PART I

BASIC CONCEPTS

This part is meant to provide enough background and basic information to anyone coming from non-technical fields that they can get the most out of the rest of the book. It starts by providing a clear introduction to machine learning (ML) and various related concepts, along with computational thinking. At this point, all of these are quite conceptual, but we still work through some hands-on examples. Of course, to further our investigation in ML, we need to pick up some tools. We start this by learning Python. If you have worked with Python before you can skip this chapter or just skim over it quickly. Next, there is a chapter on using cloud-based services for doing ML. This is becoming increasingly popular as it circumvents the need to buy one's own high-end hardware or worry about scalability of one's ML apps. The three popular platforms – Amazon Web Services (AWS), Google Cloud, and Microsoft Azure – are introduced here.

1 Teaching Computers to Write Programs

What do you need?
- A general understanding of computer and data systems.
- Familiarity with a few basic computer apps, such as spreadsheets.

What will you learn?
- Definitions and example applications of machine learning.
- Solving a small machine learning problem by hand.
- Computation thinking – a way to solve problems systematically.
- Introduction to tools, techniques, and skills for doing machine learning.

1.1 What Is Machine Learning?

Before we answer the question of what machine learning (ML) is, let's consider learning itself. The *New Oxford American Dictionary* (third edition) defines "to learn" as: "To get knowledge of something by study, experience, or being taught; to become aware by information or from observation; to commit to memory; to be informed of or to ascertain; to receive instruction."

All these meanings have limitations when they are associated with computers or machines. With the first two meanings, it is virtually impossible to test whether learning has been achieved or not. How can you check whether a machine has obtained knowledge of something? You probably cannot ask it questions, and even if you could, you would not be testing its ability to learn but its ability to answer questions. How can you tell whether it has become aware of something? The whole question of whether computers can be aware is a burning philosophical issue.

As for the last three meanings, although we can see what they denote in human terms, merely committing to memory and receiving instruction seems to fall short of what we mean by ML. These tasks are too passive, and we know that they are trivial for today's computers. Instead, we are interested in improving computer performance, or at least in the potential for improving performance, in new situations. You can commit something to memory or be informed of something by rote learning without being able to apply the new knowledge to new situations. In other words, you can receive instruction without

benefiting from it at all. Therefore, it is important to come up with a new operational definition of learning in the context of the machine, which we can formulate as:

> Things learn when they change their behavior in a way that makes them perform better in the future.

This ties learning to performance rather than knowledge. You can test learning by observing present behavior and comparing it with past behavior. This is a more objective kind of definition and is more appropriate for our purposes. Therefore, we are going to focus on the concept of learning that is connected to a change in knowledge (objectively defined in some way) or performance in a task. This is important for us to understand and accept, not just for the sake of philosophical or definitional discussions, but in order to envision what ML could and should do.

Learning usually involves a change in the system or person that is supposed to be learning. This change, presumably, should be in a positive direction – whatever "positive" means in the given context. Think about learning to play a piano. It requires knowing some basic rules about piano and music, a few pointers about where to start and what to do, and a goal, or at least some metric for improvement. Using this framework you can be said to be learning a piano if you follow those rules (maybe inventing a few new ones along the way), and with repeated practice, improve your performance. Here, performance could be simply being able to play the "Happy Birthday" song without any help or hesitation; or it could be qualifying to play with the London Philharmonic. The point is – you are doing something and improving your performance. You are learning.

Can we use that same idea for computers? This is what Tom Mitchell, a renowned scholar and a leader in the field of ML provided as a definition for computer (or any automated system) learning: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." Think about this for a minute. While this definition is for machines (computer programs), it equally applies to humans. In the example above, the task (T) is to play a piano. The experience (E) is repeated practice, and the performance (P) is improvement toward the goal of being able to play a song or qualify for a professional entourage. If you are improving in doing that task with more practice, you are learning. If not, then perhaps you should reconsider your choice in the hobby, the career, or the teacher.

