrease with training set size.*

Chapter 4
Feedforward Neural Networks

Exercise 4.1

Show that substituting

$$\nabla_{ij} I_k = \begin{cases} X_j, & i = k, \\ 0, & i \neq k, \end{cases}$$

into Eq. 4.47 gives

$$\nabla_{ij} \sigma_k \equiv \frac{\partial \sigma_k}{\partial w_{ij}} = \nabla_i \sigma_k X_j = \sigma_k(\delta_{ki} - \sigma_i) X_j.$$

Solution 4.1

By the chain rule we have

$$\frac{\partial \sigma_k}{\partial w_{ij}} = \sum_m \frac{\partial \sigma_k}{\partial I_m} \frac{\partial I_m}{\partial w_{ij}}$$

Recall that the k, i component of the Jacobian of σ is:

$$\frac{\partial \sigma_k}{\partial I_i} = \sigma_k(\delta_{ki} - \sigma_i),$$

and since $\frac{\partial I_i}{\partial w_{ij}} = X_j,\ i = k$ we have the required answer

$$\frac{\partial \sigma_k}{\partial w_{ij}} = \frac{\partial \sigma_k}{\partial I_i} X_j = \sigma_k(\delta_{ki} - \sigma_i) X_j.$$

Exercise 4.2

Show that substituting the derivative of the softmax function w.r.t. w_{ij} into Eq. 4.52 gives for the special case when the output is $Y_k = 1,\ k = i$ and $Y_k = 0,\ \forall k \neq i$:

$$\nabla_{ij}\mathcal{L}(W,b) := [\nabla_W \mathcal{L}(W,b)]_{ij} = \begin{cases}(\sigma_i - 1)X_j, & Y_i = 1,\\ 0, & Y_k = 0,\ \forall k \neq i.\end{cases}$$

Solution 4.2

Recalling Eq. 4.52 with σ for the softmax activation function and $I(W,b)$ is the transformation

$$\nabla\mathcal{L}(W,b) = \nabla(\mathcal{L}\circ\sigma)(I(W,b)) = \nabla\mathcal{L}(\sigma(I(W,b)))\cdot\nabla\sigma(I(W,b))\cdot\nabla I(W,b).$$

Writing the model output as a function of W only since we are not taking the derivative of b:

$$\hat{Y}(W) = \sigma \circ I(W).$$

The loss of a K-classifier is given by the cross-entropy:

$$\mathcal{L}(W) = -\sum_{k=1}^{K} Y_k ln\hat{Y}_k.$$

However, we want to evaluate the loss for the special case when $k = i$, since $Y_i = 1$. Therefore the loss reduces to

$$\mathcal{L}(W) = -ln\hat{Y}_i = -ln\sigma_i.$$

Taking the derivative of the loss w.r.t. w_{ij} gives

$$\frac{\partial\mathcal{L}}{\partial w_{ij}} = \sum_{m,n}\frac{\partial\mathcal{L}(W)}{\partial\sigma_m}\frac{\partial\sigma_m}{\partial I_n}\frac{\partial I_n}{\partial w_{ij}}$$

or equivalently

$$\nabla_{ij}\mathcal{L}(W) = \sum_{m,n}\nabla_{\sigma_m}\mathcal{L}(W)\nabla_{mn}\sigma(I)\nabla_{ij}I_n(W).$$

The derivative of the loss w.r.t. σ_m is

$$\nabla_{\sigma_m}\mathcal{L}(W) = -\frac{\delta_{im}}{\sigma_m}.$$

The derivative of the softmax function w.r.t. to I_n gives

$$\nabla_{mn}\sigma(I(W)) = \sigma_m(I(W))(\delta_{mn} - \sigma_n(I(W))),$$

and the derivative of I_n w.r.t w_{ij} is

$$\nabla_{ij} I_n(W) = \delta_{ni} X_j.$$

Hence the derivative of the softmax function w.r.t. to w_{ij} is

$$\sigma_m(I(W))(\delta_{mn} - \sigma_n(I(W)))\delta_{ni} X_j.$$

Putting these terms together

$$\nabla_{ij}\mathcal{L}(W) = \sum_{m,n} -\frac{\delta_{im}}{\sigma_m}\sigma_m(I(W))(\delta_{mn} - \sigma_n(I(W)))\delta_{ni} X_j,$$

Contracting over the m and n indices gives for $Y_i = 1$:

$$\nabla_{ij}\mathcal{L}(W) = (\sigma_i - 1)X_j,$$

and is zero if $Y_k = 0,\ k \neq i$.

Exercise 4.3

Consider feedforward neural networks constructed using the following two types of activation functions:

- Identity
$$Id(x) := x$$
- Step function (a.k.a. Heaviside function)
$$H(x) := \begin{cases} 1, & \text{if } x \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

a. Consider a feedforward neural network with one input $x \in \mathbb{R}$, a single hidden layer with K units having step function activations, $H(x)$, and a single output with identity (a.k.a linear) activation, $Id(x)$. The output can be written as

$$\hat{f}(x) = Id\left(b^{(2)} + \sum_{k=1}^{K} w_k^{(2)} H(b_k^{(1)} + w_k^{(1)} x)\right).$$

Construct neural networks using these activation functions.

a. Consider the step function

$$u(x; a) := yH(x - a) = \begin{cases} y, & \text{if } x \geq a, \\ 0, & \text{otherwise.} \end{cases}$$

Construct a neural network with one input x and one hidden layer, whose response is $u(x; a)$. Draw the structure of the neural network, specify the activation function for each unit (either Id or H), and specify the values for all weights (in terms of a and y).

b. Now consider the indicator function

$$\mathbf{1}_{[a,b)}(x) = \begin{cases} 1, & \text{if } x \in [a, b), \\ 0, & \text{otherwise.} \end{cases}$$

Construct a neural network with one input x and one hidden layer, whose response is $y\mathbf{1}_{[a,b)}(x)$, for given real values y, a and b. Draw the structure of the neural network, specify the activation function for each unit (either Id or H), and specify the values for all weights (in terms of a, b and y).

Solution 4.3

- There are many possible solutions. However, once such choice is as follows. Set $b^{(2)} = 0, w_1^{(2)} = y, b_1^{(1)} = -a, w_1^{(1)} = 1$.
- There are many possible solutions. However, once such choice is as follows. Set $b^{(2)} = 0, w_1^{(2)} = y, w_2^{(2)} = -y, b_1^{(1)} = -a, w_1^{(1)} = 1, b_2^{(1)} = -b, w_2^{(1)} = 1$.

Exercise 4.4

A neural network with a single hidden layer can provide an arbitrarily close approximation to any 1-dimensional bounded smooth function. This question will guide you through the proof. Let $f(x)$ be any function whose domain is $[C, D)$, for real values $C < D$. Suppose that the function is Lipschitz continuous, that is,

$$\forall x, x' \in [C, D),\ |f(x') - f(x)| \leq L|x' - x|,$$

for some constant $L \geq 0$. Use the building blocks constructed in the previous part to construct a neural network with one hidden layer that approximates this function within $\epsilon > 0$, that is, $\forall x \in [C, D),\ |f(x) - \hat{f}(x)| \leq \epsilon$, where $\hat{f}(x)$ is the output of your neural network given input x. Your network should use only the identity or the Heaviside activation functions. You need to specify the number K of hidden units, the activation function for each unit, and a formula for calculating each weight w_0, w_k, $w_0^{(k)}$, and $w_1^{(k)}$, for each $k \in \{1, \ldots, K\}$. These weights may be specified in terms of C, D, L and ϵ, as well as the values of $f(x)$ evaluated at a finite number of x values of your choosing (you need to explicitly specify which x values you use). You do not need to explicitly write the $\hat{f}(x)$ function. Why does your network attain the given accuracy ϵ?

Solution 4.4

The hidden units all use g_s activation functions, and the output unit uses g_I. $K = \lfloor (D - C)\frac{L}{2\epsilon} + 1 \rfloor$. For $k \in \{1, \ldots, K\}, w_1^{(k)} = 1, w_0^{(k)} = -(C + (k-1)2\epsilon/L)$. For the second layer, we have $w_0 = 0, w_1 = f(C + \epsilon/L)$, and for $k \in \{2, \ldots, K\}$, $w_k = f(C + (2k-1)\epsilon/L) - f(C + (2k-3)\epsilon/L)$. We only evaluate $f(x)$ at points $C + (2k-1)\epsilon/L$, for $k \in \{1, \ldots, K\}$, which is a finite number of points. The function value $\hat{f}(x)$ is exactly $f(C + (2k-1)\epsilon/L)$ for $k \in \{1, \ldots, K\}$. Note that for any $x \in [C + (2k-1)\epsilon/L, C + 2k\epsilon/L]$, $|f(x) - \hat{f}(x)| \leq \epsilon$.

Exercise 4.5

Consider a shallow neural network regression model with n tanh activated units in the hidden layer and d outputs. The hidden-outer weight matrix $W_{ij}^{(2)} = \frac{1}{n}$ and the input-hidden weight matrix $W^{(1)} = 1$. The biases are zero. If the features, $X_1, \ldots, X_p$ are i.i.d. Gaussian random variables with mean $\mu = 0$, variance σ^2, show that

a. $\hat{Y} \in [-1, 1]$.
b. $\hat{Y}$ is independent of the number of hidden units, $n \geq 1$.
c. The expectation, $\mathbb{E}[\hat{Y}] = 0$ and the variance $\mathbb{V}[\hat{Y}] \leq 1$.

Solution 4.5

$$\hat{Y} = W^{(2)}\sigma(I^{(1)})$$

Each hidden variable is a vector with identical elements

$$Z_i^{(1)} = w_{ij}^{(1)} X_j = \sum_{j=1}^{p} X_j = S_i = s, i \in \{1, \ldots, n\}.$$

Using this expression gives

$$\hat{Y}_i = w_{ij}^{(2)}\sigma(S_j) = \frac{1}{n}\sum_{j=1}^{n}\sigma(s) = \sigma(s).$$

which is independent of n.

$$\begin{aligned}
\mathbb{E}[\hat{Y}] &= \int W^{(2)}\sigma(I^{(1)})f_X(\mathbf{x})d\mathbf{x}\\
&= W^{(2)}\int \sigma(I^{(1)})f_X(\mathbf{x})d\mathbf{x}\\
&= W^{(2)}\int\cdots\int \sigma(I^{(1)})f_X(x_1)\dots f_X(x_p)dx_1\dots dx_p\\
&= W^{(2)}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} \sigma(I^{(1)})f_X(x_1)\dots f_X(x_p)dx_1\dots dx_p\\
&= W^{(2)}\int_{0}^{\infty}\cdots\int_{0}^{\infty} [\sigma(I^{(1)})+\sigma(-I^{(1)})]f_X(x_1)\dots f_X(x_p)dx_1\dots dx_p\\
&= 0,
\end{aligned}$$

where we have used the property that $\sigma(I)$ is an odd function in I (and in x since $b^{(1)} = 0$) and $f_X(x)$ is an even function since $\mu = 0$. The variance is $\mathbb{E}[\hat{Y}\hat{Y}^T] = \mathbb{E}[\sigma(I^{(1)})\sigma^T(I^{(1)})]$, where $[\mathbb{E}[\sigma(I^{(1)})\sigma^T(I^{(1)})]]_{ij} = \mathbb{E}[\sigma^2(I^{(1)})]$.

$$\begin{aligned}
\mathbb{E}[\sigma^2(I^{(1)})] &= \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} \sigma^2(I^{(1)})f_X(x_1)\dots f_X(x_p)dx_1\dots dx_p\\
&= 2\int_{0}^{\infty}\cdots\int_{0}^{\infty} \sigma^2(I^{(1)})f_X(x_1)\dots f_X(x_p)dx_1\dots dx_p\\
&\leq 2\int_{0}^{\infty}\cdots\int_{0}^{\infty} f_X(x_1)\dots f_X(x_p)dx_1\dots dx_p\\
&= 1.
\end{aligned}$$

Exercise 4.6

Determine the VC dimension of the sum of indicator functions where $\Omega = [0, 1]$

$$F_k(x) = \{f : \Omega \to \{0,1\},\ f(x) = \sum_{i=0}^{k} \mathbf{1}_{x\in[t_{2i},t_{2i+1})}, 0 \leq t_0 < \cdots < t_{2k+1} \leq 1,\ k \geq 1\}.$$

Solution 4.6

F_k is the set of all functions with $k + 1$ indicator functions. Consider the worst case labeling that can be encountered. This occurs when the labels of the points are alternating $\{0, 1, 0, 1, \dots\}$. If the function can represent the labeled data in this case, then it can do so for any other labeling. In this worst case configuration, with $k + 1$ labels and therefore $2(k + 1)$ points in the set, it is clear that all points can be labeled by placing the indicator function over each of the $k + 1$ points with label 1. So $VC(F_k) \geq 2(k + 1)$. Moreover if there are $2(k + 1) + 1$ points, you can create a configuration of alternating labeled points with a total of $k + 2$ points with label 1.

This is achieved by starting and ending with a 1 label point so that the labels of the points are $\{1, 0, 1, \ldots, 0, 1\}$. This last configuration is not representable with $k + 1$ indicator functions. Therefore $VC(F_k) = 2(k + 1)$.

Exercise 4.7

Show that a feedforward binary classifier with two Heaviside activated units shatters the data $\{0.25, 0.5, 0.75\}$.

Solution 4.7

Now for all permutations of label assignments to the data, we consider values of weights and biases which classify each arrangement:

- None of the points are activated. In this case, one of the units is essentially redundant and the output of the network is a Heaviside function: $b_1^{(1)} = b_2^{(2)}$ & $w_1^{(2)} = w_2^{(2)}$
  For example:
  $$w^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, b^{(1)} = \begin{pmatrix} c \\ c \end{pmatrix}$$
  $$w^{(2)} = \begin{pmatrix} \frac{1}{2}, \frac{1}{2} \end{pmatrix}, b^{(2)} = 0$$
  where $c < -0.75$.
  - The last point is activated: choose $-0.5 > c > -0.75$
  - The last two points are activated: choose $-0.25 > c > -0.5$
  - All points are activated: choose $c > -0.25$
- The network behaves as an indicator function
  $b_1^{(1)} \neq b_2^{(1)}$ & $w_1^{(2)} = -w_1^{(2)}$

  e.g.
  $$w^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, b^{(1)} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$
  $$w^{(2)} = \begin{pmatrix} 1, -1 \end{pmatrix}, b^{(2)} = 0 \tag{21}$$

  where $c_2 < c_1$. For example if $c_1 = -0.4$ and $c_2 = -0.6$, then the second point is activated. Also the first point is activated with, say, $c_1 = 0$ and $c_2 = -0.4$. The first two points are activated with $c_1 = 0$ and $c_2 = -0.6$.

- Lastly, the inverse indicator function can be represented, e.g., only the first and third points are activated:

$$w^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, b^{(1)} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

  and

$$w^{(2)} = \begin{pmatrix} -1, 1 \end{pmatrix}, b^{(2)} = 1$$

  with, say, $c_1 = -0.4$ and $c_2 = -0.6$.

Hence the network shatters all three points and the VC-dim is 3. Note that we could have approached the first part above differently by letting the network behave as a wider indicator function (with support covering two or three data points) rather than as a single Heaviside function.

Exercise 4.8

Compute the weight and bias updates of $W^{(2)}$ and $b^{(2)}$ given a shallow binary classifier (with one hidden layer) with unit weights, zero biases, and ReLU activation of two hidden units for the labeled observation $(x = 1, y = 1)$.

Solution 4.8

This is straightforward forward pass computations to obtain the error. E.g., with ReLU activation
$I^{(1)} = W^{(1)}x$

$$Z^{(1)} = \sigma(I^{(1)}) = \begin{pmatrix} max(1,0) \\ max(1,0) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$I^{(2)} = W^{(2)}Z^{(1)} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The prediction is

$$\hat{y} = \sigma_{sigmoid}(I^{(2)}) = \frac{1}{1 + e^{-I^{(2)}}} = \frac{1}{1 + e^{-2}} = 0.8807971$$

The error is

$$\delta^{(2)} = e = y - \hat{y} = 1 - 0.8807971 = 0.1192029$$

Applying one step of the back-propagation algorithm (multivariate chain rule):

$$\text{weight update: } \Delta W^{(2)} = -\gamma\delta^{(2)}\mathrm{Z}^{(1)} = -\begin{pmatrix} 1 \\ 1 \end{pmatrix} = -\gamma \begin{pmatrix} 0.1192029 \\ 0.1192029 \end{pmatrix}$$

where γ is the learning rate. The bias update is $\Delta b^{(2)} = -\gamma\delta^{(2)} = -\gamma 0.1192029$.

