1 *Learning, Regularity, and Compression*

Overview The task of inductive inference is to find laws or regularities underlying some given set of data. These laws are then used to gain insight into the data or to classify or predict future data. The minimum description length (MDL) principle is a general method for inductive inference, based on the idea that the more we are able to *compress* (describe in a compact manner) a set of data, the more regularities we have found in it and therefore, the more we have *learned* from the data. In this chapter we give a first, preliminary and informal introduction to this principle.

Contents In Sections 1.1 and 1.2 we discuss some of the fundamental ideas relating description length and regularity. In Section 1.3 we describe what was historically the first attempt to formalize these ideas. In Section 1.4 we explain the problems with using the original formalization in practice, and indicate what must be done to make the ideas practicable. Section 1.5 introduces the practical forms of MDL we deal with in this book, as well as the crucial concept of "universal coding." Section 1.6 deals with some issues concerning *model selection*, which is one of the main MDL applications. The philosophy underlying MDL is discussed in Section 1.7. Section 1.8 shows how the ideas behind MDL are related to "Occam's razor." We end in Section 1.9 with a brief historical overview of the field and its literature.

Fast Track This chapter discusses, in an informal manner, several of the complicated issues we will deal with in this book. It is therefore essential for readers without prior exposure to MDL. Readers who are familiar with the basic ideas behind MDL may just want to look at the boxes.

1.1 Regularity and Learning

We are interested in developing a method for *learning* the laws and regularities in data. The following example will illustrate what we mean by this and give a first idea of how it can be related to descriptions of data.

Example 1.1 We start by considering binary data. Consider the following three sequences. We assume that each sequence is 10000 bits long, and we just list the beginning and the end of each sequence.

```
00010001000100010001...0001000100010001000100010001    (1.1)
01110100110100100110...1010111010111011000101100010    (1.2)
00011000001010100000...0010001000010000001000110000    (1.3)
```

The first of these three sequences is a 2500-fold repetition of `0001`. Intuitively, the sequence looks regular; there seems to be a simple "law" underlying it; it might make sense to conjecture that future data will also be subject to this law, and to predict that future data will behave according to this law. The second sequence has been generated by tosses of a fair coin. It is, intuitively speaking, as "random as possible," and in this sense there is no regularity underlying it.[1] Indeed, we cannot seem to find such a regularity either when we look at the data. The third sequence contains exactly four times as many 0s as 1s. It looks less regular, more random than the first; but it looks less random than the second. There is still some discernible regularity in these data, but of a statistical rather than of a deterministic kind. Again, noticing that such a regularity is there and predicting that future data will behave according to the same regularity seems sensible.

1.2 Regularity and Compression

What do we mean by a "regularity"? The fundamental idea behind the MDL principle is the following insight: every regularity in the data can be used to *compress* the data, i.e. to describe it using fewer symbols than the number of symbols needed to describe the data literally. Such a description should always uniquely specify the data it describes - hence given a description or

1. Unless we call "generated by a fair coin toss" a "regularity" too. There is nothing wrong with that view - the point is that, the *more* we can compress a sequence, the *more* regularity we have found. One can avoid all terminological confusion about the concept of "regularity" by making it *relative* to something called a "base measure," but that is beyond the scope of this book (Li and Vitányi 1997).

encoding D' of a particular sequence of data D, we should always be able to fully reconstruct D using D'.

For example, sequence (1.1) above can be described using only a few words; we have actually done so already: we have not given the complete sequence — which would have taken about the whole page — but rather just a one-sentence description of it that nevertheless allows you to reproduce the complete sequence if necessary. Of course, the description was done using natural language and we may want to do it in some more formal manner.

If we want to identify regularity with compressibility, then it should also be the case that nonregular sequences can *not* be compressed. Since sequence (1.2) has been generated by fair coin tosses, it should not be compressible. As we will show below, we can indeed prove that *whatever* description method C one uses, the length of the description of a sequence like (1.2) will, with overwhelming probability, be not much shorter than sequence (1.2) itself.

Note that the description of sequence (1.3) that we gave above does not uniquely define sequence (1.3). Therefore, it does not count as a "real" description: one cannot regenerate the whole sequence if one has the description. A unique description that still takes only a few words may look like this: "Sequence (1.3) is one of those sequences of 10000 bits in which there are four times as many 0s as there are 1s. In the lexicographical ordering of those sequences, it is number i." Here i is some large number that is explicitly spelled out in the description. In general, there are 2^n binary sequences of length n, while there are only $\binom{n}{\nu n}$ sequences of length n with a fraction of ν 1s. For every rational number ν except $\nu = 1/2$, the ratio of $\binom{n}{\nu n}$ to 2^n goes to 0 exponentially fast as n increases (this is shown formally in Chapter 4; see Equation (4.36) on page 129 and the text thereunder; by the method used there one can also show that for $\nu = 1/2$, it goes to 0 as $O(1/\sqrt{n})$). It follows that compared to the total number of binary sequences of length 10000, the number of sequences of length 10000 with four times as many 0s as 1s is vanishingly small. Direct computation shows it is smaller than 2^{7213}, so that the ratio between the number of sequences with four times as many 0s than 1s and the total number of sequences is smaller than 2^{-2787}. Thus, $i < 2^{7213} \ll 2^{10000}$ and to write down i in binary we need approximately $(\log_2 i) < 7213 \ll 10000$ bits.

Example 1.2 [Compressing Various Regular Sequences] The regularities underlying sequences (1) and (3) were of a very particular kind. To illustrate that *any* type of regularity in a sequence may be exploited to compress that sequence, we give a few more examples:

The Number π Evidently, there exists a computer program for generating the first n digits of π – such a program could be based, for example, on an infinite series expansion of π. This computer program has constant size, except for the specification of n which takes no more than $O(\log n)$ bits. Thus, when n is very large, the size of the program generating the first n digits of π will be very small compared to n: the π-digit sequence is deterministic, and therefore extremely regular.

Physics Data Consider a two-column table where the first column contains numbers representing various heights from which an object was dropped. The second column contains the corresponding times it took for the object to reach the ground. Assume both heights and times are recorded to some finite precision. In Section 1.5 we illustrate that such a table can be substantially compressed by first describing the coefficients of the second-degree polynomial H that expresses Newton's law; then describing the heights; and then describing the deviation of the time points from the numbers predicted by H.

Natural Language Most sequences of words are not valid sentences according to the English language. This fact can be exploited to substantially compress English text, as long as it is syntactically mostly correct: by first describing a grammar for English, and then describing an English text D with the help of that grammar (Grünwald 1996), D can be described using much less bits than are needed without the assumption that word order is constrained.

Description Methods In order to formalize our idea, we have to replace the part of the descriptions above that made use of natural language by some formal language. For this, we need to fix a *description method* that maps sequences of data to their descriptions. Each such sequence will be encoded as another sequence of symbols coming from some finite or countably infinite *coding alphabet*. An *alphabet* is simply a countable set of distinct symbols. An example of an alphabet is the binary alphabet $\mathbb{B} = \{0, 1\}$; the three data sequences above are sequences over the binary alphabet. A sequence over a binary alphabet will also be called a binary *string*. Sometimes our data will consist of real numbers rather than binary strings. In practice, however, such numbers are always truncated to some finite precision. We can then again model them as symbols coming from a finite data alphabet.

More precisely, we are given a *sample* or equivalently *data sequence* $D = (x_1, \ldots, x_n)$ where each x_i is a member of some set $\mathcal{X}$, called the *space of observations* or the *sample space for one observation*. The set of all potential samples of length n is denoted $\mathcal{X}^n$ and is called the *sample space*. We call

x_i a single *observation* or, equivalently, a *data item*. For a general note about how our terminology relates to the usual terminology in statistics, machine learning and pattern recognition, we refer to the box on page 72.

Without any loss of generality we may describe our data sequences as binary strings (this is explained in Chapter 3, Section 3.2.2). Hence all the description methods we consider map data sequences to sequences of bits. All description methods considered in MDL satisfy the *unique decodability property*: given a description D', there is at most one ("unique") D that is encoded as D'. Therefore, given any description D', one should be able to fully reconstruct the original sequence D. Semiformally:

Description Methods

Definition 1.1 A *description method* is a *one-many* relation from the sample space to the set of binary strings of arbitrary length.

A truly formal definition will be given in Chapter 3, Section 3.1. There we also explain how our notion of "description method" relates to the more common and closely related notion of a "code." Until then, the distinction between codes an description methods is not that important, and we use the symbol C to denote both concepts.

Compression and Small Subsets We are now in a position to show that strings which are "intuitively" random cannot be substantially compressed. We equate intuitively random with "having been generated by independent tosses of a fair coin." We therefore have to prove that it is virtually impossible to substantially compress sequences that have been generated by fair coin tosses. By "it is virtually impossible" we mean "it happens with vanishing probability." Let us take some arbitrary but fixed description method C over the data alphabet consisting of the set of all binary sequences of length ≥ 1. Such a code maps binary strings to binary strings. Suppose we are given a data sequence of length n (in Example 1.1, $n = 10000$). Clearly, there are 2^n possible data sequences of length n. We see that only two of these can be mapped to a description of length 1 (since there are only two binary strings of length 1: 0 and 1). Similarly, only a subset of at most 2^m sequences can have a description of length m. This means that at most $\sum_{i=1}^{m} 2^i < 2^{m+1}$ data sequences can have a description length $\leq m$. The fraction of data sequences of length n that can be compressed by more than k bits is therefore at most 2^{-k} and as such decreases exponentially in k. If data are generated by n tosses of a fair coin, then all 2^n possibilities for the data are equally probable, so the probability that we can compress the data by more than k bits is smaller than 2^{-k}. For example, the probability that we can compress the data by more than 20 bits is smaller than one in a million.