Now we understand what learning is. The next question is *why should a computer learn?* Imagine what happens in a typical programmable system. There is usually an algorithm – a set of rules and instructions – that drives that system. Think about a traffic light at a major intersection. There is an underlying algorithm that controls how each of the light-sets behave for different lanes and directions. Perhaps it changes the timing for them at night or during rush hour. No matter what, these algorithms are all pre-programmed. One could do a traffic study over several weeks to see if that program is working optimally and adjust its parameters (e.g., time to stay on, frequency of changes) as needed. In this case, a deterministic program is changed using human learning. But what if that system, that program, could figure out the optimal traffic light pattern by itself? In other words, can we have it adjust its own parameters? Figure 1.1 shows a simple schematic

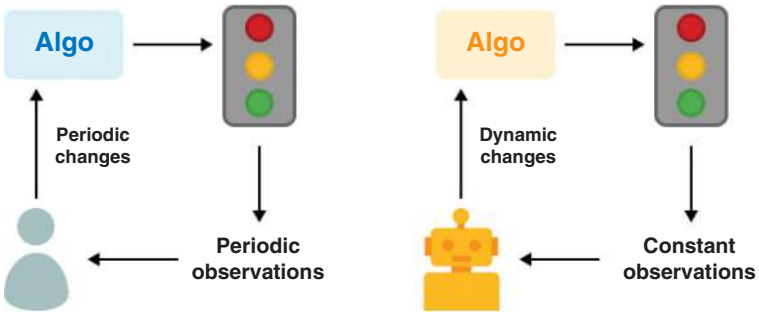

Figure 1.1 (Left) traditional (deterministic), human-operated changes to traffic light behavior; and (right) an ML-based approach to optimizing traffic light patterns.

difference between a traditional, deterministic program for traffic lights, and one where the program learns.

Hopefully you can see the difference between these two models. The first model (on the left) is costly as it involves human observations and could be less accurate since it may be prohibitive to patrol a team of humans for a very long time to observe every moment of every day and every season. The second model (on the right) may cost some extra hardware at first (e.g., cameras), but we can bypass costly observations and analysis by humans. Here, the ML-based agent or program is able to change the algorithm (and associated code) that drives the traffic light behavior.

In general, ML allows computational systems to write their own algorithms or tune the parameters for existing algorithms. According to Arthur Samuel (1959), an American pioneer in the field of artificial intelligence (AI), the goal of ML is to give "computers the ability to learn without being explicitly programmed."[1] Machine learning is a field of study that explores the use of algorithms that can learn from data and use that knowledge to make predictions on data they have not seen before. Such algorithms are designed to make data-driven predictions or decisions by building a model from sample inputs. While quite a few ML algorithms have been around for a long time, the ability to automatically apply complex mathematical calculations to big data in an efficient manner is a recent development. Following are a few widely publicized examples of ML applications you may be familiar with.

The first is the heavily hyped self-driving Google car (now rebranded as WAYMO). As shown in Figure 1.2, this car uses a real view of the road to recognize objects and patterns such as sky, road signs, and moving vehicles in a different lane. This process is quite complicated for a machine to do. A lot of things may look like a car (that blue blob in the bottom image is a car), and it may not be easy to identify where a street sign is. The self-driving car needs to not only do such object recognition, but also make decisions about navigation. There are so many unknowns involved here that it is impossible to come up with an algorithm for a car to execute. Instead, the car needs to know the rules for driving, have the ability to do object and pattern recognition, and apply these to making decisions in real time. In addition, it needs to keep improving. That is where ML comes into play.

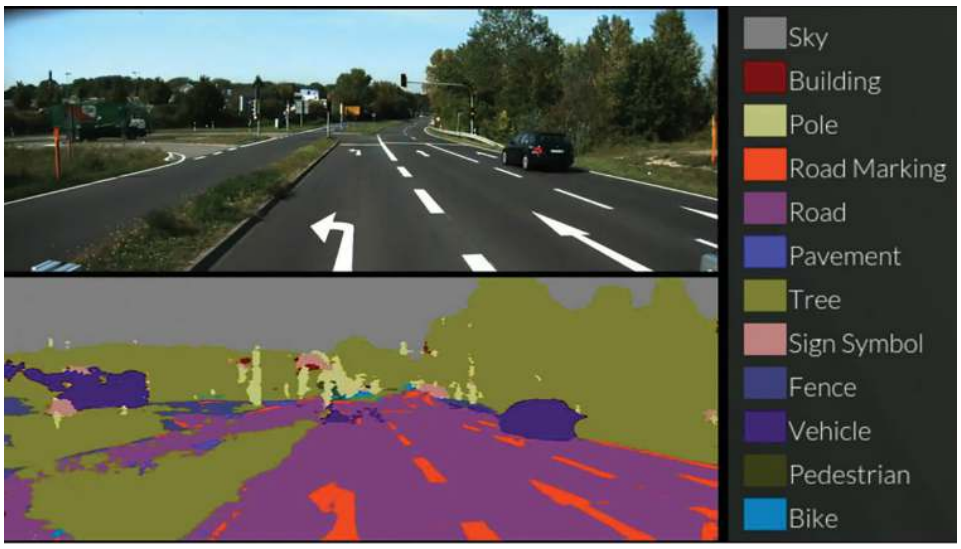

Figure 1.2 Machine learning technology behind a self-driving car (source: YouTube: Deep Learning: Technology Behind Self-Driving Car[2]).