2 Programming Related Questions*

Exercise 4.9

Consider the following dataset (taken from Anscombe's quartet):

$$\begin{aligned}(x_1, y_1) &= (10.0, 9.14), (x_2, y_2) = (8.0, 8.14), (x_3, y_3) = (13.0, 8.74),\\ (x_4, y_4) &= (9.0, 8.77), (x_5, y_5) = (11.0, 9.26), (x_6, y_6) = (14.0, 8.10),\\ (x_7, y_7) &= (6.0, 6.13), (x_8, y_8) = (4.0, 3.10), (x_9, y_9) = (12.0, 9.13),\\ (x_{10}, y_{10}) &= (7.0, 7.26), (x_{11}, y_{11}) = (5.0, 4.74).\end{aligned}$$

a. Use a neural network library of your choice to show that a feedforward network with one hidden layer consisting of one unit and a feedforward network with no hidden layers, each using only linear activation functions, do not outperform linear regression based on ordinary least squares (OLS).
b. Also demonstrate that a neural network with a hidden layer of three neurons using the tanh activation function and an output layer using the linear activation function captures the non-linearity and outperforms the linear regression.

Solution 4.9

Worked solutions are provided in the Notebook `ancombes_4_9.ipynb`.

Exercise 4.10

Review the Python notebook `deep_classifiers.ipynb`. This notebook uses Keras to build three simple feedforward networks applied to the half-moon problem: a logistic regression (with no hidden layer); a feedforward network with one hidden layer; and a feedforward architecture with two hidden layers. The half-moons problem is not linearly separable in the original coordinates. However you will observe—after plotting the fitted weights and biases—that a network with many hidden neurons gives a linearly separable representation of the classification problem in the coordinates of the output from the final hidden layer.

Complete the following questions in your own words.

a. Did we need more than one hidden layer to perfectly classify the half-moons dataset? If not, why might multiple hidden layers be useful for other datasets?

b. Why not use a very large number of neurons since it is clear that the classification accuracy improves with more degrees of freedom?
c. Repeat the plotting of the hyperplane, in Part 1b of the notebook, only without the ReLU function (i.e., activation="linear"). Describe qualitatively how the decision surface changes with increasing neurons. Why is a (non-linear) activation function needed? The use of figures to support your answer is expected.

Solution 4.10

- Only one hidden layer is needed. One reason why multiple hidden layers could be needed on other datasets is because the input data are highly correlated. For some datasets, it may be more efficient to use multiple layers and reduce the overall number of parameters.
- Adding too many neurons in the hidden layer results in a network which takes longer to train without an increase in the in-sample performance. It also leads to over-fitting on out-of-sample test sets.
- The code and results should be submitted with activation="linear" in the hidden layer. Plots of the hidden variables will show a significant difference compared to using ReLU. Additionally, the performance of the network with many neurons in the hidden layer, without activation, should be compared with ReLU activated hidden neurons.

Exercise 4.11

Using the EarlyStopping callback in Keras, modify the notebook `Deep_Classifiers.ipynb` to terminate training under the following stopping criterion $|L^{(k+1)} - L^{(k)}| \leq \delta$ with $\delta = 0.1$.

Exercise 4.12***

Consider a feedforward neural network with three inputs, two units in the first hidden layer, two units in the second hidden layer, and three units in the output layer. The activation function for hidden layer 1 is ReLU, for hidden layer 2 is sigmoid, and for the output layer is softmax.

The initial weights are given by the matrices

$$W^{(1)} = \begin{pmatrix} 0.1 & 0.3 & 0.7 \\ 0.9 & 0.4 & 0.4 \end{pmatrix}, W^{(2)} = \begin{pmatrix} 0.4 & 0.3 \\ 0.7 & 0.2 \end{pmatrix}, W^{(3)} = \begin{pmatrix} 0.5 & 0.6 \\ 0.6 & 0.7 \\ 0.3 & 0.2 \end{pmatrix},$$

and all the biases are unit vectors.

Assuming that the input $\begin{pmatrix}0.1 & 0.7 & 0.3\end{pmatrix}$ corresponds to the output $\begin{pmatrix}1 & 0 & 0\end{pmatrix}$, manually compute the updated weights and biases after a single epoch (forward + backward pass), clearly stating all derivatives that you have used. You should use a learning rate of 1.

As a practical exercise, you should modify the implementation of a stochastic gradient descent routine in the back-propagation Python notebook.

Note that the notebook example corresponds to the example in Sect. 5, which uses sigmoid activated hidden layers only. Compare the weights and biases obtained by `TensorFlow` (or your ANN library of choice) with those obtained by your procedure after 200 epochs.

Solution 4.12

Worked solutions are provided in the back-propagation solution notebook.

Chapter 5
Interpretability

Exercise 5.1*

Consider the following data generation process

$$Y = X_1 + X_2 + \epsilon,\ X_1, X_2, \epsilon \sim N(0, 1),$$

i.e. $\beta_0 = 0$ and $\beta_1 = \beta_2 = 1$.

a. For this data, write down the mathematical expression for the sensitivities of the fitted neural network when the network has
   - zero hidden layers;
   - one hidden layer, with n unactivated hidden units;
   - one hidden layer, with n tanh activated hidden units;
   - one hidden layer, with n ReLU activated hidden units; and
   - two hidden layers, each with n tanh activated hidden units.

Solution 5.1

$$Y = X_1 + X_2 + \epsilon, \quad X_1, X_2, \epsilon \sim N(0, 1),\ \beta_0 = 0, \beta_1 = \beta_2 = 1$$

- Zero hidden layers:

  In this situation, the sensitivities are simply expressed as:

  $$\frac{\partial \hat{Y}}{\partial X_i} = w_i^{(1)} \quad i \in \{1, 2\}$$

  Note, when there is one hidden layer, the sensitivities can be generally expressed as:
  $\partial_X \hat{Y} = W^{(2)} J(I^{(1)}) = W^{(2)} D(I^{(1)}) W^{(1)}$,

where $D_{i,i}(I) = \sigma'(I_i), D_{i,j\neq i} = 0$

- One hidden layers, with n unactivated hidden units:

  As the hidden units are unactivated, we can write
  $\sigma(x) = x \rightarrow \sigma'(x) = 1$
  Therefore, $D(I^{(1)})$ is the identity matrix.
  Thus,
  $$\frac{\partial \hat{Y}}{\partial X} = W^{(2)}W^{(1)}$$
- One hidden layer, with tanh activated hidden units:

  As the hidden units are tanh activated, we can write:
  $\sigma(x) = tanh(x) \rightarrow \sigma'(x) = 1 - tanh^2(x) = 1 - \sigma^2(x)$
  Therefore, we have that
  $$\frac{\partial \hat{Y}}{\partial X_j} = \sum_i w_i^{(2)}(1 - \sigma^2(I_i^{(1)}))w_{i,j}^{(1)}$$
- One hidden layer, with n ReLU activated hidden units:

  As the hidden units are ReLU activated, it follows that
  $\sigma(x) = ReLU(x) \rightarrow \sigma'(x) = H(x)$, the Heaviside function
  Therefore, we have that
  $$\frac{\partial \hat{Y}}{\partial X_j} = \sum_i w_i^{(2)}(H(I_i^{(1)}))w_{i,j}^{(1)}$$
- Two hidden layers, each with n tanh activated hidden units:

  Since we have two hidden layers, our sensitivities can be expressed as:
  $$\frac{\partial \hat{Y}}{\partial X} = W^{(3)}D^{(2)}W^{(2)}D^{(1)}W^{(1)}$$
  As we mentioned previously, we can write:
  $\sigma(x) = tanh(x) \rightarrow \sigma'(x) = 1 - tanh^2(x)$
  Therefore, we can express the sensitivities as
  $$\frac{\partial \hat{Y}}{\partial X} = W^{(3)}D^{(2)}W^{(2)}D^{(1)}W^{(1)}$$
  where $D_{i,i}^{(2)} := D_{i,i}^{(2)}(I_i^{(2)}) = 1 - tanh^2(I_i^{(2)}), D_{i,j\neq i}^{(2)} = 0$
  $D_{i,i}^{(1)} := D_{i,i}^{(1)}(I_i^{(1)}) = 1 - tanh^2(I_i^{(1)}), D_{i,j\neq i}^{(1)} = 0.$

Exercise 5.2**

Consider the following data generation process

$$Y = X_1 + X_2 + X_1 X_2 + \epsilon,\ X_1, X_2 \sim N(0,1),\ \epsilon \sim N(0, \sigma_n^2),$$

i.e. $\beta_0 = 0$ and $\beta_1 = \beta_2 = \beta_{12} = 1$, where β_{12} is the interaction term. σ_n^2 is the variance of the noise and $\sigma_n = 0.01$.

a. For this data, write down the mathematical expression for the interaction term (i.e. the off-diagonal components of the Hessian matrix) of the fitted neural network when the network has
   - zero hidden layers;
   - one hidden layer, with n unactivated hidden units;
   - one hidden layer, with n tanh activated hidden units;
   - one hidden layer, with n ReLU activated hidden units; and
   - two hidden layers, each with n tanh activated hidden units.

Why is the ReLU activated network problematic for estimating interaction terms?

Solution 5.2

See the notebook solution.

3 Programming Related Questions*

Exercise 5.3*

For the same problem in the previous exercise, use 5000 simulations to generate a regression training set dataset for the neural network with one hidden layer. Produce a table showing how the mean and standard deviation of the sensitivities β_i behave as the number of hidden units is increased. Compare your result with $tanh$ and $ReLU$ activation. What do you conclude about which activation function to use for interpretability? Note that you should use the notebook `Deep-Learning-Interpretability.ipynb` as the starting point for experimental analysis.

Solution 5.3

See the notebook solution.

Exercise 5.4*

Generalize the `sensitivities` function in Exercise 5.3 to L layers for either tanh or ReLU activated hidden layers. Test your function on the data generation process given in Exercise 5.1.

Solution 5.4

See the notebook solution.

Exercise 5.5**

Fixing the total number of hidden units, how do the mean and standard deviation of the sensitivities β_i behave as the number of layers is increased? Your answer should compare using either tanh or ReLU activation functions. Note, do not mix the type of activation functions across layers. What you conclude about the effect of the number of layers, keeping the total number of units fixed, on the interpretability of the sensitivities?

Solution 5.5

$Y = X_1 + X_2 + X_1 X_2 + \epsilon \; X_1, X_2 \sim N(0,1), \epsilon \sim N(0, \sigma_n^2)$

$\beta_0 = 0, \beta_1 = \beta_2 = \beta_{12} = 1, \sigma_n = 0.01$

- Zero hidden layers:

  In this situation, the interaction term is zero as the sensitivity is independent of X.
  Note that when there is just one hidden (activated) layer, the interaction term can be generally expressed as:
  $\partial^2_{X_i X_j} \hat{Y} = W^{(2)} diag(W_i^{(1)}) D'(I^{(1)}) W_j^{(1)}$
- One hidden layer with n unactivated hidden units:

  As the hidden units are unactivated, we have:

$$\frac{\partial^2 \hat{Y}}{\partial X_i X_j} = 0$$

- One hidden layer with tanh activated hidden units:

  As the hidden units are tanh activated, we can write:
  $\sigma(x) = tanh(x)$

$$\frac{\partial \sigma}{\partial x} = 1 - tanh^2(x)$$

$$\frac{\partial^2 \sigma}{\partial x^2} = -2tanh(x)(1 - tanh^2(x))$$

Therefore, we have that

$$\frac{\partial \hat{Y}}{\partial X_i X_j} = W^{(2)} diag(W_i^{(1)}) D'(I^{(1)}) W_j^{(1)}$$

where $D'_{i,i}(I_i^{(1)}) = -2tanh(I_i^{(1)})(1 - tanh^2(I_i^{(1)})), D'_{i,j \neq i}(I_i^{(1)}) = 0$

- One hidden layer, with n ReLU activated hidden units:

  As the hidden units are ReLU activated, we write
  $\sigma(x) = ReLU(x)$

  $$\frac{\partial \sigma}{\partial x} = H(x), \text{the Heaviside function.}$$

  $$\frac{\partial^2 \sigma}{\partial x^2} = \delta(x), \text{the Dirac Delta function.}$$

  Therefore, it follows that

  $$\frac{\partial^2 \hat{Y}}{\partial X_i X_j} = W^{(2)} diag(W_i^{(1)}) D'(I^{(1)}) W_j^{(1)}$$

  where $D'_{i,i}(I_i) = \delta(I_i), D'_{i,j \neq i}(I_i) = 0$

- Two hidden layers, each with n tanh activated hidden units:

  Since we have two hidden layers, the interaction term can be expressed as:

  $$\frac{\partial \hat{Y}}{\partial X_i X_j} = W^{(3)} W^{(2)} \left(\frac{\partial D^{(2)}}{\partial X_j} D^{(1)} + \frac{\partial D^{(1)}}{\partial X_j} D^{(2)} \right) w_i^{(1)}$$

  where applying the chain rule we have

  $$\frac{\partial D^{(2)}}{\partial X_j} = D'(I^{(2)}) W^{(2)} D^{(1)} diag(w_j^{(1)})$$

  $$\frac{\partial D^{(1)}}{\partial X_j} = D'(I^{(1)}) diag(w_j^{(1)}).$$

  As we mentioned previously, for tanh:
  $\sigma(x) = tanh(x)$

  $$\frac{\partial \sigma}{\partial x} = 1 - tanh^2(x)$$

$$\frac{\partial^2 \sigma}{\partial x^2} = -2tanh(x)(1 - tanh^2(x))$$

Therefore substitute in the expressions for D' and D
where $D^{'}_{i,i}(I^{(2)}_i) = -2tanh(I^{(2)}_i)(1 - tanh^2(I^{(2)}_i)), D^{'}_{i,j\neq i}(I^{(2)}_i) = 0$
$D^{'}_{i,i}(I^{(1)}_i) = -2tanh(I^{(1)}_i)(1 - tanh^2(I^{(1)}_i)), D^{'}_{i,j\neq i}(I^{(1)}_i) = 0.$

- ReLU does not exhibit sufficient smoothness for the interaction term to be continuous.

Exercise 5.6**

For the same data generation process as the previous exercise, use 5000 simulations to generate a regression training set for the neural network with one hidden layer. Produce a table showing how the mean and standard deviation of the interaction term behave as the number of hidden units is increased, fixing all other parameters. What do you conclude about the effect of the number of hidden units on the interpretability of the interaction term? Note that you should use the notebook `Deep-Learning-Interaction.ipynb` as the starting point for experimental analysis.

Solution 5.6

See the notebook solution.

Part II
Sequential Learning

Chapter 6
Sequence Modeling

Exercise 6.1

Calculate the mean, variance, and autocorrelation function (ACF) of the following zero-mean AR(1) process:

$$y_t = \phi_1 y_{t-1} + \epsilon_t$$

where $\phi_1 = 0.5$. Determine whether the process is stationary by computing the root of the characteristic equation $\Phi(z) = 0$.

