Part I

Theory of MDL

Introduction to Part I

In 1996, the present author attended a conference at which a well-known statistician gave a talk on Bayesian networks, a particular kind of probabilistic model. He artificially generated a set of data by sampling from a quite complicated Bayesian network. He used a 'learning algorithm' to find a good model for the data, and this learning algorithm came up with a much simpler Bayesian network. While this network was obviously not the 'correct' one, it did work quite well in predicting and classifying future data. Since the predictions of the overly simple model could be calculated much more efficiently than those of the correct model, the statistician suggested to use the simple model instead of the correct one for doing such predictions.

A member of the audience (actually, another well-known statistician) strongly objected. He said that modeling data with an overly simple model can potentially lead to disastrous results (he even claimed that the explosion of the Challenger space shuttle in 1987 was caused by using overly simple models). Of course, he said, 'Occam's Razor' is useful: if we do not have any further knowledge about the process generating our data, it makes sense to pick a reasonably simple model with small error rather than a complex model with zero error. But if we really *know* that the data is better modeled by a complex model, we should certainly not use an overly simple one.

This led to fierce discussions among the audience. On the one side there were those who said that there was nothing wrong in principle with using a model that is overly simple; on the other side were those who agreed that using an overly simple model is always a risky thing to do. Unfortunately, neither of the two sides in this debate had any means of *proving* they were right: all arguments were based on intuition.

When observations arrive sequentially, the MDL Principle will often pick an overly simple model for the first few observations. In the polynomial example (Figure 1 on page x), MDL may prefer a straight line for the first n observations. Only when there is enough data available will the lower error achieved by a third degree polynomial outweigh its extra complexity. Hence, the discussion at the conference is highly relevant to MDL: is the MDL Principle only useful in that it selects a good model *when enough data is available*? Or can the overly simple model that will be selected in the presence of few data be fruitfully used as well, without potentially 'disastrous' results?

It is this question which we attempt to answer in part I of this thesis. Rather than being based merely on intuition, our answer will be mathematically precise. The question is relevant not only to MDL, but actually to most of current statistical practice. For most models we use in practice are evidently too simple: we fit straight lines to data that are clearly not really linearly related (as an example, think about models in economics); we model several parts of data as if they were independent while we know they are not independent at all (an example is speech recognition [148]). Does this lead to potentially disastrous results? If so, how come we have got away with this for so long?

A second goal of part I of this thesis is to provide an introduction to MDL that can be read without any previous knowledge of either information theory or statistics; this introduction will also provide the necessary background needed for parts II and III. Chapter 1 gives a self-contained introduction to the MDL Principle that avoids complicated mathematics. In Chapter 2, we discuss the fundamental concept of MDL: the *stochastic complexity*. Chapter 3 gives an introduction to the *Maximum Entropy Principle* that we mentioned on page xi and connects it to the MDL Principle. Both Chapter 1 and Chapter 2 mention some problematic issues that, we think, are not fully satisfactorily handled by the existing theory. In Chapter 1 this is the issue of what exactly a model says about the modeled situation. In Chapter 2 this is the question of whether and how non-probabilistic model classes can be recast in probabilistic terms. Chapter 3 reviews an argument that has led many researchers to reject Maximum Entropy. In chapters 4 and 5 we show that all three issues are just aspects of the basic question whether or not one can justify the use of an overly simple model (or more generally, a model that captures some, but not all regularities underlying the data), and we make an attempt to answer the question.

Chapter 1

A First Introduction to the MDL Principle

This chapter aims to give a self-contained introduction to the MDL Principle. In Section 1.1 we explain the fundamental ideas and intuitions behind MDL. Section 1.2 provides the information-theoretic background that is necessary to formalize MDL. We proceed to formalize the MDL Principle for probabilistic model classes (Section 1.3) and for non-probabilistic model classes (Section 1.4). Section 1.5 discusses an important aspect of the MDL *Philosophy*, namely, that the goal of inductive inference should be to look for a 'good', and not for a 'true' model.

In order to make the introduction self-contained and at the same time not too complicated we avoid discussion of two very important topics: both the treatment of the *stochastic complexity* and of the exact relation between MDL and Bayesian statistics are deferred to Chapter 2.

1.1 The Fundamental Idea

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 in the data or to classify or predict future data. In the statistics literature, the data we are given are usually called the 'sample' or the 'observations'. The aim of finding laws underlying the data is usually cast in terms of finding a good *model* for the data. The process of finding such a model is called *statistical inference*. In the machine learning and neural network literature [110, 111, 72], the sample is usually called the *training set*, a model is often called a *hypothesis* and future data is often called the *test set*. The process of finding a good model is called 'learning'. Throughout this thesis, we use statistical and machine learning terminology interchangeably.

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.

$$\texttt{00010001000100010001}\ldots\texttt{000100010001000100010001} \quad (1.1)$$
$$\texttt{01110100110100001010}\ldots\texttt{1010111010111011000101100010} \quad (1.2)$$
$$\texttt{00011000001010010100}\ldots\texttt{0100010000010000001000110000} \quad (1.3)$$

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; this means that there is definitely no 'law' or regularity underlying it. Indeed, we cannot seem to find such a law 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 this 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 like a sensible thing to do.

1.1.1 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 less 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 *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 non-regular 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 sequence (1.2) will, with overwhelming probability, be not much shorter than sequence (1.2) itself.

Notice that our description of sequence (1.3) as given above does not uniquely define sequence 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 $\binom{n}{\gamma n}$ sequences of length n with a fraction of γ 1s. For every rational number γ except $\gamma = 1/2$, the ratio of $\binom{n}{\gamma n}$ to 2^n goes to 0 exponentially fast as n increases (this is shown formally in Chapter 3, page 59; by the method used there one can also show that for $\gamma = 1/2$, 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 than 1s is vanishingly small. Therefore, $i \ll 2^{10000}$ and to write down i in binary we need approximately $(\log_2 i) \ll 10000$ bits.

Description Methods and Codes In order to formalize our idea, we have to replace the part of the descriptions 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. Throughout this thesis, we assume that our data D always consists of a sequence of symbols coming from some fixed finite or countably infinite *data alphabet*. 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 $\mathbf{B} = \{0, 1\}$; 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 d. We can then again model them as symbols coming from a finite data alphabet.

We let $\mathbf{A}^n$ stand for the n-fold Cartesian product of alphabet $\mathbf{A}$. We define λ as the empty sequence and define $\mathbf{A}^0 = \{\lambda\}$. We let $\mathbf{A}^* = \bigcup_{i=0}^{\infty} \mathbf{A}^i$ and finally we write $\mathbf{A}^+$ as an abbreviation of $\mathbf{A}^* \setminus \mathbf{A}^0$.

In our problem setting we are given a *sample* or equivalently *data sequence* $D = (x_1, \ldots, x_n) \in \mathbf{A}^*$ where each $x_i \in \mathbf{A}$. We call x_i a single observation or, equivalently, a *data item*. We sometimes use the notation x^i for $(x_1, \ldots, x_i)$. Similarly, whenever the *sample size* or equivalently *data sequence length* n is not clear from the context, we write x^n in stead of D. For convenience, it will sometimes be useful to write $x^0 \equiv \lambda$ ($A \equiv B$ is to be read as 'A is defined to be equal to B').

As argued in [128], without any loss of generality we can always describe our data sequences as binary strings. Hence all the description methods we consider map data sequences to sequences of bits. We use the term 'code' as a specific case of a 'description method[1]'.

Definition 1.2 *Let* $\mathbf{A}$ *be some data alphabet. A* code C *is a one-to-one map from* $\mathbf{A}^+$ *to* $\mathbf{B}^+$. *A* description method *is a one-many relation over* $\mathbf{A}^+ \times \mathbf{B}^+$.

Codes and description methods have in common that given any encoding, it is always possible to reconstruct the data sequence that it describes. Description methods are a more general notion than codes in that for a code, there should always be only *one* possible encoding of every data sequence. For a description method, there may be data sequences that can be encoded in several ways.

[1]Our definition of 'description method' coincides with what Rissanen [128] calls a 'coding system'. Our definition of 'code' coincides with Rissanen's [128]. In the standard textbook [30], what we here call a code is called 'uniquely decodable code'.

Here is a simple example (copied from Rissanen [128]) of a code C_1 for a data alphabet $\mathbf{A}_1 = \{a, b, c\}$. C_1 is defined as follows: $C_1(a) = 0, C_1(b) = 10, C_1(c) = 11$; for all $x, y \in \mathbf{A}_1^+, C_1(xy) = C_1(x)C_1(y)$. We call $C_1(a)$ the *codeword* of a. For example, data sequence $aabac$ is encoded as $C_1(aabac) = 0010011$. It is easy to see that no two different data sequences can have the same code word; hence from an encoded sequence $C_1(x)$ we can always retrieve the original sequence x.

Compression and Small Subsets We are now in a position to prove our claim that it is impossible to substantially compress sequences that have been generated by fair coin tosses. Let us take some arbitrary but fixed description method C over the binary data alphabet $\mathbf{A} = \{0, 1\}$. Suppose we are given a data sequence of length n (in our example, $n = 10000$). 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 $\sum_{i=1}^{m} 2^m < 2^{m+1}$ 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.

Seen in this light, having a short code 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.

1.1.2 Solomonoff's Breakthrough

It seems that what data is compressible and what 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 [145] – Ray Solomonoff suggested to use 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 code that maps each data sequence D to the *shortest program* that prints D and then halts[2]. Clearly, this is a code in the sense of Definition 1.2 above in that it defines a 1-1 mapping from sequences over the data alphabet to binary sequences. The shortest program for a sequence D is then interpreted as the *optimal model* 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
```

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

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 (2) is, with extremely high probability, not much shorter than the following:

```
print '01110100110100001010........1110101110110001011000010'; 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 [93] 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 [145] showed that it does not matter exactly 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 $L_U(D)$ the length of the shortest program for D using universal language U. We can show that for every two universal languages U_1 and U_2, the difference between the two lengths $L_{U_1}(D) - L_{U_2}(D)$ is bounded by a constant that depends on U_1 and U_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 by Solomonoff [145], Kolmogorov [86] (hence the name 'Kolmogorov' complexity) and Chaitin [23]. The proof is based on the fact that one can write a compiler for every universal language U_1 in every other universal language U_2. Such a compiler is a computer program with length $L_{1,2}$. 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 $L_{\text{C,Pascal}}$. We can simulate each program P_1 written in language U_1 by a program P_2 written in U_2 as follows: P_2 consists of the compiler from U_1 to U_2, followed by P_1. The length of program P_2 is bounded by the length of P_1 plus $L_{1,2}$. Hence for all data D, the maximal difference between $L_{U_1}(D)$ and $L_{U_2}(D)$ is bounded by $\max\{L_{1,2}, L_{2,1}\}$, a constant which only depends on U_1 and U_2 but not on D.

1.1.3 Making the Idea Applicable

There are two problems with applying Kolmogorov Complexity to practical learning problems. First, the Kolmogorov Complexity as such cannot be computed – there is no computer program that, for every sequence of data D, when given D as input, returns a shortest program that prints D and halts. Neither can there be a program, that for every set of 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 [93]. Another problem is 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 $L_{1,2}$.

The idea behind the MDL Principle is now to scale down Solomonoff's approach so that it does become applicable: instead of a universal computer language as a description method, we should use description methods C which still allow us to compress many of the intuitively 'regular' sequences but which nevertheless also allow us to compute, for every D, the length of the shortest description of D attainable using C. 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 maximally 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, we will see below that it is possible to pick a code C with which we can compress all sets of 2-dimensional data which are 'regular' in the sense that the x-values are related to the y-values by some simple polynomial. If we have a priori reasons to believe that our data points are related by *some* polynomial, but we just do not know which, then this C may be good choice.

1.1.4 Two-Part Codes; First Instantiation of the MDL Principle

Let $\mathcal{M}$ be some given class of models, for example, the class of all polynomials. To make our idea work, we need a code that compresses those data sets that are well-described by one of the models in $\mathcal{M}$. For the model class $\mathcal{M}_U$ consisting of all computer programs written in some universal language U, the corresponding code $C_{\mathcal{M}}$ would simply be the code mapping each D to the shortest program that prints D and then halts. For model classes like the polynomials, whose elements do not directly define a description method, one can construct the codes $C_{\mathcal{M}}$ in several ways. In this chapter, we concentrate on the so-called two-part codes, which are just one possible instantiation of $C_{\mathcal{M}}$. We should stress at the outset that there are more possibilities. Since the two-part codes are conceptually the simplest (and also historically the first ones to have appeared [161, 126]), we defer discussion of other possibilities to Chapter 2.

The idea behind the two-part code is to describe a data sequence D by first describing some hypothesis $H \in \mathcal{M}$ and then describing the data with the help of the hypothesis, which usually amounts to specifying the *errors* the hypothesis makes on the data.

MDL Principle (two-part code version) Among the set of candidate hypotheses $\mathcal{M}$, the best hypothesis to explain a set of data is the one which minimizes the sum of

- the length, in bits, of the description of the hypothesis; and
- the length, in bits, of the description of the data when encoded with the help of the hypothesis.

A hypothesis with a long description length is what we would intuitively call a 'complex' one. A description of the data 'with the help of' a hypothesis means that the better the 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 (we will see a specific instance of this phenomenon in the example below). The better the fit, the more information the hypothesis gives about the data and the less bits we need to describe the data with the help of the hypothesis. The MDL Principle thus gives us a trade-off between hypothesis complexity and goodness-of-fit on the data set. We proceed with an example.

