1 Introduction

The subject of this book is automated learning, or, as we will more often call it, Machine Learning (ML). That is, we wish to program computers so that they can "learn" from input available to them. Roughly speaking, learning is the process of converting experience into expertise or knowledge. The input to a learning algorithm is training data, representing experience, and the output is some expertise, which usually takes the form of another computer program that can perform some task. Seeking a formal-mathematical understanding of this concept, we'll have to be more explicit about what we mean by each of the involved terms: What is the training data our programs will access? How can the process of learning be automated? How can we evaluate the success of such a process (namely, the quality of the output of a learning program)?

1.1 What Is Learning?

Let us begin by considering a couple of examples from naturally occurring animal learning. Some of the most fundamental issues in ML arise already in that context, which we are all familiar with.

Bait Shyness – Rats Learning to Avoid Poisonous Baits: When rats encounter food items with novel look or smell, they will first eat very small amounts, and subsequent feeding will depend on the flavor of the food and its physiological effect. If the food produces an ill effect, the novel food will often be associated with the illness, and subsequently, the rats will not eat it. Clearly, there is a learning mechanism in play here – the animal used past experience with some food to acquire expertise in detecting the safety of this food. If past experience with the food was negatively labeled, the animal predicts that it will also have a negative effect when encountered in the future.

Inspired by the preceding example of successful learning, let us demonstrate a typical machine learning task. Suppose we would like to program a machine that learns how to filter spam e-mails. A naive solution would be seemingly similar to the way rats learn how to avoid poisonous baits. The machine will simply *memorize* all previous e-mails that had been labeled as spam e-mails by the human user. When a new e-mail arrives, the machine will search for it in the set

of previous spam e-mails. If it matches one of them, it will be trashed. Otherwise, it will be moved to the user's inbox folder.

While the preceding "learning by memorization" approach is sometimes useful, it lacks an important aspect of learning systems – the ability to label unseen e-mail messages. A successful learner should be able to progress from individual examples to broader *generalization*. This is also referred to as *inductive reasoning* or *inductive inference*. In the bait shyness example presented previously, after the rats encounter an example of a certain type of food, they apply their attitude toward it on new, unseen examples of food of similar smell and taste. To achieve generalization in the spam filtering task, the learner can scan the previously seen e-mails, and extract a set of words whose appearance in an e-mail message is indicative of spam. Then, when a new e-mail arrives, the machine can check whether one of the suspicious words appears in it, and predict its label accordingly. Such a system would potentially be able correctly to predict the label of unseen e-mails.

However, inductive reasoning might lead us to false conclusions. To illustrate this, let us consider again an example from animal learning.

Pigeon Superstition: In an experiment performed by the psychologist B. F. Skinner, he placed a bunch of hungry pigeons in a cage. An automatic mechanism had been attached to the cage, delivering food to the pigeons at regular intervals with no reference whatsoever to the birds' behavior. The hungry pigeons went around the cage, and when food was first delivered, it found each pigeon engaged in some activity (pecking, turning the head, etc.). The arrival of food reinforced each bird's specific action, and consequently, each bird tended to spend some more time doing that very same action. That, in turn, increased the chance that the next random food delivery would find each bird engaged in that activity again. What results is a chain of events that reinforces the pigeons' association of the delivery of the food with whatever chance actions they had been performing when it was first delivered. They subsequently continue to perform these same actions diligently.[1]

What distinguishes learning mechanisms that result in superstition from useful learning? This question is crucial to the development of automated learners. While human learners can rely on common sense to filter out random meaningless learning conclusions, once we export the task of learning to a machine, we must provide well defined crisp principles that will protect the program from reaching senseless or useless conclusions. The development of such principles is a central goal of the theory of machine learning.

What, then, made the rats' learning more successful than that of the pigeons? As a first step toward answering this question, let us have a closer look at the bait shyness phenomenon in rats.