Solution 6.1

- The mean of an AR(1) process with no drift term is:

  $$\mathbb{E}[y_t] = \mathbb{E}[\phi_1 y_{t-1} + \epsilon_t] = \phi_1 \mathbb{E}[y_{t-1}] + \mathbb{E}[\epsilon_t] = \phi_1 \mathbb{E}[y_{t-1}].$$

  Since $y_{t-1} = \phi_1 y_{t-2} + \epsilon_t$ and $\mathbb{E}[\epsilon_t] = 0$, then $\mathbb{E}[y_t] = \phi_1^2 \mathbb{E}[y_{t-2}] = \cdots = \phi_1^n \mathbb{E}[y_{t-n}]$.
  By the property of stationary, we have $|\phi| < 1$, which gives $\lim_{n\to\infty} \phi_1^n \mathbb{E}[y_{t-n}] = 0$.
  Hence $\mathbb{E}[y_t] = 0$.
- The variance of an AR(1) process with no drift term is as follows. First note that $y_t = \phi_1 L[y_t] + \epsilon_t$ can be written as a MA(∞) process by back-substitution:

  $$y_t = (1 + \phi_1 L + \phi_1^2 L^2 + \ldots)[\epsilon_t].$$

  Taking the expectation of y_t^2:

  $$\begin{aligned}\mathbb{E}[y_t^2] &= \mathbb{E}\left[(1 + \phi_1 L + \phi_1^2 L^2 + \ldots)^2[\epsilon_t^2]\right] \\ &= (1 + \phi_1^2 + \phi_1^4 + \ldots)\mathbb{E}[\epsilon_t^2] + \mathbb{E}[\text{cross terms}] = \sigma^2/(1 - \phi_1^2).\end{aligned}$$

- The lag 1 auto-correlation of an AR(1) processs with no drift term is:

$$\begin{aligned}
\gamma_1 &:= \mathbb{E}[y_t y_{t-1}] \\
&= \mathbb{E}[(\epsilon_t + \phi_1\epsilon_{t-1} + \phi_1^2\epsilon_{t-2} + \dots)(\epsilon_{t-1} + \phi_1\epsilon_{t-2} + \phi_1^2\epsilon_{t-3} + \dots)] \\
&= \mathbb{E}[\phi_1\epsilon_{t-1}^2 + \phi_1^3\epsilon_{t-2}^2 + \phi_1^5\epsilon_{t-3}^2 + \cdots + \text{cross terms}] \\
&= \phi_1\sigma^2 + \phi_1^3\sigma^2 + \phi_1^5\sigma^2 + \dots \\
&= \phi_1\sigma^2/(1-\phi_1^2)
\end{aligned}$$

The lag 2 auto-correlation is:

$$\begin{aligned}
\gamma_2 &:= \mathbb{E}[y_t y_{t-2}] \\
&= \mathbb{E}[(\epsilon_t + \phi_1\epsilon_{t-1} + \phi_1^2\epsilon_{t-2} + \dots)(\epsilon_{t-2} + \phi_1\epsilon_{t-3} + \phi_1^2\epsilon_{t-4} + \dots)] \\
&= \mathbb{E}[\phi_1^2\epsilon_{t-2}^2 + \phi_1^4\epsilon_{t-3}^2 + \phi_1^6\epsilon_{t-4}^2 + \cdots + \text{cross terms}] \\
&= \phi_1^2\sigma^2 + \phi_1^4\sigma^2 + \phi_1^6\sigma^2 + \dots \\
&= \phi_1^2\sigma^2/(1-\phi_1^2).
\end{aligned}$$

Keep iterating, to arrive at $\gamma_s = \phi_1^s\sigma^2/(1-\phi_1^2)$. Since the autocorrelation is $\tau_s = \gamma_s/\gamma_0$, we have $\tau_s = \frac{\phi_1^s\sigma^2/(1-\phi_1^2)}{\sigma^2/(1-\phi_1^2)} = \phi_1^s$.

- Finally we can write the AR(1) process as

$$\Phi(L)[y_t] = \epsilon_t$$

where $\Phi(L) := (1-\phi_1 L)$. The root of $\Phi(z)$ for $\phi_1 = 0.5$ is $z = 2$. Since $|z| = 2 > 1$, the process has no unit root and is stationary.

Exercise 6.2

You have estimated the following ARMA(1,1) model for some time series data

$$y_t = 0.036 + 0.69y_{t-1} + 0.42u_{t-1} + u_t,$$

where you are given the data at time $t-1$, $y_{t-1} = 3.4$ and $\hat{u}_{t-1} = -1.3$. Obtain the forecasts for the series y for times $t, t+1, t+2$ using the estimated ARMA model.

If the actual values for the series are $-0.032, 0.961, 0.203$ for $t, t+1, t+2$, calculate the out-of-sample Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Solution 6.2

$$\begin{aligned}
\mathbb{E}[y_t|\Omega_{t-1}] &= 0.036 + 0.69\mathbb{E}[y_{t-1}|\Omega_{t-1}] + 0.42\mathbb{E}[u_{t-1}|\Omega_{t-1}] + \mathbb{E}[u_t|\Omega_{t-1}] \\
&= -0.036 + 0.69 \times 3.4 + 0.42 \times -1.3 + 0 = 1.836 \\
\mathbb{E}[y_{t+1}|\Omega_{t-1}] &= 0.036 + 0.69\mathbb{E}[y_t|\Omega_{t-1}] + 0.42\mathbb{E}[u_t|\Omega_{t-1}] + \mathbb{E}[u_{t+1}|\Omega_{t-1}] \\
&= 0.036 + 0.69 \times 1.836 + 0.42 \times 0 + 0 = 1.30284 \\
\mathbb{E}[y_{t+2}|\Omega_{t-1}] &= 0.036 + 0.69\mathbb{E}[y_{t+1}|\Omega_{t-1}] + 0.42\mathbb{E}[u_{t+1}|\Omega_{t-1}] + \mathbb{E}[u_{t+2}|\Omega_{t-1}] \\
&= 0.036 + 0.69 \times 1.30284 + 0.42 \times 0 + 0 = 0.9350
\end{aligned}$$

Forecasted	Actual	Absolute	Diff. Squared	Diff
t	1.836	-0.032	3.489	1.868
t+1	1.30284	0.961	0.117	0.342
t+2	0.9350	0.203	0.536	0.732

$$MSE = \frac{1}{3}(3.489 + 0.117 + 0.536) = 1.381.$$

$$MAE = \frac{1}{3}(1.868 + 0.342 + 0.732) = 0.981.$$

Exercise 6.3

Derive the mean, variance, and autocorrelation function (ACF) of a zero mean MA(1) process.

Solution 6.3

The mean of a MA(1) series with no drift term is:

$$\mathbb{E}[y_t] = \mathbb{E}[\theta_1 u_{t-1} + u_t] = \theta_1\mathbb{E}[u_{t-1}] + \mathbb{E}[u_t] = 0.$$

The variance of a MA(1) series with no drift term is:

$$\begin{aligned}
\mathbb{V}[y_t] &= \mathbb{E}[y_t^2] \\
\mathbb{E}[y_t^2] &= \mathbb{E}[(\theta_1 u_{t-1} + u_t)^2] = \theta_1^2\mathbb{E}[u_{t-1}^2] + \mathbb{E}[u_t^2] \\
&= (1 + \theta_1^2)\sigma^2.
\end{aligned}$$

The lag-1 autocovariance of a MA(1) series with no drift term is:

$$\gamma_1 = \mathbb{E}[y_t y_{t-1} = \mathbb{E}[(\theta_1 u_{t-1} + u_t)(\theta_1 u_{t-2} + u_{t-1})] = \theta_1 \mathbb{E}[u_{t-1}^2] = \theta_1 \sigma^2$$

The lag-1 autocorrelation is:

$$\tau_1 = \tau_1/\tau_0 = \theta_1/(1+\theta_1^2).$$

Exercise 6.4

Consider the following log-GARCH(1,1) model with a constant for the mean equation

$$\begin{aligned} y_t &= \mu + u_t,\ u_t \sim N(0, \sigma_t^2) \\ ln(\sigma_t^2) &= \alpha_0 + \alpha_1 u_{t-1}^2 + \beta_1 ln \sigma_{t-1}^2 \end{aligned}$$

- What are the advantages of a log-GARCH model over a standard GARCH model?
- Estimate the unconditional variance of y_t for the values $\alpha_0 = 0.01, \alpha_1 = 0.1, \beta_1 = 0.3$.
- Derive an algebraic expression relating the conditional variance with the unconditional variance.
- Calculate the half-life of the model and sketch the forecasted volatility.

Solution 6.4

The log-GARCH model prevents a negative volatility whereas this is possible under a GARCH model.

The unconditional variance of a GARCH model is

$$\begin{aligned} \sigma^2 := var(u_t) &= \frac{\alpha_0}{1 - (\sum_{i=1}^q \alpha_i + \sum_{i=1}^p \beta_i)} \\ &= \frac{0.01}{1 - (0.1 + 0.3)} = 0.01667 \end{aligned}$$

A necessary condition for stationarity of GARCH(p,q) models is that

$$(\sum_{i=1}^q \alpha_i + \sum_{i=1}^p \beta_i) < 1$$

which is satisfied since $0.1 + 0.3 = 0.4 < 1$.

The l-step ahead forecast using a GARCH(1,1) model:

$$\begin{aligned}
\sigma_t^2 &= \alpha_0 + \alpha_1 u_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \\
\hat{\sigma}_{t+1}^2 &= \alpha_0 + \alpha_1 \mathbb{E}[u_t^2 | \Omega_{t-1}] + \beta_1 \sigma_t^2 \\
&= \sigma^2 + (\alpha_1 + \beta_1)(\sigma_t^2 - \sigma^2) \\
\hat{\sigma}_{t+2}^2 &= \alpha_0 + \alpha_1 \mathbb{E}[u_{t+1}^2 | \Omega_{t-1}] + \beta_1 \mathbb{E}[\sigma_{t+1}^2 | \Omega_{t-1}] \\
&= \sigma^2 + (\alpha_1 + \beta_1)^2(\sigma_t^2 - \sigma^2) \\
\hat{\sigma}_{t+l}^2 &= \alpha_0 + \alpha_1 \mathbb{E}[u_{t+l-1}^2 | \Omega_{t-1}] + \beta_1 \mathbb{E}[\sigma_{t+l-1}^2 | \Omega_{t-1}] \\
&= \sigma^2 + (\alpha_1 + \beta_1)^l(\sigma_t^2 - \sigma^2),
\end{aligned}$$

which provides a relationship between the conditional variance σ_t^2 and the unconditional variance σ_2.

The half life is given by

$$K = ln(0.5)/ln(\alpha_1 + \beta_1) = ln(0.5)/ln(0.4) = 0.7564.$$

.

Exercise 6.5

Consider the simple moving average (SMA)

$$S_t = \frac{X_t + X_{t-1} + X_{t+2} + \ldots + X_{t-N+1}}{N},$$

and the exponential moving average (EMA), given by $E_1 = X_1$ and, for $t \geq 2$,

$$E_t = \alpha X_t + (1 - \alpha) E_{t-1},$$

where N is the time horizon of the SMA and the coefficient α represents the degree of weighting decrease of the EMA, a constant smoothing factor between 0 and 1. A higher α discounts older observations faster.

a. Suppose that, when computing the EMA, we stop after k terms, instead of going after the initial value. What fraction of the total weight is obtained?
b. Suppose that we require 99.9% of the weight. What k do we require?
c. Show that, by picking $\alpha = 2/(N + 1)$, one achieves the same center of mass in the EMA as in the SMA with the time horizon N.
d. Suppose that we have set $\alpha = 2/(N + 1)$. Show that the first N points in an EMA represent about 87.48% of the total weight.

Solution 6.5

a.

$$\frac{\text{weight omitted by stopping after } k \text{ terms}}{\text{total weight}}$$
$$= \frac{\alpha[(1-\alpha)^k + (1-\alpha)^{k+1} + (1-\alpha)^{k+2} + \ldots]}{\alpha[1 + (1-\alpha) + (1-\alpha)^2 + \ldots]}$$
$$= \frac{\alpha(1-\alpha)^k \frac{1}{1-(1-\alpha)}}{\frac{\alpha}{1-(1-\alpha)}}$$
$$= (1-\alpha)^k.$$

b. To have 99.9% of the weight, set the above ratio equal to 0.1% and solve for k:

$$k = \frac{\ln(0.001)}{\ln(1-\alpha)}$$

to determine how many terms should be used.

c. The weights of an N-day SMA have a center of mass on the R_{SMA}th day, where

$$R_{\text{SMA}} = \frac{N+1}{2}$$

or

$$R_{\text{SMA}} = \frac{N-1}{2},$$

if we use zero-based indexing. We shall stick with the one-based indexing.
The weights of an EMA have the center of mass

$$R_{\text{EMA}} = \alpha \sum_{k=1}^{\infty} k(1-\alpha)^{k-1}.$$

From the Maclaurin series

$$\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k,$$

taking derivatives of both sides, we get

$$(x-1)^{-2} = \sum_{k=0}^{\infty} k \cdot x^{k-1}.$$

Substituting $x = 1 - \alpha$, we get

$$R_{\text{EMA}} = \alpha(\alpha)^{-2} = (\alpha)^{-1}.$$

So the value of α that sets $R_{\text{EMA}} = R_{\text{SMA}}$ is

$$\alpha_{\text{EMA}} = \frac{2}{N_{\text{SMA}} + 1}.$$

d. From the formula for the sum of a geometric series, we obtain that the sum of the weights of all the terms (i.e., the infinite number of terms) in an EMA is 1. The sum of the weights of N terms is $1-(1-\alpha)^{N+1}$. The weight omitted after N terms is given by $1-[1-(1-\alpha)^{N+1}] = (1-\alpha)^{N+1}$. Substituting $\alpha = \frac{2}{N+1}$ and making use of $\lim_{n\infty}\left(1+\frac{a}{n}\right)^n = e^a$, we get $\lim_{N\infty}\left[1-\left(1-\frac{2}{N+1}\right)^{N+1}\right] = 1-e^{-2} \approx 0.8647$.

Exercise 6.6

Suppose that, for the sequence of random variables $\{y_t\}_{t=0}^{\infty}$ the following model holds:

$$y_t = \mu + \phi y_{t-1} + \epsilon_t, \quad |\phi| \leq 1, \quad \epsilon_t \sim \text{i.i.d.}(0, \sigma^2).$$

Derive the conditional expectation $\mathbb{E}[y_t \mid y_0]$ and the conditional variance $\mathbb{V}[y_t \mid y_0]$.

Solution 6.6

As

$$y_1 = \mu + \phi y_0 + \epsilon_1,$$

$$\begin{aligned} y_2 &= \mu + \phi y_1 + \epsilon_2 \\ &= \mu + \phi(\mu + \phi y_0 + \epsilon_1) + \epsilon_2 \\ &= \mu + \phi\mu + \phi^2 y_0 + \phi\epsilon_1 + \epsilon_2 \end{aligned}$$

$$\begin{aligned} y_3 &= \mu + \phi y_2 + \epsilon_3 \\ &= \mu + \phi(\mu + \phi\mu + \phi^2 y_0 + \phi\epsilon_1 + \epsilon_2) + \epsilon_3 \\ &= \mu + \phi\mu + \phi^2\mu + \phi^3 y_0 + \phi^2\epsilon_1 + \phi\epsilon_2 + \epsilon_3, \end{aligned}$$

proceeding on, in general,

$$y_t = \mu \sum_{i=0}^{t-1} \phi^i + \phi^t y_0 + \sum_{i=1}^{t} \phi^{t-i}\epsilon_i.$$

Hence

$$\mathbb{E}[y_t] = \mu \sum_{i=0}^{t-1} \phi^i + \phi^t y_0 + \sum_{i=1}^{t} \phi^{t-i}\mathbb{E}[\epsilon_i] = \mu \sum_{i=0}^{t-1} \phi^i + \phi^t y_0,$$

and

$$\mathbb{V}[y_t \mid y_0] = \mathbb{V}[\sum_{i=1}^{t} \phi^{t-i}\epsilon_i] = \sum_{i=1}^{t} \mathbb{V}[\phi^{t-i}\epsilon_i] = \sum_{i=1}^{t} \phi^{2(t-i)}\mathbb{V}[\epsilon_i] = \sum_{i=1}^{t} \phi^{2i}.$$

Chapter 7
Probabilistic Sequence Modeling

Exercise 7.1: Kalman Filtering of Autoregressive Moving Average $ARMA(p, q)$ Model

The *autoregressive moving average ARMA*(p, q) model can be written as

$$y_t = \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} + \eta_t + \theta_1 \eta_{t-1} + \ldots + \theta_q \eta_{t-q},$$

where $\eta_t \sim \mathcal{N}(0, \sigma^2)$, and includes as special cases all AR(p) and MA(q) models. Such models are often fitted to financial time series. Suppose that we would like to filter this time series using a Kalman filter. Write down a suitable process and the observation models.