Example 1.3 [**under- and overfitting**] Let $D = ((x_1, y_1), \ldots, (x_n, y_n))$ be a sequence of data consisting of pairs (x, y) where the x and y are real numbers. We are interested in finding a functional relationship between x and y: we look for a function H such that $H(x)$ predicts y reasonably well. Again we may want to use the H we found for predicting future data. The classical statistical solution to this problem is to do a standard regression: we select the linear function that is optimal in the least-squares sense, i.e. the linear function H for which the sum of the squared errors $\sum_{1 \leq i \leq n}(H(x_i) - y_i)^2$ is as small as possible (Figure 1.1 on page x). We end up with a line that seems to capture *some* of the regularity in the data, but definitely not very much - it seems to *underfit* the data. A more interesting task is to look for the best curve within a broader class of possible candidates. For example, let us look for the best *polynomial*. A naive extension of the classical statistical solution will now typically pick a polynomial of degree $n - 1$ that completely covers the data: it will go *exactly* through all the points (x_i, y_i) (Figure 1.2). Intuitively, this may not be the polynomial we are looking for – we run a large risk of *overfitting*. Again intuitively, we might prefer a third-degree polynomial (Figure 1.3): one that has small (but not 0) error and still is relatively simple (i.e. has few coefficients). In other words, we are looking for an optimal trade-off between model complexity (i.e. the number of coefficients in the polynomial) and goodness-of-fit. The more coefficients a polynomial H has, the more bits we need to describe it. On the other hand, we can use a polynomial H to compress the description of data points (x^n, y^n). Rather than describing them literally by a list $(x_1, \ldots, x_n, y_1, \ldots, y_n)$ (encoded in binary), we encode the list $(x_1, \ldots, x_n)$ and the list of errors $(y_1 - H(x_1))^2, \ldots, (y_n - H(x_n))^2)$. If the polynomial H fits the data well, the *range* of the errors will be much smaller than the range of y_i. Therefore, we need less bits to encode the errors than to encode the y_i. The two-part code MDL Principle opts for the hypothesis that minimizes the sum of the two description lengths involved and therefore achieves the desired trade-off. It can be shown [14] that such a trade-off is not just *intuitively*, but in many cases also *provably* a good choice.

1.2 Probabilistic Preliminaries

We have informally presented the basic ideas behind the MDL Principle. To make these ideas more formal, we review some basic facts of probability theory and information theory. We first introduce much of the notation that will be used throughout this thesis:

Notational Conventions We work with probability distributions P defined over a *sample space* $\mathbf{E}$. Throughout this thesis, $\mathbf{E}$ will always be either finite, countably infinite or a subset of $\mathbf{R}^k$. Here $\mathbf{R}$ stands for the real numbers and $k \geq 1$. If $\mathbf{E} \subset \mathbf{R}^k$ then we call $\mathbf{E}$ *continuous*.

Let X^n be a random variable that can take on values in $\mathbf{E}^n$. In the case of countable $\mathbf{E}$, we abbreviate $P(X^n = x^n)$ to $P(x^n)$. In the case of continuous $\mathbf{E}$, $f : \mathbf{E} \rightarrow [0,1]$ denotes the *density function* of P. We often deal with sample spaces that are allowed to be either countable or continuous. In that case, we will give the formulas only for the countable case. The appropriate formulas for the continuous case can be arrived at by substituting $f(x)$ for $P(x)$ and replacing all sums by corresponding integrals. For example, $\sum_{x \in \mathbf{E}} P(x)\phi(x)$ becomes $\int_{x \in \mathbf{E}} f(x)\phi(x)dx$.

The definition of $\mathbf{A}^+$ and $\mathbf{A}^*$ (page 7) is extended to sample spaces $\mathbf{E}$. We use 'log' for logarithm to base two, and 'ln' for natural logarithm. For $x \in \mathbf{R}$, the *ceiling* of x, denoted by $\lceil x \rceil$, stands for the smallest integer that is greater than or equal to x. For a finite set $\mathbf{X}$, we use $|\mathbf{X}|$ to denote the number of elements in $\mathbf{X}$.

For all $D \in \mathbf{E}^*$, we denote by $L_C(D)$ the length (number of bits) of the description of D when the description is done using code C. In some cases we will use indexed codes C_i. We then use L_i as shorthand for L_{C_i}.

We extend the definition of codes to *partial* odes:

Definition 1.4 *Let* $\mathbf{A}$ *be some alphabet. Let* $\mathbf{A}'$ *be some subset of* $\mathbf{A}^+$. *A* partial code *for the set* $\mathbf{A}'$ *is a one-to-one map from* $\mathbf{A}'$ *to* $\mathbf{B}^+$.

If, for a partial code C, $C(D)$ is not defined, then we write $L_C(D) = \infty$. Henceforth, we use the word 'code' to denote both proper (as in Definition 1.2) and partial codes.

1.2.1 Probabilistic Models

Many interesting model classes are *probabilistic* in the sense that each model in the class represents a probability distribution over all possible data sequences. Examples are the class of all Bernoulli processes, the class of all Markov Chains, the class of all normal distributions etc. Intuitively, a *probabilistic model* is a probability distribution defined over arbitrarily long samples[3]. To formally define probabilistic *models* as a particular

[3]Our definition of a probabilistic model coincides with what Rissanen [128] calls an *information source* and what is called a 'random process' in probability theory.

class of probability *distributions* requires measure theory, which we want to avoid here. We therefore adopt a standard alternative definition:

Definition 1.5 *Let* $\mathbf{E}$ *be a sample space. A* probabilistic model *over* $\mathbf{E}$ *is a function* $P : \mathbf{E}^* \to [0,1]$ *such that for all* $n \geq 0$*, all* $x^n \in \mathbf{E}^n$ *we have:*

1. $\sum_{z \in \mathbf{E}} P(x^n z) = P(x^n)$ *(this is usually called the* compatibility condition*).*

2. $P(\lambda) = 1$ *where* λ *is the empty sequence.*

The two conditions above simply say that the 'event' that data $x^n z$ arrives is equivalent to the event that data x^n arrives first and data z arrives afterwards [128]. If $\mathbf{E}$ is continuous then the sum in the definition above is replaced by an integral but otherwise nothing changes.

Let P be a probabilistic model over $\mathbf{E}^*$. If for all n, all $x^n \in \mathbf{E}^n$ and $x_{n+1} \in \mathbf{E}$, $P(x_{n+1}|x^n) = P(x_{n+1})$, we say that the data are *i.i.d.* (independently and identically distributed) according to P. We call a class of probabilistic models an *i.i.d. model class* if it fully consists of probabilistic models which render the data i.i.d.

1.2.2 Connecting Codes and Probability Distributions

In this subsection we establish a very fundamental connection between codes and probability distributions; it is this connection which brings MDL into the realm of probability theory and traditional statistics.

We first reconsider the code C_1 for alphabet $\mathbf{A}_1 = \{a, b, c\}$ introduced in Section 1.1.1. An interesting feature of this code is that we do not need any commas separating the code words. This is possible because no extension of a code word can itself be a code word. Hence if we de-code from left-to-right, we always know at what point the subsequence used to encode one particular data symbol ends and the encoding of the next symbol starts, thus eliminating the need for commas. Codes with this property are usually called *instantaneous* or *prefix* codes (since no code word can be a *prefix* of any other code word).

Notice that if it were necessary to use commas in our description, we would really be working with a ternary, not a binary alphabet. So once we have decided to encode all our data in binary, we should use only zeroes and ones and no commas. We already implicitly demanded this in Definition 1.2 where a code was defined as a mapping into $\mathbf{B}^+ = \{0,1\}^+$, not $\{0, 1, `,'\}^+$. Not all such comma-free codes are also prefix codes. However, one can show (see [30], Theorem 5.2.2, combined with Theorem 5.5.1) that, for every comma-free code $C : \mathbf{A}^+ \to \mathbf{B}^+$, there exists a prefix code C' such that for all $D \in \mathbf{A}^+$, $L_C(D) = L_{C'}(D)$. Since MDL is only concerned with code*lengths* rather than actual encodings, it follows that we may restrict ourselves to prefix codes without any loss of generality. Henceforth, whenever we use the word 'code', we therefore really mean a 'prefix code'. One can relate prefix codes to probability distributions, and this has some very important consequences. At this point, the reader who only wants to get a quick overview of MDL may wish to skip the technicalities below and only read about these consequences; they are summarized in Section 1.2.3.

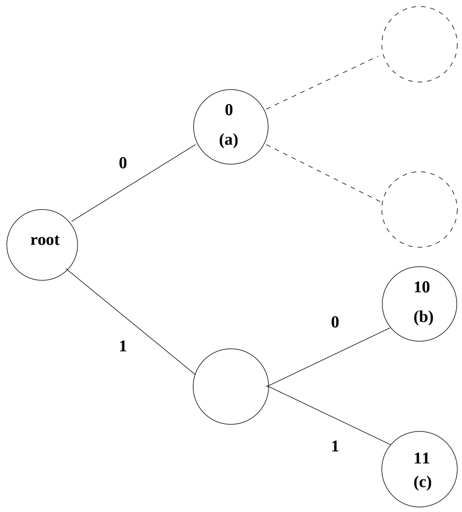


Figure 1.1: Binary code tree for the Kraft inequality using alphabet $\{a, b, c\}$ and code $C_1(a) = 0; C_1(b) = 10; C_1(c) = 11$

The Kraft Inequality

As was discussed on page 8, a crucial property of codes is the fact that each code for $\mathbf{E}$ can assign short code lengths only to very few elements in $\mathbf{E}$.

Probability distributions P over $\mathbf{E}$ have the property that $\sum_{x \in \mathbf{E}} P(x) = 1$. Hence, each P can only assign high probability to very few elements in $\mathbf{E}$.

There is an analogy here. Indeed, it turns out that short code lengths can be formally related to high probabilities. The exact relation is given by the *Kraft Inequality* [88]:

Theorem 1.6 (Kraft Inequality) *For any instantaneous code (prefix code)* $C : \mathbf{A} \to \mathbf{B}^*$ *for a finite alphabet* $\mathbf{A} = \{1, \ldots, k\}$*, the codeword lengths* $L_C(1), \ldots, L_C(k)$ *must satisfy the inequality*

$$\sum_{x \in \mathbf{A}} 2^{-L_C(x)} \leq 1. \tag{1.4}$$

Conversely, given a set of codeword lengths that satisfy this inequality, there exists an instantaneous code with these word lengths.

Proof: [adapted from [30]] We write l_{max} for the length of the longest codeword in the set of codewords: $l_{max} = \max_{x \in \mathbf{A}} L_C(x)$. We write l_i as an abbreviation of $L_C(i)$. Consider the full binary tree of depth l_{max}, where each branch is associated with either 0 or 1 (see Figure 1.1). In this tree, each codeword corresponds to a unique path starting at the root node: the first (leftmost) bit of the codeword determines which of the two children of the root node is visited; the second bit determines which of the two children of the first visited node is visited etc. The length of the path is equal to the length of the codeword. As an example, Figure 1.1 shows the paths for our example code C_1 of the previous subsection. With each codeword we associate the node at the end of the path that coincides with the codeword. For example, the node '10' is the node at the end of the path taking branch 1 at the root node and then taking branch 0.

We define the i-th *level* (denoted by LEVEL_i) of the tree to be the set of all nodes of the tree at depth i. The prefix condition on the codewords implies that no codeword is an ancestor of any other codeword on the tree. Hence, each codeword eliminates its descendants as possible codewords. The codeword i, together with its descendants, defines a subtree of possible extensions of the codeword up to depth l_{max}. If $l_i = l_{max}$, this subtree coincides with the node i itself. If $l_i = l_{max} - 1$, the subtree consists of the node i together with its two children, and so on. Let D_i be the set of leaves of the subtree belonging to i. We have $2^{-l_i} = |\text{D}_i| \cdot 2^{-l_{max}}$. Also $|\text{LEVEL}_{l_{max}}| 2^{-l_{max}} = 2^{l_{max}} 2^{-l_{max}} = 1$. Since the prefix property implies that all D_i are disjoint, we have

$$\sum_{i \in A} 2^{-L_C(i)} = \sum_{1 \leq i \leq m} 2^{-l_i} = \sum_{1 \leq i \leq m} |\text{D}_i| \cdot 2^{-l_{max}} \leq |\text{LEVEL}_{l_{max}}| 2^{-l_{max}} = 1$$

which implies the Kraft inequality (1.4).

Conversely, given any set of codeword lengths $l_1, \ldots, l_m$ satisfying the Kraft inequality, we can always construct a tree like the one in Figure 1.1. Label the first node (lexicographically) of depth l_1 as codeword of 1. Label the first node (lexicographically) that has depth l_2 and that is not a descendant from the node 1 as codeword of 2 etc. In this way, we construct a prefix code with the specified $l_1, \ldots, l_m$. $\Box$

Let $\mathbf{E}$ be finite and C be a code for $\mathbf{E}$. From the Kraft inequality we immediately see that there exists a probability distribution P defined over the space $\mathbf{E} \cup \{\Box\}$ such that for all $x \in \mathbf{E}$, $P(x) = 2^{-L_C(x)}$ (we need the extra symbol '$\Box$' to ensure that $\sum_x P(x) = 1$).

Similarly, let P be a probability distribution over the finite space $\mathbf{E}$. By the Kraft inequality (1.4) we see immediately that there exists a code C for $\mathbf{E}$ such that for all $x \in \mathbf{E}$: $L_C(x) = \lceil -\log P(x) \rceil$. This code is called the *Shannon-Fano code* [30].

We will ignore the effect of rounding up and drop the integer requirement for code lengths. Once we have done this, we obtain a *a correspondence between probability distributions and prefix codes*: to every P, there is a corresponding C such that the code lengths $L_C(x)$ are equal to $-\log P(x)$ for *all* $x \in \mathbf{E}$. At the same time, for every C there is a corresponding P such that the code lengths $L_C(x)$ are equal to $-\log P(x)$ for *all* $x \in \mathbf{E}$.