Bait Shyness revisited – rats fail to acquire conditioning between food and electric shock or between sound and nausea: The bait shyness mechanism in

[1] See: http://psychclassics.yorku.ca/Skinner/Pigeon

rats turns out to be more complex than what one may expect. In experiments carried out by Garcia (Garcia & Koelling 1996), it was demonstrated that if the unpleasant stimulus that follows food consumption is replaced by, say, electrical shock (rather than nausea), then no conditioning occurs. Even after repeated trials in which the consumption of some food is followed by the administration of unpleasant electrical shock, the rats do not tend to avoid that food. Similar failure of conditioning occurs when the characteristic of the food that implies nausea (such as taste or smell) is replaced by a vocal signal. The rats seem to have some "built in" prior knowledge telling them that, while temporal correlation between food and nausea can be causal, it is unlikely that there would be a causal relationship between food consumption and electrical shocks or between sounds and nausea.

We conclude that one distinguishing feature between the bait shyness learning and the pigeon superstition is the incorporation of *prior knowledge* that biases the learning mechanism. This is also referred to as *inductive bias*. The pigeons in the experiment are willing to adopt *any* explanation for the occurrence of food. However, the rats "know" that food cannot cause an electric shock and that the co-occurrence of noise with some food is not likely to affect the nutritional value of that food. The rats' learning process is biased toward detecting some kind of patterns while ignoring other temporal correlations between events.

It turns out that the incorporation of prior knowledge, biasing the learning process, is inevitable for the success of learning algorithms (this is formally stated and proved as the "No-Free-Lunch theorem" in Chapter 5). The development of tools for expressing domain expertise, translating it into a learning bias, and quantifying the effect of such a bias on the success of learning is a central theme of the theory of machine learning. Roughly speaking, the stronger the prior knowledge (or prior assumptions) that one starts the learning process with, the easier it is to learn from further examples. However, the stronger these prior assumptions are, the less flexible the learning is – it is bound, a priori, by the commitment to these assumptions. We shall discuss these issues explicitly in Chapter 5.

1.2 When Do We Need Machine Learning?

When do we need machine learning rather than directly program our computers to carry out the task at hand? Two aspects of a given problem may call for the use of programs that learn and improve on the basis of their "experience": the problem's complexity and the need for adaptivity.

Tasks That Are Too Complex to Program.

- *Tasks Performed by Animals/Humans:* There are numerous tasks that we human beings perform routinely, yet our introspection concerning how we do them is not sufficiently elaborate to extract a well

defined program. Examples of such tasks include driving, speech recognition, and image understanding. In all of these tasks, state of the art machine learning programs, programs that "learn from their experience," achieve quite satisfactory results, once exposed to sufficiently many training examples.

- *Tasks beyond Human Capabilities:* Another wide family of tasks that benefit from machine learning techniques are related to the analysis of very large and complex data sets: astronomical data, turning medical archives into medical knowledge, weather prediction, analysis of genomic data, Web search engines, and electronic commerce. With more and more available digitally recorded data, it becomes obvious that there are treasures of meaningful information buried in data archives that are way too large and too complex for humans to make sense of. Learning to detect meaningful patterns in large and complex data sets is a promising domain in which the combination of programs that learn with the almost unlimited memory capacity and ever increasing processing speed of computers opens up new horizons.

Adaptivity. One limiting feature of programmed tools is their rigidity – once the program has been written down and installed, it stays unchanged. However, many tasks change over time or from one user to another. Machine learning tools – programs whose behavior adapts to their input data – offer a solution to such issues; they are, by nature, adaptive to changes in the environment they interact with. Typical successful applications of machine learning to such problems include programs that decode handwritten text, where a fixed program can adapt to variations between the handwriting of different users; spam detection programs, adapting automatically to changes in the nature of spam e-mails; and speech recognition programs.

1.3 Types of Learning

Learning is, of course, a very wide domain. Consequently, the field of machine learning has branched into several subfields dealing with different types of learning tasks. We give a rough taxonomy of learning paradigms, aiming to provide some perspective of where the content of this book sits within the wide field of machine learning.

We describe four parameters along which learning paradigms can be classified.

Supervised versus Unsupervised Since learning involves an interaction between the learner and the environment, one can divide learning tasks according to the nature of that interaction. The first distinction to note is the difference between supervised and unsupervised learning. As an

illustrative example, consider the task of learning to detect spam e-mail versus the task of anomaly detection. For the spam detection task, we consider a setting in which the learner receives training e-mails for which the label `spam/not-spam` is provided. On the basis of such training the learner should figure out a rule for labeling a newly arriving e-mail message. In contrast, for the task of anomaly detection, all the learner gets as training is a large body of e-mail messages (with no labels) and the learner's task is to detect "unusual" messages.

