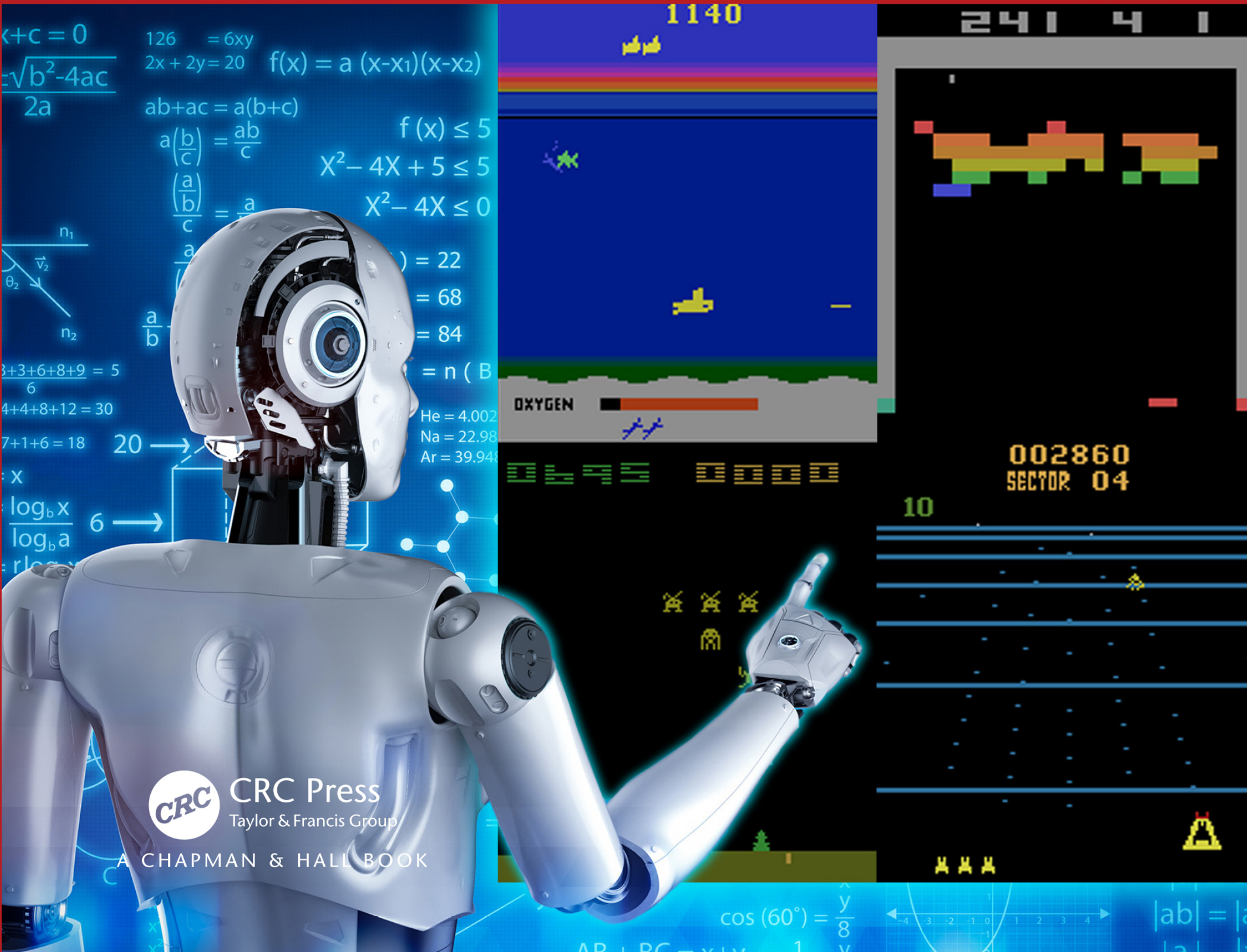
CHAPMAN & HALL/CRC
MACHINE LEARNING & PATTERN RECOGNITION

Machine Learning, Animated

Mark Liu



CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Machine Learning, Animated

The release of ChatGPT has kicked off an arms race in Machine Learning (ML), however, ML has also been described as a black box and very hard to understand. *Machine Learning, Animated* eases you into basic ML concepts and summarizes the learning process in three words: *initialize*, *adjust* and *repeat*. This is illustrated step by step with animation to show how machines learn: from initial parameter values to adjusting each step, to the final converged parameters and predictions.

This book teaches readers to create their own neural networks with dense and convolutional layers, and use them to make binary and multi-category classifications. Readers will learn how to build deep learning game strategies and combine this with reinforcement learning, witnessing AI achieve super-human performance in Atari games such as Breakout, Space Invaders, Seaquest and Beam Rider.

Written in a clear and concise style, illustrated with animations and images, this book is particularly appealing to readers with no background in computer science, mathematics or statistics.

Dr. Mark Liu is a tenured finance professor and founding director of the Master of Science in Finance program at the University of Kentucky, where he teaches Python Predictive Analytics to graduate students. He has more than 20 years of coding experience and is the author of *Make Python Talk* (No Starch Press, 2021).