Solution 7.1

[Kalman Filtering of Autoregressive Moving Average $ARMA(p, q)$ Model] Set $m := \max(p, q + 1)$, $\phi_i := 0$ for $i > p$, $\theta_i := 0$ for $i > q$. Then we obtain our process model as

$$\mathbf{X}_t = \mathbf{F}_t \mathbf{X}_{t-1} + \mathbf{a}_t + \mathbf{W}_t \mathbf{w}_t,$$

and the observation model as

$$\mathbf{Y}_t = \mathbf{H}_t \mathbf{X}_t + \mathbf{b}_t + \mathbf{V}_t \mathbf{v}_t,$$

where

$$\mathbf{X}_t = \begin{pmatrix} y_t \\ \phi_2 y_{t-1} + \ldots + \phi_p y_{t-m+1} + \theta_1 \eta_t + \ldots + \theta_{m-1} \eta_{t-m+2} \\ \phi_3 y_{t-1} + \ldots + \phi_p y_{t-m+2} + \theta_2 \eta_t + \ldots + \theta_{m-1} \eta_{t-m+3} \\ \vdots \\ \phi_m y_{t-1} + \theta_{m-1} \eta_t \end{pmatrix} \in \mathbb{R}^{m \times 1},$$

$$\mathbf{F} = \begin{pmatrix} \phi_1 & 1 & 0 & \cdots & 0 \\ \phi_2 & 0 & 1 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \phi_{m-1} & 0 & 0 & & 1 \\ \phi_m & 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{m \times m},$$

$$\mathbf{W} = \begin{pmatrix} 1 & \theta_1 & \cdots & \theta_{m-1} \end{pmatrix}^\intercal \in \mathbb{R}^{m \times 1},$$

$$w_t = \eta_t, \quad Q_t = \sigma^2, \quad \mathbf{H} = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix} \in 1 \times m, \quad b_t = 0, \quad V_t = 0.$$

If y_t is stationary, then $\mathbf{X}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$ with $\mathbf{P}$ given by the equation $\mathbf{P} = \mathbf{F}\mathbf{P}\mathbf{F}^\intercal + \sigma^2 \mathbf{W}\mathbf{W}^\intercal$, so we can set the initial state and error covariance to $\mathbf{0}$ and $\mathbf{P}$, respectively.

Exercise 7.2: The Ornstein-Uhlenbeck Process

Consider the one-dimensional *Ornstein–Uhlenbeck (OU)* process, the stationary Gauss–Markov process given by the SDE

$$dX_t = \theta(\mu - X_t)\,dt + \sigma\,dW_t,$$

where $X_t \in \mathbb{R}$, $X_0 = x_0$, and $\theta > 0$, μ, and $\sigma > 0$ are constants. Formulate the Kalman process model for this process.

Solution 7.2

[The Ornstein–Uhlenbeck Process] The solution to the SDE is given by

$$X_t = x_0 e^{-\theta t} + \mu(1 - e^{-\theta t}) + \int_0^t \sigma e^{-\theta(t-u)}\,dW_u.$$

An Itô integral, $\int_s^t f(u)\,dW_u$, of a deterministic integrand, $f(u)$, is a Gaussian random variable with mean 0 and variance $\int_0^t f^2(u)\,du$. In our case, $f(u) = \sigma e^{-\theta(t-u)}$, and $\int_0^t f^2(u)\,du = \frac{\sigma^2}{2\theta}(1 - e^{-2\theta t})$.

Since this Markov process is homogeneous, its transition density depends only upon the time difference. Setting, for $s \le t$, $h_k := t - s$ as the time interval between the time ticks $k-1$ and k, we obtain a discretized process model

$$X_k = F_k X_{k-1} + a_k + w_k,$$

with $F_k = e^{-\theta h_k}$, $a_k = \mu(1 - e^{-\theta h_k})$, $w_k \sim \mathcal{N}\left(0, \frac{\sigma^2}{2\theta}(1 - e^{-2\theta h_k})\right)$.

As a further exercise, consider extending this to a multivariate OU process.

Exercise 7.3: Deriving the Particle Filter for Stochastic Volatility with Leverage and Jumps

We shall regard the log-variance x_t as the hidden states and the log-returns y_t as observations. How can we use the particle filter to estimate x_t on the basis of the observations y_t?

a. Show that, in the absence of jumps,

$$x_t = \mu(1-\phi) + \phi x_{t-1} + \sigma_v \rho y_{t-1} e^{-x_{t-1}/2} + \sigma_v \sqrt{1-\rho^2}\xi_{t-1}$$

for some $\xi_t \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$.

b. Show that

$$p(\epsilon_t \mid x_t, y_t) = \delta(\epsilon_t - y_t e^{-x_t/2})\mathbb{P}[J_t = 0 \mid x_t, y_t] + \phi(\epsilon_t; \mu_{\epsilon_t \mid J_t=1}, \sigma^2_{\epsilon_t \mid J_t=1})\mathbb{P}[J_t = 1 \mid x_t, y_t].$$

where

$$\mu_{\epsilon_t \mid J_t=1} = \frac{y_t \exp(x_t/2)}{\exp(x_t) + \sigma_J^2}$$

and

$$\sigma^2_{\epsilon_t \mid J_t=1} = \frac{\sigma_J^2}{\exp(x_t) + \sigma_J^2}.$$

c. Explain how you could implement random sampling from the probability distribution given by the density $p(\epsilon_t \mid x_t, y_t)$.

d. Write down the probability density $p(x_t \mid x_{t-1}, y_{t-1}, \epsilon_{t-1})$.

e. Explain how you could sample from this distribution.

f. Show that the observation density is given by

$$p(y_t \mid \hat{x}^{(i)}_{t\mid t-1}, p, \sigma_J^2) = (1-p)\left[\left(2\pi e^{\hat{x}^{(i)}_{t\mid t-1}}\right)^{-1/2} \exp\left(-y_t^2/(2e^{\hat{x}^{(i)}_{t\mid t-1}})\right)\right] +$$
$$p\left[\left(2\pi(e^{\hat{x}^{(i)}_{t\mid t-1}} + \sigma_J^2)\right)^{-1/2} \exp\left(-y_t^2/(2e^{\hat{x}^{(i)}_{t\mid t-1}} + 2\sigma_J^2)\right)\right],$$

Solution 7.3

[Deriving the Particle Filter for Stochastic Volatility with Leverage and Jumps]

a. The Cholesky decomposition of

$$\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

can be written as $\Sigma = \mathbf{L}\mathbf{L}^\intercal$ where

$$\mathbf{L} = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{pmatrix}.$$

Set $\xi_t \sim \mathcal{N}(0,1)$, independent of ϵ_t and η_t. Then we can write

$$\begin{pmatrix} \epsilon_t \\ \eta_t \end{pmatrix} = \mathbf{L} \begin{pmatrix} \epsilon_t \\ \xi_t \end{pmatrix},$$

so

$$\eta_t = \rho\epsilon_t + \sqrt{1-\rho^2}\xi_t$$

and

$$\mathrm{Var}\left[\begin{pmatrix} \epsilon_t \\ \eta_t \end{pmatrix}\right] = \mathrm{Var}\left[\mathbf{L}\begin{pmatrix} \epsilon_t \\ \xi_t \end{pmatrix}\right] = \mathbf{L}\mathrm{Var}\left[\begin{pmatrix} \epsilon_t \\ \xi_t \end{pmatrix}\right]\mathbf{L}^\intercal = \mathbf{L}\mathbf{I}\mathbf{L}^\intercal = \Sigma,$$

as required. Substituting this into the process model, we get

$$\begin{aligned} x_{t+1} &= \mu(1-\phi) + \phi x_t + \sigma_v \eta_t \\ &= \mu(1-\phi) + \phi x_t + \sigma_v \left(\rho\epsilon_t + \sqrt{1-\rho^2}\xi_t\right) \\ &= \mu(1-\phi) + \phi x_t + \sigma_v\rho\epsilon_t + \sigma_v\sqrt{1-\rho^2}\xi_t. \end{aligned}$$

Since, in the absence of jumps, $\epsilon_t = y_t e^{-x_t/2}$, the result follows.

b. By the Law of Total Probability, we can write

$$p(\epsilon_t \mid x_t, y_t) = p(\epsilon_t \mid J_t = 0, x_t, y_t)\mathbb{P}\left[J_t = 0 \mid x_t, y_t\right] + p(\epsilon_t \mid J_t = 1, x_t, y_t)\mathbb{P}\left[J_t = 1 \mid x_t, y_t\right].$$

It is clear that, if the process does not jump, there is a Dirac delta mass at a single point, $\frac{y_t}{\exp(x_2/2)}$, so

$$p(\epsilon_t \mid J_t = 0, x_t, y_t) = \delta(\epsilon_t - y_t e^{-x_t/2}).$$

Let us find $p(\epsilon_t \mid J_t = 0, x_t, y_t)$. By Bayes's theorem,

$$\begin{aligned} p(\epsilon_t \mid J_t = 1, x_t, y_t) &\propto p(y_t \mid J_t = 1, x_t, \epsilon_t)p(\epsilon_t) \\ &= \phi\left(y_t; \epsilon_t \exp(x_t/2), \sigma_J^2\right)\phi(\epsilon_t; 0, 1). \end{aligned}$$

Taking the natural logarithm,

$$\ln p(\epsilon_t \mid J_t = 1, x_t, y_t) = C - \frac{1}{2} \cdot \frac{(y_t - \epsilon_t \exp(x_t/2))^2}{\sigma_J^2} - \frac{1}{2}\epsilon_t^2 \tag{22}$$

for some constant C. We recognise in this form the log-pdf of the normal distribution,

$$\ln \phi\left(\epsilon_t; \mu_{\epsilon_t \mid J_t=1}, \sigma^2_{\epsilon_t \mid J_t=1}\right) = C - \frac{1}{2} \cdot \frac{(\epsilon_t - \mu_{\epsilon_t \mid J_t=1})^2}{\sigma^2_{\epsilon_t \mid J_t=1}}. \tag{23}$$

We find the parameters $\mu_{\epsilon_t \mid J_t=1}$ and $\mu_{\epsilon_t \mid J_t=1}$ by equating the coefficients of ϵ^2 and ϵ in the above two equations as follows. From the first equation,

$$\ln \phi \left(\epsilon_t; \mu_{\epsilon_t \mid J_t=1}, \sigma^2_{\epsilon_t \mid J_t=1}\right) = \epsilon_t^2 \left[-\frac{\exp(x_t)}{2\sigma_J^2} - \frac{1}{2}\right] + \epsilon_t \left[\frac{y_t \exp(x_t/2)}{\sigma_J^2}\right] + \text{constant term}$$

From the second,

$$\ln \phi \left(\epsilon_t; \mu_{\epsilon_t \mid J_t=1}, \sigma^2_{\epsilon_t \mid J_t=1}\right) = \epsilon_t^2 \left[-\frac{1}{2\sigma^2_{\epsilon_t \mid J_t=1}}\right] + \epsilon_t \left[\frac{\mu_{\epsilon_t \mid J_t=1}}{\sigma^2_{\epsilon_t \mid J_t=1}}\right] + \text{constant term}$$

Equating the coefficients of ϵ_t^2 in the above two equations, we get

$$-\frac{\exp(x_t)}{2\sigma_J^2} - \frac{1}{2} = -\frac{1}{2\sigma^2_{\epsilon_t \mid J_t=1}},$$

hence

$$\sigma^2_{\epsilon_t \mid J_t=1} = \frac{\sigma_J^2}{\exp(x_t) + \sigma_J^2}.$$

Equating the coefficients of ϵ_t, we get

$$\frac{y_t \exp(x_t/2)}{\sigma_J^2} = \frac{\mu_{\epsilon_t \mid J_t=1}}{\sigma^2_{\epsilon_t \mid J_t=1}} \overset{\text{by above result}}{=} \frac{\mu_{\epsilon_t \mid J_t=1}(\exp(x_t) + \sigma_J^2)}{\sigma_J^2},$$

hence

$$\mu_{\epsilon_t \mid J_t=1} = \frac{y_t \exp(x_t/2)}{\exp(x_t) + \sigma_J^2}.$$

The result follows.

c. Since

$$x_t = \mu(1-\phi) + \phi x_{t-1} + \sigma_v \rho \epsilon_{t-1} + \sigma_v \sqrt{1-\rho^2}\xi_{t-1},$$

for some $\xi_t \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$, we have

$$p(x_t \mid x_{t-1}, y_{t-1}, \epsilon_{t-1}) = \phi \left(x_t; \mu(1-\phi) + \phi x_{t-1} + \sigma_v \rho \epsilon_{t-1}, \sigma_v^2(1-\rho^2)\right).$$

d. We first sample ϵ_{t-1} from $p(\epsilon_{t-1} \mid x_{t-1}, y_{t-1})$, then sample from the above normal pdf.

e. At t, the jump occurs with probability p and does not occur with probability $p-1$. When the jump does not occur, y_t is normally distributed with mean 0 and variance e^{x_t}. When the jump does occur, it is also normally distributed, since the sum $Z = X + Y$ of two normal random variables, $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$, is itself a normal random variable, $Z \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$. In our case, the two normal random variables are $\epsilon_t e^{x_t/2} \sim \mathcal{N}(0, e^{x_t})$ and $\varpi_t \sim \mathcal{N}(0, \sigma_J^2)$. The result follows by the Law of Total Probability.

Exercise 7.4: The Viterbi algorithm and an occasionally dishonest casino

The dealer has two coins, a fair coin, with $\mathbb{P}(\text{Heads}) = \frac{1}{2}$, and a loaded coin, with $\mathbb{P}(\text{Heads}) = \frac{4}{5}$. The dealer starts with the fair coin with probability $\frac{3}{5}$. The dealer then tosses the coin several times. After each toss, there is a $\frac{2}{5}$ probability of a switch to the other coin. The observed sequence is Heads, Tails, Tails, Heads, Tails, Heads, Heads, Heads, Tails, Heads. Run the Viterbi algorithm to determine which coin the dealer was most likely using for each coin toss.

Solution 7.4

The solution is presented in notebook `viterbi_7_4.ipynb`.

```
max_probability': 1.146617856e-05,
 'steps': ['Fair','Fair','Fair','Fair','Fair',
 'Loaded','Loaded','Loaded','Fair','Loaded']
```

Chapter 8
Advanced Neural Networks

Exercise 8.1*

Calculate the half-life of the following univariate RNN

$$\begin{aligned}\hat{x}_t &= W_y z_{t-1} + b_y,\\ z_{t-1} &= \tanh(W_z z_{t-2} + W_x x_{t-1}),\end{aligned}$$

where $W_y = 1$, $W_z = W_x = 0.5$, $b_h = 0.1$ and $b_y = 0$.

Solution 8.1

The lag 1 unit impulse gives $\hat{x}_t = \tanh(0.5 + 0.1)$ and the intermediate calculation of the ratios, $r^{(k)}$, are shown in Table 2, showing that the half-life is 4 periods.

Lag k	$r^{(k)}$
1	1.00
2	0.657
3	0.502
4	0.429

Table 2: *The half-life characterizes the memory decay of the architecture by measuring the number of periods before a lagged unit impulse has at least half of its effect at lag 1. The calculation of the half-life involves nested composition of the recursion relation for the hidden layer until $r^{(k)}$ is less than a half. In this example, the half-life is 4 periods.*

Question 8.2: Recurrent Neural Networks

- State the assumptions needed to apply plain RNNs to time series data.
- Show that a linear RNN(p) model with bias terms in both the output layer and the hidden layer can be written in the form

$$\hat{y}_t = \mu + \sum_{i=1}^{p} \phi_i y_{t-i}$$

  and state the form of the coefficients $\{\phi_i\}$.
- State the conditions on the activation function and weights in a plain RNN is the model stable? (i.e. lags do not grow)

Solution 8.2

- Data should not be i.i.d., but autocorrelated. The data should be stationary.
- Start by assuming that $W_z^{(1)} = \phi_z, |\phi_z| < 1, W_x^{(1)} = \phi_x, W_y = 1$, but b_h and b_y are not zero. Then follow the example in the chapter but keeping the extra b_h term so that $\mu = \sum_{i=1}^{p} \phi^i b_y$ so that

$$\hat{y}_t = \mu + \sum_{i=1}^{p} \phi_i y_{t-i}$$

  and $\phi_i = \phi_x \phi_z^{i-1}$.
- The following constraint on the activation function $|\sigma(x)| \leq 1$ must hold for stability of the RNN.