More abstractly, viewing learning as a process of "using experience to gain expertise," supervised learning describes a scenario in which the "experience," a training example, contains significant information (say, the `spam/not-spam` labels) that is missing in the unseen "test examples" to which the learned expertise is to be applied. In this setting, the acquired expertise is aimed to predict that missing information for the test data. In such cases, we can think of the environment as a teacher that "supervises" the learner by providing the extra information (labels). In unsupervised learning, however, there is no distinction between training and test data. The learner processes input data with the goal of coming up with some summary, or compressed version of that data. Clustering a data set into subsets of similar objets is a typical example of such a task.

There is also an intermediate learning setting in which, while the training examples contain more information than the test examples, the learner is required to predict even more information for the test examples. For example, one may try to learn a value function that describes for each setting of a chess board the degree by which White's position is better than the Black's. Yet, the only information available to the learner at training time is positions that occurred throughout actual chess games, labeled by who eventually won that game. Such learning frameworks are mainly investigated under the title of *reinforcement learning*.

Active versus Passive Learners Learning paradigms can vary by the role played by the learner. We distinguish between "active" and "passive" learners. An active learner interacts with the environment at training time, say, by posing queries or performing experiments, while a passive learner only observes the information provided by the environment (or the teacher) without influencing or directing it. Note that the learner of a spam filter is usually passive – waiting for users to mark the e-mails coming to them. In an active setting, one could imagine asking users to label specific e-mails chosen by the learner, or even composed by the learner, to enhance its understanding of what spam is.

Helpfulness of the Teacher When one thinks about human learning, of a baby at home or a student at school, the process often involves a helpful teacher, who is trying to feed the learner with the information most use-

ful for achieving the learning goal. In contrast, when a scientist learns about nature, the environment, playing the role of the teacher, can be best thought of as passive – apples drop, stars shine, and the rain falls without regard to the needs of the learner. We model such learning scenarios by postulating that the training data (or the learner's experience) is generated by some random process. This is the basic building block in the branch of "statistical learning." Finally, learning also occurs when the learner's input is generated by an adversarial "teacher." This may be the case in the spam filtering example (if the spammer makes an effort to mislead the spam filtering designer) or in learning to detect fraud. One also uses an adversarial teacher model as a worst-case scenario, when no milder setup can be safely assumed. If you can learn against an adversarial teacher, you are guaranteed to succeed interacting any odd teacher.

Online versus Batch Learning Protocol The last parameter we mention is the distinction between situations in which the learner has to respond online, throughout the learning process, and settings in which the learner has to engage the acquired expertise only after having a chance to process large amounts of data. For example, a stockbroker has to make daily decisions, based on the experience collected so far. He may become an expert over time, but might have made costly mistakes in the process. In contrast, in many data mining settings, the learner – the data miner – has large amounts of training data to play with before having to output conclusions.

In this book we shall discuss only a subset of the possible learning paradigms. Our main focus is on supervised statistical batch learning with a passive learner (for example, trying to learn how to generate patients' prognoses, based on large archives of records of patients that were independently collected and are already labeled by the fate of the recorded patients). We shall also briefly discuss online learning and batch unsupervised learning (in particular, clustering).

1.4 Relations to Other Fields

As an interdisciplinary field, machine learning shares common threads with the mathematical fields of statistics, information theory, game theory, and optimization. It is naturally a subfield of computer science, as our goal is to program machines so that they will learn. In a sense, machine learning can be viewed as a branch of AI (Artificial Intelligence), since, after all, the ability to turn experience into expertise or to detect meaningful patterns in complex sensory data is a cornerstone of human (and animal) intelligence. However, one should note that, in contrast with traditional AI, machine learning is not trying to build automated imitation of intelligent behavior, but rather to use the strengths and

special abilities of computers to complement human intelligence, often performing tasks that fall way beyond human capabilities. For example, the ability to scan and process huge databases allows machine learning programs to detect patterns that are outside the scope of human perception.