Machine Learning, Animated

Mark Liu

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

First edition published 2024
by CRC Press
6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

CRC Press is an imprint of Taylor & Francis Group, LLC

Library of Congress Cataloging-in-Publication Data

Names: Liu, Mark (Mark H.), author.
Title: Machine learning, animated / Mark Liu.
Description: First edition. | Boca Raton, FL : CRC Press, 2024. | Includes bibliographical references and index.
Identifiers: LCCN 2023016272 (print) | LCCN 2023016273 (ebook) | ISBN 9781032462141 (hbk) | ISBN 9781032462134 (pbk) | ISBN 9781003441281 (ebk) | ISBN 9781003380580 (eBook+)
Subjects: LCSH: Neural networks (Computer science)--Problems, exercises, etc. | Computer animation--Problems, exercises, etc. | Machine learning--Problems, exercises, etc. | Video games--Programming--Problems, exercises, etc.
Classification: LCC QA76.87 .L576 2024 (print) | LCC QA76.87 (ebook) | DDC 006.3/107--dc23/eng/20230905
LC record available at https://lccn.loc.gov/2023016272
LC ebook record available at https://lccn.loc.gov/2023016273

ISBN: 978-1-032-46214-1 (hbk)
ISBN: 978-1-032-46213-4 (pbk)
ISBN: 978-1-003-44128-1 (ebk)
ISBN: 978-1-003-38058-0 (eBook+)

DOI: 10.1201/b23383

Typeset in Latin Modern font
by KnowledgeWorks Global Ltd.

Publisher's note: This book has been prepared from camera-ready copy provided by the authors.

Access the Support Material: https://www.routledge.com/Machine-Learning-Animated/Liu/p/book/9781032462141 and https://github.com/markhliu/MLA

To Ivey and Andrew.

Contents

Preface

Artificial Intelligence, deep learning, machine learning — whatever you're doing if you don't understand it — learn it. Because otherwise you're going to be a dinosaur within 3 years.

–Mark Cuban, 2021

Machine learning (ML) is redefining the way we live nowadays: it's integrated into an increasing number of products and services in the economy, from recommender systems to language translations, from voice assistants, medical imaging, to self-driving cars... ML, especially deep learning, has made great strides in the last couple of decades, largely due to the advancements in computing power (such as graphics processing unit (GPU) training and distributed computing) and the exploding amount of data available to train deep neural networks.

The recent release of ChatGPT by OpenAI has upped the ante in the game, forcing Google and other competitors to release large language models of their own [5]. Different organizations and institutions have realized that an arms race in the field of ML and artificial intelligence (AI) is on. Everyone in every profession must adapt or face the risk of becoming a dinosaur and getting left behind. A case in point is a recent announcement by the Chartered Financial Analyst (CFA) Institute on March 17, 2023, to add ML and AI in CFA exams to prepare candidates in these fields [26]. The change comes after the CFA Institute "speaking with employers who were bemoaning that while what's in the program was very practical, when they hire new charterholders, they're not quite job ready." [11]

The need to incorporate ML into the college curriculum was clear long before the release of ChatGPT. In the Master of Science in Finance (MSF) program at the University of Kentucky, we have kept a close eye on the market demand for skill sets in these fields so as to keep our graduates competitive on the job market. I created and taught a Python Predictive Analytics course for our finance master students, involving state-of-the-art ML models such as deep neural networks, random forests, gradient boosting machines, and so on. While students are generally amazed by what ML can accomplish, they complain that the learning process in ML is like a black box and hard to understand. To help explain the inner workings of ML algorithms, I have simplified the learning process into three words: initialize, adjust, and repeat.

- Step 1: A machine learning model assigns values to the model parameters (initialize).
- Step 2: It makes predictions based on the current parameters and compares predictions with the actual values; it changes the parameters so that the predictions in the next iteration will move closer to the actual values (adjust).
- Step 3: It repeats step 2 until the parameters converge (repeat).

The teaching experience has sowed the seed for this book. In the early part of the book, I'll discuss the building blocks of ML such as loss functions, activation functions, the gradient descent optimization algorithm, the learning rate... Better yet, the book will use animations to show step by step how machines learn: the initial parameter values, the adjustment in each step, and the final converged parameters and predictions. I attempt to fill the void in the market for an ML book for college students and young professionals with no background in computer science, mathematics, or statistics. As such, the book takes a practical rather than technical approach to ML. The book provides an intuitive explanation of concepts such as deep learning, Q-learning, or the policy-gradient algorithm. You'll learn how to implement these algorithms by following the examples and how to apply them to your own field, be that business, biology, medicine, or something else entirely. While most models are built by using the TensorFlow Keras API, you also learn to create ML models from scratch on your own, without resorting to any API. Along the way, you'll know how ML models are constructed, how the parameter values are initialized and then gradually adjusted during the training process, how parameters converge, and how the trained models make accurate predictions.

This book is divided into six parts. Part I discusses how to install Python and how to create animations with Python libraries. Part II introduces you to ML basics such as the gradient descent optimization algorithm, the learning rate, loss functions, and activation functions. Part III covers binary and multi-category classifications and introduces you to neural networks. In Part IV, we build deep learning game strategies in OpenAI Gym games as well as in multi-player games such as Tic Tac Toe and Connect Four. Part V introduces you to the basics of reinforcement learning. In Part VI, we combine deep learning with reinforcement learning to create deep reinforcement learning game strategies, so you can create a double deep Q-network to train all Atari games (Breakout, Space Invaders, Seaquest, and so on).

Here's an overview of the book:

Part I: Installing Python and Learning Animations

Chapter 1: Installing Anaconda and Jupyter Notebook

This chapter guides you through installing the Python software based on your operating system, whether that's Windows, Mac, or Linux. You'll create a virtual environment just for projects in this book and install Jupyter Notebook as the integrated development environment (IDE). You'll set up a directory to manage files in this book.

Chapter 2: Creating Animations

You learn to create graphics and animations in Python. This prepares you to create graphic representations and animations of the intermediate stages of the ML process later in this book.