Exercise 8.3*

Using Jensen's inequality, calculate the lower bound on the partial autocovariance function of the following zero-mean RNN(1) process:

$$y_t = \sigma(\phi y_{t-1}) + u_t,$$

for some monotonically increasing, positive and convex activation function, $\sigma(x)$ and positive constant ϕ. Note that Jensen's inequality states that $\mathbb{E}[g(X)] \geq g(\mathbb{E}[X])$ for any convex function g of a random variable X.

Solution 8.3

The lag 1 partial autocovariance is the same as the lag-1 autocovariance:

$$\begin{aligned}
\mathbb{E}[y_t, y_{t-1}] &= \mathbb{E}[\sigma(\phi y_{t-1}) + u_t, y_{t-1}] \\
&= \mathbb{E}[\sigma(\phi y_{t-1}), y_{t-1}] \\
&= \mathbb{E}[g(y_{t-1})] \\
&\geq g(\mathbb{E}[y_{t-1}]) = \sigma(\phi \mathbb{E}[y_{t-1}])\mathbb{E}[y_{t-1}],
\end{aligned}$$

where we have made use of Jensen's inequality under the property that $g(x) = \sigma(\phi x)x$ is convex in x. Convexity holds if $\sigma(x)$ is monotonic increasing and convex and x is non-negative.

Exercise 8.4*

Show that the discrete convolution of the input sequence $X = \{3, 1, 2\}$ and the filter $F = \{3, 2, 1\}$ given by $Y = X * F$ where

$$y_i = X * F_i = \sum_{j=-\infty}^{\infty} x_j F_{i-j}$$

is $Y = \{9, 9, 11, 5, 2\}$.

Solution 8.4

$$\begin{aligned}
y_0 &= x_0 \times F_0 = 3 \times 3 = 9, \\
y_1 &= x_0 \times F_1 + x_1 \times F_0 = 3 \times 1 + 2 \times 3 = 9, \\
y_2 &= x_0 \times F_2 + x_1 \times F_1 + x_2 \times F_0 = 3 \times 2 + 2 \times 1 + 1 \times 3 = 11, \\
y_3 &= x_1 \times F_2 + x_2 \times F_1 + x_3 \times F_0 = 2 \times 2 + 1 \times 1 = 5, \\
y_4 &= x_2 \times F_2 = 1 \times 2 = 2,
\end{aligned}$$

Exercise 8.5*

Show that the discrete convolution $\hat{x}_t = F * x_t$ defines a univariate $AR(p)$ if a p-width filter is defined as $F_j := \phi^j$ for some constant parameter ϕ.

Solution 8.5

Substituting the definition of filter and expanding the discrete convolution gives:

$$\hat{x}_t = F * x_t = \sum_{j=1}^{p} F_j x_{t-j} = \phi x_{t-1} + \phi^2 x_{t-2} + \cdots + \phi^p x_{t-p},$$

which is a special case of a $AR(p)$ with geometrically decaying coefficients when $|\phi| < 1$.

4 Programming Related Questions*

Exercise 8.6***

Modify the RNN notebook to predict coindesk prices using a univariate RNN applied to the data `coindesk.csv`. Then complete the following tasks

a. Determine whether the data is stationary by applying the augmented Dickey-Fuller test.
b. Estimate the partial autocorrelation and determine the optimum lag at the 99% confidence level. Note that you will not be able to able to draw conclusions if your data is not stationarity. Choose the sequence length to be equal to this optimum lag.
c. Evaluate the MSE in-sample and out-of-sample as you vary the number of hidden neurons. What do you conclude about the level of over-fitting?
d. Apply L_1 regularization to reduce the variance.
e. How does the out-of-sample performance of a plain RNN compare with that of a GRU?
f. Determine whether the model error is white noise or is auto-correlated by applying the Ljung Box test.

Solution 8.6

See the notebook solution.

Exercise 8.7***

Modify the CNN 1D time series notebook to predict high frequency mid-prices with a single hidden layer CNN, using the data `HFT.csv`. Then complete the following tasks

a. Confirm that the data is stationary by applying the augmented Dickey-Fuller test.
b. Estimate the partial autocorrelation and determine the optimum lag at the 99% confidence level.
c. Evaluate the MSE in-sample and out-of-sample using 4 filters. What do you conclude about the level of over-fitting as you vary the number of filters?
d. Apply L_1 regularization to reduce the variance.
e. Determine whether the model error is white noise or is auto-correlated by applying the Ljung-Box test.

Hint: You should also review the HFT RNN notebook before you begin this exercise.

Solution 8.7

See the notebook solution.

Part III
Sequential Data with Decision-Making

Chapter 9
Introduction to Reinforcement learning

Exercise 9.1

Consider an MDP with a reward function $r_t = r(s_t, a_t)$. Let $Q^\pi(s,a)$ be an action-value function for policy π for this MDP, and $\pi_\star(a|s) = \arg\max_\pi Q^\pi(s,a)$ be an optimal greedy policy. Assume we define a new reward function as an affine transformation of the previous reward: $\tilde{r}(t) = wr_t + b$ with constant parameters b and $w > 0$. How does the new optimal policy $\tilde{\pi}_\star$ relate to the old optimal policy $\pi_\star$?

Solution 9.1

Using the definition of the action-value function as a conditional expectation of discounted rewards, we find that under an affine transformation of rewards $r_t \to ar_t + b$ (where $a > 0$ and b are fixed numbers), the action-value function transforms as follows: $Q^\pi(s,a) \to \tilde{Q}^\pi(s,a) = aQ^\pi(s,a) + f(s)$ where $f(s)$ is a function of the current state s. As $f(s)$ does not depend on actions a, optimal greedy policies obtained with $Q^\pi(s,a)$ and $\tilde{Q}^\pi(s,a)$ are clearly the same:

$$\pi_\star = \arg\max_\pi Q^\pi(s,a) = \arg\max_\pi \tilde{Q}^\pi(s,a)$$

Exercise 9.2

With True/False questions, give a short explanation to support your answer.

- True/False: Value iteration always find the optimal policy, when run to convergence. [3]
- True/False: Q-learning is an on policy learning (value iteration) algorithm and estimates updates to the state-action-value function, $Q(s,a)$ using actions taken under the current policy π. [5]

- For Q-learning to converge we need to correctly manage the exploration vs. exploitation tradeoff. What property needs to hold for the exploration strategy? [4]
- True/False: Q-learning with linear function approximation will always converge to the optimal policy. [2]

Solution 9.2

- True: Value iteration always find the optimal policy, when run to convergence (This is a result of the Bellman equations being a contraction mapping).
- False: Q-learning is an *off-policy* learning algorithm: the $Q(s,a)$ function is learned from different actions (for example, random actions). We even don't need a policy at all to learn $Q(s,a)$.
  $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$, where a' are *all* actions, that were probed in state s' (not actions under the current policy).
- In the limit, *every action* needs to be tried sufficiently often in every possible state. This can be guaranteed with an sufficiently permissive exploration strategy.
- False. It may not even be able to represent the optimal policy due to approximation error.

Exercise 9.3*

Consider the following toy **cash buffer problem**. An investor owns a stock, initially valued at $S_{t_0} = 1$, and must ensure that their wealth (stock + cash) is not less than a certain threshold K at time $t = t_1$. Let $W_t = S_t + C_t$ denote their at time t, where C_t is the total cash in the portfolio. If the wealth $W_{t_1} < K = 2$ then the investor is penalized with a -10 reward.

The investor chooses to inject either 0 or 1 amounts of cash with a respective penalty of 0 or -1 (which is not deducted from the fund).

Dynamics The stock price follows a discrete Markov chain with $P(S_{t+1} = s \mid S_t = s) = 0.5$, i.e. with probability 0.5 the stock remains the same price over the time interval. $P(S_{t+1} = s+1 \mid S_t = s) = P(S_{t+1} = s-1 \mid S_t = s) = 0.25$. If the wealth moves off the grid it simply bounces to the nearest value in the grid at that time. The states are grid squares, identified by their row and column number (row first). The investor always starts in state (1,0) (i.e. the initial wealth $W_{t_0} = 1$ at time $t_0 = 0$—there is no cash in the fund) and both states in the last column (i.e., at time $t = t_1 = 1$) are terminal (Table 3).

Using the Bellman equation (with generic state notation), give the first round of value iteration updates for each state by completing the table below. You may ignore the time value of money, i.e., set $\gamma = 1$.

$$V_{i+1}(\mathbf{s}) = \max_a\Big(\sum_{\mathbf{s}'} T(\mathbf{s},a,\mathbf{s}')(R(\mathbf{s},a,\mathbf{s}') + \gamma V_i(\mathbf{s}'))\Big)$$

w	t_0	t_1
2	0	0
1	0	-10

Table 3: The reward function depends on fund wealth w and time.

(w,t)	(1,0)	(2,0)
$V_0(w)$	0	0
$V_1(w)$	?	NA

Solution 9.3

The stock price sequence $\{S_{t_0}, S_{t_1}\}$ is a Markov chain with one transition period. The transition probabilities are denoted $p_{ji} = P(S_{t+1} = s_j | S_t = s_i)$ with states $s_1 = 1, s_2 = 2$.

The Wealth sequence, $\{W_{t_0}, W_{t_1}\}$ is an MDP. To avoid confusion in notation with the stock states, let us denote the wealth states $w \in \{w_1 = 1, w_2 = 2\}$ (instead of the usual $s_1, s_2, ..$). The actions $a \in A := \{a_0 = 0, a_1 = 1\}$. The transition matrix for this MDP is

$$T_{ijk} := T(w_i, a_k, w_j) := P(W_{t+1} = w_j | W_t = w_i, a_t = a_k).$$

From the problem constraints and the Markov chain transition probabilities p_{ij} we can write for action a_0 (no cash injection):

$$T_{,,0} = \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix}$$

and for action a_1 (one dollar of cash)

$$T_{,,1} = \begin{bmatrix} 0.25 & 0.75 \\ 0.0 & 1.0 \end{bmatrix}$$

We can now evaluate the rewards for the case when $W_{t_0} = w_1 = 1$ (which is the only starting state). The rewards are $R(w_1, 0, w_1) = -10, R(w_1, 0, w_2) = 0, R(w_1, 1, w_1) = -11, R(w_1, 1, w_2) = -1$

And using the Bellman equation, the value updates from state $W_{t_0} = w_1$ is

$$\begin{aligned}
V_1(w_1) &= \max_{a \in A} \sum_{j=1}^{2} T(w_1, a, w_j) R(w_1, a, w_j) \\
&= \max\{T(w_1, 0, w_1)R(w_1, 0, w_1) + T(w_1, 0, w_2)R(w_1, 0, w_2), T(w_1, 1, w_1)R(w_1, 1, w_1) + T(w_1, 1, w_2)R(w_1, 1, w_2)\} \\
&= \max(0.75 \times -10 + 0.25 \times 0, 0.25 \times -11 + 0.75 \times -1) \\
&= -3.5
\end{aligned}$$

for action $a_{t_0} = 1$. Note that $V_1(w_2)$ does not exist as the wealth can not transition from w_2 at time t_0.

Exercise 9.4*

Consider the following toy **cash buffer problem**. An investor owns a stock, initially valued at $S_{t_0} = 1$, and must ensure that their wealth (stock + cash) does not fall below a threshold $K = 1$ at time $t = t_1$. The investor can choose to either sell the stock or inject more cash, but not both. In the former case, the sale of the stock at time t results in an immediate cash update s_t (you may ignore transactions costs). If the investor chooses to inject a cash amount $c_t \in \{0, 1\}$, there is a corresponding penalty of $-c_t$ (which is not taken from the fund).

Let $W_t = S_t + C_t$ denote their wealth at time t, where C_t is the total cash in the portfolio.

Dynamics The stock price follows a discrete Markov chain with $P(S_{t+1} = s \mid S_t = s) = 0.5$, i.e., with probability 0.5 the stock remains the same price over the time interval. $P(S_{t+1} = s + 1 \mid S_t = s) = P(S_{t+1} = s - 1 \mid S_t = s) = 0.25$. If the wealth moves off the grid it simply bounces to the nearest value in the grid at that time. The states are grid squares, identified by their row and column number (row first). The investor always starts in state (1,0) (i.e. the initial wealth $W_{t_0} = 1$ at time $t_0 = 0$—there is no cash in the fund) and both states in the last column (i.e., at time $t = t_1 = 1$) are terminal.

w	t_0	t_1
1	0	0
0	0	-10

Table 4: The reward function depends on fund wealth w and time.

Using the Bellman equation (with generic state notation), give the first round of value iteration updates for each state by completing the table below. You may ignore the time value of money, i.e., set $\gamma = 1$.

$$V_{i+1}(\mathbf{s}) = \max_a \left(\sum_{\mathbf{s}'} T(\mathbf{s}, a, \mathbf{s}')(R(\mathbf{s}, a, \mathbf{s}') + \gamma V_i(\mathbf{s}'))\right)$$

(w,t)	(0,0)	(1,0)
$V_0(w)$	0	0
$V_1(w)$	?	?

Solution 9.4

The stock price sequence $\{S_{t_0}, S_{t_1}\}$ is a Markov chain with one transition period. The transition probabilities are denoted $p_{ji} = P(S_{t+1} = s_j | S_t = s_i)$ with states $s_0 = 0, s_1 = 1, s_2 = 2$.

The Wealth sequence, $\{W_{t_0}, W_{t_1}\}$ is an MDP. Let us denote the wealth states $w \in \{w_0 = 0, w_1 = 1\}$. The actions $a := \{a_0, a_1, a_2\}$ respectively denote (i) do not inject cash or sell the stock; (ii) sell the stock but do not inject cash; and (iii) inject cash (\$ 1) but do not sell the stock.

The transition matrix for this MDP is

$$T_{ijk} := T(w_i, a_k, w_j) := P(W_{t+1} = w_j | W_t = w_i, a_t = a_k).$$

From the problem constraints and the Markov chain transition probabilities p_{ij} we can write for action a_0 (no cash injection or sale):

$$T_{,,0} = \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix}$$

For the sale of the stock under action a_1:

$$T_{,,1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

For action a_2 (one dollar of cash added)

$$T_{,,2} = \begin{bmatrix} 0.25 & 0.75 \\ 0.0 & 1.0 \end{bmatrix}$$

We can now evaluate the rewards for the case when $W_{t_0} = w_1 = 1$ (which is the only starting state). The rewards are

$$\begin{aligned} R(w_1, a_0, w_1) = 0, \;\; R(w_1, a_0, w_0) = -10, \\ R(w_1, a_1, w_1) = 0, \;\; R(w_1, a_1, w_0) = -10 \\ R(w_1, a_2, w_1) = -1, \;\; R(w_1, a_2, w_0) = -11 \end{aligned}$$

And using the Bellman equation, the value updates from state $W_{t_0} = w_1$ is

$$\begin{aligned} V_1(w_1) &= \max_{a \in A} \sum_{j=0}^{1} T(w_1, a, w_j) R(w_1, a, w_j) \\ &= \max\{T(w_1, a_0, w_1)R(w_1, a_0, w_1) + T(w_1, a_0, w_0)R(w_1, a_0, w_0), \\ &\quad T(w_1, a_1, w_1)R(w_1, a_1, w_1) + T(w_1, a_1, w_0)R(w_1, a_1, w_0), \\ &\quad T(w_1, a_2, w_1)R(w_1, a_2, w_1) + T(w_1, a_2, w_0)R(w_1, a_2, w_0)\} \\ &= \max(0.75 \times 0 + 0.25 \times -10, 1.0 \times 0 + 0 \times -10, 1.0 \times -1 + 0 \times -11) \\ &= \max(-2.5, 0, -1) \end{aligned}$$

which states that taking action $a_{t_0} = a_1$ (i.e. sell the stock) is the most valuable state.

The other empty value in the table is not filled in. $V_1(w_0)$ does not exist as the wealth can not transition from w_0 at time t_0.