The component of experience, or training, in machine learning often refers to data that is randomly generated. The task of the learner is to process such randomly generated examples toward drawing conclusions that hold for the environment from which these examples are picked. This description of machine learning highlights its close relationship with statistics. Indeed there is a lot in common between the two disciplines, in terms of both the goals and techniques used. There are, however, a few significant differences of emphasis; if a doctor comes up with the hypothesis that there is a correlation between smoking and heart disease, it is the statistician's role to view samples of patients and check the validity of that hypothesis (this is the common statistical task of hypothesis testing). In contrast, machine learning aims to use the data gathered from samples of patients to come up with a description of the causes of heart disease. The hope is that automated techniques may be able to figure out meaningful patterns (or hypotheses) that may have been missed by the human observer.

In contrast with traditional statistics, in machine learning in general, and in this book in particular, algorithmic considerations play a major role. Machine learning is about the execution of learning by computers; hence algorithmic issues are pivotal. We develop algorithms to perform the learning tasks and are concerned with their computational efficiency. Another difference is that while statistics is often interested in asymptotic behavior (like the convergence of sample-based statistical estimates as the sample sizes grow to infinity), the theory of machine learning focuses on finite sample bounds. Namely, given the size of available samples, machine learning theory aims to figure out the degree of accuracy that a learner can expect on the basis of such samples.

There are further differences between these two disciplines, of which we shall mention only one more here. While in statistics it is common to work under the assumption of certain presubscribed data models (such as assuming the normality of data-generating distributions, or the linearity of functional dependencies), in machine learning the emphasis is on working under a "distribution-free" setting, where the learner assumes as little as possible about the nature of the data distribution and allows the learning algorithm to figure out which models best approximate the data-generating process. A precise discussion of this issue requires some technical preliminaries, and we will come back to it later in the book, and in particular in Chapter 5.

1.5 How to Read This Book

The first part of the book provides the basic theoretical principles that underlie machine learning (ML). In a sense, this is the foundation upon which the rest

of the book is built. This part could serve as a basis for a minicourse on the theoretical foundations of ML.

The second part of the book introduces the most commonly used algorithmic approaches to supervised machine learning. A subset of these chapters may also be used for introducing machine learning in a general AI course to computer science, Math, or engineering students.

The third part of the book extends the scope of discussion from statistical classification to other learning models. It covers online learning, unsupervised learning, dimensionality reduction, generative models, and feature learning.

The fourth part of the book, Advanced Theory, is geared toward readers who have interest in research and provides the more technical mathematical techniques that serve to analyze and drive forward the field of theoretical machine learning.

The Appendixes provide some technical tools used in the book. In particular, we list basic results from measure concentration and linear algebra.

A few sections are marked by an asterisk, which means they are addressed to more advanced students. Each chapter is concluded with a list of exercises. A solution manual is provided in the course Web site.

1.5.1 Possible Course Plans Based on This Book

A 14 Week Introduction Course for Graduate Students:

1. Chapters 2–4.
2. Chapter 9 (without the VC calculation).
3. Chapters 5–6 (without proofs).
4. Chapter 10.
5. Chapters 7, 11 (without proofs).
6. Chapters 12, 13 (with some of the easier proofs).
7. Chapter 14 (with some of the easier proofs).
8. Chapter 15.
9. Chapter 16.
10. Chapter 18.
11. Chapter 22.
12. Chapter 23 (without proofs for compressed sensing).
13. Chapter 24.
14. Chapter 25.

A 14 Week Advanced Course for Graduate Students:

1. Chapters 26, 27.
2. (continued)
3. Chapters 6, 28.
4. Chapter 7.
5. Chapter 31.

6. Chapter 30.
7. Chapters 12, 13.
8. Chapter 14.
9. Chapter 8.
10. Chapter 17.
11. Chapter 29.
12. Chapter 19.
13. Chapter 20.
14. Chapter 21.

1.6 Notation

Most of the notation we use throughout the book is either standard or defined on the spot. In this section we describe our main conventions and provide a table summarizing our notation (Table 1.1). The reader is encouraged to skip this section and return to it if during the reading of the book some notation is unclear.