Most Data Sets Are Incompressible
Suppose our goal is to encode a binary sequence of length n. Then

- No matter what description method we use, only a fraction of at most 2^{-k} sequences can be compressed by more than k bits.
- Thus, if data are generated by fair coin tosses, then no matter what code we use, the probability that we can compress a sequence by more than k bits is at most 2^{-k}.
- This observation will be generalized to data generated by an arbitrary distribution in Chapter 3. We then call it the *no-hypercompression inequality*. It can be found in the box on page 103.

Seen in this light, having a short description length for the data is equivalent to identifying the data as belonging to a tiny, very *special* subset out of all a priori possible data sequences; see also the box on page 31.

1.3 Solomonoff's Breakthrough – Kolmogorov Complexity

It seems that what data are compressible and what are not is extremely dependent on the specific description method used. In 1964 – in a pioneering paper that may be regarded as the starting point of all MDL-related research (Solomonoff 1964) – Ray Solomonoff suggested the use of a *universal computer language* as a description method. By a universal language we mean a computer language in which a universal Turing machine can be implemented. All commonly used computer languages, like Pascal, LISP, C, are "universal." Every data sequence D can be encoded by a computer program P that prints D and then halts. We can define a description method that maps each data sequence D to the *shortest program* that prints D and then halts.[2] Clearly, this is a description method in our sense of the word in that it defines a 1-many (even 1-1) mapping from sequences over the data alphabet to a subset of the binary sequences.

The shortest program for a sequence D is then interpreted as the *optimal hypothesis* for D. Let us see how this works for sequence (1.1) above. Using a language similar to C, we can write a program

```
for i = 1 to 2500 ; do {print '0001'} ; halt
```

which prints sequence (1.1) but is clearly a lot shorter than it. If we want to make a fair comparison, we should rewrite this program in a binary alphabet; the resulting number of bits is still much smaller than 10000. The shortest program printing sequence (1.1) is at least as short as the program above, which means that sequence (1.1) is indeed highly compressible using Solomonoff's code. By the arguments of the previous section we see that, given an arbitrary description method C, sequences like (1.2) that have been generated by tosses of a fair coin are very likely not substantially compressible using C. In other words, the shortest program for sequence (1.1) is, with extremely high probability, not much shorter than the following:

```
print '01110100110100001010........10111011000101100010'; halt
```

This program has size about equal to the length of the sequence. Clearly, it is nothing more than a repetition of the sequence.

Kolmogorov Complexity We define the *Kolmogorov complexity* of a sequence as the length of the shortest program that prints the sequence and then halts. Kolmogorov complexity has become a large subject in its own right; see (Li and Vitányi 1997) for a comprehensive introduction.

The lower the Kolmogorov complexity of a sequence, the *more regular* or equivalently, the *less random*, or, yet equivalently, the *simpler* it is. Measuring regularity in this way confronts us with a problem, since it depends on the particular programming language used. However, in his 1964 paper, Ray Solomonoff (Solomonoff 1964) showed that *asymptotically* it does not matter what programming language one uses, as long as it is universal: for every sequence of data $D = (x_1, \ldots, x_n)$, let us denote by $L_{\text{UL}}(D)$ the length of the shortest program for D using universal language UL. We can show that for

2. If there exists more than one shortest program, we pick the one that comes first in enumeration order.

every two universal languages UL_1 and UL_2, the difference between the two lengths $L_{\text{UL}_1}(D) - L_{\text{UL}_2}(D)$ is bounded by a constant that depends on UL_1 and UL_2 but not on the length n of the data sequence D. This implies that if we have a lot of data (n is large), then the difference in the two description lengths is negligible compared to the size of the data sequence. This result is known as the *invariance theorem* and was proved independently in (Solomonoff 1964), (Kolmogorov 1965) (hence the name Kolmogorov complexity), and (Chaitin 1969). The proof is based on the fact that one can write a compiler for every universal language UL_1 in every other universal language UL_2. Such a compiler is a computer program with length $L_{1\to 2}$. For example, we can write a program in Pascal that translates every C program into an equivalent Pascal program. The length (in bits) of this program would then be $L_{\text{C}\to\text{Pascal}}$. We can simulate each program P_1 written in language UL_1 by program P_2 written in UL_2 as follows: P_2 consists of the compiler from UL_1 to UL_2, followed by P_1. The length of program P_2 is bounded by the length of P_1 plus $L_{1\to 2}$. Hence for all data D, the maximal difference between $L_{\text{UL}_1}(D)$ and $L_{\text{UL}_2}(D)$ is bounded by $\max\{L_{1\to 2}, L_{2\to 1}\}$, a constant which only depends on UL_1 and UL_2 but not on D.

1.4 Making the Idea Applicable

Problems There are two major problems with applying Kolmogorov complexity to practical learning problems:

1. **Uncomputability.** The Kolmogorov complexity cannot be computed in general;

2. **Large constants.** The description length of any sequence of data involves a constant depending on the description method used.

By "Kolmogorov complexity cannot be computed" we mean the following: there is no computer program that, for every sequence of data D, when given D as input, returns the shortest program that prints D and halts. Neither can there be a program, that for every data D returns only the *length* of the shortest program that prints D and then halts. Assuming such a program exists leads to a contradiction (Li and Vitányi 1997).

The second problem relates to the fact that in many realistic settings, we are confronted with very small data sequences for which the invariance theorem is not very relevant since the length of D is small compared to the constant $L_{1\to 2}$.

"Idealized" or "Algorithmic" MDL If we ignore these problems, we may use Kolmogorov complexity as our fundamental concept and build a theory of idealized inductive inference on top of it. This road has been taken by Solomonoff (1964, 1978), starting with the 1964 paper in which he introduced Kolmogorov complexity, and by Kolmogorov, when he introduced the *Kolmogorov minimum sufficient statistic* (Li and Vitányi 1997; Cover and Thomas 1991). Both Solomonoff's and Kolmogorov's ideas have been substantially refined by several authors. We mention here P. Vitányi (Li and Vitányi 1997; Gács, Tromp, and Vitányi 2001; Vereshchagin and Vitányi 2002; Vereshchagin and Vitányi 2004; Vitányi 2005), who concentrated on Kolmogorov's ideas, and M. Hutter (2004), who concentrated on Solomonoff's ideas. Different authors have used different names for this area of research: "ideal MDL," "idealized MDL," or "algorithmic statistics." It is closely related to the celebrated theory of *random sequences* due to P. Martin-Löf and Kolmogorov (Li and Vitányi 1997). We briefly return to idealized MDL in Chapter 17, Section 17.8.

Practical MDL Like most authors in the field, we concentrate here on non-idealized, practical versions of MDL that explicitly deal with the two problems mentioned above. The basic idea is to scale down Solomonoff's approach so that it does become applicable. This is achieved by using description methods that are less expressive than general-purpose computer languages. Such description methods C should be restrictive enough so that for any data sequence D, we can always compute the length of the shortest description of D that is attainable using method C; but they should be general enough to allow us to compress many of the intuitively "regular" sequences. The price we pay is that, using the "practical" MDL principle, there will always be some regular sequences which we will not be able to compress. But we already know that there can be *no* method for inductive inference at all which will always give us all the regularity there is — simply because there can be no automated method which for any sequence D finds the shortest computer program that prints D and then halts. Moreover, it will often be possible to guide a suitable choice of C by a priori knowledge we have about our problem domain. For example, below we consider a description method C that is based on the class of all polynomials, such that with the help of C we can compress all data sets which can meaningfully be seen as points on some polynomial.

1.5 Crude MDL, Refined MDL and Universal Coding

Let us recapitulate our main insights so far:

MDL: The Basic Idea
The goal of statistical inference may be cast as trying to find regularity in the data. "Regularity" may be identified with "ability to compress." MDL combines these two insights by *viewing learning as data compression*: it tells us that, for a given set of hypotheses $\mathcal{H}$ and data set D, we should try to find the hypothesis or combination of hypotheses in $\mathcal{H}$ that compresses D most.

This idea can be applied to all sorts of inductive inference problems, but it turns out to be most fruitful in (and its development has mostly concentrated on) problems of *model selection* and, more generally, dealing with *overfitting*. Here is a standard example (we explain the difference between "model" and "hypothesis" after the example).

Example 1.3 [Model Selection and Overfitting] Consider the points in Figure 1.1. We would like to learn how the y-values depend on the x-values. To this end, we may want to fit a polynomial to the points. Straightforward linear regression will give us the leftmost polynomial - a straight line that seems overly simple: it does not capture the regularities in the data well. Since for any set of n points there exists a polynomial of the $(n-1)$st degree that goes exactly through all these points, simply looking for the polynomial with the least error will give us a polynomial like the one in the second picture. This polynomial seems overly complex: it reflects the random fluctuations in the data rather than the general pattern underlying it. Instead of picking the overly simple or the overly complex polynomial, it seems more reasonable to prefer a relatively simple polynomial with small but nonzero error, as in the rightmost picture. This intuition is confirmed by numerous experiments on real-world data from a broad variety of sources (Rissanen 1989; Vapnik 1998; Ripley 1996): if one naively fits a high-degree polynomial to a small sample (set of data points), then one obtains a very good fit to the data. Yet if one *tests* the inferred polynomial on a second set of data coming from the same source, it typically fits this test data very badly in the sense that there is a large distance between the polynomial and the new data points. We say that the polynomial *overfits* the data. Indeed, all model selection methods that are used in practice either implicitly or explicitly choose

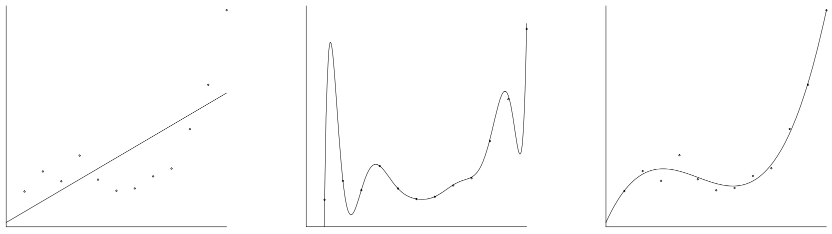

Figure 1.1 A simple, a complex and a tradeoff (third-degree) polynomial.

a tradeoff between goodness-of-fit and complexity of the models involved. In practice, such tradeoffs lead to much better predictions of test data than one would get by adopting the "simplest" (one degree) or most "complex"[3] ($n - 1$-degree) polynomial. MDL provides one particular means of achieving such a tradeoff.