Since an analogue to the Kraft Inequality can be proven for countably infinite alphabets, the correspondence still holds for countably infinite $\mathbf{E}$ ([30]). We now consider the case where $\mathbf{E} \subset \mathbf{R}^k$. As first noted on page 7, in practice we will always be confronted with data that is recorded up to a finite precision d. That is, instead of using a probability distribution P with density function P defined over $\mathbf{E}$, in reality we use a discretized version P' of P, with probability mass function P', defined on a large but finite subset $\mathbf{E}'$ of $\mathbf{E}$. For this P' and all $x \in \mathbf{E}'$, we have $P'(x) = (1/K)P(x)$ for some constant K which depends on the precision that is being used. Using the code lengths of the code C' corresponding to P' we find that $L_{C'}(x) = -\log P(x) + \log K$ for all $x \in \mathbf{E}'$. Since $\log K$ does not depend on x, we may safely neglect it in our analyses (see [128] for details). We can then once more regard $-\log P(x)$ as an 'idealized' code length.

1.2.3 Identifying Codes and Probability Distributions

The above can be summarized as follows:

Correspondence between Probability Distributions and Prefix Codes
Let $\mathbf{E}$ be a sample space. Let P be a probability distribution over $\mathbf{E}^n$, the set of sequences of length n. Then there exists a prefix code C for $\mathbf{E}^n$ such that for all $x^n \in \mathbf{E}^n$, $L_C(x^n) = -\log P(x^n)$. C is called the *code corresponding to* P.
Similarly, let C' be a prefix code for $\mathbf{E}^n$. Then there exists a probability distribution P' such that for all $x^n \in \mathbf{E}^n$, $-\log P'(x^n) = L_{C'}(x^n)$. P' is called the *probability distribution corresponding to* C'.

The correspondence allows us to *identify* codes and probability distributions, such that a short code length corresponds to a high probability and vice versa.

In this correspondence, probability distributions are treated as mathematical objects and *nothing else*. If we use a code C to encode our data, this definitely does *not* necessarily mean that we assume our data is drawn according to the probability distribution corresponding to C.

The identification of probability distributions and codes allows for a probabilistic reinterpretation of the *minimum description length* principle as a *maximum probability* principle, since short code lengths correspond to high probabilities and vice versa. This reinterpretation will be worked out in the sections to come.

1.3 Formalizing the Two-Part Code

Let $\mathcal{M}$ be some class of models and let D be a given data sequence. As we saw in Section 1.1.4, the (two-part code) MDL Principle tells us to look for the hypothesis $H \in \mathcal{M}$ that minimizes the sum of the description length of H plus the description length of the data when encoded with the help of H. This means that two different codes are involved: a code C_1 to encode our hypotheses, and a code C_2 to encode our data D using the hypothesis. In mathematical notation, the MDL Principle then becomes: *we should pick the* $H_{mdl} \in \mathcal{M}$ *with*

$$H_{\mathrm{mdl}} = \arg\min_{H \in \mathcal{M}} \{L_{C_2}(D|H) + L_{C_1}(H)\} \tag{1.5}$$

Here the notation $\arg\min f(x)$ denotes the x minimizing $f(x)$. If there is more than one $H \in \mathcal{M}$ for which $L_{C_2}(D|H) + L_{C_1}(H)$ is minimized, we pick the one with minimum hypothesis complexity $L_{C_1}(H)$. If this still leaves us with several possible H, we do not have a further preference among them. For each different H, C_2 will encode the data D in a different way: it encodes it 'with the help of H'. Therefore, C_2 is not really one single code but rather a function that maps each $H \in \mathcal{M}$ to a different code $C_2(\cdot|H)$. The notation $L_{C_2}(D|H)$ in Equation. (1.5) is used to denote the length of encoding $C_2(D|H)$.

The single code over data sequences D that results from coding each D using the H_{mdl} that is optimal for this specific D is called the *two-part MDL-code*. We denote the code lengths resulting from this code by $L_{\text{2-p}}(\cdot|\mathcal{M})$:

$$L_{\text{2-p}}(D|\mathcal{M}) = L_{C_2}(D|H_{\mathrm{mdl}}) + L_{C_1}(H_{\mathrm{mdl}}) \tag{1.6}$$

where H_{mdl} depends on D and is given by (1.5).

1.3.1 Two-Part Codes for Probabilistic Model Classes

Most of traditional statistics is concerned with classes consisting of probabilistic models. We assume such classes to be parameterized by a vector θ coming from some set of possible parameter vectors Γ. The class of models can then be written as

$$\mathcal{M} = \{P(\cdot|\theta) \mid \theta \in \Gamma\}$$

where the $P(\cdot|\theta)$ are all probabilistic models for the same sample space $\mathbf{E}$. $P(\cdot|\theta)$ can be read as 'the probability of the data given that the model used is θ'; examples will be given below. Notice that, strictly speaking, θ does not denote a model but a *name* for a model. This difference will become relevant in later chapters. For now, we simply use H to denote non-probabilistic models and θ to denote probabilistic models and write, with some slight abuse of notation, $\theta \in \mathcal{M}$ instead of $\theta \in \Gamma$.

By the remarks in Section 1.2.3, we know that there exists a code C such that $-\log P(D|\theta) = L_C(D)$ for all D of length n. It is this code we will use for C_2: for every $\theta \in \mathcal{M}$, we get $L_{C_2}(D|\theta) = -\log P(D|\theta)$ for all D. Notice that this choice for C_2 gives a short code length to sequences which have high probability according to θ while it gives a high code length to sequences with low probability. The code length thus reflects the goodness-of-fit of the data with respect to θ, since the higher the probability of D according to θ, the better θ fits D. As explained at the beginning of this section, our aim is a trade-off between goodness-of-fit and complexity. This makes the choice of C_2 such that $L_{C_2}(D|\theta) = -\log P(D|\theta)$ a natural one. Whether this is really the only reasonable possibility for C_2 is a subtle issue. It will be discussed further in Chapter 2 (Sections 2.2) and then, at length, in Chapter 5, Section 5.4). For now, we assume C_2 is defined as indicated above. The MDL Principle for probabilistic model classes thus tells us to pick the following model θ_{mdl} when given data D:

$$\theta_{\mathrm{mdl}} = \arg\min_{\theta \in \mathcal{M}} \{-\log P(D|\theta) + L_{C_1}(\theta)\} \tag{1.7}$$

where C_1 still has to be specified.

MDL and ML In the well-known method of *Maximum Likelihood (ML) Estimation* (see any introductory textbook on statistics, for example [6]), we estimate a probability distribution from a given set of data D by taking the probabilistic model θ that maximizes the probability of D. We denote this θ by $\hat{\theta}$. Sometimes we write $\hat{\theta}(D)$ rather than $\hat{\theta}$, in order to stress that $\hat{\theta}$ actually depends on D. If we pick our models from model class $\mathcal{M}$, we can write

$$\hat{\theta}(D) = \hat{\theta} = \arg\max_{\theta \in \mathcal{M}} P(D|\theta) = \arg\min_{\theta \in \mathcal{M}} \{-\log P(D|\theta)\} \tag{1.8}$$

where the last equality follows from the fact that log and '$-$' are monotonous transformations. Combining (1.7) and (1.8), we see that the ML estimator $\hat{\theta}$ can be seen as an approximation of the *MDL estimator* θ_{mdl}: the MDL estimator becomes the ML estimator $\hat{\theta}$ if we neglect the complexity term $L_{C_1}(\theta)$.

Example 1.1 (continued) We consider the class of *Bernoulli models* for the binary sequences given in Example 1.1 on page 5. Each model is identified with a parameter $\theta \in [0, 1]$:

Definition 1.7 *Let* $\mathbf{E} = \{0, 1\}$. *The Bernoulli model* $P(\cdot|\theta)$ *is defined as follows: for all* n, *all* $x^n \in \mathbf{E}^n$, $P(X_{n+1} = 1|X^n = x^n) = P(X_1 = 1) = \theta$.

In other words, we regard the data as being generated by independent tosses of a biased coin with probability θ of coming up 'heads' (which we identify with '1'). Let us consider sequence (1.3) of Section 1.1 again. This sequence contains four times as many 0s than 1s. Clearly, the ML estimator for this sequence is $\hat{\theta} = 1/5$. In general, for a sequence D of length n with a fraction of γ 1s, we have $\hat{\theta}(D) = \gamma$: the ML Bernoulli model coincides with the frequency of 1s. We have:

$$-\log P(D|\hat{\theta}(D)) = -\log \gamma^{\gamma \cdot n}(1-\gamma)^{(1-\gamma)\cdot n} \approx \log \binom{n}{\gamma n}. \tag{1.9}$$

where the rightmost (approximate) equality will be shown to hold in Chapter 3, Section 3.5, page 59. In Section 1.1, we suggested to code sequence (1.3) by first stating γ, the frequency of 1s in the data string D, and then giving its index in the lexicographical ordering of all sequences with the same fraction γ of 1s. We saw that the second part of this code would take approximately $\log \binom{n}{\gamma n}$ bits. We now see that if we code the data D using the code corresponding to a Bernoulli model, we are really doing something similar: we first, using code C_1, encode the number $\hat{\theta}$, which corresponds to the frequency of 1s in D. Then we code the data 'with the help of' $\hat{\theta}$, which takes approximately $\log \binom{n}{\hat{\theta} \cdot n}$ bits, about the same amount of bits as it would take to specify the index in the ordering referred to above.

However, there is also a big difference between coding by giving an index and coding using the code corresponding to a distribution θ: in the second case, we can also encode data strings D' for which the frequency of 1s is *not* equal to θ, and this will take us $-\log P(D'|\theta)$ bits. To fully describe the data D, we also need to describe the parameter θ_{mdl} itself. θ_{mdl} is *not* necessarily equal to $\hat{\theta}$. To encode $\hat{\theta}$, we need to work with a fairly high precision. Often, there is another $\theta \in \mathcal{M}$ for which $-\log P(D|\theta)$ is just a little lower but which can be encoded using much lower precision. For such a θ, the sum (1.7) will be lower than for $\hat{\theta}$. We now show how to encode values of θ.

The Code C_1 - Descriptions as Messages To explain the code C_1, it will be useful to interpret descriptions as messages. We can always think of a description as a message that some sender or *encoder*, say Mr. A, sends to some receiver or *decoder*, say Mr. B. Before sending any messages, Mr. A. and Mr. B. meet in person. They agree upon a description method that will be used by A to send his messages to B. Once this has been done, A and B go back to their respective homes and A sends his messages to B in the form of binary strings. The fact that a description method must be one-to-many (page 7) implies that, when B receives a message, he should always be able to decode it in a unique manner.

In the 2-part coding scheme, A first sends a parameter value $\theta \in \mathcal{M}$ to B; only then does he send an encoding of the data D. If $\mathcal{M}$ contains only one single model θ_1, then B knows beforehand what A wants to send. So A does not have to send anything and the description of θ_1 takes 0 bits.

If we consider a model class containing a finite number of elements k, then there are k different possible messages. Hence A and B must agree on a description method that assigns different encodings to all these k messages. If $k = 2^d$ for some d, then there exists a description method for doing this that assigns code length d to all k possibilities. Similarly, if, as in our Bernoulli example, we want to describe a parameter $\theta \in \mathbf{R}$, we have to truncate θ to a finite precision. If we use a fixed precision d, then we need d bits to send one of the 2^d possible truncated parameter values.

Since, for every $\theta \in \mathcal{M}$ ($\mathcal{M}$ still denoting the Bernoulli models), we would like to be able to describe parameter values that are as close to θ as we want, we need to work with a precision that is *not* fixed. Therefore, when sending a value θ_d, we first need to encode the particular value d with which θ will be encoded, otherwise a decoder would not know how many bits the description of θ would take. We have to encode d using a prefix code, otherwise the decoder cannot decide once more at what point he has finished decoding the number d and should start decoding the value of θ itself. To see how to code d, first consider a primitive prefix code which encodes the natural numbers $n > 0$ as follows: the number n is encoded as $0^{n-1}1$, i.e. $(n-1)$ 0s followed by a 1. We call this the *trivial* code for n. Clearly, this code has the prefix property. We can turn this code into a more efficient one that is still prefix as follows: we first encode, in the 'trivial' way, the *length* that the description of n would have if it were written in binary in the standard (non-prefix) manner (that is, we write '1' as 0, '2' as 1, '3' as 10 and so on). This length is $\lceil \log n \rceil$. We let this first encoding be followed by the encoding of n in this standard non-prefix manner. This 2-part message will take $2\lceil \log n \rceil$ bits. A decoder can now first decode the length of n; he then knows how many bits the coding of n will take so he can start decoding it and will know exactly when he is done decoding it: the resulting code still has the prefix property. Hence the number n is encoded using $2 \cdot \lceil \log n \rceil$ bits. But we can also repeat the trick, and first encode *the length of the length* of n in the trivial manner, and then encode first the length of n and then n itself in the non-prefix free manner. Again, the resulting code will be prefix free. In this way, we need $2 \cdot \lceil \log \lceil \log n \rceil \rceil + \lceil \log n \rceil \approx 2 \log \log n + \log n$ bits.