Part II: Machine Learning Basics

Chapter 3: Machine Learning: An Overview

You'll learn what ML is and how it's different from the traditional algorithms in artificial intelligence (AI). We'll discuss three types of ML: supervised learning, unsupervised learning, and reinforcement learning. The three types also differ in terms of data, methodologies, and applications.

Chapter 4: Gradient Descent – Where Magic Happens

You'll use animations to show step by step how the parameter values in ML models change based on the gradient descent algorithm so that the ML models make predictions with the lowest forecasting error possible. The forecasting errors are measured by a loss function. Training an ML model is finding parameter values that minimize the loss function. The optimization process is achieved through gradient descent or some variant of it. You'll also know what the learning rate is and how it affects the training process.

Chapter 5: Introduction to Neural Networks

This chapter discusses how neural networks learn from the data and make predictions. You learn to construct a simple neural network from scratch to learn the relation between ten pairs of input and output variables. You use the three steps that we have outlined in ML: initialize, adjust, and repeat. You'll animate the learning process by extracting the parameter values and predictions in each step of the training process in this simple neural network.

Chapter 6: Activation Functions

You'll use the rectified linear unit (ReLU) activation function in a neural network to approximate a nonlinear relationship. The Sigmoid activation function squashes a number to the range between 0 and 1 so that it can be interpreted as the probability of an outcome. The Softmax activation function squeezes a group of numbers into the range [0, 1] so they can be interpreted as the probability distribution of multiple outcomes.

Part III: Binary and Multi-Category Classifications

Chapter 7: Binary Classifications

Binary classification is an ML algorithm to classify samples into one of two categories. In this chapter, you learn binary classifications by classifying images into horses and deer using a neural network. You create an animation to demonstrate how the model weights and the predicted probabilities change in different stages of training.

Chapter 8: Convolutional Neural Networks

A convolutional layer treats an image as a two-dimensional object and finds patterns on the image. It then associates these patterns with the image labels. This significantly improves the predictive power of the model. In this chapter, you learn the basic concepts related to a convolutional layer such as the number of filters, kernel size, zero-padding, strides... Better yet, you learn to create animations to show step by step how to apply a filter on an image and how the convolution operations are conducted.

Chapter 9: Multi-Category Image Classifications

When the target label is a multi-category variable with more than two possible values, we call the machine learning algorithm a multi-category classification problem. In this chapter, you learn to classify images in CIFAR-10 into one of the ten labels using a deep neural network with augmentations and convolutional layers.

Part IV: Developing Deep Learning Game Strategies

Chapter 10: Deep Learning Game Strategies

You learn to use deep learning to train intelligent game strategies in the Frozen Lake game in OpenAI Gym. You first generate game data for training purposes. You then create a deep neural network to train game strategies. The agent picks the action with the highest probability of winning based on the trained model.

Chapter 11: Apply Deep Learning to the Cart Pole Game

You learn to train deep learning game strategies to play the Cart Pole game in OpenAI Gym. You learn to creatively redefine what's considered "winning" in a game so that there are roughly evenly distributed numbers of winning and losing games in the simulated data. You feed the re-labelled data into a deep neural network to train the model. The trained model wins the Cart Pole game 100% of the time.

Chapter 12: Deep Learning in Multi-Player Games

You learn to create a game environment for Tic Tac Toe. You then apply deep learning to Tic Tac Toe with the aim of developing intelligent game strategies. We'll also animate the decision-making process of the agent so we can look under the hood at how deep learning game strategies work.

Chapter 13: Deep Learning in Connect Four

You create a game environment for Connect Four and use simulated games to train a deep neural network. At each step of the game, the deep learning agent iterates through all possible next moves and selects the move with the highest probability of winning. You animate the decision-making process by showing all possible next moves and the associated probabilities of winning in each step of the game.

Part V: Reinforcement Learning

Chapter 14: Introduction to Reinforcement Learning

In reinforcement learning, an agent interacts with an environment through trial and error. The agent learns to achieve the optimal outcome by receiving feedback from the environment in the form of rewards and punishments. In this chapter, you'll train the Q-table in the Frozen Lake game. You create an animation to demonstrate how tabular Q-learning works. In each state, you put the game board on the left and the Q-table on the right. You highlight the row corresponding to the state and compare the Q-values under the four actions. The best action is highlighted in red. The animation repeats this process until the game ends.

Chapter 15: Q-Learning with Continuous States

Tabular Q-learning can solve problems in which both the number of actions and the number of states are finite. In the Mountain Car game, the state variable is continuous so the number of states is infinite. You use a finite number of discrete values to represent the state space and train the Q-table for the game effectively.

Chapter 16: Solving Real-World Problems with Machine Learning

You learn to solve an Amazon Delivery Route problem by using tabular Q-learning. You first find the shortest route between any two households in town by training a Q-table. You need to deliver eight packages a day. You consider all permutations and calculate the total distance traveled with each permutation. You select the one with the shortest total distance.

Part VI: Deep Reinforcement Learning

Chapter 17: Deep Q-Learning

You learn to use a neural network to approximate a Q-table. A deep Q-learning agent chooses an action in a given state by feeding the current game state into a deep Q-network. The network returns Q-values associated with different actions. The agent selects the action with the highest Q-value. You learn to successfully apply deep Q-learning to the Cart Pole game.

Chapter 18: Policy-Based Deep Reinforcement Learning

You learn policy-based reinforcement learning in this chapter: instead of estimating the value functions associated with different actions, you directly train a policy that tells the agent which action to take in a given state. You use the policy gradient method to play the Atari Pong game, earning a perfect score of 21 to 0.