It will be useful to distinguish between "model", "model class" and "(point) hypothesis." This terminology is explained in the box on page 15, and will be discussed in more detail in Section 2.4, page 69. In our terminology, the problem described in Example 1.3 is a "point hypothesis selection problem" if we are interested in selecting both the degree of a polynomial and the corresponding parameters; it is a "model selection problem" if we are mainly interested in selecting the degree.

To apply MDL to polynomial or other types of hypothesis and model selection, we have to make precise the somewhat vague insight "learning may be viewed as data compression." This can be done in various ways. We first explain the earliest and simplest implementation of the idea. This is the so-called *two-part code* version of MDL:

3. Strictly speaking, in our context it is not very accurate to speak of "simple" or "complex" polynomials; instead we should call the *set* of first degree polynomials "simple," and the *set* of 100th-degree polynomials "complex."

Crude Two-Part Version of MDL Principle (Informally Stated)
Let $\mathcal{H}_1, \mathcal{H}_2, \ldots$ be a list of candidate models (e.g., $\mathcal{H}_\gamma$ is the set of γth degree polynomials), each containing a set of point hypotheses (e.g., individual polynomials). The best point hypothesis $H \in \mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cup \ldots$ to explain the data D is the one which minimizes the sum $L(H)+L(D|H)$, where

- $L(H)$ is the length, in bits, of the description of the hypothesis; and
- $L(D|H)$ is the length, in bits, of the description of the data when encoded with the help of the hypothesis.

The best *model* to explain D is the smallest model containing the selected H.

The terminology "crude MDL" is explained in the next subsection. It is not standard, and it is introduced here for pedagogical reasons.

Example 1.4 [Polynomials, cont.] In our previous example, the candidate hypotheses were polynomials. We can describe a polynomial by describing its coefficients at a certain precision (number of bits per parameter). Thus, the higher the degree of a polynomial or the precision, the more bits we need to describe it and the more "complex" it becomes. A description of the data "with the help of" a hypothesis means that the better the hypothesis fits the data, the shorter the description will be. A hypothesis that fits the data well gives us a lot of *information* about the data. Such information can always be used to compress the data. Intuitively, this is because we only have to code the *errors* the hypothesis makes on the data rather than the full data. In our polynomial example, the better a polynomial H fits D, the fewer bits we need to encode the discrepancies between the actual y-values y_i and the predicted y-values $H(x_i)$. We can typically find a very complex point hypothesis (large $L(H)$) with a very good fit (small $L(D|H)$). We can also typically find a very simple point hypothesis (small $L(H)$) with a rather bad fit (large $L(D|H)$). The sum of the two description lengths will be minimized at a hypothesis that is quite (but not too) "simple," with a good (but not perfect) fit.

1.5.1 From Crude to Refined MDL

Crude MDL picks the H minimizing the sum $L(H) + L(D|H)$. To make this procedure well defined, we need to agree on precise definitions for the

Models and Model Classes; (Point) Hypotheses
We use the word *model* to refer to a *set* of probability distributions or functions of the same functional form. E.g., the “first-order Markov model” is the set of all probability distributions that are first-order Markov chains. The “model of kth degree polynomials” is the set of all kth degree polynomials for some fixed k.
We use the word *model class* to refer to a family (set) of models, e.g. “the model class of all polynomials” or “the model class of all Markov chains of each order.” The definitions of “model” and “model class” are chosen so that they agree with how these words are used in statistical practice. Therefore they are intentionally left somewhat imprecise.
We use the word *hypothesis* to refer to an *arbitrary* set of probability distributions or functions. We use the word *point hypothesis* to refer to a *single* probability distribution (e.g. a Markov chain with all parameter values specified) or function (e.g. a particular polynomial). In parametric inference (Chapter 2), a point hypothesis corresponds to a particular parameter value. A point hypothesis may also be viewed as an *instantiation* of a model.

What we call “point hypothesis” is called “*simple* hypothesis” in the statistics literature; our use of the word “model (selection)” coincides with its use in much of the statistics literature; see Section 2.3, page 62 where we give several examples to clarify our terminology.

Figure 1.2 Models and Model Classes; (Point) Hypotheses.

codes (description methods) giving rise to lengths $L(D|H)$ and $L(H)$. We now discuss these codes in more detail. We will see that the definition of $L(H)$ is problematic, indicating that we somehow need to “refine” our crude MDL principle.

Definition of $\boldsymbol{L(D|H)}$ Consider a two-part code as described above, and assume for the time being that all H under consideration define probability distributions. If H is a polynomial, we can turn it into a distribution by making the additional assumption that the Y-values are given by $Y = H(X) + Z$, where Z is a normally distributed noise term with mean 0.

For each H we need to define a code with length $L(\cdot \mid H)$ such that $L(D|H)$ can be interpreted as "the codelength of D when encoded with the help of H." It turns out that for probabilistic hypotheses, there is only one reasonable choice for this code; this is explained at length in Chapter 5. It it is the so-called *Shannon-Fano code*, satisfying, for all data sequences D, $L(D|H) = -\log P(D|H)$, where $P(D|H)$ is the probability mass or density of D according to H. Such a code always exists, as we explain in Chapter 3, in the box on page 96.

Definition of $L(H)$: A Problem for Crude MDL It is more problematic to find a good code for hypotheses H. Some authors have simply used "intuitively reasonable" codes in the past, but this is not satisfactory: since the description length $L(H)$ of any fixed point hypothesis H can be very large under one code, but quite short under another, our procedure is in danger of becoming arbitrary. Instead, *we need some additional principle for designing a code for $\mathcal{H}$.*

In the first publications on MDL (Rissanen 1978; Rissanen 1983), it was implicitly advocated to choose some sort of *minimax code* for each $\mathcal{H}_\gamma$, minimizing the shortest worst-case total description length $L(H) + L(D|H)$, where the worst-case is over all possible data sequences. Thus, the MDL principle is employed at a "meta-level" to choose a code for $\mathcal{H}_\gamma$. This idea, already implicit in Rissanen's early work abut perhaps for the first time stated and formalized in a completely precise way Barron and Cover (1991), is the first step towards "refined" MDL.

More Problems for Crude MDL We can use crude MDL to code any sequence of data D with a total description length $L(D) := \min_H \{L(D|H) + L(H)\}$. But it turns out that this code is *incomplete*: one can show that there exist other codes L' which for some D achieve strictly smaller codelength $(L'(D) < L(D))$, and for no D achieve larger codelength (Chapter 6, Example 6.4). It seems strange that our "minimum description length" principle should be based on codes which are incomplete (inefficient) in this sense. Another, less fundamental problem with two-part codes is that, if designed in a minimax way as indicated above, they require a cumbersome discretization of the model space $\mathcal{H}$, which is not always feasible in practice. The final problem we mention is that, while it is clear how to use crude two-part codes for point hypothesis and model selection, it is not immediately clear how they can be used for *prediction*.

Later, Rissanen (1984) realized that these problems could be side-stepped by using *one-part* rather than *two-part codes*. As we explain below, it depends on the situation at hand whether a one-part or a two-part code should be used. Combining the idea of designing codes so as to achieve essentially minimax optimal codelengths with the combined use of one-part and two-part codes (whichever is appropriate for the situation at hand) has culminated in a theory of inductive inference that we call *refined MDL*. We discuss it in more detail in the next subsection.

Crude Two-Part MDL (Part I, Chapter 5 of this book)
In this book, we use the term "crude MDL" to refer to applications of MDL for model and hypothesis selection of the type described in the box on page 14, as long as the hypotheses $H \in \mathcal{H}$ are encoded in "intuitively reasonable" but ad-hoc ways.
Refined MDL is sometimes based on one-part codes, sometimes on two-part codes, and sometimes on a combination of these, but, in contrast to crude MDL, the codes are invariably designed according to some minimax principles. If there is a choice, one should always prefer refined MDL, but in some exotic modeling situations, the use of crude MDL is inevitable.
Part I of this book first discusses all probabilistic, statistical and information-theoretic preliminaries (Chapters 2–4) and culminates in a description of crude two-part MDL (Chapter 5). Refined MDL is described only in Part III.