If we encode the precision d in this way, the decoder can first decode d. He then knows that d more bits follow for the parameter value itself, so he also knows at what point he will be done decoding θ and can start decoding the data using θ: the resulting code C_1 takes

$$L_{C_1}(\theta) \approx d + \log d + 2 \log \log d \text{ bits.} \tag{1.10}$$

One can show that asymptotically, the optimal precision d for encoding a sample of size n is given by

$$d(n) = \frac{1}{2} \log n + c \tag{1.11}$$

where c is a constant that can also be computed ([128], page 57). So, alternatively, in our encoder-decoder setting, sender and receiver may opt for a code in which the precision is given by (1.11). Then the receiver always knows what precision the sender will use in coding the value θ and can uniquely decode it (we assume here that the sample size n is known to both sender and receiver in advance). In this way, the code length $L_{C_1}(\theta)$ becomes equal to $d(n)$ as given in (1.11): coding the parameter θ takes $1/2 \log n + c$ bits, while d itself does not have to be encoded any more, so the $\log d + 2 \log \log d$ term disappears. For large n, (1.11) will lead to a slightly shorter code length than the code with lengths (1.10).

In this case of a model class with a single parameter, $L_{C_1}(\theta)$ (growing logarithmically in n) will in general be much smaller than $L_{C_2}(D|\theta)$ which clearly grows linearly in n for all $0 < \theta < 1$. Hence, neglecting $L_{C_1}(\theta)$ will not lead to much trouble. This is in sharp contrast with the many-parameter setting, as shown in the next example.

There is one important detail in the coding procedure that we have glossed over so far: how do we map parameters to their encoded values? If we use precision $d = 2$, we can encode four possible parameter values $(\theta_{00}, \theta_{01}, \theta_{10}, \theta_{11})$. Which four should we take? $(0, 1/3, 2/3, 1)$? Or $(1/5, 2/5, 3/5, 4/5)$? Or something else? This tricky question has been given a full answer only very recently [129]; see also Chapter 7. In most practical situations, codes with equal spacing between the parameter values will work reasonably well.

1.4 Two-Part Codes for Non-Probabilistic Model Classes; Supervised Learning

We now discuss how to construct two-part codes for non-probabilistic model classes. As in the probabilistic case, we must first construct the code C_2 and then the code C_1, where C_1 and C_2 are used as in Equation 1.6. We show how to do this using our polynomial example:

Example 1.3 (continued) Let the model class $\mathcal{M}$ consist of the class of all polynomials. We seek a polynomial H that provides a good trade-off between complexity and goodness-of-fit as measured by the total squared error $\sum_{1 \leq i \leq n} (H(x_i) - y_i)^2$. In Section 1.5 (and more completely in Chapter 5, Section 5.3.5) we discuss why it makes sense to take the squared error function here.

Supervised Learning The code C_1 will be used to encode polynomials; the code C_2 will be used to encode data $D = (x^n, y^n)$ with the help of a polynomial H. Since we are only interested in finding out how the y_i *depend* on the x_i, we may regard the x_i as given. Indeed, after one has selected a polynomial H on the basis of (x^n, y^n) and one is asked to use it to predict future data, one will usually also be *given* some new value of x and one will then predict y as $H(x)$. Learning functions $H : \mathbf{E}_x \to \mathbf{E}_y$ with the goal of predicting y-values on the basis of $H(x)$ is called *supervised learning* in the machine learning/neural network literature [72]; it corresponds to *regression* in statistics [6].

Regarding the x_i as given means that we do not have to encode them[4]. In the encoder-decoder setting introduced in the previous section, this corresponds to the situation that $x_1, \ldots, x_n$ are known both to encoder and decoder.

The Code C_2 In the case of probabilistic model classes, we turned each model into a code such that high probability of the data corresponded to a short code length of the data; our goal in doing this was to get the code length of data D to reflect the goodness-of-fit of D with respect to the hypothesis: the higher the probability of the data according to a hypothesis, the better the hypothesis *fits* the data. In the present (polynomial) example, goodness-of-fit is defined in terms of squared errors: polynomial H gives a better fit on the set of points (x_i, y_i) than on the set of points (x_i, z_i) iff $\text{ER}_{sq}(y^n|H, x^n) < \text{ER}_{sq}(z^n|H, x^n)$ where ER_{sq} is the total squared error: $\text{ER}_{sq}(y^n|H, x^n) = \sum_{i=1}^{n}(y_i - H(x_i))^2$. In analogy to the case of probabilistic models, we would like the description length of the data $y_1, \ldots, y_n$ given hypothesis H and $x_1, \ldots, x_n$ to reflect how well H fits data (x^n, y^n): the shorter the code length, the better the fit. This will be achieved if we can manage to construct a code C_2 such that for all $D = ((x_1, y_1), \ldots, (x_n, y_n))$ and all H:

$$L_{C_2}(y_1, \ldots, y_n | H, x_1, \ldots, x_n) = \text{ER}_{sq}(y^n | H, x^n) + K \tag{1.12}$$

where K is a constant that may depend on n but *not* on H or any of the y_i. To see why a code with lengths given by (1.12) is what we want, note that (1.12) implies for all $(y_1, \ldots, y_n)$ and all $(z_1, \ldots, z_n)$ that

$$\begin{array}{rcll} \Delta L_{C_2} & = & L_{C_2}(z^n|H, x^n) - L_{C_2}(y^n|H, x^n) & = \\ = \Delta\text{ER}_{sq} & = & \text{ER}_{sq}(z^n|H, x^n) - \text{ER}_{sq}(y^n|H, x^n) & \end{array} \tag{1.13}$$

which means that the difference in goodness-of-fit between every two datasets is precisely reflected in the difference in code length. It turns out that we can indeed construct such a code C_2. The reason is that for each $H \in \mathcal{M}$ there exists a probability distribution $P(\cdot|H)$ such that, for all (x^n, y^n),

$$-\log P(y^n|H, x^n) = \text{ER}_{sq}(y^n|H, x^n) + K \tag{1.14}$$

We know from Section 1.2.3 that the existence of $P(\cdot|H, x^n)$ implies the existence of a code $C(\cdot|H)$ with code lengths given by $L_C(y^n|H, x^n) = \text{ER}_{sq}(y^n|H, x^n) + K$. Below

[4] Encoding the x_i after all does not change anything really essential though, as will be shown at the end of this section.

we verify that the probability distribution P which achieves this for us is simply the n-fold product distribution of a conditional

Gaussian (normal) distribution. More precisely, it has density function f given by:

$$f(y^n|H,x^n) = \prod_{i=1}^{n} f(y_i|H,x_i) \text{ where}$$
$$f(y|H,x) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(y-H(x))^2}{2\sigma^2}) \text{ with variance } \sigma^2 = (2\ln 2)^{-1}. \tag{1.15}$$

Hence, conditioned on x_i the y_i are independently normally distributed with mean $\mu_i = H(x_i)$ and constant variance σ^2. We can now verify at once that (1.14) holds:

$$-\log f(y^n|H,x^n) = -\sum_{i=1}^{n} \log f(y_i|H,x_i) = \text{ER}_{sq}(y^n|H,x^n) + K$$

where both equalities follow by plugging in the expressions for $f(y_i|H,x_i)$ given in (1.15).

The Code C_1 and the Sum Encoding a polynomial $H_{k,d}$ of the $(k-1)$-st degree up to d bits precision per parameter is done by a straightforward extension of the procedure used for Bernoulli models: we first code d and k in a prefix-free manner, and then we encode the k parameter values using kd bits. We omit the details. Combining the resulting L_{C_1} with L_{C_2} as given by (1.12), the sum of the two description lengths becomes:

$$L(y^n;H_{k,d}|x^n) = \overbrace{\sum_{i=1}^{n}(y_i - H_{k,d}(x_i))^2}^{\text{error term}} + \overbrace{2(\log\log k + \log\log d) + \log kd + kd}^{\text{complexity term}} + K \tag{1.16}$$

Here K is a term that does not depend on $H_{k,d}$ and that therefore does not play any role when finding the $H_{k,d}$ that minimizes (1.16) (but see below!). According to the MDL Principle, we should now prefer the $H_{k,d}$ for which (1.16) is minimized. This is the optimal trade-off between the error term $(y_i - H_{k,d}(x_i))^2$ and the complexity terms involving k and d. The overall optimum depends on both the *degree* of the polynomial and the *precision* with which to encode the parameter values.

If we want to optimize the behaviour of MDL, the constant K becomes important after all. Matters get a lot more complicated once we start involving K, and we defer their discussion to Chapter 5, Section 5.2.

Let $\alpha_1, \ldots, \alpha_k$ be the parameters of a $(k-1)$-st degree polynomial. Just like in the

Bernoulli example, it can be shown that asymptotically, the optimal precision for each parameter α_i is given by $d_i = 1/2 \log n + c_i$ for some constant c_i ([128], page 56). Using this precision (which can be derived from the sample size n) the value d does not have to be encoded any more and (1.16) becomes:

$$L(y^n; H_{k,d}|x^n) = \sum_{i=1}^{n} (y_i - H_{k,d}(x_i))^2 + \tfrac{k}{2} \log n + 2 \log\log k + \log k + K' \tag{1.17}$$

for some constant K'.

When x_i are not regarded as given The MDL Principle tells us that we should find a short description for *all* our data (Section 1.1.1). One may argue that, since the data include x^n, we should also describe x^n which would lead to a code length larger than (1.16). However, this will still lead us to pick the same optimal $H_{k,d}$ as shown by the following argument: suppose we first encode the x_i using some code C_3. We then code a polynomial H using code C_1, followed by an encoding of the y_i based on the code C_2. We then get as a total code length:

$$L(x^n, y^n; H_{k,d}) = L(y^n; H_{k,d}|x^n) + L_{C_3}(x^n) \tag{1.18}$$

where $L(y^n; H_{k,d}|x^n)$ is as in (1.16). Clearly, for every fixed D, the $H_{k,d}$ minimizing (1.18) coincides with the $H_{k,d}$ minimizing (1.16).

Semi-Probabilistic Interpretation of Non-probabilistic Model Classes

The squared error function led to a specific code C_2. We have seen that this code coincided with the code corresponding to a distribution that makes the y_i independently distributed around $H(x_i)$ with Gaussian noise of a specific variance. It *seems* as if using the squared error function implies an implicit assumption of Gaussian noise. But this is not true: at this point we should stress once again that this probability distribution is just a means of describing, or equivalently *modeling* our data. We discuss in the next section (and prove in Chapter 5) that even if the data is *generated* (drawn) according to a completely different distribution, it may still be useful and harmless to *model* it using a normal distribution.

1.4.1 MDL and the model that best fits the data

It can be shown [128] that an analogue to Equation 1.17 holds in great generality; for all sufficiently regular model classes the optimal number of bits needed to encode the parameters is $1/2 \log n$ for each parameter. If we use a class $\mathcal{M}_k$ containing only models with a fixed number of parameters k (for example, the class of $(k-1)$-st degree polynomials), then the number k does not have to be encoded, and we get:

$$L(y^n; \theta_d|x^n) = -\log P(y^n|\theta_d, x^n) + \frac{k}{2} \log n + O(1) \tag{1.19}$$

This formula can be used both for the case of probabilistic and non-probabilistic model classes; for, as we shall see in the next chapter, the trick we applied to the squared error to turn our hypotheses into codes, and hence probability distributions, can be employed in general.

From (1.19) we see that for a class with a fixed number of parameters, as n increases, the model in $\mathcal{M}$ that maximizes the likelihood and the model θ_{mdl} in $\mathcal{M}$ that minimizes the description length (1.19) *converge*: for increasing n, the precision with which parameters θ_d are encoded becomes higher and higher, while the complexity term $k/2 \log n$ is independent of θ_d if k is fixed. Therefore, θ_{mdl} and $\hat{\theta}$ must become closer and closer.

However, if k is not fixed (for example, when the model class of *all* polynomials is used), and when the data comes from a noisy source, the model in $\mathcal{M}$ that maximizes the likelihood or, if a non-probabilistic model class is used, minimizes the error will generally 'jump around' as the size of the data increases: one can show that if we have k data points, we will generally select a k-th degree polynomial (as, for example, in Figure 1.2 on page x).

In such a case though, the MDL estimator will keep choosing simple polynomials. One can show that if the data are truely 'generated' by some polynomial H^* of degree k with Gaussian noise, then the MDL estimator will generally, once the size of the data set becomes large enough, converge to the 'true' polynomial H^* with probability 1 [14]. However, if the true degree k is high, then for small samples, MDL will usually 'underfit': it will pick a polynomial of degree lower than k. This can be seen from Equation 1.17: the lower error term for the high degree polynomial does not yet outweigh its higher complexity term. In the next section we will consider the question of whether such a provisional, overly simple model can still be useful.

1.5 MDL is Looking for a Good, not for a True Model

indexreliable!prediction We just saw that, if not enough data is available, MDL will sometimes pick a model that is 'too simple'. It is important to realize that this does *not* mean that MDL assumes that 'simple models are a priori more likely to be true', or that 'nature prefers simplicity' or anything like that: the rationale behind picking only simple models when few data is available is rather that *the data set is too small to identify a complex model with any reliability*. This is nicely illustrated for *very* complex models in Figure 1.2 on page x: if the model class $\mathcal{M}$ contains just as many parameters as your sample size n, then for each possible realization of the data you will find a model with a perfect fit to the data; if the data is generated by a process that is prone to small random fluctuations, such a model clearly does not give you any information about the data: the probability that you will hit upon the right model is about 0. On the other hand, if you pick the model that best fits the data within a small (simple) enough model class then the model you pick will typically be close to the model *within your class* that will give the best predictions of future data (in Chapter 2, Section 2.7 we make this more precise and consider how the error you make when predicting on the basis of a model inferred from a k-dimensional class depends on k).

The fact that MDL may pick an overly simple model if only few data are available raises the question whether we can 'provisionally' use the overly simple model to make reasonable predictions of future data. We give an example:

Example 1.8 [Bernoulli and Markov] Suppose you are given a sequence of 0s and 1s where the 0s and 1s are not really independent. You are not aware of this fact, and you decide to model your data using the Bernoulli model class (Definition 1.7) which treats the data items as if they were independent.