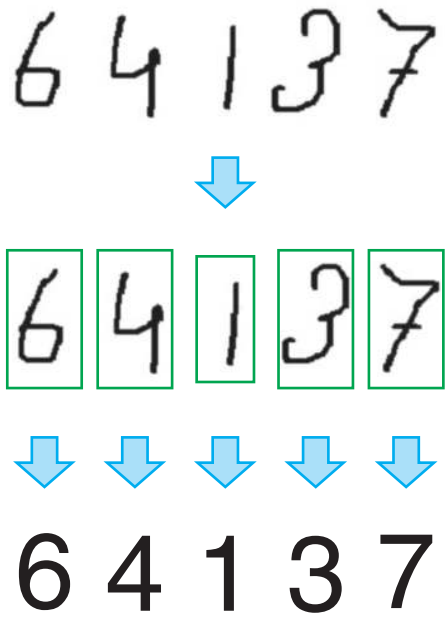

Figure 1.3 The problem of OCR.

Another classic example of ML is optical character recognition (OCR). Humans are good at recognizing handwritten characters, but computers are not. Why? Because there are too many variations in the way that any one character can be written, and there is no way we could teach a computer all those variations. There may also be noise – an

unfinished character, a join with another character, some unrelated stuff in the background, uneven or disjoint lines or curves, etc. So, once again, what we need is to give the computer a basic set of rules that says what "A," "a," "5," etc., look like, and then have it make a decision based on pattern recognition. The way this happens is by showing many versions of a character to the computer so it *learns* that character, just like a child would through repetition, and then have it go through the recognition process (Figure 1.3).

This is an example of a classification problem. In such a problem, the objective is to learn how different patterns (here, handwritten characters) fall under one of the classes of interest (here, digits or characters). As a reference, classification falls under what is known as supervised learning. The self-driving car example we saw before represents unsupervised learning and reinforcement learning categories of ML. In the following section we will discuss these branches in more detail. Before that, let's work through a few examples for our hands-on practice.

Hands-on Example 1.1: Machine Learning by Hand

Let's take an example that is perhaps more relevant to everyday life. If you have used any online services, the chances are you have come across recommendations. Take, for instance, services such as Amazon and Netflix. How do they know what products to recommend? We understand that they are *monitoring* our activities, that they have our past records, and that is how they are able to give us suggestions. But how exactly? They use something called **collaborative filtering** (CF). This is a method that uses your past behavior and its similarities with the behaviors of other users in that community to figure out what you may like in the future.

Look at Table 1.1. There is data about four people's ratings for different movies. The objective for the system is to figure out if Person 5 will like a movie or not based on that data as well as her own movie ratings from the past. In other words, it is trying to *learn* what kinds of things Person 5 likes (and dislikes) based on what others similar to Person 5 like, and uses that *knowledge* to make new recommendations. On top of that, as Person 5 accepts or rejects its recommendations, the system extends its learning to include knowledge about how Person 5 responds to its suggestions, and further corrects its models.

Table 1.1 Machine learning-based CF for movie recommendations.

		Movie name				
		Sherlock Holmes	*Avengers*	*Titanic*	*La La Land*	*WALL-E*
Rating	Person 1	4	5	3	4	2
	Person 2	3	4	3	2	4
	Person 3	2	3	4	5	3
	Person 4	3	4	4	5	2
	Person 5	4	?	3	?	4

Can we solve this by hand? Let's try. First off, what is your guess? Is there another person in this dataset that Person 5 is close to in terms of their movie tastes? We have data for only three movies for Person 5. They seem to like two of the movies (*Sherlock Holmes* and *WALL-E*), but not so much the third one (*Titanic*). Do we know anyone else with the same or similar taste? What about Person 2? It is not an exact match, but a close one. Based on that match, if we were to guess, we could recommend *Avengers* to Person 5, but not *La La Land*. Of course, in reality, things are not as straightforward, and we do not have to rely on a simple technique like the one we just used to figure out what to recommend. But I hope this simple exercise gave you a glimpse into how ML can work in real-life scenarios. What we just did is an example of applying a classification or clustering technique to a recommendation application. We will learn about classification in Chapters 5 and 6, clustering in Chapter 7, and practical recommendation systems in Chapters 12.