Exercise 9.5*

Deterministic policies such as the greedy policy $pi_\star(a|s) = \arg\max_\pi Q^\pi(s,a)$ are invariant with respect to a shift of the action-value function by an arbitrary function of a state $f(s)$: $\pi_\star(a|s) = \arg\max_\pi Q^\pi(s,a) = \arg\max_\pi \tilde{Q}^\pi(s,a)$ where $\tilde{Q}^\pi(s,a) = Q^\pi(s,a) - f(s)$. Show that this implies that the optimal policy is also invariant with respect to the following transformation of an original reward function $r(s_t, a_t, s_{t+1})$:

$$\tilde{r}(s_t, a_t, s_{t+1}) = r(s_t, a_t, s_{t+1}) + \gamma f(s_{t+1}) - f(s_t)$$

This transformation of a reward function is known as *reward shaping* (Ng, Russell 1999). It has been used in reinforcement learning to accelerate learning in certain settings. In the context of inverse reinforcement learning, reward shaping invariance has far-reaching implications, as we will discuss later in the book.

Solution 9.5

Consider the Bellman optimality equation for the action-value function where we subtract an arbitrary function of the state $f(s)$ from both sides of the equation. We can write it as follows:

$$Q(s,a) - f(s) = \mathbb{E}\left[r(s,a,s') + \gamma f(s') - f(s) + \gamma \max_{a'}\left(Q(s',a') - f(s')\right)\right]$$

This is equivalent to an MDP with a modified reward $\tilde{r}_t = r(s,a,s') + \gamma f(s') - f(s)$ and a new action-value function $\tilde{Q}(s,a) = Q(s,a) - f(s)$. Because the MDP with the new reward is obtained by the identity transformation of the original Bellman optimality equation, the optimal policy obtained with the "shaped" reward $\tilde{r}_t$ will be identical to the optimal policy obtained with the original reward $r_t = r(s_t, a_t, s_{t+1})$.

Exercise 9.6**

Define the occupancy measure $\rho_\pi \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ by the relation

$$\rho_\pi(s,a) = \pi(a|s) \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s|\pi)$$

where $\Pr(s_t = s|\pi)$ is the probability density of the state $s = s_t$ at time t following policy π. The occupancy measure $\rho_\pi(s,a)$ can be interpreted as an unnormalized density of state-action pairs. It can be used e.g. to specify the value function as an expectation value of the reward: $V =< r(s, a) >_\rho$.

a. Compute the policy in terms of the occupancy measure ρ_π.

b. Compute a normalized occupancy measure $\tilde{\rho}_\pi(s, a)$. How different the policy will be if we used the normalized measure $\tilde{\rho}_\pi(s, a)$ instead of the unnormalized measure ρ_π?

Solution 9.6

a. The policy is

$$\pi(a|s) = \frac{\rho_\pi(s, a)}{\sum_s \rho_\pi(s, a)}$$

b. The occupance measure is

$$\tilde{\rho}(s, a) = \frac{\rho_\pi(s, a)}{\sum_{s,a} \rho_\pi(s, a)} = \frac{\rho_\pi(s, a)}{\sum_{t=0}^{\infty} \gamma^t} = (1 - \gamma)\rho(s, a)$$

Therefore, if we rescale the occupance measure ρ_π by a constant factor $1 - \gamma$, we obtain a normalized probability density $\tilde{\rho}(s, a)$ of state-action pairs. On the other hand, an optimal policy is invariant under a rescaling of all rewards by a constant factor (see Exercise 9.1). This means that we can always consider the occupancy measure ρ_π to be a valid normalized probability density, as any mismatch in the normalization could always be re-absorbed in rescaled rewards.

Exercise 9.7**

Theoretical models for reinforcement learning typically assume that rewards $r_t := r(s_t, a_t, s_{t+1})$ are bounded: $r_{min} \leq r_t \leq r_{max}$ with some fixed values r_{min}, r_{max}. On the other hand, some models of rewards used by practitioners may produce (numerically) unbounded rewards. For example, with linear architectures, a popular choice of a reward function is a linear expansion $r_t = \sum_{k=1}^{K} \theta_k \Psi_k(s_t, a_t)$ over a set of K basis functions Ψ_k. Even if one chooses a set of bounded basis functions, this expression may become unbounded via a choice of coefficients θ_t.

a. Use the policy invariance under linear transforms of rewards (see Exercise 9.1) to equivalently formulate the same problem with rewards that are bounded to the unit interval $[0, 1]$, so they can be interpreted as probabilities.

b. How could you modify a linear unbounded specification of reward $r_\theta(s, a, s') = \sum_{k=1^K} \theta_k \Psi_k(s, a, s')$ to a bounded reward function with values in a unit interval $[0, 1]$?

Solution 9.7

a. As greedy policies are invariant under affine transformations of a reward function $r_t \rightarrow w r_t + b$ with fixed parameters $w > 0$ and b (see Exercise 9.1), by a proper choice of parameters w, b, we can always map a finite interval $[r_{min}, r_{max}]$ onto a unit interval $[0, 1]$. Upon such transformation, rewards can be interpreted as probabilities.

b. Once we realize that any MDP with bounded rewards can be mapped onto an MDP with rewards $r_t \in [0, 1]$ without any loss of generality, a simple alternative to a linear reward would be a logistic reward:

$$r_\theta(s_t, a_t, s_{t+1}) = \frac{1}{1 + \exp\left(-\sum_{k=1}^{K} \theta_k \Psi_k(s_t, a_t, s_{t+1})\right)}.$$

Clearly, while a logistic reward is one simple choice of a bounded function on a unit interval, other specifications are possible as well.

Exercise 9.8

Consider an MDP with a finite number of states and actions in a real-time setting where the agent learns to act optimally using the *ε-greedy policy*. The ε-greedy policy amounts to taking an action $a_\star = \operatorname{argmax}_{a'} Q(s, a')$ in each state s with probability $1 - \varepsilon$, and taking a random action with probability ε. Will SARSA and Q-learning converge to the same solution under such policy, using a constant value of ε? What will be different in the answer if ε decays with the epoch, e.g. as $\varepsilon_t \sim 1/t$?

Solution 9.8

If ε is fixed, then SARSA will converge to an optimal ε-greedy policy, while Q-learning with converge to an optimal policy. If ε is gradually reduced with the epoch, e.g. $\varepsilon_t \sim 1/t$, then both SARSA and Q-learning converge to the same optimal policy.

Exercise 9.9

Consider the following single-step random cost (negative reward)

$$C\left(s_t, a_t, s_{t+1}\right) = \eta a_t + \left(K - s_{t+1} - a_t\right)_+,$$

where η and K are some parameters. You can use such a cost function to develop an MDP model for an agent learning to invest. For example, s_t can be the current assets in a portfolio of equities at time t, a_t be an additional cash added to or subtracted from the portfolio at time t, and s_{t+1} be the portfolio value at the end of time interval $[t, t+1)$. The second term is an option-like cost of a total portfolio (equity and cash) shortfall by time $t+1$ from a target value K. Parameter η controls the relative importance of paying costs now as opposed to delaying payment.

a. What is the corresponding expected cost for this problem, if the expectation is taken w.r.t. to the stock prices and a_t is treated as deterministic?
b. Is the expected cost a convex or concave function of the action a_t?
c. Can you find an optimal one-step action $a_t^\star$ that minimizes the one-step expected cost?

Hint: For Part (i), you can use the following property:

$$\frac{d}{dx}[y-x]_+ = \frac{d}{dx}[(y-x)H(y-x)]$$

, where $H(x)$ is the Heaviside function.

Solution 9.9

a. The expected cost is

$$C(s_t, a_t, s_{t+1}) = \mathbb{E}[\eta a_t + (K - s_{t+1} - a_t)_+] = \eta a_t + \mathbb{E}[(K - s_{t+1} - a_t)_+].$$

b. To find the maximum of the expected one-step reward, we use the relation $\frac{d}{dx}[y-x]_+ = \frac{d}{dx}[(y-x)H(y-x)] = -H(y-x) - (y-x)\delta(y-x) = -H(y-x)$, where $H(x)$ is the Heaviside step function. Note that the $(y-x)\delta(y-x)$ term is zero everywhere. Using this relation, the derivative of the expected reward is

$$\frac{\partial C}{\partial a_t} = \eta - \mathbb{E}[H(K - s_{t+1} - a_t)] = \eta - \Pr(s_{t+1} \leq K - a_t).$$

Taking the second derivative and assuming that the distribution of s_{t+1} has a pdf $p(s_{t+1})$, we obtain $\frac{\partial^2 C}{\partial a_t^2} = p(K - a_t) \geq 0$. This shows that the expected cost $C(s_t, a_t, s_{t+1})$ is convex in action a_t.

c. An implicit equation for the optimal action $a_t^\star$ is obtained at a zero of the derivative of the expected cost:

$$\Pr\left(s_{t+1} \leq K - a_t^\star\right) = \eta.$$

Exercise 9.10

Exercise 9.9 presented a simple single-period cost function that can be used in the setting of model-free reinforcement learning. We can now formulate a model based formulation for such an option-like reward. To this end, we use the following specification of the random end-of-period portfolio state:

$$\begin{aligned} s_{t+1} &= (1 + r_t)s_t \\ r_t &= G(\mathbf{F}_t) + \varepsilon_t. \end{aligned}$$

In words, the initial portfolio value $s_t + a_t$ in the beginning of the interval $[t, t+1)$ grows with a random return r_t given by a function $G(\mathbf{F}_t)$ of factors $\mathbf{F}_t$ corrupted by noise ε with $\mathbb{E}[\varepsilon] = 0$ and $\mathbb{E}\left[\varepsilon^2\right] = \sigma^2$.

a. Obtain the form of expected cost for this specification in Exercise 9.9.
b. Obtain the optimal single-step action for this case.
c. Compute the sensitivity of the optimal action with respect to the i-th factor $\mathbf{F}_{it}$ assuming the sigmoid link function $G(\mathbf{F}_t) = \sigma\left(\sum_i \omega_i F_{it}\right)$ and a Gaussian noise ε_t.

Solution 9.10

a. The model dependent expected cost in this case is

$$C(s_t, a_t) = \eta a_t + \mathbb{E}\left[(K - (1 + G(\mathbf{F}_t) + \sigma\varepsilon)s_t - a_t)_+\right].$$

b. Equating the partial derivative $\partial C / \partial a_t$ to zero, we obtain an implicit equation for the optimal action $a_t^\star$

$$\mathbf{P}\left[\varepsilon_t \le \frac{K - a_t^\star - (1 + G(\mathbf{F}_t))s_t}{\sigma s_t}\right] = \eta.$$

c. If $\varepsilon \sim N(0,1)$, the last formula becomes

$$\mathcal{N}\left(\frac{K - a_t^\star - (1 + G(\mathbf{F}_t))s_t}{\sigma s_t}\right) = \eta.$$

where $\mathcal{N}(\cdot)$ is the cumulative normal distribution. Differentiating this expression with respect to factor F_{it} using the sigmoid link function $G(\mathbf{F}_t) = \sigma\left(\sum_i \omega_i F_{it}\right) = \sigma(F)$, we obtain

$$\frac{\partial a_t^\star}{\partial F_{it}} = -s_t \frac{\partial G(\mathbf{F}_t)}{\partial F_{it}} = -s_t \omega_i \sigma(F)\left(1 - \sigma(F)\right).$$

Exercise 9.11

Assuming a discrete set of actions $a_t \in \mathcal{A}$ of dimension K—show that deterministic policy optimization by greedy policy of Q-learning $Q(s_t, a_t^\star) = \max_{a_t \in \mathcal{A}} Q(s_t, a_t)$ can be equivalently expressed as maximization over a set *probability distributions* $\pi(a_t)$ with probabilities π_k for $a_t = A_k$, $k = 1, \ldots K$ (this relation is known as

Fenchel duality):

$$\max_{a_t \in \mathcal{A}} Q(s_t, a_t) = \max_{\{\pi\}_k} \sum_{k=1}^{K} \pi_k Q\left(s_t, A_k\right) \quad \text{s.t. } 0 \le \pi_i \le 1, \ \sum_{k=1}^{K} \pi_k = 1$$

Solution 9.11

As all weights are between zero and one, the solution to the maximization over the distribution $\pi = \{\pi_k\}_{k=1}^{K}$ is to put all weights into a value $k = k_\star$ such that $a_t = A_{k_\star}$ maximizes $Q(s_t, a_t)$: $\pi_{k_\star} = 1, \ \pi_k = 0, \forall k \neq k_\star$. This means that maximization over all possible actions for a discrete action set can be reduced to a linear programming problem.

Exercise 9.12**

The reformulation of a deterministic policy search in terms of search over probability distributions given in Exercise 9.8 is a mathematical identity where the end result is still a deterministic policy. We can convert it to a probabilistic policy search if we modify the objective function

$$\max_{a_t \in \mathcal{A}} Q(s_t, a_t) = \max_{\{\pi\}_k} \sum_{k=1}^{K} \pi_k Q\left(s_t, A_k\right) \quad \text{s.t. } 0 \le \pi_i \le 1, \ \sum_{k=1}^{K} \pi_k = 1$$

by adding to it a KL divergence of the policy π with some reference ("prior") policy ω:

$$G^\star(s_t, a_t) = \max_{\pi} \sum_{k=1}^{K} \pi_k Q\left(s_t, A_k\right) - \frac{1}{\beta} \sum_{k=1}^{K} \pi_i \log \frac{\pi_k}{\omega_k}$$

where β is a regularization parameter controlling the relative importance of the two terms that enforce, respectively, maximization of the action-value function and a preference for a previous reference policy ω with probabilities ω_k. When parameter $\beta < \infty$ is finite, this produces a stochastic rather than deterministic optimal policy $\pi^\star(a|s)$.

Find the optimal policy $\pi^\star(a|s)$ from the entropy-regularized functional $G(s_t, a_t)$ (Hint: use the method of Lagrange multipliers to enforce the normalization constraint $\sum_k \pi_k = 1$).

Solution 9.12

By changing the sign to replace maximization by minimization, rescaling, and using the Lagrange multiplier method to enforce the normalization constraint, we have the following Lagrangian function

$$\mathcal{L} = \sum_{k=1}^{K} \pi_i \log \frac{\pi_k}{\omega_k} - \beta \sum_{k=1}^{K} \pi_k Q\left(s_t, A_k\right) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right),$$

where λ is the Lagrange multiplier. Computing its variational derivative with respect to π_k, we obtain

$$\pi_k = \omega_k e^{\beta Q(s_t, A_k) - \lambda - 1}$$

The Lagrange multiplier λ can now be found by substituting this expression into the normalization condition $\sum_{k=1}^{K} \pi_k = 1$. This produces the final result

$$\pi_k = \frac{\omega_k e^{\beta Q(s_t, A_k)}}{\sum_k \omega_k e^{\beta Q(s_t, A_k)}}.$$

Exercise 9.13**

Regularization by KL-divergence with a reference distribution ω introduced in the previous exercise can be extended to a multi-period setting. This produces maximum entropy reinforcement learning which augments the standard RL reward by an additional entropy penalty term in the form of KL divergence. The optimal value function in MaxEnt RL is

$$F^\star(s) = \max_\pi \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t \left(r(s_t, a_t, s_{t+1}) - \frac{1}{\beta} \log \frac{\pi(a_t|s_t)}{\pi_0(a_t|s_t)} \right) \middle| s_0 = s \right] \tag{24}$$

where $\mathbb{E}[\cdot]$ stands for an average under a stationary distribution $\rho_\pi(a) = \sum_s \mu_\pi(s)\pi(a|s)$ where $\mu_\pi(s)$ is a stationary distribution over states induced by the policy π, and π_0 is some reference policy. Show that the optimal policy for this entropy-regularized MDP problem has the following form:

$$\pi^\star(a|s) = \frac{1}{Z_t} \pi_0(a_t|s_t) e^{\beta G_t^\pi(s_t, a_t)}, \quad Z_t \equiv \sum_{a_t} \pi_0(a_t|s_t) e^{\beta G_t^\pi(s_t, a_t)} \tag{25}$$

where $G_t^\pi(s_t, a_t) = \mathbb{E}^\pi\left[r(s_t, a_t, s_{t+1})\right] + \gamma \sum_{s_{t+1}} p(s_{t+1}|s_t, a_t) F_{t+1}^\pi(s_{t+1})$. Check that the limit $\beta \to \infty$ reproduces the standard deterministic policy, that is $\lim_{\beta\to\infty} V^\star(s) = \max_\pi V^\pi(s)$, while in the opposite limit $\beta \to 0$ we obtain a random and uniform policy. We will return to entropy-regularized value-based RL and stochastic policies such as (25) (which are sometimes referred to as Boltzmann policies) in later chapters of this book.