We denote scalars and abstract objects with lowercase letters (e.g. x and λ). Often, we would like to emphasize that some object is a vector and then we use boldface letters (e.g. $\mathbf{x}$ and $\boldsymbol{\lambda}$). The ith element of a vector $\mathbf{x}$ is denoted by x_i. We use uppercase letters to denote matrices, sets, and sequences. The meaning should be clear from the context. As we will see momentarily, the input of a learning algorithm is a sequence of training examples. We denote by z an abstract example and by $S = z_1, \ldots, z_m$ a sequence of m examples. Historically, S is often referred to as a training *set*; however, we will always assume that S is a *sequence* rather than a set. A sequence of m vectors is denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_m$. The ith element of $\mathbf{x}_t$ is denoted by $x_{t,i}$.

Throughout the book, we make use of basic notions from probability. We denote by $\mathcal{D}$ a distribution over some set,[2] for example, Z. We use the notation $z \sim \mathcal{D}$ to denote that z is sampled according to $\mathcal{D}$. Given a random variable $f : Z \to \mathbb{R}$, its expected value is denoted by $\mathbb{E}_{z\sim\mathcal{D}}[f(z)]$. We sometimes use the shorthand $\mathbb{E}[f]$ when the dependence on z is clear from the context. For $f : Z \to \{\texttt{true}, \texttt{false}\}$ we also use $\mathbb{P}_{z\sim\mathcal{D}}[f(z)]$ to denote $\mathcal{D}(\{z : f(z) = \texttt{true}\})$. In the next chapter we will also introduce the notation $\mathcal{D}^m$ to denote the probability over Z^m induced by sampling $(z_1, \ldots, z_m)$ where each point z_i is sampled from $\mathcal{D}$ independently of the other points.

In general, we have made an effort to avoid asymptotic notation. However, we occasionally use it to clarify the main results. In particular, given $f : \mathbb{R} \to \mathbb{R}_+$ and $g : \mathbb{R} \to \mathbb{R}_+$ we write $f = O(g)$ if there exist $x_0, \alpha \in \mathbb{R}_+$ such that for all $x > x_0$ we have $f(x) \leq \alpha g(x)$. We write $f = o(g)$ if for every $\alpha > 0$ there exists

[2] To be mathematically precise, $\mathcal{D}$ should be defined over some σ-algebra of subsets of Z. The user who is not familiar with measure theory can skip the few footnotes and remarks regarding more formal measurability definitions and assumptions.