Movie recommendation is perhaps something we all encounter and can relate to as we can see those recommendations coming our way in an explicit manner, but there are many other applications where recommendations happen in subtle ways. For example, Facebook uses ML to personalize each member's news feed. Most financial institutions use ML algorithms to detect fraud. Intelligence agencies use ML to sift through mounds of information to look for credible threats of terrorism.

There are many other applications that we encounter in daily life that involve ML in one way or another. In fact, it is almost impossible to finish our day without having used something that is driven by ML. Did you do any online browsing or searching today? Did you go to a grocery store? Did you use a social media app on your phone? Then you have used ML applications.

Try It Yourself 1.1: Machine Learning by Hand

The recommendation example we tried above used CF, which uses data about similar people to make recommendations to a given person. But we can also use **content filtering**, which uses information about similar items to decide what to recommend. In Table 1.2 we have ratings for one person's movies and various characteristics of those movies. Based on this information, would you recommend *Avengers* to this person? Explain your process and reasoning.

Table 1.2 Machine learning-based filtering for movie recommendations.

		Characteristics			
		Rating	Year	Genre	Lead
Movies	*Sherlock Holmes*	4	2009	Action	Robert Downey, Jr.
	Avengers	?	2012	Action	Robert Downey, Jr.
	Titanic	3	1997	Drama	Leonardo DiCaprio
	La La Land	2	2016	Comedy	Ryan Gosling
	WALL-E	4	2008	Animation	Ben Burtt

In most cases the application of ML is entwined with the application of statistical analysis. Therefore, it is important to remember the differences in the nomenclature of these two fields:

- A variable in statistics is called a **feature** in ML.
- In statistics a target is called a dependent variable, whereas in ML it is called a **label**.
- A transformation in statistics is called **feature creation** in ML.

As we move through various chapters, especially Chapter 4, you will see more ML terminology used. If you are coming from a strong statistics background or even have taken an introductory level statistics class before, these differences are important to note to get you comfortable with ML. If you did not have a chance to learn basic statistics, or it has gotten a bit rusty, I strongly recommend you brush up on it before continuing, or refer to appropriate appendices in this book as needed.

1.2 Branches of Machine Learning

In the previous section we looked at different problems that we could solve using ML techniques. Now, let's look at these techniques within a systematic framework. Figure 1.4 shows the three main branches of ML: supervised, unsupervised, and reinforcement.

If we are trying to predict a value by learning (from data) how various predictors relate to the response variable, we are performing **supervised learning**. Within that branch, if the response variable is continuous, the problem is **regression**. Think about knowing someone's age and occupation and predicting their income. If, on the other hand, the response variable is **discrete** (having a few possible values or labels), this is a **classification** problem. For instance, using someone's age and occupation to learn whether they are a high-earner, medium-earner, or low-earner (three classes) is classification. We will cover regression in Chapter 4 and classification in Chapters 5 and 6.

Supervised learning problems require us to know the truth first. For example, in order to learn how age and occupation predict one's earning class, we need to know the true value

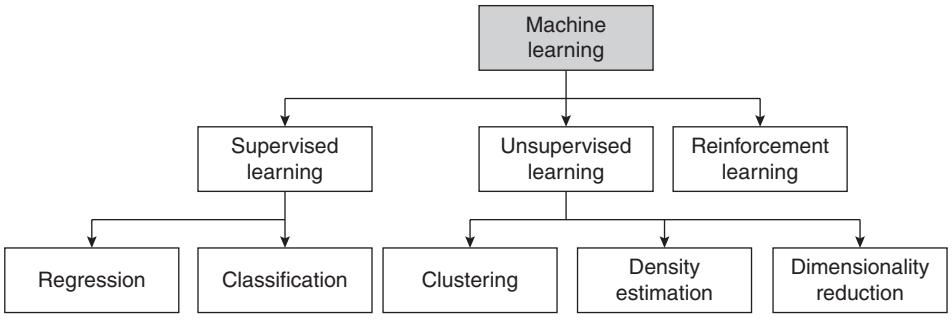

Figure 1.4 An outline of the core ML branches.

of someone's class – are they a high-earner, medium-earner, or low-earner? But there are times when the data given to us do not have clear labels or true values. And yet, we are tasked with exploring and explaining that data. In this case, we are dealing with an **unsupervised learning** problem. Within that, if we want to organize data into various groups, that is a **clustering** problem. Clustering is similar to classification, but unlike classification we do not know how many classes there are or what they are called. On the other hand, if we are trying to explain the data by estimating underlying processes that may be responsible for how the data is distributed, that is a **density estimation** problem. We will cover clustering and density estimation in Chapter 7.