Chapter 19: The Policy Gradient Method in Breakout

You generalize the policy gradient method you learned in Chapter 18 to another Atari game: Breakout. You animate how the agent learns to dig a tunnel on the side of the wall to send the ball to the back of the wall to score more efficiently.

Chapter 20: Double Deep Q-Learning

Q-learning has a well-known problem of overestimating Q-values. To overcome this problem, you learn to use the double Q-learning method in which one deep Q-network is used for training (the training network) and another for prediction (the target network). You animate how the trained agent in Breakout sends the ball to the back of the wall multiple times.

Chapter 21: Space Invaders with Double Deep Q-Learning

You tweak the Q-network you used in Chapter 20 and apply it to another Atari game, Space Invaders. Even though the agent does not know the rules of the Space Invaders game, it can eliminate all invaders on the screen, just by learning from the rewards via repeated interactions with the game environment.

Chapter 22: Scaling Up Double Deep Q-Learning

You scale up the double deep Q-network to play any Atari game. A model with the same network architecture, same hyperparameters, and same training procedure is created that can be applied to any Atari game. You apply the model on two new Atari games: Seaquest and Beam Rider. With these skills, you are ready to train and test any Atari game by using the same model.

All Python programs, along with answers to some end-of-the-chapter questions, are provided in the GitHub repository https://github.com/markhliu/MLA.

Acknowledgments

I have many people to thank for along the journey of making this book a reality. A portion of this book was developed while I was teaching Python Predictive Analytics to Master of Science in Finance students at the University of Kentucky in the past few years. I'd like to thank all MS Finance students for keeping me motivated to find novel ways to explain how Machine Learning works. I'd also like to thank Randi Slack and Solomon Pace-McCarrick at CRC Press for guiding me through the editorial process. Thanks to the production team of Michele Dimont and Riya Bhattacharya for helping me cross the finish line. Finally, I'd like to thank my wife Ivey Zhang and my son Andrew Liu for being so supportive in this journey.

I

Installing Python and Learning Animations

Installing Anaconda and Jupyter Notebook

The mechanic, who wishes to do his work well, must first sharpen his tools.
–*Ancient Chinese Proverb*

IN THIS CHAPTER, you'll first learn why Python is a great tool for machine learning (ML). After that, I'll guide you through installing the Python software you need to start running Python programs for this book. There are different ways of installing Python and managing packages on your computer. We'll be using Anaconda as our Python distribution and development environment for this book. I'll guide you through the installation process based on your operating system, whether that's Windows, Mac, or Linux. I'll also discuss the advantages of choosing Anaconda over other ways of installing Python.

You'll learn to create a virtual environment just for projects in this book. After that, you'll install Jupyter Notebook as your integrated development environment (IDE) and start coding in it. At the end of the chapter, you'll set up a directory to manage files in this book.

New Skills in This Chapter

- Setting up Python on your computer by installing Anaconda
- Creating a virtual environment for projects in this book
- Starting coding in Python by using Jupyter Notebook
- Setting up a file system for this book

DOI: 10.1201/b23383-1

1.1 WHY PYTHON FOR MACHINE LEARNING?

In this section, I'll briefly discuss why Python is popular as a programming language in general and why it's the preferred language for ML nowadays in particular.

1.1.1 The Rise of Python

Python has been the world's most popular programming language since late 2018, according to The Economist [25]. Once you start to code in Python, it's easy to see why. Python is a user-friendly, open-source, and cross-platform programming language. Python code is relatively close to plain English, so with only a little experience, you can often guess what a block of code is trying to accomplish.

Python is open source, meaning not only that the software is free to use for everyone but also that other users can create and alter libraries. In fact, Python has a vast ecosystem from which you can get resources and help from members in the community. Python programmers can share their code with one another, so instead of building everything from scratch, you can import modules designed by others, as well as share your modules with others in the Python community.

Python is a cross-platform programming language, meaning you can code in Python whether you use Windows, Mac, or Linux. However, the installation of software and libraries can be slightly different depending on your operating system. I'll show you how to install various libraries in your operating system. Once these are properly installed, Python code works the same in different operating systems.

Python is a high-level interpreted language. It allows users to abstract away from details of the computer such as data type, memory management, and pointers. As a result, the execution of Python code is slower than lower-level compiled languages such as C, C++, or Java. However, nowadays, with the advancements in computer hardware, you'll hardly notice the difference.

Ways to Learn Python Basics

This book assumes you have some basic understanding of the Python programming language. If not, a great place to start is the free online Python tutorial provided by W3Schools. Go to https://www.w3schools.com/python/ and follow the examples and exercises in the tutorial. They also provide a "Try it Yourself" editor and online compiler for you to run the Python code without installing Python on your computer. Alternatively, you can pick up a Python basics book and go over it. The Michigan State University's Professor Charles Severance has a book called Python for Everyone [22], and there is a printed version as well as a free online version https://www.py4e.com/.

1.1.2 Python for Machine Learning

All the ML algorithms in this book are in Python. We choose Python for several reasons.

First, as we mentioned above, Python is an expressive high-level language for general application development. Python's syntax structure is easy to follow. It is easy for ML enthusiasts to understand and process what the code is trying to accomplish. As a result, Python users can focus on solving ML problems without spending too much time and effort on the coding part. The simplicity of Python also allows programmers to collaborate with each other easily because understanding each other's code is not as difficult.

Second, you can easily get support from the Python ML community. There is a large online community with various groups and forums where programmers post their errors or other types of problems and help each other out. You can get resources and help from members in the Python ML community. If you encounter issues for the ML libraries in this book, you can search the forums for the Python packages you are using, or go to sites such as Stack Overflow to look for answers. In the rare case that you couldn't find an answer, feel free to reach out to me for help.