Solution 9.13

The entropy-regulated value function for policy π is

$$F_t^\pi(s_t) = \mathbb{E}^\pi \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \left(r(s_t, a_t, s_{t+1}) - \frac{1}{\beta} \log \frac{\pi(a_t|s_t)}{\pi_0(a_t|s_t)} \right) \middle| s_t \right]$$

Note that this expression coincides with the usual definition of the value function in the limit $\beta \to \infty$. Consider a similar extension to the action-value function:

$$G_t^\pi(s_t, a_t) = \mathbb{E}^\pi \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} \left(r(s_t, a_t, s_{t+1}) - \frac{1}{\beta} \log \frac{\pi(a_t|s_t)}{\pi_0(a_t|s_t)} \right) \middle| s_t, a_t \right]$$

From these two expressions, and using the fact that the entropy term vanishes for the first time step where the action a_0 is fixed, we obtain

$$F_t^\pi(s_t) = \sum_{a_t} \pi(a_t|s_t) \left[G_t^\pi(s_t, a_t) - \frac{1}{\beta} \log \frac{\pi(a_t|s_t)}{\pi_0(a_t|s_t)} \right]$$

Maximizing this expression with respect to $\pi(a_t|s_t)$, we obtain

$$\pi(a_t|s_t) = \frac{1}{Z_t} \pi_0(a_t|s_t) e^{\beta G_t^\pi(s_t, a_t)}, \quad Z_t \equiv \sum_{a_t} \pi_0(a_t|s_t) e^{\beta G_t^\pi(s_t, a_t)}$$

where $G_t^\pi(s_t, a_t) = \mathbb{E}^\pi \left[r(s_t, a_t, s_{t+1}) \right] + \gamma \sum_{s_{t+1}} p(s_{t+1}|s_t, a_t) F_{t+1}^\pi(s_{t+1})$.

Exercise 9.14*

Show that the solution for the coefficients W_{tk} in the LSPI method (see Eq.(9.71)) is

$$\mathbf{W}_t^\star = \mathbf{S}_t^{-1} \mathbf{M}_t$$

where $\mathbf{S_t}$ is a matrix and $\mathbf{M_t}$ is a vector with the following elements:

$$\begin{aligned} S_{nm}^{(t)} &= \sum_{k=1}^{N} \Psi_n \left(X_t^{(k)}, a_t^{(k)} \right) \Psi_m \left(X_t^{(k)}, a_t^{(k)} \right) \\ M_n^{(t)} &= \sum_{k=1}^{N} \Psi_n \left(X_t^{(k)}, a_t^{(k)} \right) \left(R_t \left(X_t^{(k)}, a_t^{(k)}, X_{t+1}^{(k)} \right) + \gamma Q_{t+1}^\pi \left(X_{t+1}^{(k)}, \pi \left(X_{t+1}^{(k)} \right) \right) \right) \end{aligned}$$

Solution 9.14

The objective function is

$$\mathcal{L}_t \left(\mathbf{W}_t \right) = \sum_{k=1}^{N} \left(R_t \left(X_t^{(k)}, a_t^{(k)}, X_{t+1}^{(k)} \right) + \gamma Q_{t+1}^\pi \left(X_{t+1}^{(k)}, \pi \left(X_{t+1}^{(k)} \right) \right) - \mathbf{W}_t \mathbf{\Psi} \left(X_t^{(k)}, a_t^{(k)} \right) \right)^2$$

Differentiating this expression with respect to parameters $\mathbf{W}_t$, equating it to zero, and re-arranging terms, we obtain the result shown in the exercise.

Exercise 9.15**

Consider the Boltzmann weighted average of a function $h(i)$ defined on a binary set $I = \{1, 2\}$:

$$\text{Boltz}_\beta\, h(i) = \sum_{i \in I} h(i) \frac{e^{\beta h(i)}}{\sum_{i \in I} e^{\beta h(i)}}$$

a. Verify that this operator smoothly interpolates between the max and the mean of $h(i)$ which are obtained in the limits $\beta \to \infty$ and $\beta \to 0$, respectively.
b. By taking $\beta = 1$, $h(1) = 100$, $h(2) = 1$, $h'(1) = 1$, $h'(2) = 0$, show that Boltz_β is not a non-expansion.
c. (Programming) Using operators that are not non-expansions can lead to a loss of a solution in a generalized Bellman equation. To illustrate such phenomenon, we use the following simple example.Consider the MDP problem on the set $I = \{1, 2\}$ with two actions a and b and the following specification: $p(1|1, a) = 0.66$, $p(2|1, a) = 0.34$, $r(1, a) = 0.122$ and $p(1|1, b) = 0.99$, $p(1|1, b) = 0.01$, $r(1, b) = 0.033$. The second state is absorbing with $p(1|2) = 0$, $p(2|2) = 1$. The discount factor is $\gamma = 0.98$. Assume we use the Boltzmann policy

$$\pi(a|s) = \frac{e^{\beta \hat{Q}(s,a)}}{\sum_a e^{\beta \hat{Q}(s,a)}}$$

Show that the SARSA algorithm

$$\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha \left[r(s, a) + \gamma \hat{Q}(s', a') - \hat{Q}(s, a) \right],$$

where a, a' are drawn from the Boltzmann policy with $\beta = 16.55$ and $\alpha = 0.1$, leads to oscillating solutions for $\hat{Q}(s_1, a)$ and $\hat{Q}(s_1, a)$ that do not achieve stable states with an increased number of iterations.

Solution 9.15

Part 2:
To verify the non-expansion property, we compute

$$|\text{Boltz}_T\, h(i) - \text{Boltz}_T\, h'(i)| = \left| \sum_{i \in I} h(i) \frac{e^{h(i)/T}}{\sum_{i \in I} e^{h(i)/T}} - \sum_{i \in I} h'(i) \frac{e^{h'(i)/T}}{\sum_{i \in I} e^{h'(i)/T}} \right|$$
$$\simeq |100 + 0 - 0.731 - 0| = 99.269 > 99 = \max_{i \in I} |h(i) - h'(i)|$$

Exercise 9.16**

An alternative continuous approximation to the intractable *max* operator in the Bellman optimality equation is given by the *mellowmax* function:

$$mm_\omega(\mathbf{X}) = \frac{1}{\omega}\log\left(\frac{1}{n}\sum_{i=1}^{n} e^{\omega x_i}\right)$$

a. Show that the mellowmax function recovers the *max* function in the limit $\omega \to \infty$.
b. Show that mellowmax is a non-expansion.

Solution 9.16

a. Let $m = \max(\mathbf{X})$ and $W \geq 1$ is the number of values in $\mathbf{X}$ that are equal to m. We obtain

$$\lim_{\omega\to\infty} mm_\omega(\mathbf{X}) = \lim_{\omega\to\infty}\frac{1}{\omega}\log\left(\frac{1}{n}\sum_{i=1}^{n} e^{\omega x_i}\right) = \lim_{\omega\to\infty}\frac{1}{\omega}\log\left(\frac{1}{n}e^{\omega m}\sum_{i=1}^{n} e^{\omega(x_i-m)}\right)$$
$$= \lim_{\omega\to\infty}\frac{1}{\omega}\log\left(\frac{1}{n}e^{\omega m}W\right) = m = \max(\mathbf{X})$$

b. Let $\mathbf{X}$ and $\mathbf{Y}$ be two vectors of length n, and $\Delta_i = X_i - Y_i$ be the difference of their i-th components. Let $i^\star$ be the index with the maximum component-wise difference: $i^\star = \text{argmax}_i \Delta_i$. Without loss of generality, we assume that $x_{i^\star} - y_{i^\star} \geq 0$. We obtain

$$|mm_\omega(\mathbf{X}) - mm_\omega(\mathbf{Y})| = \left|\frac{1}{\omega}\log\frac{\frac{1}{n}\sum_{i=1}^{n} e^{\omega x_i}}{\frac{1}{n}\sum_{i=1}^{n} e^{\omega y_i}}\right| = \left|\frac{1}{\omega}\log\frac{\sum_{i=1}^{n} e^{\omega(y_i+\Delta_i)}}{\sum_{i=1}^{n} e^{\omega y_i}}\right| \leq \left|\frac{1}{\omega}\log\frac{\sum_{i=1}^{n} e^{\omega(y_i+\Delta_{i^\star})}}{\sum_{i=1}^{n} e^{\omega y_i}}\right|$$
$$= \left|\frac{1}{\omega}\log e^{\omega\Delta_{i^\star}}\right| = |\Delta_{i^\star}| = \max_i |X_i - Y_i|$$

Chapter 10
Applications of Reinforcement Learning

Exercise 10.1

Derive Eq.(10.46) that gives the limit of the optimal action in the QLBS model in the continuous-time limit.

Solution 10.1

The first term in Eq.(10.46) is evaluated in the same way as Eq.(10.22) and yields

$$\lim_{\Delta t \to 0} \frac{\mathbb{E}_t\left[\Delta \hat{S}_t \hat{\Pi}_{t+1}\right]}{\mathbb{E}_t\left[\left(\Delta \hat{S}_t\right)^2\right]} = \frac{\partial \hat{C}_t}{\partial S_t}.$$

The second term is

$$\lim_{\Delta t \to 0} \frac{\mathbb{E}_t\left[\frac{1}{2\gamma\lambda}\Delta S_t\right]}{\mathbb{E}_t\left[\left(\Delta \hat{S}_t\right)^2\right]} = \frac{\mu - r}{2\lambda\sigma^2}\frac{1}{S_t}.$$

Exercise 10.2

Consider the expression (10.121) for optimal policy obtained with G-learning

$$\pi(\mathbf{a}_t|\mathbf{y}_t) = \frac{1}{Z_t}\pi_0(\mathbf{a}_t|\mathbf{y}_t)e^{\hat{R}(\mathbf{y}_t,\mathbf{a}_t)+\gamma\mathbb{E}_{t,\mathbf{a}_t}\left[F^{\pi}_{t+1}(\mathbf{y}_{t+1})\right]}$$

where the one-step reward is quadratic as in Eq.(10.91):

$$\hat{R}(\mathbf{y}_t,\mathbf{a}_t) = \mathbf{y}_t^T\mathbf{R}_{yy}\mathbf{y}_t + \mathbf{a}_t^T\mathbf{R}_{aa}\mathbf{a} + \mathbf{a}_t^T\mathbf{R}_{ay}\mathbf{y}_t + \mathbf{a}_t^T\mathbf{R}_a.$$

How does this relation simplify in two cases: (a) when the conditional expectation $\mathbb{E}_{t,\mathbf{a}}\left[F^{\pi}_{t+1}(\mathbf{y}_{t+1})\right]$ does not depend on the action a_t, and (b) when the dynamics are linear in a_t as in Eq.(10.125)?

Solution 10.2

(a) When the conditional expectation $\mathbb{E}_{t,\mathbf{a}}\left[F^{\pi}_{t+1}(\mathbf{y}_{t+1})\right]$ does not depend on the action $\mathbf{a}_t$, this term trivially cancels out with the denominator Z_t. Therefore, the optimal policy in this case is determined by the one-step reward: $\pi(\mathbf{a}_t|\mathbf{y}_t) \sim e^{\hat{R}(\mathbf{y}_t,\mathbf{a}_t)}$.

(b) When the dynamics are linear in action $\mathbf{a}_t$ as in Eq.(10.125), the optimal policy for a multi-step problem has the same Gaussian form as an optimal policy for a single step problem, however this time, parameters $\mathbf{R}_{aa}$, $\mathbf{R}_{ay}$, $\mathbf{R}_{a}$ of the reward function are re-scaled.

Exercise 10.3

Derive relations (10.141).

Solution 10.3

The solution is obtained by a straightforward algebraic manipulation.

Exercise 10.4

Consider G-learning for a time-stationary case, given by Eq.(10.122):

$$G^{\pi}(\mathbf{y},\mathbf{a}) = \hat{R}(\mathbf{y},\mathbf{a}) + \frac{\gamma}{\beta}\sum_{\mathbf{y}'}\rho(\mathbf{y}'|\mathbf{y},\mathbf{a})\log\sum_{\mathbf{a}'}\pi_0(\mathbf{a}'|\mathbf{y}')e^{\beta G^{\pi}(\mathbf{y}',\mathbf{a}')}$$

Show that the high-temperature limit $\beta \to 0$ of this equation reproduces the fixed-policy Bellman equation for $G^{\pi}(\mathbf{y},\mathbf{a})$ where the policy coincides with the prior policy, i.e. $\pi = \pi_0$.

Solution 10.4

Expand the expression in the exponent to the first order in parameter $\beta \to 0$:

$$e^{\beta G^{\pi}(\mathbf{y}',\mathbf{a}')} = 1 + \beta G^{\pi}(\mathbf{y}',\mathbf{a}') + O(\beta^2).$$

Plugging it into Eq.(10.122) and using the first-order Taylor expansion for the logarithm $\log(1+x) = x + O(x^2)$, we obtain

$$G^{\pi}(\mathbf{y}, \mathbf{a}) = \hat{R}(\mathbf{y}, \mathbf{a}) + \gamma \sum_{\mathbf{y}', \mathbf{a}'} \rho(\mathbf{y}'|\mathbf{y}, \mathbf{a}) \pi_0(\mathbf{a}'|\mathbf{y}') G^{\pi}(\mathbf{y}', \mathbf{a}').$$

This is the same as the fixed-policy Bellman equation for the G-function $G^{\pi}(\mathbf{y}, \mathbf{a})$ with $\pi = \pi_0$ and transition probability $\rho(\mathbf{y}', \mathbf{a}'|\mathbf{y}, \mathbf{a}) = \rho(\mathbf{y}'|\mathbf{y}, \mathbf{a}) \pi_0(\mathbf{a}'|\mathbf{y}')$.

Exercise 10.5

Consider policy update equations for G-learning given by Eqs.(10.174):

$$\tilde{\Sigma}_p^{-1} = \Sigma_p^{-1} - 2\beta \mathbf{Q}_t^{(uu)}$$

$$\tilde{\mathbf{u}}_t = \tilde{\Sigma}_p \left(\Sigma_p^{-1} \bar{\mathbf{u}}_t + \beta \mathbf{Q}_t^{(u)} \right)$$

$$\tilde{\mathbf{v}}_t = \tilde{\Sigma}_p \left(\Sigma_p^{-1} \bar{\mathbf{v}}_t + \beta \mathbf{Q}_t^{(ux)} \right)$$

(a) Find the limiting forms of these expressions in the high-temperature limit $\beta \to 0$ and low-temperature limit $\beta \to \infty$.

(b) Assuming that we know the stable point $(\bar{\mathbf{u}}_t, \bar{\mathbf{v}}_t)$ of these iterative equations, as well as the covariance $\tilde{\Sigma}_p$, invert them to find parameters of Q-function in terms of stable point values $\bar{\mathbf{u}}_t, \bar{\mathbf{v}}_t$. Note that only parameters $\mathbf{Q}_t^{(uu)}$, $\mathbf{Q}_t^{(ux)}$, and $\mathbf{Q}_t^{(u)}$ can be recovered. Can you explain why parameters $\mathbf{Q}_t^{(xx)}$ and $\mathbf{Q}_t^{(x)}$ are lost in this procedure? (Note: this problem can be viewed as a prelude to the topic of inverse reinforcement learning covered in the next chapter.)