1.5.2 Universal Coding and Refined MDL

In refined MDL, we associate a code for encoding D *not with a single* $H \in \mathcal{H}$, but with the full model $\mathcal{H}$. Thus, given model $\mathcal{H}$, we encode data not in two parts but we design a single *one-part code* with lengths $\bar{L}(D|\mathcal{H})$. This code is designed such that *whenever there is a member of (parameter in)* $\mathcal{H}$ *that fits the data well, in the sense that* $L(D \mid H)$ *is small, then the codelength* $\bar{L}(D|\mathcal{H})$ *will also be small*. Codes with this property are called *universal codes* in the information-theoretic literature (Barron, Rissanen, and Yu 1998):

Universal Coding (Part II of This Book)
There exist at least four types of universal codes:

1. The normalized maximum likelihood (NML) code and its variations.
2. The Bayesian mixture code and its variations.
3. The prequential plug-in code
4. The two-part code

These codes are all based on entirely different coding schemes, but in practice, lead to very similar codelengths $\bar{L}(D|\mathcal{H})$. Part II of this book is entirely devoted to universal coding. The four types of codes are introduced in Chapter 6. This is follows by a separate chapter for each code.

For each model $\mathcal{H}$, there are many different universal codes we can associate with $\mathcal{H}$. When applying MDL, we have a preference for the one that is *minimax optimal* in a sense made precise in Chapter 6. For example, the set $\mathcal{H}_3$ of third-degree polynomials is associated with a code with lengths $\bar{L}(\cdot \mid \mathcal{H}_3)$ such that, the better the data D are fit by the best-fitting third-degree polynomial, the shorter the codelength $\bar{L}(D \mid \mathcal{H})$. $\bar{L}(D \mid \mathcal{H})$ is called the *stochastic complexity* of the data given the model.

Refined MDL is a general theory of inductive inference based on universal codes that are designed to be minimax, or close to minimax optimal. It has mostly been developed for model selection, estimation and prediction. To give a first flavor, we initially discuss model selection, where, arguably, it has the most new insights to offer:

1.5.3 Refined MDL for Model Selection

Parametric Complexity A fundamental concept of refined MDL for model selection is the *parametric complexity* of a parametric model $\mathcal{H}$ which we denote by $\text{COMP}(\mathcal{H})$. This is a measure of the "richness" of model $\mathcal{H}$, indicating its ability to fit random data. This complexity is related to the number of degrees-of-freedom (parameters) in $\mathcal{H}$, but also to the geometrical structure of $\mathcal{H}$; see Example 1.5. To see how it relates to stochastic complexity, let, for given data D, $\hat{H}$ denote the distribution in $\mathcal{H}$ which maximizes the probability, and hence minimizes the codelength $L(D \mid \hat{H})$ of D. It turns out that

$$\bar{L}(D \mid \mathcal{H}) = \text{stochastic complexity of } D \text{ given } \mathcal{H} = L(D \mid \hat{H}) + \text{COMP}(\mathcal{H}).$$

Refined MDL model selection between two parametric models $\mathcal{H}_1$ and $\mathcal{H}_2$ (such as the models of first and second degree polynomials) now proceeds as follows. We encode data D in two stages. In the first stage, we encode a number $j \in \{1, 2\}$. In the second stage, we encode the data using the universal code with lengths $\bar{L}(D \mid \mathcal{H}_j)$. As in the two-part code principle, we then select the $\mathcal{M}_j$ achieving the minimum total two-part codelength,

$$\min_{j \in \{1,2\}} \{L(j) + \bar{L}(D \mid \mathcal{H}_j)\} = \min_{j \in \{1,2\}} \{L(j) + L(D \mid \hat{H}) + \text{COMP}(\mathcal{H})\}. \quad (1.4)$$

Since the worst-case optimal code to encode j needs only 1 bit to encode either $j = 1$ or $j = 2$, we use a code for the first-part such that $L(1) = L(2) = 1$. But this means that $L(j)$ plays no role in the minimization, and we are effectively selecting the model such that the stochastic complexity of the given data D is smallest.[4] Thus, in the end we select the model *minimizing the one-part codelength of the data*. Nevertheless, refined MDL model selection involves a tradeoff between two terms: a goodness-of-fit term $L(D \mid \hat{H})$ and a complexity term COMP$(\mathcal{H})$. However, because we do not explicitly encode hypotheses H anymore, there is no potential for arbitrary codelengths anymore. The resulting procedure can be interpreted in several different ways, some of which provide us with rationales for MDL model selection beyond the pure coding interpretation (Chapter 14):

1. **Counting/differential geometric interpretation** The parametric complexity of a model is the logarithm of the number of *essentially different*, *distinguishable* point hypotheses within the model.

2. **Two-part code interpretation** For large samples, the stochastic complexity can be interpreted as a two-part codelength of the data after all, where hypotheses H are encoded with a special code that works by first discretizing the model space $\mathcal{H}$ into a set of "maximally distinguishable hypotheses," and then assigning equal codelength to each of these.

3. **Bayesian interpretation** In many cases, refined MDL model selection coincides with Bayes factor model selection based on a *noninformative prior* such as *Jeffreys' prior* (Bernardo and Smith 1994).

4. The reason we include $L(j)$ at all in (1.4) is to maintain consistency with the case where we need to select between an infinite number of models. In that case, it is necessary to include $L(j)$.

4. Prequential interpretation MDL model selection can be interpreted as selecting the model with the best predictive performance when sequentially predicting *unseen* test data, in the sense described in Chapter 6, Section 6.4 and Chapter 9. This makes it an instance of Dawid's (1984) *prequential* model validation and also relates it to *cross-validation* methods; see Chapter 17, Sections 17.5 and 17.6.

In Section 1.6.1 we show that refined MDL allows us to compare models of different functional form. It even accounts for the phenomenon that different models with the same number of parameters may not be equally "complex."

1.5.4 General Refined MDL: Prediction and Hypothesis Selection

Model selection is just one application of refined MDL. The two other main applications are *point hypothesis selection* and *prediction*. These applications can also be interpreted as methods for parametric and nonparametric *estimation*. In fact,it turns out that large parts of MDL theory can be reinterpreted as a theory about *sequential prediction of future data given previously seen data*. This "prequential" interpretation of MDL (Chapter 15) is at least as important as the coding interpretation. It is based on the fundamental correspondence between probability distributions and codes via the Shannon-Fano code that we alluded to before, when explaining the code with lengths $L(D \mid H)$; see the box on page 96. This correspondence allows us to view any universal code $\bar{L}(\cdot \mid \mathcal{H})$ as a strategy for sequentially predicting data, such that the better $\mathcal{H}$ is suited as a model for the data, the better the predictions will be.

MDL prediction and hypothesis selection are mathematically cleaner than MDL model selection: in Chapter 15, we provide theorems (Theorem 15.1 and Theorem 15.3) which, in the respective contexts of prediction and hypothesis selection, express that, in full generality, *good data compression implies fast learning*, where "learning" is defined as "finding a hypothesis that is in some sense close to an imagined "true state of the world." There are similar theorems for model selection, but these lack some of the simplicity and elegance of Theorem 15.1 and Theorem 15.3.

Probabilistic vs. Nonprobabilistic MDL Like most other authors on MDL, in this book we confine ourselves to *probabilistic hypotheses*, also known as *probabilistic sources*. These are hypotheses that take the form of *probability distributions* over the space of possible data sequences. The examples we give in this chapter (Examples 1.3 and 1.5) involve hypotheses H that are functions from some space $\mathcal{X}$ to another space $\mathcal{Y}$; at first sight, these are not "probabilistic." We will usually assume that for any given x, we have $y = H(x) + Z$ where Z is a *noise term* with a known distribution. Typically, the noise Z will be assumed to be Gaussian (normally) distributed. With such an additional assumption, we may view "functional" hypotheses $H : \mathcal{X} \to \mathcal{Y}$ as "probabilistic" after all. Such a technique of turning functions into probability distributions is customary in statistics, and we will use it throughout large parts of this book. Whenever we refer to MDL, we implicitly assume that we deal with probabilistic models. We should note though that there exists variations of MDL that *directly* work with universal codes relative to functional hypotheses such as polynomials (see Section 1.9.1, and Chapter 17, Section 17.10).

Fixing Notation
We use the symbol H for general point hypotheses, that may either represent a probabilistic source or a deterministic function. We use $\mathcal{H}$ for sets of such general point hypotheses. We reserve the symbol $\mathcal{M}$ for probabilistic models and model classes. We denote probabilistic point hypotheses by P, and point hypotheses that are deterministic functions by h.

Individual-Sequence vs. Expectation-based MDL Refined MDL is based on minimax optimal universal codes. Broadly speaking, there are two different ways to define what we mean by minimax optimality. One is to look at the worst-case codelength over *all possible sequences*. We call this *individual-sequence MDL*. An alternative is to look at *expected codelength*, where the expectation is taken over some probability distribution, usually but not always assumed to be a member of the model class $\mathcal{M}$ under consideration. We call this *expectation-based MDL*. We discuss the distinction in detail in Part III of the book; see also the box on page 407. The individual-sequence approach is the one taken by Rissanen, the main originator of MDL, and we will mostly follow it throughout this book.

The Luckiness Principle In the individual-sequence approach, the minimax optimal universal code is given by the normalized maximum likelihood (NML) code that we mentioned above. A problem is that for many (in fact, most) practically interesting models, the NML code is not well defined. In such cases, a minimax optimal code does not exist. As we explain in Chapter 11, in some cases one can get around this problem using so-called "conditional NML" codes, but in general, one needs to use codes based on a modified minimax principle, which we call the *luckiness principle*. Although it has been implicitly used in MDL since its inception, I am the first to use the term "luckiness principle" in an MDL context; see the box on page 92, Chapter 3; the developments in Chapter 11, Section 11.3, where we introduce the concept of a *luckiness function*; and the discussion in Chapter 17, Section 17.2.1.