To make the example concrete, suppose the data are really distributed according to a first-order Markov chain [47]: a distribution such that the probability of 1 at the $(i+1)$-st outcome depends on the i-th outcome. Let us assume that the 'true' Markov chain P^* is given by

$$P^*(X_{i+1} = 1 | X_i = 1) = \frac{2}{3} \quad ; \quad P^*(X_{i+1} = 1 | X_i = 0) = \frac{1}{6} \tag{1.20}$$

It is easy to show that under such a distribution the frequency of 1s in a sequence $x_1, x_2, \ldots$ will, as its length n increases, tend to $1/3$ with probability 1.

Therefore, if enough data is available and you model the data using the Bernoulli model class, your optimal model $\theta_{\mathrm{mdl}}(x^n)$ will converge to $1/3$ with probability 1 (recall that the MDL and the maximum likelihood estimator converge). On the other hand, if data were truely distributed according to a Bernoulli distribution with $P(X = 1) = 1/3$, then your model would also, with probability 1, converge to $\theta_{\mathrm{mdl}}(x^n) = 1/3$: in both cases you would end up with the same optimal model, which, also in both cases, you would consider to be a reasonable model for your data since it allows for considerable compression: you would need approximately

$$-\frac{n}{3} \log \frac{1}{3} - \frac{2n}{3} \log \frac{2}{3} + O(\log n) \tag{1.21}$$

bits to code the data[5] (see page 18), which is smaller (by a linear amount) than n, the number of bits you would have needed if you had encoded the data literally.

However, in the case where data are distributed according to the Markov chain, you would be able to achieve a lot of additional compression if you had used the model class consisting of all Markov chains rather than the Bernoulli model class. One can show [30] that with probability 1, coding the data using the code based on the 'true' Markov chain will allow for additional (by an amount linear in n) compression of the data. The first-order Markov chain itself can be encoded by two parameters. Again, the optimal precision for doing this will be on the order of $1/2 \log n$ per parameter, which means that as n gets larger, the total two-part codelength based on the class of Markov chains will be smaller than the two-part code length based on the Bernoulli model class by an amount linear in n.

In practice one will often use a model class $\mathcal{M}_1$ and not be aware that there exists another model class $\mathcal{M}_2$ with which one can achieve much more compression. The

[5] Equation 1.21 needs some additional justification since both the model $\theta_{\mathrm{mdl}}(x^n)$ and the frequency of 1's will, for finite n, not be completely equal to $1/3$. However, using results from [26] it is not hard to show that asymptotically, (1.21) will indeed hold with probability one.

only way to avoid this would be to use the class of all computer programs relative to some universal computer language; but we have already seen that we cannot do that in practice. If one uses a more restricted class, it can always happen that one does not capture all regularity in the data (Section 1.1.3): if one uses the Bernoulli model class, it may be that the class of all Markov chains provides a better (more compressing) model[6]. If one uses the class of all Markov chains, then it may be that some completely different model class achieves more compression. One can of course 'lump together' several different model classes into one big model class, but in general, it will simply be unknown what the optimal model class is for the data at hand and one will use a suboptimal one.

Such a suboptimal model class will contains models *that allow for substantial compression of the data but that are (a) not in any respect 'true' and (b) lead to much less compression of the data than a shortest computer program that prints the data and then halts.*

According to Rissanen's MDL Philosophy [128], this is the situation we will typically be in - and we agree. In fact, in every application of MDL we know of, the model class is immediately seen to be not completely correct or even clearly incorrect. A concrete example of this can be found in part II of this thesis, where we use the very simple 'naive Bayes' model class to model economic, biological and medical data, which obviously has not been generated by any 'naive Bayes' model. Nevertheless, the naive Bayes model allows us to discern regularities in the data which can be used to predict and classify future data with surprisingly high accuracy.

The MDL philosophy even goes one step further and says that there is no such thing as a 'true model' at all: all our models will always be wrong to some extent. Nevertheless, it is of interest to study the hypothetical situation where some 'true model' exists but is quite different from any of the models in our class. If we can show that MDL is well-behaved in such a situation, then this can serve as an indication that it will work well in practical settings. In this thesis we will study the situation where data are distributed according to some unknown P^* that is not necessarily related to any of the models in the model class $\mathcal{M}$ used (the models in $\mathcal{M}$ may even be non-probabilistic). Such a situation has been studied before [14, 167]. It is relatively easy to show that, under quite general conditions, the MDL estimator $\theta_{\text{mdl}}(x^n)$ will converge to the model $\tilde{\theta}$ in $\mathcal{M}$ that in some precisely defined sense is closest to P^*.

However, one may ask what will happen if one uses such a model $\tilde{\theta}$ - which is partially wrong - instead of P^* for *predicting* aspects of future data. It is this question which we will study further in chapters 4 and 5. Roughly, in those chapters we show that a model that is quite different from the 'true' P^* can nevertheless be 'safe' to use in the sense that it gives you a correct impression of how well you can predict future data. Let us assume once more, as in Example 1.8, that data are generated by a Markov chain but modeled by the Bernoulli class. If we used the model $\theta_{\text{mdl}}(x^n) = 1/3$ to predict the value of the next bit, we would predict that a 0 will occur rather than a 1. Our model tells us that this will be right in about 2/3 of the cases. If future data

[6] Of course there are also cases where one has additional reasons for assuming that the data are independent (for example, they are outcomes of a repeatable experiment). But generally speaking it is nearly always the case that there may exist some alternative model class for the data at hand that is better than the one one is using.

will still be distributed according to the Markov chain, then this will give a correct impression of the frequency of mistakes we will make. As is clear from (1.20), if we use the 'true' Markov chain instead for predicting future data, we will predict a 1 if our last seen outcome was a 1 and a 0 if it was a 0. This will lead to a much lower frequency of error in predicting - the complex model is better. However, using the simpler Bernoulli model we never *expected* to predict better than with accuracy 1/3. Therefore, though the model is too simple, it is so in a relatively harmless way, since the model itself ***tells*** us that it is not very accurate: its *average* performance over future data will be reasonably close to its *expected* performance over future data. In other words, when using the overly simple Bernoulli model for the data generated by the Markov chain, *you do not know very much about the data generating process, but you* know *that you do not know very much.*

It turns out (Chapter 5) that a similar statement holds in general: the optimal model in $\mathcal{M}$ for present data will give a correct impression of the error you will make if you use it to predict future data – provided present and future data are generated by the same probability distribution. However, this distribution need not be identical or even 'close' to any of the models in $\mathcal{M}$.

Let us give another example, involving once again the class of polynomials. We assume throughout this example that data are actually sampled independently from the third-degree polynomial depicted in Figure 1 with some noise (not necessarily Gaussian), and that we model the data using only the class of first degree polynomials. In that case, the optimal two-part code MDL model as well as the model maximizing the fit to the data will, with probability 1, converge to the same first-degree polynomial $\widetilde{H}$ (this will be shown in Chapter 5, Lemma 5.14). This polynomial will look like the one depicted in Figure 1.1. It is clearly too simple; if we look at the average error per data item (x_i, y_i), we see that it is quite high; the third degree polynomial would be a better model for the data at hand. Nevertheless, as we will show in Chapter 5 (Theorem 5.19), if we use $\widetilde{H}$ as a basis of predicting future data, then, with high probability, we will make approximately the same error on average as we have made on the data D we have seen so far: the error we will make in predicting future data is close to the error we would *expect* to make *if our model $\widetilde{H}$ were actually true*. Hence, also in this case, using the overly simple model is 'safe' in that the model does not promise you more than you will get.

Modeling versus Generating Distributions Model classes that do not contain the 'true' probabilistic model are usually called 'misspecified'. We show in Chapter 5 that under a wide variety of probabilistic underlying laws, basing estimates on a 'misspecified' model class will be 'safe' in the sense above. Also, it turns out (Chapter 5, Theorem 5.19) that if we use the squared error function and hence model the data *as if* it were subject to Gaussian noise, we will still obtain good, 'safe' (in the sense above) results even if the data *generating* distribution is not Gaussian at all!

We obtain this result (which in itself is not new) as a special case. It partially explains why modeling errors using a Gaussian distribution generally leads to good results. People have often wondered how this is possible, since when we inspect observational data, we find more often than not that the empirical distribution of the errors is not really Gaussian. We quote from Jaynes' forthcoming book ([77], Chapter 7):

> "In the middle 1950's the writer heard an after-dinner speech by Professor Willy Feller (writer of the classic [47], PG) , in which he roundly denounced the practice of using Gaussian *probability* distributions for errors, on the grounds that the *frequency* distributions of real errors are almost never Gaussian. Yet in spite of Feller's disapproval, we continued to use them, and their ubiquitous success in parameter estimation continued."

By always keeping in mind that our *modeling* probabilities are monotone transforms of code lengths and not frequencies, we see that Feller's reasoning is not necessarily correct.

What does a model say about the world? Once we acknowledge that our models always have a chance of being partially wrong, a new difficulty arises, for it follows that *we should not just act as if our model were true in all respects.* Let us return to the example where we assume data to be generated by the Markov chain (1.20) and we inferred the Bernoulli model $\theta_{\text{mdl}}(x^n) = 1/3$. According to our Bernoulli model, which models the data as being independently distributed, the frequency of '11' over future data will be $1/3 \cdot 1/3 = 1/9$. But this will not be the case (neither in the given sample nor in future data): an easy calculation shows that it tends to $2/9$ instead. So the conclusion that the frequency of 1's in future data will be approximately $1/3$ seems more 'safe' than the conclusion that the frequency of '11' will be approximately $1/9$: it is less sensitive to whether or not our model is really 'true' in any sense. In general, it may be unwise to draw just *every* conclusion which would follow if an inferred model for data really were 'true' and *would* generate the data. You could, of course, for each new conclusion you would like to draw from your model inspect the given data D on which your model is based to see whether it really holds for D, but then you would not be able to use your model as a 'stand-alone' which is what you usually want. Apparently, some conclusions about the model are much more sensitive to the model 'really being true' than others. This raises the important question of what can and what cannot be reliably inferred from a statistical model for the data: what exactly does a model say about the modeled situation? We deal with this question at length in Chapters 4 and 5; but in order to do so, we need to introduce the fundamental concept of MDL, the *stochastic complexity.* This will be the subject of the next chapter.

Chapter 2

Stochastic Complexity

In this chapter we introduce the fundamental concept of MDL: the *stochastic complexity*. We start by motivating the need for the concept. In Section 2.2 we prepare the formal definition of stochastic complexity by arguing that *every* model together with every reasonable error function can be re-interpreted as a probabilistic model or, equivalently, a description method.

In Section 2.3 the stochastic complexity is introduced formally. This is followed by discussions of various interpretations and applications of stochastic complexity. Section 2.7 provides an alternative derivation of stochastic complexity by reinterpreting the whole theory in terms of proportional betting. Stochastic complexity is closely related to the 'marginal distribution' of Bayesian statistics. The relation is discussed in some detail in Section 2.8. The chapter ends with a short summary.

2.1 Motivation

We can distinguish between the general MDL Principle and its instantiations. The general MDL Principle tells us the following: the more we can compress our data, the better we have captured the regular features of the data and hence the more we have learned about it. In Chapter 1 we instantiated this to the two-part code MDL Principle given on page 11, where a particular code $C_{2\text{-}p}$ was defined relative to a model class $\mathcal{M}$ so as to allow compression of those data sequences that are well-described by one of the models in $\mathcal{M}$.

Let $\mathbf{E}$ be a sample space and let C and C' be codes over $\mathbf{E}$. We call C' *more efficient* than C if for all $D \in \mathbf{E}$, $L_{C'}(D) \leq L_C(D)$ while for at least one $D \in \mathbf{E}$, $L_{C'}(D) < L_C(D)$. If, for a code C, there exists a more efficient code C', we call C *redundant*.

According to the general MDL Principle, a code C' that is more efficient than the two-part code $C_{2\text{-}p}$ is preferable over $C_{2\text{-}p}$: it will compress some data sequences more, and no data sequences less. We now show that $C_{2\text{-}p}$ is indeed redundant, so such a preferable C' must exist.

Note first that the two-part code instantiation of MDL is based on a description method that is actually *not* a code: it is based on a one-many relation and not on a one-to-one mapping (see Chapter 1, Definition 1.2). For example, using the model class of polynomials we can describe each set of points $D = ((x_1, y_1), \ldots, (x_n, y_n))$ in an infinite number of ways: for each k, we can describe D by first describing a polynomial H_k of degree k and then describing the data with the help of that polynomial. By picking always the k for which the overall description length is minimized, we turned our description method into a code, but the fact remains that the coding scheme as a whole allows each data set to be encoded in an infinite number of ways. This means that an infinite number of codewords are reserved for each data sequence where only one is needed; it follows that there must exist some code C' which is more efficient (in the sense defined above) than $C_{2\text{-}p}$.

This leads us to search for an alternative code which does give the *shortest possible code lengths* to those data sequences for which there exists a good model in $\mathcal{M}$. According to MDL, this would be the truly optimal code to use in inductive inference; the two-part code would serve only as approximation. To find this code, we have to make mathematically precise the notion of 'the shortest possible code lengths for all data sequences for which a good model in $\mathcal{M}$ exists'. This is tricky but possible; we will do it in Section 2.3. Once this has been done, the optimally compressing code can be defined explicitly. *The code length of D when encoded using this MDL-optimal code is called the* **stochastic complexity** *of D with respect to the model class $\mathcal{M}$.*

This gives only a first idea; once we will have formally established the concept, we will see that it can be interpreted in several different ways. This is done in Section 2.4. We will then also see how stochastic complexity can be applied to two very important problems that arise in statistical practice: *model class selection* and *prediction*.