Third, Python is one of the most popular languages for ML. This is mainly because the Python ML ecosystem provides a wide collection of libraries that enable users to create ML models easily. In particular, you'll use extensively the following three libraries in this book: NumPy, TensorFlow, and Keras. Below, I'll briefly discuss what these libraries can accomplish. In a later chapter, we'll go into more details when we use these libraries to create various ML models.

NumPy stands for numerical Python. The NumPy library provides efficient data structures to represent numerical vectors and matrices, which allows Python to handle high-dimensional array objects and perform efficient mathematical operations. It is the bedrock of many of Python's numerical computing libraries such as pandas, matplotlib, and TensorFlow. For example, as you'll see later in this book, pictures are represented as three-dimensional NumPy arrays in Python: the first dimension is the width of the picture, the second the height, and the third the color channels. Even though NumPy is a Python library, most of the code in it is written in C or C++, and this allows for faster execution of the code.

Keras is a deep learning application programming interface (API) developed by Google. It makes the implementations of deep neural networks easy. Specifically, it provides the building blocks for developing state-of-the-art deep neural networks. It provides a convenient way for you to specify neural networks. You can easily add or remove a layer of neurons from the network as you tune your model. When you add a new layer of neurons, you can specify how many neurons to include in the layer, what activation function to use, and so on. You can also choose different types of layers of neurons such as dense layers or convolutional layers. Later chapters cover more details.

TensorFlow is an ML library developed by Google. It uses data flow and differentiable programming to perform different tasks. It allows users to pre-process data. The library takes input data as high-dimensional arrays known as tensors. The TensorFlow library allows you to perform mathematical operations such as matrix multiplications, convolutional operations, and so on. For example, we'll use TensorFlow to calculate the gradients of a function at the current parameter values so that we know how much to adjust the parameters based on the rule of gradient descent. We'll discuss how to implement all these (along with the terminologies I mentioned here) in later chapters.

1.2 INSTALLING ANACONDA

There are different ways of running Python programs and managing packages on your computer. This book uses Anaconda. Anaconda is an open-source Python distribution, package, and environment manager. It is user-friendly and provides for the easy installation of many useful Python libraries and packages that otherwise can be quite a pain (or downright impossible) to compile and install yourself. Specifically, Anaconda allows users to conda install packages in addition to pip installing packages (if you don't know the difference between the two, don't panic; I'll explain later in this chapter). As a matter of fact, many packages and libraries used in this book will be conda installed. Some of them cannot be pip installed. Therefore, if you don't install Anaconda on your computer, many projects in this book won't work. I urge you to follow the instructions in this chapter and install Anaconda so that you can enjoy all projects in this book.

Below, I'll guide you through the process of installing Anaconda on your computer based on your operating system.

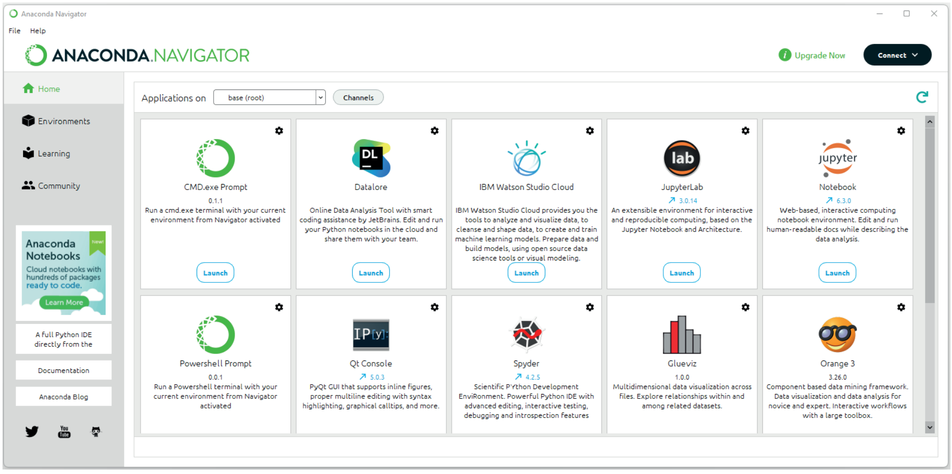
1.2.1 Installing Anaconda in Windows

To install Anaconda in Windows, go to https://www.anaconda.com/products/individual/. Scroll down to the section Anaconda Installers. Download the latest version of Python 3 graphical installer for Windows. Make sure you download the appropriate 32- or 64-bit package for your machine. Run the installer and follow the instructions all the way through.

To check if Anaconda is properly installed on your computer, search for the Anaconda Navigator app on your computer. If you can open the app, Anaconda is successfully installed on your computer. The Anaconda Navigator app looks like what you see in Figure 1.1.

1.2.2 Installing Anaconda in macOS

To install Anaconda in macOS, go to https://www.anaconda.com/products/individual/. Scroll down to the section Anaconda Installers. Download the latest



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

Datalore
Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

JupyterLab
↗ 3.0.14
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
↗ 6.3.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
↗ 5.0.3
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
↗ 4.2.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Glueviz
1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.26.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Figure 1.1 The Anaconda Navigator app

version of Python 3 graphical installer for Mac. There is a command line installer option as well. I recommend using the graphical installer instead of the command line installer, especially for beginners, to avoid mistakes. Run the installer and follow the instructions all the way through.