The luckiness principle reintroduces some subjectivity in MDL code design. This seems to bring us back to the ad-hoc codes used in crude two-part MDL. The difference however is that with luckiness functions, we can precisely quantify the effects of this subjectivity: for each possible data sample D that we may observe, we can indicate how "lucky" we are on the sample, i.e. how many extra bits we need compared to encode D compared to the best hypothesis that we have available for D. This idea significantly extends the applicability of refined MDL methods.

MDL is a Principle Contrary to what is often thought, MDL, and even, "modern, refined MDL" is *not* a unique, single method of inductive inference. Rather, it represents a general *principle* for doing inductive inference. The principle may (and will) be formulated precisely enough to allow us to establish, for many given methods (procedures, learning algorithms) "this method is an instance of MDL" or "this is *not* an instance of MDL. But nevertheless:

MDL Is a Principle, Not a Unique Method
Being a *principle*, MDL gives rise to several *methods* of inductive inference. There is no single "uniquely optimal MDL method/procedure/algorithm." Nevertheless, in *some special situations* (e.g. simple parametric statistical models), one can clearly distinguish between good and not so good versions of MDL, and something close to "an optimal MDL method" exists.

Summary: Refined MDL (Part III of This Book)
Refined MDL is a method of inductive inference based on *universal codes* which are designed to have some *minimax optimality properties*. Each model $\mathcal{H}$ under consideration is associated with a corresponding universal code. In this book we restrict ourselves to probabilistic $\mathcal{H}$. Refined MDL has mainly been developed for model selection, point hypothesis selection and prediction.
Refined MDL comes in two versions: individual-sequence and expectation-based refined MDL, depending on whether the universal codes are designed to be optimal in an individual-sequence or in an expected sense. If the minimax optimal code relative to a model $\mathcal{M}$ is not defined, some element of subjectivity is introduced into the coding using a *luckiness function*. A more precise overview is given in the box on page 406.

In the remainder of this chapter we will mostly concentrate on MDL for model selection.

1.6 Some Remarks on Model Selection

Model selection is a controversial topic in statistics. Although most people agree that it is important, many say it can only be done on external grounds, and never by merely looking at the data. Still, a plethora of automatic model selection methods has been suggested in the literature. These can give wildly different results on the same data, one of the main reasons being that they have often been designed with different goals in mind. This section starts with a further example that motivates the need for model selection, and it then discusses several goals that one may have in mind when doing model selection. These issues are discussed in a lot more detail in Chapter 14. See also Chapter 17, especially Section 17.3, where we compare MDL model selection to the standard model selection methods AIC and BIC.

1.6.1 Model Selection among Non-Nested Models

Model selection is often used in the following context: two researchers or research groups A and B propose entirely different models $\mathcal{M}_A$ and $\mathcal{M}_B$ as an explanation for the same data D. This situation occurs all the time in applied sciences like econometrics, biology, experimental psychology, etc. For example, group A may have some general theory about the phenomenon at hand which prescribes that the trend in data D is given by some polynomial. Group B may think that the trend is better described by some neural network; a concrete case will be given in Example 1.3 below. A and B would like to have some way of deciding which of their two models is better suited for the data at hand. If they simply decide on the model containing the hypothesis (parameter instantiation) that best fits the data, they once again run the risk of overfitting: if model $\mathcal{M}_A$ has more degrees of freedom (parameters) than model $\mathcal{M}_B$, it will typically be able to better fit random noise in the data. It may then be selected even if $\mathcal{M}_B$ actually better captures the underlying trend (regularity) in the data. Therefore, just as in the hypothesis selection example, deciding whether $\mathcal{M}_A$ or $\mathcal{M}_B$ is a better explanation for the data should somehow depend on how well $\mathcal{M}_A$ and $\mathcal{M}_B$ fit the data and on the respective "complexities" of $\mathcal{M}_A$ and $\mathcal{M}_B$.

In the polynomial case discussed before, there was a countably infinite number of "nested" $\mathcal{M}_\gamma$ (i.e. $\mathcal{M}_\gamma \subset \mathcal{M}_{\gamma+1}$). In contrast, we now deal with a finite number of entirely unrelated models $\mathcal{M}_\gamma$. But there is nothing that stops us from using MDL model selection as "defined" above.

Example 1.5 [Selecting Between Models of Different Functional Form] Consider two models from psychophysics describing the relationship between physical dimensions (e.g., light intensity) and their psychological counterparts (e.g. brightness) (Myung, Balasubramanian, and Pitt 2000): $y = ax^b + Z$ (Stevens's model) and $y = a\ln(x+b) + Z$ (Fechner's model) where Z is a normally distributed noise term. Both models have two free parameters; nevertheless, according to the refined version of MDL model selection to be introduced in Part III, Chapter 14 of this book, Stevens's model is in a sense "more complex" than Fechner's (see page 417). Roughly speaking, this means there are a lot more data patterns that can be *explained* by Stevens's model than can be explained by Fechner's model. Somewhat more precisely, the number of data patterns (sequences of data) of a given length that can be fit well by Stevens's model is much larger than the number of data patterns of the same length that can be fit well by Fechner's model. Therefore, using Stevens's model we run a larger risk of "overfitting."

In the example above, the goal was to select between a power law and a logarithmic relationship. In general, we may of course come across model selection problems involving neural networks, polynomials, Fourier or wavelet expansions, exponential functions - anything may be proposed and tested.

1.6.2 Goals of Model vs. Point Hypothesis Selection

The goal of point hypothesis selection is usually just to infer a hypothesis from the data and use that to make predictions of, or decisions about, future data coming from the same source. Model selection may be done for several reasons:

1. **Deciding between "general" theories.** This is the application that was illustrated in the example above. Often, the research groups A and B are only interested in the models $\mathcal{M}_A$ and $\mathcal{M}_B$, and not in particular hypotheses (corresponding to parameter settings) within those models. The reason is that the models $\mathcal{M}_A$ and $\mathcal{M}_B$ are proposed as *general* theories for the phenomenon at hand. The claim is that they work not only under the exact circumstances under which the experiment giving rise to data D took place but in many other situations as well. In our case, the research group proposing model $\mathcal{M}_A$ may claim that the functional relationship underlying model $\mathcal{M}_A$ provides a good description of the relationship between light intensity and brightness under a variety of circumstances; however, the specific parameter settings may vary from situation to situation. For example, it should be an appropriate model both in daylight (for parameter setting (a_0, b_0)) and in artificial light (for parameter setting (a_1, b_1)).

2. **Gaining insight.** Sometimes, the goal is not to make specific predictions but just to get a first idea of the process underlying the data. Such a rough, first impression may then be used to guide further experimentation about the phenomenon under investigation.

3. **Determining relevant variables.** In Example 1.3 the instances x_i were all real numbers. In practice, the y_i may often depend on several quantities, which may be modeled by taking the x_i to be *real vectors* $x_i = x_{i1}, \ldots, x_{ik}$. We say that there are *k regressor variables*. In such a setting, an important model selection problem is to determine which variables are relevant and which are not. This is sometimes called the *selection-of-variables problem*. Often, for each j, there is a cost associated with measuring x_{ij}. We would therefore like to learn, from some given set of empirical data, which of the regressor variables are truly relevant for predicting the values of y. If there are k regressor variables, this involves model selection between 2^k different models. Each model corresponds to the set of all linear relationships between a particular subset of the regressor variables and y.

4. **Prediction by weighted averaging.** Even if our sole goal is prediction of future data, model selection may be useful. In this context, we first infer a model (set of hypotheses) for the data at hand. We then predict future data by *combining all the point hypotheses within the model to arrive at a prediction*. Usually this is done by taking a weighted average of the predictions that would be optimal according to the different hypotheses within the model. Here the weights of these predictions are determined by the performance of the corresponding hypotheses on past data. There are abundant examples in the literature on Bayesian statistics (Lee 1997; Berger 1985; Bernardo and Smith 1994) which show that, both in theory and in "the real world," prediction by weighting averaging usually works substantially better than prediction by a single hypothesis. In Chapter 15, we discuss model-based MDL prediction, which is quite similar to Bayesian prediction.

1.7 The MDL Philosophy

The first central MDL idea is that every regularity in data may be used to compress that data; the second central idea is that learning can be equated with finding regularities in data. Whereas the first part is relatively straightforward, the second part of the idea implies that *methods for learning from data must have a clear interpretation independent of whether any of the models under consideration is "true" or not*. Quoting J. Rissanen (1989), the main originator of MDL:

> "We never want to make the false assumption that the observed data actually were generated by a distribution of some kind, say Gaussian, and then go on to analyze the consequences and make further deductions. Our deductions may be entertaining but quite irrelevant to the task at hand, namely, to learn useful properties from the data."
>
> *Jorma Rissanen [1989]*

Based on such ideas, Rissanen has developed a radical philosophy of learning and statistical inference that is considerably different from the ideas underlying mainstream statistics, both frequentist and Bayesian. We now describe this philosophy in more detail; see also Chapter 17, where we compare the MDL philosophy to the ideas underlying other statistical paradigms.

1. Regularity as Compression According to Rissanen, the goal of inductive inference should be to "squeeze out as much regularity as possible" from the given data. The main task is to distill the meaningful information present in the data, i.e. to separate structure (interpreted as the regularity, the "meaningful information") from noise (interpreted as the "accidental information"). For the three sequences of Example 1.1, this would amount to the following: the first sequence would be considered as entirely regular and "noiseless." The second sequence would be considered as entirely random - all information in the sequence is accidental, there is no structure present. In the third sequence, the structural part would (roughly) be the pattern that 4 times as many 0s as 1s occur; given this regularity, the description of exactly which one among all sequences with four times as many 0s as 1s actually occurs, is the accidental information.