2.2 The Concept of 'Model'

In this section we will argue that all models (hypotheses) can be interpreted as codes and therefore also as probability distributions. We start with a discussion on the notion of 'model'.

2.2.1 Models are Fit-measurers

In Chapter 1 we considered probabilistic and non-probabilistic models. The latter were implicitly taken to be functions over the data (in our example we considered the class of polynomials). Now a function H is really an incomplete specification of a model, since, given data D, H in itself does not tell us how well it fits D. It is for this reason that a function becomes a model only if it is accompanied by an *error function* which specifies for each possible realization of the data how well H fits that data, or, equivalently, how well that data is *explained* by H. In the polynomial example, this was achieved by associating with each polynomial the squared error function. Henceforth we shall call a model that gives a 'fit' to *all* possible realizations of the data *complete*:

Definition 2.1 *A* complete model *M for sample space* $\mathbf{E}$ *is a function* $M : \mathbf{E}^* \to \mathbf{R}$. *For each realization of the data D, $M(D)$ measures how well M fits D.*

We use the convention that the higher $M(D)$, *the higher the error* M *makes on* D *and hence the* worse M *fits* D.

We note that a probabilistic model is a special case of a complete model. Specifically, for a given probabilistic model P we can define $M_P(D) = g(P(D))$ where g is some monotone decreasing function. In this way, the higher the probability, the better M_P fits the data.

Definition 2.2 *Let $\mathcal{M}$ be a set and $\mathbf{E}$ be a sample space. An* error function *for $\mathcal{M}$ is a total function $\text{ER} : \mathbf{E} \times \mathcal{M} \rightarrow \mathbf{U}$ where $\mathbf{U} \subseteq \mathbf{R}$. For $x \in \mathbf{E}$ and $H \in \mathcal{M}$, we write $\text{ER}(x|H)$ rather than $\text{ER}(x, H)$. We restrict ourselves to* additive *error functions*[1]*: ER is extended to outcomes $x^n \in \mathbf{E}^n$ by $\text{ER}(x^n|H) = \sum_{i=1}^{n} \text{ER}(x_i|H)$.*

Obviously, we can define a mapping such that each $H \in \mathcal{M}$ is mapped to a complete model M_H with for all $D \in \mathbf{E}^*$:

$$M_H(D) = \text{ER}(D|H) \tag{2.1}$$

The set containing M_H for all $H \in \mathcal{M}$ forms a class of complete models. In the supervised setting (Chapter 1, Section 1.4), $\mathbf{E} = \mathbf{E}_x \times \mathbf{E}_y$ and the data D can be written as $D = (x^n, y^n)$ where the x^n may be regarded as given. The models $\mathcal{M}$ will then be functions from $\mathbf{E}_x$ to $\mathbf{E}_y$. In this case, we should consider *conditional* complete models $M_H(y^n|x^n)$ where, for each (x^n, y^n), $M_H(y^n|x^n)$ measures the error that H makes on y^n given x^n. Here is an example:

Example 2.3 Let $\mathbf{E}_y = \mathbf{R}$. Let $\mathcal{M}$ be a class of functions from $\mathbf{E}_x$ to $\mathbf{E}_y$ (for example, the polynomials). The typical error function to use in this case is the squared error with $\text{ER}_{sq}(x^n, y^n|H) = \sum_{i=1}^{n}(y_i - H(x_i))^2$. We usually write $\text{ER}_{sq}(y|H, x)$ rather than $\text{ER}_{sq}(x, y|H)$ in order to express the fact that x may be regarded as given. For every H, we can arrive at a complete model M_H by setting $M_H(y^n|x^n) := \text{ER}_{sq}(y^n|H, x^n)$.

Here is another example:

Example 2.4 **[concept learning]** An important form of 'supervised learning' is *concept learning*. Here $\mathbf{E} = \mathbf{E}_x \times \mathbf{E}_y$, $\mathbf{E}_y = \{0, 1\}$ and the model class $\mathcal{M}$ consists of functions $H : \mathbf{E}_x \rightarrow \mathbf{E}_y$ called 'concepts'. If $H(x) = 1$, then x 'belongs to the concept'; otherwise x falls outside the concept. Most of the model classes usually considered in *Computational Learning Theory* [2] are classes of concepts. A natural way to measure how well a concept H fits given data $D = (x^n, y^n)$ is simply to count the number of mistakes H makes on D. For this, we define the *0/1-error function* ER_{01} (using the same notational convention as for ER_{sq}) as

$$\text{ER}_{01}(y|H, x) = \begin{cases} 0 & \text{if } H(x) = y \\ 1 & \text{otherwise.} \end{cases}$$

[1]We can, of course, define non-independent error functions such that the error H makes on x_i depends on one or more of the outcomes $x_{i-1}, x_{i-2}, \ldots$ but we will not consider such cases in this thesis.

This is extended to sequences of outcomes by

$$\text{ER}_{01}(y^n|H,x^n) := \sum_{i=1}^{n} \text{ER}_{01}(y_i|H,x_i).$$

Using Equation 2.1 we can turn each concept H into an equivalent complete model M_H.

Henceforth, we assume each model class to consist either of complete models or to be associated with an error function ER so that it can be turned into an equivalent class of complete models in the way indicated above.

Discussion Is this reasonable? One may argue that, in realistic situations, we want to learn a model from the data and we then want to use this model to give good predictions under all kinds of loss functions. For example, we may want to find a polynomial that allows for good prediction of future data not only in the squared error sense, but also when other loss functions are used. Nevertheless, as long as we are in the learning stage, we must have *some* means to determine how well a model fits the given data, otherwise there may appear data D such that, for two different models H_1 and H_2, we have no way of deciding which of the two better fits the data. Therefore we must use some error function when *learning* from a sample D - irrespective of whether or not that same function will be used later to measure prediction loss over future data. Indeed, in Chapter 4 we will distinguish between 'error functions', used in the learning phase, and 'loss functions', used for prediction of future data. In the Epilogue to Part I of this thesis (page 269), we will see that under some circumstances, learning using error function ER will give good predictions under some loss functions LOSS $\neq$ ER. However, there are also circumstances under which using a loss function different from the error function may lead to bad results.

2.2.2 Complete Models correspond to Probability Distributions

et $\mathcal{M}$ be a class of complete models. It so happens that, under quite general conditions on $\mathcal{M}$ (we will identify these conditions in Chapter 5, Section 5.2), there exists a corresponding class of probabilistic models $\mathcal{M}_{pr}$ that is 'equivalent' to $\mathcal{M}$ in the following sense: there exists a bijection g that maps each $M \in \mathcal{M}$ to a $P(\cdot|M) \in \mathcal{M}_{pr}$ such that, for all n, all $x^n \in \mathbf{E}^n$,

$$-\log P(x^n|M) = M(x^n) + K_n \tag{2.2}$$

where K_n depends only on n, and not on either M or x^n (see [127, 128]).

In the case that $\mathcal{M}$ itself is the complete model class corresponding to a class of functions $\mathcal{M}_{\text{H}}$ and an error function ER, this gives by (2.1) that for all $x^n \in \mathbf{E}^n$:

$$-\log P(x^n|M_H) = M_H(x^n) + K_n = \text{ER}(x^n|H) + K_n \tag{2.3}$$

Hence $-\log P(x^n|M_H)$ is equal to the error H makes on x^n up to an additive constant. In the case of supervised learning, where $D = (x^n, y^n)$ and $\mathcal{M}_H$ consists of functions from $\mathbf{E}_x$ to $\mathbf{E}_y$, we should regard the x^n as given. Therefore, we should map M_H to a *conditional* distribution yielding:

$$-\log P(y^n|M_H, x^n) = M_H(y^n|x^n) + K_n = \text{ER}(y^n|H, x^n) + K_n \tag{2.4}$$

Example 2.5 In the case of the squared error, the probability distribution $P(y^n|M_H, x^n)$ in (2.4) is the conditional normal distribution that was defined in Chapter 1, Equation 1.15 on page 22.

Example 2.6 For the 0/1-error, let us define

$$P(y^n|H, x^n) = \prod_{i=1}^{n} P(y_i|H, x_i) \text{ where}$$
$$P(y|H, x) = \frac{1}{Z} 2^{-\text{ER}_{01}(y|H,x)} \text{ with } Z = \textstyle\sum_{y \in \{0,1\}} 2^{-\text{ER}_{01}(y|H,x)} = 3/2.$$

which is immediately seen to satisfy (2.4).

2.2.3 Probability Distributions are Codes are Models

Rissanen [127, 128] claims that *every* reasonable model class can be transformed into an associated class of probabilistic models in the manner indicated above. This requires that, for every model class $\mathcal{M}$ and every n, a suitable constant K_n exists such that (2.3) holds. Since the constant K_n plays no role in the inferences made[2], this leads directly to the further claim that, in developing a theory of inductive inference, we need consider only probabilistic models[3]. Bayesian statisticians [17, 38] arrive at that same conclusion on quite different grounds. For the time being, we will assume that Rissanen and the Bayesians are right, since this greatly facilitates the treatment of stochastic complexity. However, we will have much more to say on this issue in the Epilogue to Part I of this thesis, page 119.

Let us assume then that every reasonable model class has a corresponding class of probabilistic models. By the arguments of Chapter 1, Section 1.2.3, for fixed sample size n such a class can be seen as a class of probability distributions P over $\mathbf{E}^n$ all of which have a corresponding code C_P such that $L_{C_P}(x^n) = -\log P(x^n)$ for all $x^n \in \mathbf{E}^n$. Putting everything together, we can interpret all models, probabilistic or not, as *codes*. The obvious advantage of doing this is that it allows for a unified treatment of probabilistic and non-probabilistic model classes.

2.2.4 All models are 'probabilistic'

From now on, following Rissanen [128], we *identify* models, codes and probability distributions. Therefore, all model classes (unless explicitly stated otherwise) will be assumed to consist of *probabilistic models*. For example, we will regard the class

[2]This is actually true only if the constant has the same value among all the models being compared. Let us assume for the moment that it is true (it is immediately seen to be true for the squared error and the 0/1-error). We discuss this further in Chapter 5, Section 5.2.

[3]See [128], page 18/19.

of polynomials as being probabilistic too, identifying each polynomial H with the associated $P(\cdot|H)$ as given by Equation 1.15 on page 22.

In Chapter 1, pages 11 and 16, we defined the two-part code length $L_{2\text{-}p}(D|\mathcal{M})$ as the sum of the code length of a hypothesis θ and the code length of the data D when encoded 'with the help of θ'. The latter term was denoted $L_{C_2}(D|\theta)$. From now on, we regard θ itself simply as a code and drop the subscript C_2. Coding the data 'with the help of θ' now becomes equivalent to coding the data 'using the code θ'.

We stress once more that it is a subtle issue whether one should associate non-probabilistic models with probabilistic ones on the basis of Equation 2.3 (rather than in some other way) and whether one should associate probabilistic models θ with codes with lengths $L(x^n|\theta) = -\log P(x^n|\theta)$ (rather than in some other way). In Chapter 5, sections 5.4 and the Epilogue to Part I of this thesis this is discussed further. For now, we will simply go along with it and assume that it is unproblematic.

Modeling vs. Generating Distributions It is very important to keep in mind that our probabilistic models are not necessarily related to the traditional notion of probability distributions 'according to which the data are drawn'. Indeed, as we show in Chapter 4, only in some special cases will probabilities have anything to do with frequencies. From this it follows that we may not just take expectations over probabilistic models the way we are used to. What we can and what we cannot do with our probabilistic models will be discussed in Chapter 4.

Sometimes we do want to speak of a 'classical' probability distribution. Whenever we do that, we call it a (data) *generating* distribution; we denote such a distribution by P^* and a parameter vector that indexes it by θ^*. In contexts where there could arise confusion, we call probability distributions that are used as hypotheses '*modeling* probability distributions'.

Model achieving Least Error for given data D =
Probability Distribution achieving Maximum Likelihood of D =
Code achieving Minimum Codelength for D

If the data D is then such that the maximum likelihood estimator $\hat{\theta}(D) \in \mathcal{M}$ exists, we see by Equation 1.8 (Chapter 1) that $\hat{\theta}(D) \in \mathcal{M}$ coincides with the code in $\mathcal{M}$ which yields the shortest code length of D. In case that $\mathcal{M}$ is a probabilistic version of a non-probabilistic model class $\mathcal{M}'$, we see by Equation 2.3 that $\hat{\theta}(D)$ corresponds to the model in $\mathcal{M}'$ under which D has the least error. Hence we can identify the best-fitting model with the maximum likelihood model and the minimum-code length code.

2.3 Stochastic Complexity

From the previous section we know that, without any loss of generality, we may confine ourselves to probabilistic model classes. We also know that, for given data D, the model H in model class $\mathcal{M}$ that maximizes the likelihood of D is also the model that, interpreted as a code, assigns the shortest description length to D. These are the two necessary ingredients for a formal treatment of stochastic complexity, which now follows.