To check if Anaconda is properly installed on your computer, search for the Anaconda Navigator app on your computer. If you can open the app, Anaconda is successfully installed on your computer. The Anaconda Navigator app looks like what you see in Figure 1.1.

1.2.3 Installing Anaconda in Linux

The installation of Anaconda in Linux involves more steps than for other operating systems: there is no graphical installer for Linux. First, go to https://www.anaconda.com/products/individual/, scroll down, and find the latest Linux version. Choose the appropriate x86 or Power8 and Power9 package. Click and download the latest installer bash script. For example, the installer bash script during my installation was https://repo.anaconda.com/archive/Anaconda3-2022.05-Linux-x86_64.sh. This link will change over time, but we'll use this version as our example.

Open a terminal on your computer. By default, the installer bash script is downloaded and saved to the Downloads folder on your computer. You should then install Anaconda by issuing the following command in the terminal (use the path for your bash script if it is different):

```
bash ~/Downloads/Anaconda3-2022.05-Linux-x86_64.sh
```

After pressing the ENTER key on your keyboard, you'll be prompted to review and approve the license agreement. The last question in the installation process is this:

```
installation finished. Do you wish the installer to prepend the
Anaconda3 install location to PATH in your /home/mark/.bashrc ?
[yes|no] [no] >>>
```

You should type yes and press ENTER in order to use the conda command to open Anaconda in a terminal.

Now you need to activate the installation by executing this command:

```
source ~/.bashrc
```

To open Anaconda navigator, enter the following command in the terminal:

```
anaconda-navigator
```

You should see the Anaconda navigator on your machine, similar to Figure 1.1.

1.2.4 Difference between Conda-install and Pip-install

Many people think pip install and conda install are the same, but they're not. Pip is the Python packaging authority's recommended tool for installing packages from the Python packaging index. Pip can be used to install only Python software. In contrast, Conda is a cross-platform package and environment manager that installs not only Python software but also packages in C or C++ libraries, R packages, or other software. One case in point is that the portaudio package is a C package, which cannot be installed using Pip, but can be installed using Conda. In order to make Python connect to your computer microphone, you need the portaudio package. Installing Anaconda is the only way to make speech recognition work in Python. See my book Make Python Talk for details if you are interested [14].

1.3 VIRTUAL ENVIRONMENT FOR THIS BOOK

As you build more and more projects in Python, you'll install many libraries. Some libraries may interfere with other libraries, and different projects may use different versions of the same library. To avoid problems of clashing libraries, I recommend you build a virtual environment for each project. A virtual environment is a way to isolate projects from each other.

1.3.1 Create the Virtual Environment MLA

We'll create a virtual environment to contain all projects in this book. Let's name the virtual environment MLA, as in Machine Learning, Animated.

How to Open the Anaconda Prompt in Windows

Don't confuse the Anaconda prompt in the Windows operating system with the command prompt. To open Anaconda prompt in Windows, search for the Anaconda prompt app and click on the app to open it.

To create a virtual environment, open the Anaconda prompt (in Windows) or a terminal (in Mac or Linux). Enter the following command:

```
conda create -n MLA python==3.9.12
```

After pressing ENTER, follow the instructions onscreen and press y when the prompt asks you y/n. Once you have created the virtual environment on your machine, you need to activate it.

1.3.2 Activate the Virtual Environment

To activate the virtual environment MLA, open the Anaconda prompt (in Windows) or a terminal (in Mac or Linux). Execute the following command:

```
conda activate MLA
```

In Windows, you'll see the following on your Anaconda prompt:

```
(MLA) C:\>
```

You can see the (MLA) prompt, which indicates that the command line is now in the virtual environment MLA that you've just created.

On a Mac, you should see something similar to the following in the terminal (the username will be different):

```
(MLA) Macs-MacBook-Pro:~ macuser$
```

In Linux, you should see something similar to this on your terminal (the username will be different):

```
(MLA) mark@mark-OptiPlex-9020:~$
```

1.3.3 De-activate the Virtual Environment

When the command line is in the virtual environment MLA, there are two ways you can deactivate it.

The first way is to issue the following command:

```
conda deactivate
```

Note that you don't need to put the environment name MLA in the command. Conda automatically goes to the base environment after deactivation. In Windows, you'll see the following on your Anaconda prompt:

```
(base) C:\>
```

You can see the (base) prompt, which indicates that the command line is in the default Python environment.

On a Mac, you should see something similar to the following in the terminal:

```
(base) Macs-MacBook-Pro:~ macuser$
```

In Linux, you should see something similar to this in your terminal:

```
(base) mark@mark-OptiPlex-9020:~$
```

The second way is to issue the following command:

```
conda activate base
```

The above command activates the base environment, which is the default Python environment. This effectively deactivates the virtual environment MLA the command line was in before.

1.4 SET UP JUPYTER NOTEBOOK IN THE VIRTUAL ENVIRONMENT

Now we need to set up Jupyter Notebook in the newly created virtual environment on your computer. First, activate the virtual environment MLA by running the following line of code in the Anaconda prompt (in Windows) or a terminal (in Mac or Linux):

```
conda activate MLA
```

To install Jupyter Notebook in the virtual environment, run the command:

```
conda install notebook==6.4.8
```

To launch Jupyter Notebook, execute the following command in the same terminal with the virtual environment activated:

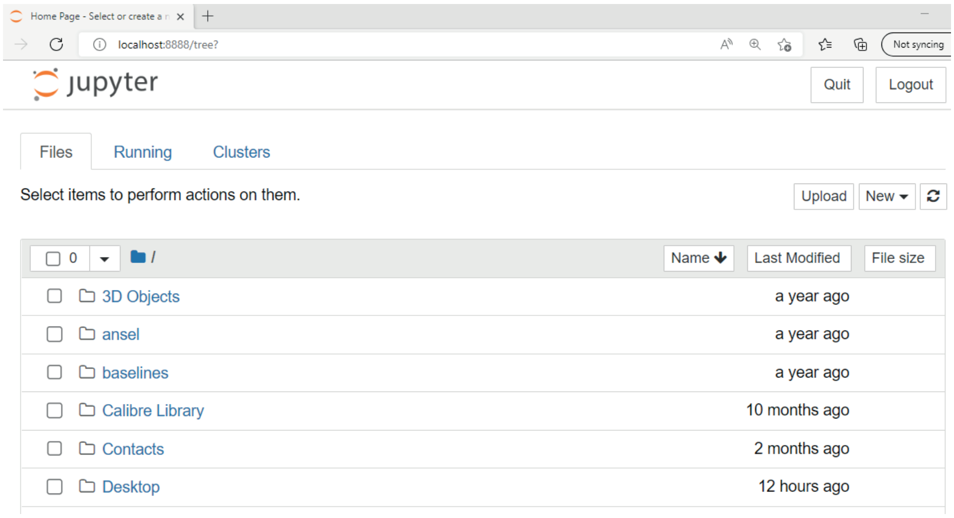
```
jupyter notebook
```

Jupyter Notebook should open in your default browser. If not, open a browser and put http://localhost:8888 in the address bar, and you should open the Jupyter Notebook.

The Jupyter Notebook app is shown in Figure 1.2.

1.4.1 Write Python in Jupyter Notebook

To get you up and running, I'll show you how to run Python programs in Jupyter Notebook.



Select items to perform actions on them.

	Name	Last Modified
☐	3D Objects	a year ago
☐	ansel	a year ago
☐	baselines	a year ago
☐	Calibre Library	10 months ago
☐	Contacts	2 months ago
☐	Desktop	12 hours ago

Figure 1.2 The Jupyter Notebook app

How to Download an .ipynb or .py file from GitHub

To download an individual file from GitHub with .ipynb or .py extension, first go to the file's url using your browser. Click on the Raw button and you'll be redirected to a new url that shows the raw code of file. Press CTRL and S simultaneously on your keyboard and a dialog box pops up. Select All Files (*.*) from the Save as type drop-down menu and save the file on your computer.

Download the template file *tmp.ipynb* from the book's GitHub repository https://github.com/markhliu/MLA/blob/main/files/tmp.ipynb. Save it in the folder /mla in your computer's /Desktop folder. Go back to the Jupyter Notebook in your browser, click on Desktop, then the mla folder, then the file *tmp.ipynb*. You should see a cell with the following lines of code in it:

```
[2]: print("I love Python!")
```

```
I love Python!
```

Put your mouse cursor inside the cell and click on the Run button at the top menu (the icon with a black triangle and the word Run in it). You should see a message below the cell as the output. The message says, "I love Python!"

Two Ways to Run the Code in a Cell in Jupyter Notebook

There are two ways you can run the code in a cell in Jupyter Notebook: press the Run button, or press the ENTER and SHIFT keys simultaneously.

1.4.2 Issue Commands in Jupyter Notebook

You can issue certain commands in Jupyter Notebook without going to the Anaconda prompt (in Windows) or a terminal (in Mac or Linux).

For example, if you want to pip install the **matplotlib** library in the virtual environment MLA, you can do it in two different ways. The first way is to open the Anaconda prompt (in Windows) or a terminal (in Mac or Linux) and issuing the following two lines of commands:

```
conda activate MLA
```

```
pip install matplotlib==3.5.2
```

The second way, a shortcut, is to enter the following line of code in a cell:

```
[3]: !pip install matplotlib==3.5.2
```

Run the above cell will install the *matplotlib* library on your computer. Note that since you have opened the Jupyter Notebook in the MLA virtual environment, you have installed the *matplotlib* library in the MLA virtual environment, not in the base Python environment.

Make sure you put an exclamation mark (!) in front of the code in the cell. This tells Python to use the cell as a shortcut to the command line.

Not All Commands Can Be Executed in a Jupyter Notebook Cell

Not all commands can be executed in a Jupyter Notebook cell. For example, you cannot conda install a package by issuing commands in a Jupyter Notebook cell. As a matter of fact, you cannot execute any Conda command in a Jupyter Notebook cell.

1.5 FILE SYSTEM FOR THE BOOK

First, make sure that you have a subfolder /mla in your computer's /Desktop folder. We'll use the folder /mla to contain all files for this book. We'll use a subfolder /utils within the /mla folder for all local packages that we use for this book. We'll use another subfolder /files to contain other files such as graphs and videos. Within the /files directory, we create a sub-directory for each chapter such as /ch01, /ch02, and so on.

Python Modules, Packages, and Libraries

Python modules, packages, and libraries differ slightly. A Python module is a single file with the .py extension. In contrast, a Python package is a collection of Python modules contained in a single directory. The directory must have a file named **__init__**.py to distinguish it from a directory that happens to have .py extension files in it. A Python library is a collection of Python packages. We'll use the terms modules, packages, and libraries loosely and sometimes interchangeably.

Next, open a Jupyter notebook in the MLA virtual environment by following instructions earlier in this chapter. Save it as *ch01.ipynb* in the folder /Desktop/mla/. Enter the following lines of code in it and run the cell:

```
[4]: import os

     os.makedirs("utils", exist_ok=True)
     os.makedirs("files/ch01", exist_ok=True)
```

The *makedirs()* method in the *os* library creates a directory on your computer. The exist_ok=True option tells Python not to return an error message if such a directory already exists. Download the file *__init__.py* from the book's GitHub repository https://github.com/markhliu/MLA and save it in the folder /Desktop/mla/utils/ on your computer.