2. Models as Languages Rissanen interprets models (sets of hypotheses) as nothing more than languages for describing useful properties of the data – a model $\mathcal{H}$ is *identified* with its corresponding universal code $\bar{L}(\cdot \mid \mathcal{H})$. Different individual hypotheses within the models express different regularities in the data, and may simply be regarded as *statistics*, that is, summaries of certain regularities in the data. *These regularities are present and meaningful independently of whether some $H^* \in \mathcal{H}$ is the "true state of nature" or not*. Suppose that the model $\mathcal{H} = \mathcal{M}$ under consideration is probabilistic. In traditional theories, one typically assumes that some $P^* \in \mathcal{M}$ generates the data, and then "noise" is defined as a random quantity relative to this P^*. In the MDL view "noise" is defined relative to the model $\mathcal{M}$ as the residual number of bits needed to encode the data once the model $\mathcal{M}$ is given. Thus, noise is *not* a random variable: it is a function only of the chosen model and the *actually observed data*. Indeed, there is no place for a "true distribution" or a "true state of nature" in this view – there are only models and data. To bring out the difference to the ordinary statistical viewpoint, consider the phrase "these experimental data are quite noisy." According to a traditional interpretation, such a statement means that the data were generated by a distribution with high variance. According to the MDL philosophy, such a phrase means only that the data are not compressible with the currently hypothesized model – as a matter of principle, it can *never* be ruled out that there exists a different model under which the data are very compressible (not noisy) after all!

3. We Have Only the Data Many (but not all[5]) other methods of inductive inference are based on the idea that there exists some "true state of nature," typically a distribution assumed to lie in some model $\mathcal{M}$. The methods are then designed as a means to identify or approximate this state of nature based on as little data as possible. According to Rissanen,[6] such methods are fundamentally flawed. The main reason is that the methods are designed under the assumption that the true state of nature is in the assumed model $\mathcal{M}$, which is often not the case. Therefore, *such methods only admit a clear interpretation under assumptions that are typically violated in practice*. Many cherished statistical methods have been designed in this way - we mention hypothesis testing, minimum-variance unbiased estimation, several nonparametric methods, and even some forms of Bayesian inference – see Chapter 17, Section 17.2.1. In contrast, MDL has a clear interpretation which *depends only on the data*, and not on the assumption of any underlying "state of nature."

Example 1.6 [Models That are Wrong, Yet Useful] Even though the models under consideration are often wrong, they can nevertheless be very *useful*. Examples are the successful "Naive Bayes" model for spam filtering, hidden Markov models for speech recognition (is speech a stationary ergodic process? probably not), and the use of linear models in econometrics and psychology. Since these models are evidently wrong, it seems strange to base inferences on them using methods that are designed under the assumption that they contain the true distribution. To be fair, we should add that domains such as spam filtering and speech recognition are not what the fathers of modern statistics had in mind when they designed their procedures – they were usually thinking about much simpler domains, where the assumption that some distribution $P^* \in \mathcal{M}$ is "true" may not be so unreasonable; see also Chapter 17, Section 17.1.1.

4. MDL and Consistency Let $\mathcal{M}$ be a probabilistic model, such that each $P \in \mathcal{M}$ is a probability distribution. Roughly, a statistical procedure is called *consistent* relative to $\mathcal{M}$ if, for all $P^* \in \mathcal{M}$, the following holds: suppose data are distributed according to P^*. Then given enough data, the learning method will learn a good approximation of P^* with high probability. Many traditional statistical methods have been designed with consistency in mind (Chapter 2, Section 2.5).

5. For example, cross-validation cannot easily be interpreted in such terms of "a method hunting for the true distribution." The same holds for some – not all – Bayesian methods; see Chapter 17.

6. The present author's own views are somewhat milder in this respect, but this is not the place to discuss them.

The fact that in MDL, we do not assume a true distribution may suggest that we do not care about statistical consistency. But this is not the case: we would still like our statistical method to be such that in the *idealized* case where one of the distributions in one of the models under consideration actually generates the data, our method is able to identify this distribution, given enough data. If even in the idealized special case where a "truth" exists within our models, the method fails to learn it, then we certainly cannot trust it to do something reasonable in the more general case, where there may not be a "true distribution" underlying the data at all. So: consistency *is* important in the MDL philosophy, but it is used *as a sanity check (for a method that has been developed without making distributional assumptions) rather than as a design principle*; see also Chapter 17, Section 17.1.1.

In fact, mere consistency is not sufficient. We would like our method to converge to the imagined true P^* *fast*, based on as small a sample as possible. Theorems 15.1 and 15.3 of Chapter 15 show that this indeed happens for MDL prediction and hypothesis selection – as explained in Chapter 16, MDL convergence rates for estimation and prediction are typically either minimax optimal or within a factor $\log n$ of minimax optimal.

Summarizing this section, the MDL philosophy is agnostic about whether any of the models under consideration is "true," or whether something like a "true distribution" even exists. Nevertheless, it has been suggested (Webb 1996; Domingos 1999) that MDL embodies a naive belief that "simple models" are "a priori more likely to be true" than complex models. Below we explain why such claims are mistaken.

1.8 Does It Make Any Sense? MDL, Occam's Razor, and the "True Model"

When two models fit the data equally well, MDL will choose the one that is the "simplest" in the sense that it allows for a shorter description of the data. As such, it implements a precise form of Occam's razor – *even though as more and more data become available, the model selected by MDL may become more and more complex!* Throughout the ages, Occam's razor has received a lot of praise as well as criticism. Some of these criticisms (Webb 1996; Domingos 1999) seem applicable to MDL as well. The following two are probably heard most often:

"1. Occam's razor (and MDL) is arbitrary." Because "description length" is a syntactic notion it may seem that MDL selects an arbitrary model: different codes would have led to different description lengths, and therefore, to different models. By changing the encoding method, we can make "complex' things "simple" and vice versa.

"2. Occam's razor is false." It is sometimes claimed that Occam's razor is false: we often try to model real-world situations that are arbitrarily complex, so why should we favor simple models? In the words of G. Webb:[7] "What good are simple models of a complex world?"

The short answer to 1 is that this argument overlooks the fact that we are not allowed to use just any code we like! "Refined MDL" severely restricts the set of codes one is allowed to use. As we explain below, and in more detail in Chapter 7, this leads to a notion of complexity that can also be interpreted as a kind of "model volume," without any reference to "description lengths." The short answer to 2 is that even if the true data-generating machinery is very complex, it may often be a good *strategy* to prefer simple models for small sample sizes.. Below we give more elaborate answers to both criticisms.

1.8.1 Answer to Criticism No. 1: Refined MDL's Notion of "Complexity" Is Not Arbitrary

In "algorithmic" or "idealized" MDL (Section 1.4), it is possible to define the Kolmogorov complexity of a *point* hypothesis H as the length of the shortest program that computes the function value or probability $H(x)$ up to r bits precision when input (x, r). In our practical version of MDL, there is no single "universal" description method used to encode point hypotheses. A hypothesis with a very short description under one description method may have a very long description under another method. Therefore it is usually meaningless to say that a particular point hypothesis is "simple" or "complex." However, for many types of models, it *is* possible to define the complexity of a *model* (interrelated set of point hypotheses) in an unambiguous manner, that does not depend on the way we parameterize the model. This is the "parametric complexity" that we mentioned in Section 1.5.3.[8] It

7. Quoted with permission from KDD Nuggets 96:28, 1996.
8. The parametric complexity of a probabilistic model $\mathcal{M} = \{P\}$ that consists only of one hypothesis, is always 0, no matter how large the Kolmogorov complexity of P; see Chapter 17, Example 17.5.

will be defined for finite models in Chapter 6. In Chapter 7 we extend the definition to general models that contain uncountably many hypotheses.

> There exists a close connection between the algorithmic complexity of a hypothesis and the parametric complexity of any large model that contains the hypothesis. Broadly speaking, for *most* hypotheses that are contained in any given model, the Kolmogorov complexity of the hypothesis will be approximately equal to the parametric complexity of the model.

In Example 1.3 we did speak of a "complex" point hypothesis. This is really sloppy terminology: since only the complexity of *models* rather than hypotheses can be given an unambiguous meaning, we should instead have spoken of "a point hypothesis that, relative to the set of models under consideration, is a member only of a complex model and not of a simple model." Such sloppy terminology is commonly used in papers on MDL. Unfortunately, it has caused a lot of confusion in the past. Specifically, it has led people to think that MDL model selection is a mostly arbitrary procedure leading to completely different results according to how the details in the procedure are filled in (Shaffer 1993). At least for the refined versions of MDL we discuss in Part III of this book, this is just plain false.

Complexity of Models vs. Complexity of Hypotheses
In *algorithmic* MDL, we may define the complexity of an *individual* (i.e., *point*) hypothesis (function or probability distribution). In *practical* MDL, as studied here, this is not possible: complexity becomes a property of *models* (sets of point hypotheses) rather than individual point hypotheses (instantiations of models).
MDL-based, or *parametric* complexity, is a property of a model that does not depend on any particular description method used, or any parameterization of the hypotheses within the model. It is related to (but not quite the same as) the number of substantially different hypotheses in a model (Part II of this book, Chapter 6, Chapter 7). A "simple model" then roughly corresponds to "a small set of hypotheses."

In practice, we often use models for which the parametric complexity is undefined. We then use an extended notion of complexity, based on a "luckiness function." While such a complexity measure does have a subjective component, it is still far from arbitrary, and it cannot be used to make a complex model "simple"; see Chapter 17, Section 17.2.1.