Recall that at the beginning of this chapter we defined the stochastic complexity of data D relative to model class $\mathcal{M}$ as the description length of D obtained when it is encoded using some special code C_{sc}. We now derive a mathematical expression for C_{sc}. For simplicity, we only consider the case where $\mathcal{M}$ can be parameterized by a fixed number of parameters k. This is formalized in the following definition:

Definition 2.7 *Let $\mathcal{M}$ be a class of probabilistic models over sample space* $\mathbf{E}$ *and let* $\Gamma \subset \mathbf{R}^k$. *We say that $\mathcal{M}$ is finitely parameterized by Γ if*

1. *There exists a bijection $g : \Gamma \rightarrow \mathcal{M}$.*
2. *Let $D \in \mathbf{E}^*$ be arbitrary but fixed. Then $P(D|\theta)$ as a function of θ is the restriction to domain Γ of a continuous function $\mathbf{R}^k \rightarrow \mathbf{R}$.*
3. *If $\mathbf{E}$ is continuous, then for all n, the density function $f(x^n|\theta)$ as a function of x^n is continuous at each x^n in the interior of $\mathbf{E}^n$.*

Throughout this section we will assume a setting with a sample space $\mathbf{E}$ and a model class $\mathcal{M}$ of models over $\mathbf{E}$ such that

C1 All outcomes fall within $\mathbf{E}_* \subseteq \mathbf{E}$ where, if $\mathbf{E}$ is discrete, $\mathbf{E}_*$ contains a finite number of elements and if $\mathbf{E}$ is continuous, $\mathbf{E}_*$ is compact[4] (closed and bounded).

C2 For all $x^n \in \mathbf{E}^n$, the ML estimator $\hat{\theta}(x^n)$ exists.

C3 $\mathcal{M}$ is finitely parameterized by some $\Gamma \subset \mathbf{R}^k$. Hence $\mathcal{M}$ can be written as $\mathcal{M} = \{P(\cdot|\theta) \mid \theta \in \Gamma\}$.

Let $D \in \mathbf{E}^n$ be our observational data. The MDL Principle tells us to look for an encoding of D that is as short as possible. As noted above, the model in $\mathcal{M}$ that permits the shortest encoding is the maximum likelihood model $\hat{\theta}(D)$. It *seems* we should code our data D using the ML model, in which case the MDL Principle would reduce to the maximum likelihood method of classical statistics!

However - and this is the crucial observation which makes MDL very different from ML - we are looking for a single, *fixed* optimal code C_{sc}, which compresses *all* data samples that are well modeled by $\mathcal{M}$. But the code corresponding to $\hat{\theta}(D)$, i.e. the code that encodes any D' using $L(D'|\hat{\theta}(D)) = -\log P(D'|\hat{\theta}(D))$ bits, only gives optimal compression for *some* data sequences (among which D). For most other data sequences $D' \neq D$, $\hat{\theta}(D)$ will definitely not be optimal: if we had been given such a different data sequence D' (also of length n) instead of D, then the code corresponding to $\hat{\theta}(D')$ rather than $\hat{\theta}(D)$ would give us the optimal compression. In general, coding D' using $\hat{\theta}(D)$ (i.e. using $L(D'|\hat{\theta}(D))$ bits) may be very inefficient.

Ideally, we would like to have a single code C_1 such that $L_{C_1}(D) = L(D|\hat{\theta}(D))$ for all possible D. However, such a code does not exist as soon as our model class contains more than one element – the reason being the familiar fact that, whatever code we

[4] Note that if $\mathbf{E}$ itself is either finite or compact, we may choose $\mathbf{E}_* = \mathbf{E}$.

use, only very few data sequences can receive short code lengths (Chapter 1, Section 1.1.1). It therefore makes sense to define the *regret* $\mathcal{R}_C(\cdot|\mathcal{M})$ of a code C relative to a class $\mathcal{M}$ as the function that, for each D of length n, gives the excess code length relative to the code length based on the ML estimator:

$$\mathcal{R}_C(D|\mathcal{M}) = L_C(D) - L(D|\hat{\theta}(D)) \quad (\text{where } \hat{\theta}(D) \in \Gamma) \tag{2.5}$$

We define the *stochastic complexity code* $C_{sc}(\cdot|\mathcal{M})$ as the code that minimizes the worst-case[5] regret. The worst-case regret is

$$\max_{D \in \mathbf{E}^n} \{\mathcal{R}_C(D|\mathcal{M})\} \tag{2.6}$$

The conditions on $\mathbf{E}$ ensure that this maximum exists. In general, there may exist several codes achieving the worst-case regret but they will share the same code lengths $L_{sc}(\cdot|\mathcal{M})$. The code length $L_{sc}(D|\mathcal{M})$ is called the *stochastic complexity* of data D with respect to model class $\mathcal{M}$:

Definition 2.8 [**stochastic complexity**] *Let $\mathcal{M} = \{P(\cdot|\theta) \mid \theta \in \Gamma\}$ be a probabilistic model class over $\mathbf{E}$ such that conditions* **C1**-**C3** *hold. Let $n > 0$ and let $\mathcal{L}$ be the set of functions $L : \mathbf{E}_*^n \to \mathbf{R}$ which satisfy the Kraft Inequality $\sum_{x^n \in \mathbf{E}_*^n} 2^{-L(x^n)} \leq 1$. Let $D \in \mathbf{E}^n$. The* stochastic complexity of data D relative to model class $\mathcal{M}$ *is the code length $L_{sc}(D|\mathcal{M})$ defined by*

$$L_{sc}(\cdot|\mathcal{M}) := \arg\min_{L' \in \mathcal{L}} \max_{\substack{x^n \in \mathbf{E}_*^n \\ \hat{\theta}(x^n) \in \Gamma}} \{L'(x^n) - L(x^n|\hat{\theta}(x^n))\} \tag{2.7}$$

In the definition above, the set $\mathcal{L}$ is the set of all functions L which can be interpreted as (idealized) code lengths.

The regret $\mathcal{R}_{sc}(D|\mathcal{M}) = L_{sc}(D|\mathcal{M}) - L(D|\hat{\theta}(D))$ must be equal for all D of length n (otherwise it would be possible to construct a code C' with a smaller worst-case regret). We can therefore define $K_{sc}(n|\mathcal{M}) = \mathcal{R}_{sc}(D|\mathcal{M})$ and write

$$L_{sc}(D|\mathcal{M}) = L(D|\hat{\theta}(D)) + K_{sc}(n|\mathcal{M}) \tag{2.8}$$

Whenever the model class $\mathcal{M}$ is clear from the context, we write $L_{sc}(D)$ and $K_{sc}(n)$ rather than $L_{sc}(D|\mathcal{M})$ and $K_{sc}(n|\mathcal{M})$.

The Trade-off The term $L(D|\hat{\theta}(D))$ in (2.8) is called the *goodness-of-fit* term. It reflects for each D how well D is fitted by the model in the class that fits D best. As a function of n, $K_{sc}(\cdot|\mathcal{M})$ is called the *model cost* of $\mathcal{M}$ or equivalently the *complexity* term. It measures the extra code length needed to encode the data due to the richness of $\mathcal{M}$. The stochastic complexity thus embodies a trade-off between fit and complexity. To see this for the polynomial example, let us compare the polynomials in Figure 1 on page x of the introduction: let $\mathcal{M}_i$ be the class of all polynomials of the i-th degree.

[5]The reasons why we are interested in the worst-case over all D will be discussed in Section 2.4.2, page 39.

Let $\hat{\theta}_i(D)$ be the ML model for data D within class $\mathcal{M}_i$. By equations 1.12 and 1.13 (page 21), the difference in the first terms of $L_{sc}(D|\mathcal{M}_1)$ and $L_{sc}(D|\mathcal{M}_{12})$ is *equal* to the difference in the squared errors on D of the best-fitting models in these classes. We have

$$\begin{gathered} L_{sc}(D|\mathcal{M}_1) - L_{sc}(D|\mathcal{M}_{12}) = \\ L(D|\hat{\theta}_1(D)) - L(D|\hat{\theta}_{12}(D)) + K_{sc}(n|\mathcal{M}_1) - K_{sc}(n|\mathcal{M}_{12}) = \\ \Delta\mathrm{ER}_{sq} + \Delta\text{complexity} \end{gathered}$$

$K_{sc}(n|\mathcal{M}_1)$ will be much *smaller* than $K_{sc}(n|\mathcal{M}_{12})$: $\mathcal{M}_{12}$ contains well-fitting models for *many many more* data sequences than $\mathcal{M}_1$. Since every code can only give a short code length to very few data sets, $K_{sc}(n|\mathcal{M}_{12})$ must be much larger than $K_{sc}(n|\mathcal{M}_1)$. We give an explicit value for $K_{sc}(n|\mathcal{M}_{12})$ in Section 2.6.

Formula (2.8) is reminiscent of the two-part code (1.6) which also contains an error term and a complexity term. The difference lies in the description methods used; while the two terms in (1.6) were based on an encoding of the data in which some hypothesis θ was encoded *explicitly*, the two terms in (2.8) are arrived at by looking directly for the code lengths that are, in a certain sense, as small as possible. The result is that the total length as given by (2.8) will in general be shorter than that given by the two-part code. However, as we discuss in Section 2.6, asymptotically the difference between the two code lengths will be very small.

The Stochastic Complexity Distribution Since $L_{sc}(\cdot|\mathcal{M})$ is a code length function, we can map it to a probability distribution $P_{sc}(\cdot|\mathcal{M})$ over $\mathbf{E}_*^n$ such that for all D of length n, $L_{sc}(D|\mathcal{M}) = -\log P_{sc}(D|\mathcal{M})$. We call $P_{sc}(\cdot|\mathcal{M})$ the *stochastic complexity distribution with respect to* $\mathcal{M}$. Just like $C_{sc}(\cdot|\mathcal{M})$ is the code that gives the shortest possible code length to those data sets for which there exists a well-fitting model in $\mathcal{M}$, $P_{sc}(\cdot|\mathcal{M})$ is the distribution that gives as much probability as possible to those data sets for which there exists a good-fitting model in $\mathcal{M}$. From (2.8) we have (dropping $\mathcal{M}$ from the conditionals whenever $\mathcal{M}$ is understood from the context):

$$L_{sc}(D) = -\log P_{sc}(D) = -\log P(D|\hat{\theta}(D)) + K_{sc}(n) \tag{2.9}$$

Hence we can write $P_{sc}(D) = P(D|\hat{\theta}(D))/F(n)$, where $K_{sc}(n) = \log F(n)$. If $\mathbf{E}_*$ is finite, then the sum $S(n) = \sum_{D \in \mathbf{E}_*^n} P(D|\hat{\theta}(D))$ is finite too. $F(n) \geq S(n)$, otherwise P_{sc} is not a probability distribution. Since, by definition, L_{sc} minimizes the worst-case regret (2.6), P_{sc} must maximize

$$\min_{x^n \in \mathbf{E}_*^n} \{P(x^n)/P(x^n|\hat{\theta}(x^n))\}.$$

From this it follows that $F(n) = S(n)$, and therefore we can write:

$$P_{sc}(D) = \frac{P(D|\hat{\theta}(D))}{\sum_{D \in \mathbf{E}^n} P(D|\hat{\theta}(D))} \tag{2.10}$$

If $\mathbf{E}$ is continuous, the normalizing sum gets replaced by the corresponding integral. Our regularity conditions **C1-C3** ensure that the integral is well-defined and finite.

2.4 Interpretation of the SC Distribution as a Single Hypothesis - Applications of SC

Let $P_{sc} = P_{sc}(\cdot|\mathcal{M})$. In light of the seemodel discussion in Section 2.2, we may regard P_{sc} as a *hypothesis*, i.e. a *single* model: just like any other hypothesis, P_{sc} defines for every possible realization of the data how well that data is explained by P_{sc} - so why not look at P_{sc} as if it where a hypothesis itself? It may then be regarded as the hypothesis that 'summarizes' all individual hypotheses in $\mathcal{M}$. Note however that only in special cases will P_{sc} itself be a member of $\mathcal{M}$.

In analogy to the case for single models (see page 34), we use the terminology 'the code length of D when D is encoded with the help of $\mathcal{M}$' to denote the code length of D when D is encoded using the code $C_{sc}(\cdot|\mathcal{M})$.

A Simpler Definition of Stochastic Complexity Once we have made the conceptual step of interpreting P_{sc} as defining a single hypothesis, we can rephrase the verbal definition of stochastic complexity in a much simpler way.

> The stochastic complexity of data D relative to model class $\mathcal{M}$ is the description length of D when D is encoded with the help of class $\mathcal{M}$.

2.4.1 Applications of SC

The stochastic complexity cannot directly be used to find the best *single* hypothesis $H \in \mathcal{M}$; encodings of data using the code $C_{sc}(\cdot|\mathcal{M})$ do not contain substrings that code for a particular $H \in \mathcal{M}$, so there is no way in which $L_{sc}(\cdot|\mathcal{M})$ tells one what the optimal single H is. If the goal is to identify such a single hypothesis, then the 2-part code of Chapter 1 remains the code of choice. But if we are interested solely in *prediction* or *model class selection*, then stochastic complexity can be fruitfully applied (see Yamanishi [167] for more applications). The case of prediction will be treated in detail in Chapter 6. Here we briefly sketch the case of model class selection.

MDL Principle for Model Class Selection Given data D and two model classes $\mathcal{M}_1$ and $\mathcal{M}_2$, we should prefer model class $\mathcal{M}_1$ if and only if the stochastic complexity of D with respect to $\mathcal{M}_1$ is smaller than the stochastic complexity of D with respect to $\mathcal{M}_2$, i.e. if

$$L_{sc}(D|\mathcal{M}_1) < L_{sc}(D|\mathcal{M}_2) \tag{2.11}$$

If $\mathcal{M}_1$ has a smaller stochastic complexity than $\mathcal{M}_2$, then apparently it captures more of the regularity in the data and as such should be preferred. The larger the difference, the higher the confidence that D is better modeled using $\mathcal{M}_1$ than using $\mathcal{M}_2$. We can use this definition to find out whether, for example, the given data is better modeled by the class of second degree polynomials or by the class of third degree polynomials. More interestingly, we can also compare *completely different model classes* to each other. For example, we can decide whether the class of 3rd-degree polynomials or the class of backpropagation neural networks [72] with 5 hidden units is better for the data at hand.