Solution 10.5

(a) In the high-temperature limit $\beta \to 0$, we find that parameters of the prior policy are not updated:

$$\tilde{\Sigma}_p^{-1} = \Sigma_p^{-1}, \; \tilde{\mathbf{u}}_t = \bar{\mathbf{u}}_t, \; \tilde{\mathbf{v}}_t = \bar{\mathbf{v}}_t.$$

For the low-temperature limit $\beta \to \infty$, the policy becomes a deterministic limit of the Gaussian policy with the following parameters that are independent of the parameters of the prior policy:

$$\tilde{\Sigma}_p = -\frac{1}{2\beta} \left[\mathbf{Q}_t^{(uu)} \right]^{-1} \to 0$$

$$\tilde{\mathbf{u}}_t = -\frac{1}{2} \left[\mathbf{Q}_t^{(uu)} \right]^{-1} \mathbf{Q}_t^{(u)}$$

$$\tilde{\mathbf{v}}_t = -\frac{1}{2} \left[\mathbf{Q}_t^{(uu)} \right]^{-1} \mathbf{Q}_t^{(ux)}$$

(b) If $(\bar{\mathbf{u}}_t, \bar{\mathbf{v}}_t)$ is a fixed point of the update equations and we assume it is known, along with the covariance $\tilde{\Sigma}_p$, we can use it to obtain

$$\mathbf{Q}_t^{(u)} = \frac{1}{\beta}\left(1 - \tilde{\Sigma}_p \Sigma_p^{-1}\right)\bar{\mathbf{u}}_t$$

$$\mathbf{Q}_t^{(ux)} = \frac{1}{\beta}\left(1 - \tilde{\Sigma}_p \Sigma_p^{-1}\right)\bar{\mathbf{v}}_t$$

This implies that if we know the parameters of the policy, we can recover from it the coefficients $\mathbf{Q}_t^{(uu)}, \mathbf{Q}_t^{(ux)}$ and $\mathbf{Q}_t^{(u)}$. However, parameters $\mathbf{Q}_t^{(xx)}$ and $\mathbf{Q}_t^{(x)}$ are not recovered from this procedure. The reason for it is that the policy depends exponentially on $Q(\mathbf{s}_t, \mathbf{a}_t)$ as $\pi(\mathbf{a}_t|\mathbf{s}_t) = \frac{1}{Z(\mathbf{s}_t)} e^{\beta G(\mathbf{s}_t, \mathbf{a}_t)}$ where $Z(\mathbf{s}_t)$ is a normalization factor, therefore additive terms in $G(\mathbf{s}_t, \mathbf{a}_t)$ that depend only on $\mathbf{s}_t$ cancel out between the numerator and denominator.

Exercise 10.6***

The formula for an unconstrained Gaussian integral in n dimensions reads

$$\int e^{-\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{x}^T\mathbf{B}} d^n\mathbf{x} = \sqrt{\frac{(2\pi)^n}{|\mathbf{A}|}} e^{\frac{1}{2}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}}.$$

Show that when a constraint $\sum_{i=1}^n x_i \leq \bar{\mathbf{X}}$ with a parameter $\bar{\mathbf{X}}$ is imposed on the integration variables, a constrained version of this integral reads

$$\int e^{-\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{x}^T\mathbf{B}} \theta\left(\bar{\mathbf{X}} - \sum_{i=1}^n x_i\right) d^n\mathbf{x} = \sqrt{\frac{(2\pi)^n}{|\mathbf{A}|}} e^{\frac{1}{2}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}} \left(1 - N\left(\frac{\mathbf{B}^T\mathbf{A}^{-1}\mathbf{1} - \bar{\mathbf{X}}}{\sqrt{\mathbf{1}^T\mathbf{A}^{-1}\mathbf{1}}}\right)\right)$$

where $N(\cdot)$ is the cumulative normal distribution.

Hint: use the integral representation of the Heaviside step function

$$\theta(x) = \lim_{\varepsilon\to 0} \frac{1}{2\pi i}\int_{-\infty}^{\infty} \frac{e^{izx}}{z - i\varepsilon} dz.$$

Solution 10.6

Writing $\sum_i x_i = \mathbf{x}^T\mathbf{1}$ where $\mathbf{1}$ is a vector of ones, the integral representation of the Heaviside step function reads

$$\theta\left(\bar{\mathbf{X}} - \mathbf{x}^T\mathbf{1}\right) = \lim_{\varepsilon\to 0} \frac{1}{2\pi i}\int_{-\infty}^{\infty} \frac{e^{iz\left(\bar{\mathbf{X}}-\mathbf{x}^T\mathbf{1}\right)}}{z - i\varepsilon} dz.$$

This gives

$$\int e^{-\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{x}^T\mathbf{B}}\theta\left(\bar{\mathbf{X}}-\sum_{i=1}^{n}x_i\right)d^n\mathbf{x}=\int e^{-\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{x}^T\mathbf{B}}d^n\mathbf{x}\frac{1}{2\pi i}\int_{-\infty}^{\infty}\frac{e^{iz\left(\bar{\mathbf{X}}-\mathbf{x}^T\mathbf{1}\right)}}{z-i\varepsilon}dz$$

where we should take the limit $\varepsilon \to 0$ at the end of the calculation. Exchanging the order of integration, the inner integral with respect to $\mathbf{x}$ can be evaluated using the unconstrained formula where we substitute $\mathbf{B} \to \mathbf{B} - iz\mathbf{1}$. The integral becomes (here $a := \mathbf{1}^T\mathbf{A}^{-1}\mathbf{1}$)

$$\sqrt{\frac{(2\pi)^n}{|\mathbf{A}|}}e^{\frac{1}{2}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}}\frac{1}{2\pi i}\int\frac{e^{iz\bar{\mathbf{X}}-\frac{1}{2}z\mathbf{1}^T\mathbf{A}^{-1}\mathbf{1}z-i\mathbf{B}^T\mathbf{A}^{-1}\mathbf{1}}}{z-i\varepsilon}dz$$
$$=\sqrt{\frac{(2\pi)^n}{|\mathbf{A}|}}e^{\frac{1}{2}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}-\frac{\left(\bar{\mathbf{X}}-\mathbf{B}^T\mathbf{A}^{-1}\mathbf{1}\right)^2}{2a}}\frac{1}{2\pi i}\int\frac{e^{-\frac{1}{2}a\left(z+i\frac{\mathbf{B}^T\mathbf{A}^{-1}\mathbf{1}-\bar{\mathbf{X}}}{a}\right)^2}}{z-i\varepsilon}dz$$

The integral over z can be evaluated by changing the variable $z \to z - i\frac{\mathbf{B}^T\mathbf{A}^{-1}\mathbf{1}-\bar{\mathbf{X}}}{a}$, taking the limit $\varepsilon \to 0$, and using the following formula (here $\beta := \frac{\mathbf{B}^T\mathbf{A}^{-1}\mathbf{1}-\bar{\mathbf{X}}}{a}$):

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{e^{-\frac{1}{2}az^2}}{\beta+iz}dz=\frac{\beta}{\pi}\int_{0}^{\infty}\frac{e^{-\frac{1}{2}az^2}}{\beta^2+z^2}dz=\left(1-N\left(\beta\sqrt{a}\right)\right)e^{\frac{a\beta^2}{2}}$$

where in the second equation we used Eq.(3.466) from Gradshteyn and Ryzhik[2] Using this relation, we finally obtain

$$\int e^{-\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{x}^T\mathbf{B}}\theta\left(\bar{\mathbf{X}}-\sum_{i=1}^{n}x_i\right)d^n\mathbf{x}=\sqrt{\frac{(2\pi)^n}{|\mathbf{A}|}}e^{\frac{1}{2}\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}}\left(1-N\left(\frac{\mathbf{B}^T\mathbf{A}^{-1}\mathbf{1}-\bar{\mathbf{X}}}{\sqrt{\mathbf{1}^T\mathbf{A}^{-1}\mathbf{1}}}\right)\right).$$

Note that in the limit $\bar{\mathbf{X}} \to \infty$, we recover the unconstrained integral formula.

[2] I.S. Gradshteyn and I.M. Ryzhik, "Table of Integrals, Series, and Products", Elsevier, 2007.

Chapter 11
Inverse Reinforcement Learning and Imitation Learning

Exercise 11.1

a. Derive Eq.(11.7).
b. Verify that the optimization problem in Eq.(11.10) is convex.

Solution 11.1

To find the optimal policy corresponding to the regularized value function (11.8), we use the method of Lagrange multipliers, and add the normalization constraint to this functional:

$$F^{\pi}(s) = \int \pi(a|s) \left[r(s,a) - \frac{1}{\beta} \log \pi(a|s) \right] da + \lambda \left(\int \pi(a|s) da - 1 \right) \tag{26}$$

where λ is the Lagrange multiplier. Taking the variational derivative with respect to $\pi(a|s)$ and equating it to zero, we obtain

$$r(s,a) - \frac{1}{\beta} \log \pi(a|s) - \frac{1}{\beta} + \lambda = 0$$

Re-arranging this relation, we obtain

$$\pi(a|s) = e^{\beta(r(s,a)+\lambda)-1}$$

The value of the Lagrange multiplier λ can now be found from the normalization constraint $\int \pi(a|s)da = 1$. This produces the final result

$$\pi(a|s) = \frac{1}{Z_{\beta}(s)} e^{\beta r(s,a)}, \quad Z_{\beta}(s) \equiv \int e^{\beta r(s,a)} da$$

Consider the objective function defined in Eq.(11.10)

$$G(\lambda) = \log Z_\lambda - \lambda \bar{r}(s)$$

Taking the second derivative with respect to λ, we obtain

$$\frac{d^2G}{d\lambda^2} = \frac{Z''(\lambda)}{Z(\lambda)} - \left(\frac{Z'(\lambda)}{Z(\lambda)}\right)^2 = \int r^2(s,a)\pi(a|s)da - \left(\int r(s,a)\pi(a|s)da\right)^2 = \mathrm{Var}(r)$$

As variance $\mathrm{Var}(r)$ is always non-negative, it means that the optimization problem in Eq.(11.10) is convex.

Exercise 11.2

Consider the policy optimization problem with one-dimensional state- and action-spaces and the following parameterization of the one-step reward

$$r(s,a) = -\log\left(1 + e^{-\theta\Psi(s,a)}\right)$$

where θ is a vector of K parameters, and $\Psi(s,a)$ is a vector of K basis functions. Verify that this is a concave function of a as long as basis functions $\Psi(s,a)$ are linear in a.

Solution 11.2

If the basis functions $\Psi(s,a)$ are linear in a while having an arbitrary dependence on s, the second derivative of the reward is

$$\frac{\partial^2 r}{\partial a^2} = -\frac{\left(\theta\frac{\partial\Psi}{\partial a}\right)^2}{\left(1 + e^{-\theta\Psi(s,a)}\right)^2} \leq 0$$

Exercise 11.3

Verify that variational maximization with respect to classifier $D(s,a)$ in Eq.(11.75) reproduces the Jensen–Shannon divergence (11.72).

Solution 11.3

The GAIL loss function (11.75) can be written as

$$\Psi^\star_{GA} = \max_{D\in[0,1]^{S\times\mathcal{A}}} \int \rho_E(s,a)\log D(s,a)dsda + \int \rho_\pi(s,a)\log\left(1 - D(s,a)\right)dsda$$

Taking the variational derivative of this expression with respect to $D(s,a)$, we obtain

$$\frac{\rho_E(s,a)}{D(s,a)} - \frac{\rho_\pi(s,a)}{1-D(s,a)} = 0$$

Re-arranging terms in this expression, we obtain Eq.(11.76):

$$D(s,a) = \frac{\rho_E(s,a)}{\rho_\pi(s,a) + \rho_E(s,a)}$$

Substituting this expression into the first expression for $\Psi^\star_{GA}$, we obtain

$$\Psi^\star_{GA} = \int \rho_E(s,a) \log \frac{\rho_E(s,a)}{\rho_\pi(s,a) + \rho_E(s,a)} dsda + \int \rho_\pi(s,a) \log \frac{\rho_\pi(s,a)}{\rho_\pi(s,a) + \rho_E(s,a)} dsda$$

Dividing the numerator and denominator in logarithms in both terms by two and re-arranging terms, we obtain

$$\Psi^\star_{GA} = DK_{KL}\left(\rho_\pi || \frac{1}{2}\left(\rho_\pi + \rho_E\right)\right) + DK_{KL}\left(\rho_E || \frac{1}{2}\left(\rho_\pi + \rho_E\right)\right) - \log 4 = D_{JS}\left(\rho_\pi, \rho_E\right) - \log 4$$

where $D_{JS}\left(\rho_\pi, \rho_E\right)$ is the Jensen–Shannon distance (11.72).

Exercise 11.4

Using the definition (11.71) of the convex conjugate function $\psi^\star$ for a differentiable convex function $\psi(x)$ of a scalar variable x, show that (i) $\psi^\star$ is convex, and (ii) $\psi^{\star\star} = \psi$.

Solution 11.4

For a convex differentiable function $\psi(y)$ of a scalar variable y, the convex conjugate

$$\psi^\star(x) = \sup_y \; xy - \psi(y)$$

coincides with the Legendre transform. Taking the derivative of this expression with respect to y and equating it to zero, we have

$$\psi'(y) = x \;\Rightarrow y_x = g(x)$$

where $g = [\psi']^{-1}$ is an inverse function of ψ', so that we have

$$\psi'\left(g(x)\right) \equiv \psi'(y_x) = x$$

Differentiating both sides of this equation with respect to x, we obtain

$$g'(x) = \frac{1}{\psi''(g(x))}$$

We can therefore write the convex conjugate as a composite function of x:

$$\psi^\star(x) = \; xy_x - \psi(y_x)$$

where $y_x = g(x)$. Differentiating this expression, we have

$$\frac{d\psi^\star(x)}{dx} = y_x + x\frac{dy_x}{dx} - \frac{\psi(y_x)}{dy_x}\frac{dy_x}{dx} = y_x + x\frac{dy_x}{dx} - x\frac{dy_x}{dx} = y_x$$

Differentiating the second time, we have

$$\frac{d^2\psi^\star(x)}{dx^2} = \frac{dy_x}{dx} = \frac{1}{\psi''(g(x))} \geq 0$$

Therefore, the convex conjugate (Legendre transform) of a convex differentiable function $\psi(y)$ is convex.

To show that $\psi^{\star\star} = \psi$, first note that the original function can be written in terms of its transform:

$$\psi(y_x) = xy_x - \psi^\star(x)$$

Using this, we obtain

$$\psi^{\star\star}(x) = \left(xp_x - \psi^\star(p_x)\right)_{\frac{d\psi^\star(p)}{dp}\big|_{p=p_x}=x} = \left(xp_x - \psi^\star(p_x)\right)_{g(p_x)=x} = g(p_x)p_x - \psi^\star(p_x)$$
$$= \psi\left(g(p_x)\right) = \psi(x)$$

where we replaced x by $g(p_x)$ in the third step, and replaced $g(p_x)$ by x in the last step.

Exercise 11.5

Show that the choice $f(x) = x\log x - (x+1)\log\frac{x+1}{2}$ in the definition of the f-divergence (11.73) gives rise to the Jensen-Shannon divergence (11.72) of distributions P and Q.

Solution 11.5

Using $f(x) = x\log x - (x+1)\log\frac{x+1}{2}$ in Eq.(11.73), we obtain

$$
\begin{aligned}
&D_f\,(P||\,Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx \\
&= \int p(x) \log \frac{p(x)}{p(x)+q(x)} dx + \int q(x) \log \frac{q(x)}{p(x)+q(x)} dx \\
&= D_{JS}(p,q) - \log 4
\end{aligned}
$$

where $D_{JS}(p,q)$ is the Jensen–Shannon distance between p and q.

Exercise 11.6

In the example of learning a straight line from Sect. 5.6, compute the KL divergence $D_{KL}\,(P_\theta||P_E)$, $D_{KL}\,(P_E||P_\theta)$, and the JS divergence $D_{JS}\,(P_\theta, P_E)$.

Solution 11.6

Let P_E be a distribution corresponding to a vertical line with $x = 0$, and P_θ be a distribution corresponding to a vertical line at $x = \theta$. They can be written as

$$P_E(x,z) = \delta(x)z, \quad P_\theta = \delta(x-\theta)z$$

As they have non-overlapping supports, we have, for $\theta \neq 0$, $D_{KL}\,(P_\theta||P_E) = D_{KL}\,(P_E||P_\theta) = 0$, while for $\theta = 0$, both KL divergences are zeros. On the other hand, the JS distance is $D_{JS}\,(P_\theta, P_E) = 2\log 2$ for $\theta \neq 0$, while it vanishes for $\theta = 0$.