1.8.2 Answer to Criticism No. 2

In light of the previous discussions in this chapter, preferring "simpler" over more "complex" models seems to make a lot of sense: one should try to avoid overfitting (i.e. one should try to avoid modeling the noise rather than the "pattern" or "regularity" in the data). It seems plausible that this may be achieved by somehow taking into account the "complexity", "richness", or "(non-) smoothness" of the models under consideration. But from another viewpoint, the whole enterprise may seem misguided: as the example below shows, it seems to imply that, when we apply MDL, we are implicitly assuming that "simpler" models are somehow a priori more likely to be "true." Yet in many cases of interest, the phenomena we try to model are very complex, so then preferring simpler models for the data at hand would not seem to make a lot of sense. How can these two conflicting intuitions be reconciled?

Authors criticizing Occam's razor (Domingos 1999; Webb 1996) usually do think that in some cases "simpler" models should be preferred over more "complex" ones since the former are more understandable, and in that sense more useful. But they argue that the "simpler" model will usually not lead to better predictions of future data coming from the same source. We claim that on the contrary, for the MDL-based definitions of "simple" and "complex" we will introduce in this book, selecting the simpler model in many cases *does* lead to better predictions, even in a complex environment.

Example 1.7 [MDL Hypothesis/Model Selection for Polynomials, cont.] Let us focus on point hypothesis selection of polynomials. Let $\mathcal{H}_\gamma$ be the set of γth degree polynomials. Suppose a "truth" exists in the following sense: there exists some distribution P^*_X such that the x_i are all independently distributed according to P^*_X. We assume that all x_i must fall within some interval $[a,b]$, i.e. $P^*_X([a,b]) = 1$. There also exists a function h^* such that for all x_i generated by P^*_X, we have $y_i = h^*(x_i) + Z_i$. Here the Z_i are noise or "error" terms. We assume the Z_i to be identical, independent, normally (Gaussian) distributed random variables, with mean 0 and some variance σ^2. For concreteness we will assume $h^*(x) = x^3 - 8x^2 + 19x + 9$ and $\sigma^2 = 1$.

This is actually the polynomial/error-combination that was used to generate the points in Figure 1.1 on page 13. However, the x_i in that graph were not drawn according to some distribution such as in the present scenario. Instead, they were preset to be $0.5, 1, 1.5, \ldots, 12$. This is similar to the practical case where the *experimental design* is controlled by the experimenter: the experimenter determines the x_i values for which corresponding y_i-values will be measured. In this setup similar analyses as those below may still be made (see, e.g. (Wei 1992)).

In such a scenario with a "true" h^* and P^*_X, as more and more data pairs (x_i, y_i) are made available, with high probability something like the following will happen (Chapter 16): for very small n, MDL model selection will select a 0th-degree polynomial, i.e. $y = c$ for some constant c. Then as n grows larger, MDL will start selecting models of higher degree. At some sample size n it will select a third-order polynomial for the first time. It will then for a while (as n increases further) fluctuate between second-, third- and fourth-order polynomials. But for *all* n larger than some critical n_0, MDL will select the correct third order $\mathcal{H}_3$. It turns out that for all n, the point hypothesis $\ddot{h}_n$ selected by two-part MDL hypothesis selection is (approximately) the polynomial within the selected $\mathcal{H}_\gamma$ that best fits data $D = ((x_1, y_1), \ldots, (x_n, y_n))$. As n goes to infinity, $\ddot{h}_n$ will with high probability converge to h^* in the sense that all coefficients converge to the corresponding coefficients of h^*.

Two-part code MDL behaves like this not just when applied to the model class of polynomials, but for most other potentially interesting model classes at well. The upshot is that, *for small sample sizes*, MDL has a built-in preference for "simple" models. This preference may seem unjustified, since "reality" may be more complex. We claim that on the contrary, such a preference (if implemented carefully) has ample justification.

To back our claim, we first note that MDL (and the corresponding form of Occam's razor) is just a *strategy* for inferring models from data ("choose simple models at small sample sizes"), not a statement about how the world works ("simple models are more likely to be true") – indeed, a strategy cannot be true or false, it is "clever" or "stupid." And the strategy of preferring simpler models is clever even if the data-generating process is highly complex, as illustrated by the following example:

Example 1.8 ["Infinitely" Complex Sources] Suppose that data are subject to the law $Y = g(X) + Z$ where g is some continuous function and Z is some noise term with mean 0. If g is not a polynomial, but X only takes values in a finite interval, say $[-1, 1]$, we may still approximate g arbitrarily well by taking higher and higher degree polynomials. For example, let $g(x) = \exp(x)$. Then, if we use MDL to learn a polynomial for data $D = ((x_1, y_1), \ldots, (x_n, y_n))$, the degree of the polynomial $\ddot{h}_n$ selected by MDL at sample size n will increase with n, and with high probability, $\ddot{h}_n$ converges to $g(x) = \exp(x)$ in the sense that $\max_{x \in [-1,1]} |\ddot{h}_n(x) - g(x)| \to 0$ (Chapter 16). Of course, if we had better prior knowledge about the problem we could have tried to learn g using a model class $\mathcal{H}$ containing the function $y = \exp(x)$. But in general, both our imagination and our computational resources are limited, and we may be forced to use imperfect models.

If, based on a small sample, we choose the best-fitting polynomial $\hat{h}$ within the set of *all* polynomials, then, even though $\hat{h}$ will fit the data very well, it is likely to be quite unrelated to the "true" g, and $\hat{h}$ may lead to disastrous predictions of future data. The reason is that, for small samples, the set of all polynomials is very large compared to the set of possible data patterns that we might have observed. Therefore, any particular data pattern can only give us very limited information about which high-degree polynomial best approximates g. On the other hand, if we choose the best-fitting $\hat{h}^\circ$ in some much smaller set such as the set of second-degree polynomials, then it is highly probable that the prediction quality (mean squared error) of $\hat{h}^\circ$ on future data is about the same as its mean squared error on the data we observed: the size (complexity) of the contemplated model is relatively small compared to the set of possible data patterns that we might have observed. Therefore, the particular pattern that we do observe gives us a lot of information on what second-degree polynomial best approximates g.

Thus, (a) $\hat{h}^\circ$ typically leads to better predictions of future data than $\hat{h}$; and (b) unlike $\hat{h}$, $\hat{h}^\circ$ is *reliable* in that it gives a correct impression of how good it will predict future data *even if the "true" g is "infinitely" complex*. This idea does not just appear in MDL, but is also the basis of the structural risk minimization approach (Vapnik 1998) and many standard statistical methods for nonparametric inference; see Chapter 17, Section 17.10. In such approaches one acknowledges that the data-generating machinery can be infinitely complex (e.g., not describable by a finite degree polynomial). Nevertheless, it is still a good strategy to approximate it by simple hypotheses (low-degree polynomials) as long as the sample size is small. Summarizing:

The Inherent Difference between Under- and Overfitting
If we choose an overly simple model for our data, then the best-fitting point hypothesis within the model is likely to be almost the best predictor, within the simple model, of future data coming from the same source. If we overfit (choose a very complex model) and there is noise in our data, then, *even if the complex model contains the "true" point hypothesis*, the best-fitting point hypothesis within the model may lead to very bad predictions of future data coming from the same source.

This statement is very imprecise and is meant more to convey the general idea than to be completely true. The fundamental consistency theorems for MDL prediction and hypothesis selection (Chapter 15, Theorem 15.1 and Theorem 15.3), as well as their extension to model selection (Chapter 16), are essentially just variations of this statement that are provably true.

The Future and The Past Our analysis depends on the data items (x_i, y_i) to be probabilistically independent. While this assumption may be substantially weakened, we can justify the use of MDL and other forms of Occam's razor *only* if we are willing to adopt some (possibly very weak) assumption of the sort "training data and future data are from the same source": future data should (at least with high probability) be subject to some of the same regularities as training data. Otherwise, D and D' may be completely unrelated and *no* method of inductive inference can be expected to work well. This is indirectly related to the *grue*-paradox (Goodman 1955).

MDL and Occam's Razor
While MDL does have a built-in preference for selecting "simple" models (with small "parametric complexity"), this does *not at all* mean that applying MDL only makes sense in situations where simpler models are more likely to be true. MDL is a *methodology for inferring models from data, not a statement about how the world works!* For small sample sizes, it prefers simple models. It does so not because these are "more likely to be true" (they often are not). Instead, it does so because this tends to select the model that leads to the best predictions of future data from the same source. For small sample sizes this may be a model much simpler than the model containing the "truth" (assuming for the time being that such a model containing the "truth" exists in the first place).

In fact, some of MDL's most useful and successful applications are in nonparametric statistics where the "truth" underlying data is typically assumed to be "infinitely" complex (see Chapter 13 and Chapter 15).

1.9 History and Forms of MDL

The practical MDL principle that we discuss in this book has mainly been developed by J. Rissanen in a series of papers starting with (Rissanen 1978). It has its roots in the theory of Kolmogorov complexity (Li and Vitányi 1997), developed in the 1960s by Solomonoff (1964), Kolmogorov (1965) and Chaitin (1966, 1969). Among these authors, Solomonoff (a former student of the famous philosopher of science, Rudolf Carnap) was explicitly interested in inductive inference. The 1964 paper contains explicit suggestions on how the underlying ideas could be made practical, thereby foreshadowing some of the later work on two-part MDL. While Rissanen was not aware of Solomonoff's work at the time, Kolmogorov's [1965] paper did serve as an inspiration for Rissanen's (1978) development of MDL. Still, Rissanen's practical MDL is quite different from the idealized forms of MDL that have been directly based on Kolmogorov complexity, which we discussed in Section 1.4.