With that, you are all set up. You'll learn how to create pictures and animations in Python in the next chapter.

1.6 GLOSSARY

- **Activate A Virtual Environment:** Go into the subdirectory for a virtual environment so that you can write programs using packages within the virtual environment.
- **Anaconda:** An open-source software distribution, package, and environment manager.
- **Anaconda Navigator:** An app in Windows, Mac, or Linux that you can install on your computer to manage virtual environments, packages, and programs using a user interface.
- **Anaconda Prompt:** An app for the Windows operating system that you can install on your computer to manage virtual environments, packages, and programs via command lines.
- **Conda Install:** Installing packages for Python on your computer. The packages are provided by Anaconda and can be written in either Python, C, C++, or R.

- **IDE:** An integrated development environment. A comprehensive application for computer programming software development. It usually provides a source code editor, a complier, and a debugger.
- **Pip Install:** Installing packages on your computer from the Python Package Index. Only packages written in Python can be installed.

- **Virtual Environment:** An isolated environment on your computer to contain all needed files for a project.

1.7 EXERCISES

1.1 Install Anaconda on your computer. Open the Anaconda Navigator app on your computer.

1.2 Create a new virtual environment and name it MLA.

1.3 Activate the virtual environment MLA you just created in Exercise 2. Then deactivate the virtual environment and go to the base environment using the two methods discussed in this chapter.

1.4 Install Jupyter Notebook in the virtual environment MLA.

1.5 Download the file *tmp.ipynb* from the book's GitHub repository https://github.com/markhliu/MLA/blob/main/files/tmp.ipynb. Open the Jupyter Notebook app on your computer, and open the file *tmp.ipynb* in the Jupyter Notebook app.

1.6 Continue the previous exercise, in a new cell in Jupyter Notebook, write a line of Python code so that the output is a message "Machine Learning is fun!"

1.7 Continue the previous exercise, put the command !pip install matplotlib==3.5.2 in a new cell in Jupyter Notebook, run the code in the cell and see what happens.

1.8 Continue the previous exercise, put the command !conda install yt in a new cell in Jupyter Notebook, run the code in the cell and see what happens. Hint: you should get an error message because you cannot run any Conda command in a Jupyter Notebook cell.

1.9 Use the *makedirs()* method from the *os* library to create a folder named mla on your computer's desktop. Then create two subfolders utils and files inside the mla folder. Make sure you put the exist_ok=True option in the *makedirs()* method so that Python won't return an error message if such a directory already exists.

References

Andrej Karpathy. Blog: http://karpathy.github.io/2016/05/31/rl/, 2016.

Richard Bellman. Adaptive Control Processes: A Guided Tour. Princeton University Press, 1961.

Ian Goodfellow Yoshua Bengio and Aaron Courville. Deep Learning. The MIT Press, 2016.

Saheli Roy Choudhury. bit.ly/43QnZWB. *CNBC*, 2016.

Martin Coulter and Greg Bensiger. Alphabet shares dive after google ai chatbot bard flubs answer in ad: bit.ly/4602vc2, 2023.

DeepMind. Blog: deepmind.com/blog/deep-reinforcement-learning, 2016.

History.com Editors. https://www.history.com/this-day-in-history/deep-blue-defeats-garry-kasparov-in-chess-match. *HISTORY*, 2009.

Richard Feynman. The Meaning of It All: Thoughts of a Citizen-Scientist. Basic Books, 2015 Reprint.

Y.W. Teh G.E. Hinton, S. Osindero. A fast learning algorithm for deep belief nets. Neural Computation, 2006.

Hado Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, Advances in Neural Information Processing Systems, volume 23. Curran Associates, Inc., 2010.

Institutional Investor. CFA Institute Makes Biggest Single Package of Changes in Its History: bit.ly/42AyFb4, 2023.

Jocob Chapman and Mathias Lechner. Deep Q-Learning for Atari Breakout: https://keras.io/examples/rl/deep_q_network_breakout/, 2020.

Leaders. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data. *The Economist*, 2017.

Mark Liu. Make Python Talk: Build Apps with Voice Control and Speech Recognition. No Starch Press, 2021.

Scott MacFarland. https://www.huffpost.com/entry/if-a-picture-video-production_b_4996655/. *The Huffington Post*, 2014.

Bernard Marr. https://www.forbes.com/sites/bernardmarr/2018/12/31/the-most-amazing-artificial-intelligence-milestones-so-far/?sh=6698f5c37753. *Forbes*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, February 2015.

OpenAI. https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html, 2018.

Max Pumperla and Kevin Ferguson. Deep Learning and the Game of Go. Manning, 2019.

Barry Schwartz. The Paradox of Choice: Why More Is Less. Harper Perennial, 2005.

Eric Seigel. Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die. Wiley, 2015.

Charles Severance. *Python for Everyone: Exploring Data in Python 3*. 2016.

Daniel J. Siegel. The Developing Mind: Toward a Neurobiology of Interpersonal Experience. The Guilford Press, 1999.

J.F. Gusella T. Green, S.F. Heinemann. Molecular neurobiology and genetics: Investigation of neural function and dysfunction. *Neuron*, 1998.

The Data Team. econ.st/43PiYOb. *The Economist*, 2018.

The CFA Institute. Hands-On Learning: evolve.cfainstitute.org/practical-skills-modules.html, 2023.