There are times when the problem we are dealing with does not have labeled data (supervised learning), nor does it involve organizing the data or observations in some way (unsupervised learning). More specifically, there are situations where the given context or environment is so complex or dynamic that we have to learn on the go without having labels or clear boundaries. That brings us to the third branch of ML – **reinforcement learning**. This used to be exclusively popular in robotics, but has gotten a lot of attention in other kinds of ML applications due to its versatility and appeal in dynamically changing environments. For example, some of the best chess-playing programs in the world are built using reinforcement learning, and so are core components in self-driving cars. We will cover reinforcement learning in Chapter 11.

These branches will appear in the chapters on neural networks (Chapters 9 and 10) as well. A neural network model can be built for solving regression or classification (supervised learning) as well as clustering, density estimation, and dimensionality reduction (unsupervised learning). A neural network-based model can be combined with reinforcement learning processes to build a dynamic and adaptive intelligent system.

FYI: AI vs. ML

It is not uncommon to see many people and places using AI (artificial intelligence) and ML (machine learning) interchangeably. But it is important to understand that ML is typically understood to be a subset of AI. Of course, these days, ML techniques and applications have been so predominant and successful that they overwhelm the field of AI. Perhaps it may be better to think about what is in AI that is not in ML.

Areas of knowledge representation, constraint optimization, logical and probabilistic reasoning, and planning are examples of core AI that are typically outside of ML. Most expert systems and rule-based intelligent entities are also in AI, but are not considered to be in ML. In general, AI is about getting systems to perform tasks that typically require human cognition and decision-making abilities. A lot of this is being done using various ML techniques these days, but not all.

In the end, this difference may not matter much to most. Certainly, in this book, as our goal is to apply various techniques to solve problems, we would not be concerned about this. For the most part, we are focused on ML, but as we get more reflective and start contextualizing our work toward the end, we will reconnect with AI in the last chapter.

1.3 What Do You Need for Machine Learning?

Several things are required for a student to successfully learn and practice ML. One needs to, of course, be able to read, write, and think logically. But what are some of the specialized requirements? Here, we divide them up into two parts: skills and tools. In this book, we will cover both of those parts, but keep in mind that some skills are better covered in other parts of a curricula. For example, while we will cover some required math, especially statistics, it is expected that the reader has already received a comprehensive and formal education in statistics. Similarly, there are many tools for doing ML, but in this book we will cover only some of them – namely, Python and cloud computing. If you are interested in diving deeper into those tools or platforms you will have to find some other resources outside of this book. Some of these resources are listed at the end of the corresponding chapters.

1.3.1 Skills

Given that ML is about teaching computers to program themselves, one needs to know how to program in the first place. Programming is an essential skill for doing ML. But what else do you need? Historically, ML was almost exclusively taught in computer science (CS) programs. A lot of what is needed for ML is also already covered in any traditional statistics program. And this means, traditionally, the skills needed for ML were those that were typically found in CS and statistics programs.

Things have changed now. First, the entry barrier for ML has come down substantially. Before, one needed expensive hardware and sophisticated tools and programming skills to be able to develop ML solutions. That is no longer the case. Most ML techniques can be learned and practiced with nothing more than a modern laptop and online resources. There are existing libraries or packages that make building ML models as easy as writing your first "Hello, World!" program (see the next subsection). Lots of datasets are now available freely and easily for training and testing models. There are plenty of online forums and discussion groups where one can seek and receive help. ML is no longer reserved for the few and the privileged, nor should it be.

That brings us to the second point. Machine learning is not just about solving a few complex and sometimes abstract problems. Given how ML is integrated in our lives, solving ML problems and applying ML techniques goes way beyond traditional CS boundaries. In fact, as we will see in the section on ethics, bias, and privacy later in this chapter, and in the chapter on responsible AI (Chapter 13), we need people from different disciplines to join the ML revolution. We need people who can think not just with a computational lens, but also a social one. And while we are on that subject, I want to reiterate and remind you that this book is explicitly written with that premise in mind – lowering the barrier for ML entry and making ML accessible to a broad audience to engage them in the future of ML and AI.