Another important inspiration for Rissanen was Akaike's AIC method for model selection (Chapter 17, Section 17.3), essentially the first model selection method based on information-theoretic ideas (Akaike 1973). Even though Rissanen was inspired by AIC, both the actual method and the underlying philosophy are substantially different from MDL.

Minimum *Message* Length MDL is much closer related to the *Minimum Message Length (MML) Principle* (Wallace 2005), developed by Wallace and his coworkers in a series of papers starting with the groundbreaking (Wallace and Boulton 1968); other milestones are (Wallace and Boulton 1975) and (Wallace and Freeman 1987). Remarkably, Wallace developed his ideas without being aware of the notion of Kolmogorov complexity. Although Rissanen became aware of Wallace's work before the publication of (Rissanen 1978), he developed his ideas mostly independently, being influenced rather by Akaike and Kolmogorov. Indeed, despite the close resemblance of both methods in practice, the underlying philosophy is very different - see Chapter 17, Section 17.4.

Refined MDL The first publications on MDL only mention two-part codes. Important progress was made by Rissanen (1984), in which prequential codes are employed for the first time and Rissanen (1987), who introduced the Bayesian mixture codes into MDL. This led to the development of the notion of stochastic complexity as the shortest codelength of the data given a model

(Rissanen 1986c; Rissanen 1987). However, the connection to Shtarkov's *normalized maximum likelihood code* was not made until 1996, and this prevented the full development of the notion of "parametric complexity." In the mean time, in his impressive Ph.D. thesis, Barron (1985) showed how a specific version of the two-part code criterion has excellent frequentist statistical consistency properties. This was extended by Barron and Cover (1991) who achieved a breakthrough for two-part codes: they gave clear prescriptions on how to design codes for hypotheses, relating codes with good minimax codelength properties to rates of convergence in statistical consistency theorems. Some of the ideas of Rissanen (1987) and Barron and Cover (1991) were, as it were, unified when Rissanen (1996) introduced the normalized maximum likelihood code. The resulting theory was summarized for the first time by Barron, Rissanen, and Yu (1998), and is the subject of this book. Whenever we need to distinguish it from other forms of MDL, we call it "refined MDL."

1.9.1 What Is MDL?

"MDL" is used by different authors in somewhat different meanings, and it may be useful to review these. Some authors use MDL as a broad umbrella term for all types of inductive inference based on finding a short codelength for the data. This would, for example, include the "idealized" versions of MDL based on Kolmogorov complexity (page 11) and Wallaces's MML principle (see above). Some authors take an even broader view and include all inductive inference that is based on data compression, even if it cannot be directly interpreted in terms of codelength minimization. This includes, for example the work on similarity analysis and clustering based on the *normalized compression distance* (Cilibrasi and Vitányi 2005).

On the other extreme, for historical reasons, some authors use the *MDL Criterion* to describe a very specific (and often not very successful) model selection criterion equivalent to BIC (see Chapter 17, Section 17.3).

As already indicated, we adopt the meaning of the term that is embraced in the survey (Barron, Rissanen, and Yu 1998), written by arguably the three most important contributors to the field: we use MDL for general *inference based on universal models*. Although we concentrate on hypothesis selection, model selection and prediction, this idea can be further extended to many other types of inductive inference. These include *denoising* (Rissanen 2000; Hansen and Yu 2000; Roos, Myllymäki, and Tirri 2005), *similarity analysis* and *clustering* (Kontkanen, Myllymäki, Buntine, Rissanen, and Tirri 2005), *outlier detection* and *transduction* (as defined in (Vapnik 1998)), and many others. In such areas there has been less research and a "definitive" universal-model based MDL approach has not yet been formulated. We do expect, however, that such research will take place in the future: one of the main strengths of "MDL" in this broad sense is that it can be applied to ever more exotic modeling situations, in which the models do not resemble anything that is usually encountered in statistical practice. An example is the model of context-free grammars, already considered by Solomonoff (1964).

Another application of universal-model based MDL is the type of problem usually studied in *statistical learning theory* (Vapnik 1998); see also Chapter 17, Section 17.10. Here the goal is to directly learn functions (such as polynomials) to predict Y given X, without making any specific probabilistic assumptions about the noise. MDL has been developed in some detail for such problems, most notably *classification* problems, where Y takes its values in a finite set – spam filtering is a prototypical example; here X stands for an email message, and Y encodes whether or not it is spam. An example is the application of MDL to decision tree learning (Quinlan and Rivest 1989; Wallace and Patrick 1993; Mehta, Rissanen, and Agrawal 1995). Some MDL theory for such cases has been developed (Meir and Merhav 1995; Yamanishi 1998; Grünwald 1998), but the existing MDL methods in this area can behave suboptimally. This is explained in Chapter 17, Section 17.10.2. Although we certainly consider it a part of "refined" MDL, we do not consider this "nonprobabilistic" MDL further in this book, except in Section 17.10.2.

1.9.2 MDL Literature

Theoretical Contributions There have been numerous contributors to refined MDL theory, but there are three researchers that I should mention explicitly: J. Rissanen, B. Yu and A. Barron, who jointly wrote (Barron, Rissanen, and Yu 1998). For example, most of the results that connect MDL to traditional statistics (including Theorem 15.1 and Theorem 15.3 in Chapter 15) are due to A. Barron. This book contains numerous references to their work.

There is a close connection between MDL theory and work in *universal coding* ((Merhav and Feder 1998); see also Chapter 6) and *universal prediction* ((Cesa-Bianchi and Lugosi 2006); see also Chapter 17, Section 17.9).

Practical Contributions There have been numerous practical applications of MDL. The only three applications we describe in detail are a crude MDL method for learning Markov chains (Chapter 5); a refined MDL method for learning densities based on histograms (Chapter 13 and Chapter 15); and MDL regression (Chapter 12 and Chapter 14). Below we give a few representative examples of other applications and experimental results that have appeared in the literature. We warn the reader that this list is by no means complete! Hansen and Yu (2001) apply MDL to a variety of practical problems involving regression, clustering analysis, and time series analysis. In (Tabus, Rissanen, and Astola 2002; Tabus, Rissanen, and Astola 2003), MDL is used for classification problems arising in genomics. Lee (2002a,b) describes additive clustering with MDL. use MDL for image denoising and apply MDL to decision tree learning. use MDL for sequential prediction. In (Myung, Pitt, Zhang, and Balasubramanian 2000; Myung, Balasubramanian, and Pitt 2000), MDL is applied to a variety of model selection problems arising in cognitive psychology. All these authors apply modern, "refined" versions of MDL. Some references to older work, in which "crude" (but often quite sensible) ad-hoc codes are used, are (Friedman, Geiger, and Goldszmidt 1997; Allen and Greiner 2000; Allen, Madani, and Greiner 2003; Rissanen and Ristad 1994; Quinlan and Rivest 1989; Nowak and Figueiredo 2000; Liu and Moulin 1998; Ndili, Nowak, and Figueiredo 2001; Figueiredo, J. Leitão, and A.K.Jain 2000; Gao and Li 1989). In these papers, MDL is applied to learning Bayesian networks, grammar inference and language acquisition, learning decision trees, analysis of Poisson point processes (for biomedical imaging applications), image denoising, image segmentation, contour estimation, and Chinese handwritten character recognition respectively. MDL has also been extensively studied in time-series analysis, both in theory (Hannan and Rissanen 1982; Gerenscér 1987; Wax 1988; Hannan, McDougall, and Poskitt 1989; Hemerly and Davis 1989b; Hemerly and Davis 1989a; Gerencsér 1994) and practice (Wei 1992; Wagenmakers, Grünwald, and Steyvers 2006).

Finally, we should note that there have been a number of applications, especially in *natural language learning*, which, although practically viable, have been primarily inspired by "idealized MDL" and Kolmogorov complexity, rather than by the Rissanen-Barron-Yu style of MDL that we consider here. These include (Adriaans and Jacobs 2006; Osborne 1999; Starkie 2001) and my own (Grünwald 1996).

Other Tutorials, Introductions and Overviews The reader who prefers a shorter introduction to MDL than the present one may want to have a look at (Barron, Rissanen, and Yu 1998) (very theoretical and very comprehensive; presumes knowledge of information theory), (Hansen and Yu 2001) (presumes knowledge of statistics; describes several practical applications), (Lanterman 2001) (about comparing MDL, MML and asymptotic Bayesian approaches to model selection), or perhaps my own (Grünwald 2005), which is part of (Grünwald, Myung, and Pitt 2005), a "source book" for MDL theory and applications that contains chapters by most of the main contributors to the field.

Rissanen (1989,2007) has written two books on MDL. While outdated as an introduction to MDL, the "little green book" (Rissanen 1989) is still very much worth reading for its clear exposition of the philosophy underlying MDL. (Rissanen 2007) contains a brief general introduction and then focuses on some recent research of Rissanen's, applying the renormalized maximum likelihood (RNML) distribution (Chapter 11) in regression and denoising, and formalizing the connection between MDL and Kolmogorov's structure function. In contrast to myself, Rissanen writes in accord with his own principle: while containing a lot of information, both texts are quite short.

1.10 Summary and Outlook

We have discussed the relationship between compression, regularity, and learning. We have given a first idea of what the MDL principle is all about, and of the kind of problems we can apply it to. In the next chapters, we present the mathematical background needed to describe such applications in detail.