Table 1.1 Summary of notation

symbol	meaning
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^d$	the set of d-dimensional vectors over $\mathbb{R}$
$\mathbb{R}_+$	the set of non-negative real numbers
$\mathbb{N}$	the set of natural numbers
$O, o, \Theta, \omega, \Omega, \tilde{O}$	asymptotic notation (see text)
$\mathbb{1}_{[\text{Boolean expression}]}$	indicator function (equals 1 if expression is true and 0 o.w.)
$[a]_+$	$= \max\{0, a\}$
$[n]$	the set $\{1, \ldots, n\}$ (for $n \in \mathbb{N}$)
$\mathbf{x}, \mathbf{v}, \mathbf{w}$	(column) vectors
x_i, v_i, w_i	the ith element of a vector
$\langle \mathbf{x}, \mathbf{v} \rangle$	$= \sum_{i=1}^d x_i v_i$ (inner product)
$\|\mathbf{x}\|_2$ or $\|\mathbf{x}\|$	$= \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ (the ℓ_2 norm of $\mathbf{x}$)
$\|\mathbf{x}\|_1$	$= \sum_{i=1}^d \|x_i\|$ (the ℓ_1 norm of $\mathbf{x}$)
$\|\mathbf{x}\|_\infty$	$= \max_i \|x_i\|$ (the ℓ_∞ norm of $\mathbf{x}$)
$\|\mathbf{x}\|_0$	the number of nonzero elements of $\mathbf{x}$
$A \in \mathbb{R}^{d,k}$	a $d \times k$ matrix over $\mathbb{R}$
$A^\top$	the transpose of A
$A_{i,j}$	the (i, j) element of A
$\mathbf{x}\,\mathbf{x}^\top$	the $d \times d$ matrix A s.t. $A_{i,j} = x_i x_j$ (where $\mathbf{x} \in \mathbb{R}^d$)
$\mathbf{x}_1, \ldots, \mathbf{x}_m$	a sequence of m vectors
$x_{i,j}$	the jth element of the ith vector in the sequence
$\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(T)}$	the values of a vector $\mathbf{w}$ during an iterative algorithm
$w_i^{(t)}$	the ith element of the vector $\mathbf{w}^{(t)}$
$\mathcal{X}$	instances domain (a set)
$\mathcal{Y}$	labels domain (a set)
Z	examples domain (a set)
$\mathcal{H}$	hypothesis class (a set)
$\ell : \mathcal{H} \times Z \to \mathbb{R}_+$	loss function
$\mathcal{D}$	a distribution over some set (usually over Z or over $\mathcal{X}$)
$\mathcal{D}(A)$	the probability of a set $A \subseteq Z$ according to $\mathcal{D}$
$z \sim \mathcal{D}$	sampling z according to $\mathcal{D}$
$S = z_1, \ldots, z_m$	a sequence of m examples
$S \sim \mathcal{D}^m$	sampling $S = z_1, \ldots, z_m$ i.i.d. according to $\mathcal{D}$
$\mathbb{P}, \mathbb{E}$	probability and expectation of a random variable
$\mathbb{P}_{z \sim \mathcal{D}}[f(z)]$	$= \mathcal{D}(\{z : f(z) = \texttt{true}\})$ for $f : Z \to \{\texttt{true}, \texttt{false}\}$
$\mathbb{E}_{z \sim \mathcal{D}}[f(z)]$	expectation of the random variable $f : Z \to \mathbb{R}$
$N(\boldsymbol{\mu}, C)$	Gaussian distribution with expectation $\boldsymbol{\mu}$ and covariance C
$f'(x)$	the derivative of a function $f : \mathbb{R} \to \mathbb{R}$ at x
$f''(x)$	the second derivative of a function $f : \mathbb{R} \to \mathbb{R}$ at x
$\frac{\partial f(\mathbf{w})}{\partial w_i}$	the partial derivative of a function $f : \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{w}$ w.r.t. w_i
$\nabla f(\mathbf{w})$	the gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{w}$
$\partial f(\mathbf{w})$	the differential set of a function $f : \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{w}$
$\min_{x \in C} f(x)$	$= \min\{f(x) : x \in C\}$ (minimal value of f over C)
$\max_{x \in C} f(x)$	$= \max\{f(x) : x \in C\}$ (maximal value of f over C)
$\operatorname{argmin}_{x \in C} f(x)$	the set $\{x \in C : f(x) = \min_{z \in C} f(z)\}$
$\operatorname{argmax}_{x \in C} f(x)$	the set $\{x \in C : f(x) = \max_{z \in C} f(z)\}$
log	the natural logarithm

x_0 such that for all $x > x_0$ we have $f(x) \leq \alpha g(x)$. We write $f = \Omega(g)$ if there exist $x_0, \alpha \in \mathbb{R}_+$ such that for all $x > x_0$ we have $f(x) \geq \alpha g(x)$. The notation $f = \omega(g)$ is defined analogously. The notation $f = \Theta(g)$ means that $f = O(g)$ and $g = O(f)$. Finally, the notation $f = \tilde{O}(g)$ means that there exists $k \in \mathbb{N}$ such that $f(x) = O(g(x) \log^k(g(x)))$.

The inner product between vectors $\mathbf{x}$ and $\mathbf{w}$ is denoted by $\langle \mathbf{x}, \mathbf{w} \rangle$. Whenever we do not specify the vector space we assume that it is the d-dimensional Euclidean space and then $\langle \mathbf{x}, \mathbf{w} \rangle = \sum_{i=1}^{d} x_i w_i$. The Euclidean (or ℓ_2) norm of a vector $\mathbf{w}$ is $\|\mathbf{w}\|_2 = \sqrt{\langle \mathbf{w}, \mathbf{w} \rangle}$. We omit the subscript from the ℓ_2 norm when it is clear from the context. We also use other ℓ_p norms, $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}$, and in particular $\|\mathbf{w}\|_1 = \sum_i |w_i|$ and $\|\mathbf{w}\|_\infty = \max_i |w_i|$.

We use the notation $\min_{x \in C} f(x)$ to denote the minimum value of the set $\{f(x) : x \in C\}$. To be mathematically more precise, we should use $\inf_{x \in C} f(x)$ whenever the minimum is not achievable. However, in the context of this book the distinction between infimum and minimum is often of little interest. Hence, to simplify the presentation, we sometimes use the min notation even when inf is more adequate. An analogous remark applies to max versus sup.