One can extend the definition of stochastic complexity to model classes that do not have a maximum number of parameters [128, 129]. Having done this, one can even use stochastic complexity to decide whether the data is better modeled by the class of *all* polynomials or by the class of *all* backpropagation neural networks. We note that in traditional statistics, this kind of inductive inference problem can be dealt with only in *ad hoc* ways [128].

2.4.2 Discussion

The definition of stochastic complexity which we gave in the previous section was a very 'pessimistic' one: we minimized the worst-case regret $\mathcal{R}(D)$, where the worst-case was taken over *all* data of length n. One reason for taking the worst-case regret is that it leads to Equation 2.8 (page 36): it precisely reflects for all D how many bits are needed to encode D on the basis of the model in $\mathcal{M}$ that fit D best. Hence it captures the intuitive idea of 'coding with the help of $\mathcal{M}$'. One may nevertheless ask whether there is no other code C' which for some, very few, data sequences gives a very long code length while giving a shorter code length than C_{sc} to all others. If one takes a more traditional point of view and assumes that the data is actually *generated* by one of the distributions in the model class, then such a code C' may be preferable over C_{sc}. One may, in fact, define the code C' that gives the shortest worst-case *expected* regret, where the expectation is taken over a distribution $P^* \in \mathcal{M}$ that is assumed to generate the data and the worst-case is taken over all distributions in the class $\mathcal{M}$ (rather than all data D as in our definition). Rissanen [129], using results from Clarke and Barron [26, 27] , shows that asymptotically the code C' which achieves this minimum worst-case expected regret is (under some regularity conditions on $\mathcal{M}$) *identical* to C_{sc}. This provides an important additional justification of using the code C_{sc} as a basis of stochastic complexity.

2.5 Former Definition of Stochastic Complexity

It turns out that if we *average* over all models in a given model class, we obtain something that is very close to the stochastic complexity. Assume, just for the moment, that our class $\mathcal{M}$ has a finite number of elements, all of which are probabilistic models $P(\cdot|\theta)$. In this case we may define a new probability distribution $P_{av}(\cdot|\mathcal{M})$ as a *mixture* of all the models in the class:

$$P_{av}(D|\mathcal{M}) = \sum_{\theta \in \Gamma} P(D|\theta) \times W(\theta) \tag{2.12}$$

Here $W(\theta)$ is a probability distribution over the models in $\mathcal{M}$ that is introduced for normalization. If we let $W(\theta) > 0$ for all $\theta \in \Gamma$, then for large samples D, $P_{av}(D|\mathcal{M})$ will be approximately equal to $P_{sc}(D|\mathcal{M})$. More precisely, whereas $-\ln P_{av}(D|\mathcal{M})$ and $-\ln P_{sc}(D|\mathcal{M})$ both grow linearly in the size n of D, there difference is bounded by some constant independent of n. Intuitively, the reason for this is that the models in the class that give the highest probability to data D automatically contribute the most to the probability $P_{av}(D|\mathcal{M})$. If D is large, then the contribution of $W(\theta)$ to each term in the summation becomes negligible compared to the contribution of $P(D|\theta)$.

For model classes indexed by parameters ranging over a continuous domain the sum gets replaced by an integral and we obtain

$$P_{av}(D|\mathcal{M}) = \int_{\theta \in \Gamma} P(D|\theta) w(\theta) d\theta \tag{2.13}$$

Here $w(\theta)$ must be some *prior density* over Γ, usually just called a 'prior'. Distribution (2.13) plays a central role in Bayesian statistics [17] where it is called the *marginal distribution* or *evidence*. Rissanen gave the verbal definition of stochastic complexity in 1986 but did not connect it to formula (2.10) until 1996. Until that time, $-\log P_{av}(D|\mathcal{M})$ served as the working definition of stochastic complexity, since it was the closest thing to his verbal definition which he could come up with - just how close depends on the prior w as we will show in the next section. Formally, we define $L_{av,w}(D|\mathcal{M}) = -\log \int P(D|\theta) w(\theta) d\theta$ as the *mixture approximation (with prior w)* to stochastic complexity.

2.6 Asymptotic Expansion of SC

In this section we give an explicit formula for $L_{sc}(x^n|\mathcal{M})$, which will allow us to quantify much more precisely the exact trade-off involved in formula (2.8).

As proved by Rissanen [129], for model classes that satisfy certain regularity conditions, we get the asymptotic expansion of $L_{sc}(x^n|\mathcal{M})$ given in formula (2.14) below. The precise regularity conditions can be found in [129]. Essentially, they state that the CLT (Central Limit Theorem) should hold for the maximum likelihood estimators for all elements in the model classes. Generally speaking, this is the case for model classes which consist of i.i.d. probabilistic models but also, for example, for some non-i.i.d. classes like the class of Markov models.

Let $\mathcal{M}_k$ be a class of probabilistic models parameterized by a set $\Gamma \subset \mathbf{R}^k$. We have, for all $n > 0$ and all x^n with $\hat{\theta}(x^n)$ in the interior of Γ:

$$L_{sc}(x^n|\mathcal{M}) = -\log P(x^n|\hat{\theta}(x^n)) + \frac{k}{2} \log \frac{n}{2\pi} + \log \int \sqrt{|I(\theta)|} d\theta + o(1) \tag{2.14}$$

where the integral goes over all $\theta \in \mathcal{M}$ and $\lim_{n\to\infty} o(1) = 0$.

The first term measures the goodness-of-fit of the data using the best-fitting model in the class. From (2.8) we see immediately that the model cost $K_{sc}(n|\mathcal{M})$ is equal to the sum of the second, third and fourth term. The second term measures the part of this complexity term that is due to the number of parameters k; note that it grows linearly in k and logarithmically in n. The third term involves the determinant $|I(\theta)|$ of the *(expected) Fisher information matrix* $||I(\theta)||$; the definition of this matrix is given in Chapter 6. It measures the part of the complexity term that is due to local geometrical properties of the model space.

To get an idea of what this notion means, note that if we have a model class consisting of an enormous number of hypotheses, all of which, however, are the same, then the model class is really not that complex at all. For example, we may have a parameter set $\Gamma = \mathbf{R}^2$ but all parameters are names for the same probabilistic model P. Hence in reality, the model class contains only one model with many different names and it is not complex at all. The regularity conditions needed for (2.14) to hold rule out this situation, but they do not rule out the following possibility: there may be regions in a (continuous) parameter space containing only extremely *similar* models in the sense that $P(\cdot|\theta_1) \approx P(\cdot|\theta_2)$ for all θ_1 and θ_2 in the region. In other regions of the space, $P(\cdot|\theta_1)$ and $P(\cdot|\theta_2)$ may be much further apart even though the regions are equally large. Intuitively, such a region is more 'dense': it contains more distinguishable, 'truly different' models. The term $\log \int \sqrt{|I(\theta)|} d\theta$ corrects for this phenomenon.

While the first two terms in (2.14) grow with n the third term does not. Hence, for very large n, it can be safely neglected. The model class selection criterion which arises if we indeed neglect it has, somewhat confusingly, been called the *MDL Model Selection Criterion* [128]. We see that this criterion amounts to the following: For given data D and competing model classes $\mathcal{M}_1, \ldots, M_m$, we should pick the $\mathcal{M}_k$ which minimizes $-\log P(D|\hat{\theta}_k(D)) + \frac{k}{2} \log n$ where $\hat{\theta}_k(D)$ is the maximum likelihood estimator of D within class $\mathcal{M}_k$. This criterion can be used as a first approximation, but since the neglected term does depend on k and can vary quite a lot for different k, one really needs a lot of data to be able to safely neglect it.

In cases where the integral in this term diverges (for example, when the class of normal distributions is used as the model class), Rissanen [129] gives a correction to the formula (2.14).

How close do the approximations get? Both the two-part code and the code based on P_{av} are good approximations to the stochastic complexity. If the model class $\mathcal{M}_k$ satisfies suitable regularity conditions similar to those needed for (2.14) to hold, one can prove that, for all x^n with $\hat{\theta}(x^n)$ in the interior of the parameter space, the two-part code length $L_{\textit{2-p}}(x^n|\mathcal{M})$ approaches $L_{sc}(x^n|\mathcal{M})$ to within a constant:

$$L_{\textit{2-p}}(x^n|\mathcal{M}) = -\log P(x^n|\hat{\theta}(x^n)) + \frac{k}{2} \log n + O(1) \tag{2.15}$$

and hence becomes a better and better approximation as n increases. In Chapter 7 we describe a modification to two-part codes (introduced in [129]) which removes their inherent redundancy that was noted in Section 2.1. For this improved 2-part code, we get the same asymptotic expansion as (2.14). We call such a code 'asymptotically perfect', since, for all x^n, the difference in description length of data x^n between this code and the stochastic complexity code C_{sc} goes to 0 as n goes to infinity. The mixture $L_{av,w}(x^n|\mathcal{M})$ can - once more under similar regularity conditions - be rewritten exactly as (2.15). A special case is obtained if we take the prior w to be *Jeffreys' prior* [78, 17]. Jeffreys' prior is given[6] by $\pi(\theta) \propto \sqrt{|I(\theta)|}$ where $|I(\theta)|$ is the determinant of the Fisher information matrix that also appeared in Equation 2.14. Balasubramanian [9] showed

[6]The symbol $\propto$ denotes proportionality: $\pi(\theta) \propto \sqrt{|I(\theta)|}$ means that there exists a constant c such that for all $\theta \in \Gamma$: $\pi(\theta) = c\sqrt{|I(\theta)|}$.

that the asymptotic expansion of $L_{av,\pi}(x^n|\mathcal{M})$ is given by (2.14). Hence the evidence approximation of stochastic complexity using Jeffreys' prior is asymptotically perfect.

Using the code based on Jeffreys' prior has some advantages over the code $C_{sc}(\cdot|\mathcal{M})$, as we will discuss in Chapter 6. Therefore, it can be regarded as an alternative definition rather than approximation of stochastic complexity [129].

2.7 A Reinterpretation in Terms of Money

In this section, we show that the MDL Principle can be obtained in a rather different way: minimizing description length of a sequence of outcomes turns out to be equivalent to sequentially placing proportional bets on the outcomes in an optimal manner. This reinterpretation forms an additional justification of the identification of hypotheses with code lengths; it also allows us to view the stochastic complexity in a different way.

Proportional Betting Imagine you take part in a betting game. You are asked to place bets on possible outcomes. The set of possible outcomes is $\mathbf{E} = \{e_1, \dots, e_k\}$. The odds in the game are k-to-1 on all outcomes e_i. This means that you are offered a ticket for e_i at the price of 1 Euro. If you buy the ticket and the outcome of the game is indeed e_i, then you receive k Euroes. Upon any other outcome $e_j \neq e_i$, you receive nothing and your investment is lost. We consider an 'idealized' game in that you are allowed to buy any number f of tickets, where f may be every rational number. If you buy f tickets for outcome e_i, then we say that you 'place a bet of f on outcome e_i'.

You are allowed to bet on several different outcomes at the same time. This implies that the bets are 'fair' in the following sense: suppose your start out with a capital f. By equally dividing f over the possible outcomes (i.e. placing a bet of f/k on all e_i) your capital after the game will always be equal to your starting capital. Hence it is always possible to place a 'safe bet' so that even if you are completely ignorant about the process generating the outcomes, you are guaranteed not to lose anything.

Notice that this setup is very similar to the game that is actually offered to you at the horse races [30]; the only essential difference is that in the actual horse race, the odds will be different for different horses, set up in such a way that they allow the book maker to make money.

Now suppose you have a probability distribution P over the k possible outcomes. You want to take part in the game described above such that the amount you bet on outcome e_i is proportional to $P(e_i)$. You decide to invest f Euroes, so your bet on e_i is $fP(e_i)$. We refer to this way of dividing your investments as *proportional betting*. Now your remaining capital after playing the game is clearly $kfP(x)$ where $x \in \mathbf{E}$ is the outcome that actually took place.

Next suppose the game involves a sequence of outcomes $x^n = (x_1, \dots, x_n) \in \mathbf{E}^n$, and you have a probability distribution $P : \mathbf{E}^n \to [0,1]$. Playing the game as before, you now place bets on every different $x^n \in \mathbf{E}^n$ proportionally to $P(x^n)$. Since there are k^n possible outcomes, the odds are 1-to-k^n and your end capital will then be

$$f \cdot k^n \cdot P(x^n) \tag{2.16}$$

where x^n is the sequence that actually occurred. But you may also play another game consisting of n sub-games. In the first sub-game, you divide all your capital among bets on the first outcome, x_1. The amount you put on outcome $e \in \mathbf{E}$ is $f \cdot P\{X_1 = e\} = \sum_{x_2,\ldots,x_n} P(ex_2 \ldots x_n)$. After this first sub-game, your remaining capital is $f \cdot kP(x_1)$ where x_1 is the outcome that actually occurred. You now invest all remaining capital in bets on the second outcome x_2 using the *conditional* probability distribution $P(x_2|x_1) = P(x_1x_2)/P(x_1)$. Hence

$$P\{X_2 = e|X_1 = x_1\} = \frac{\sum_{x_3,\ldots,x_n \in \mathbf{E}^{n-2}} P(x_1 e x_3 \ldots x_n)}{\sum_{x_2,x_3,\ldots,x_n \in \mathbf{E}^{n-1}} P(x_1 x_2 x_3 \ldots x_n)}$$

After the second outcome, your remaining capital is $f \cdot k^2 P(x_1)P(x_2|x_1)$ where x_1 and x_2 are the two outcomes that actually occurred. Clearly, if you continue betting on x_i using the conditional distribution $P(x_i|x_1, \ldots x_{i-1})$ where $x_1, \ldots, x_{i-1}$ are the first $i-1$ outcomes that actually occurred, then after the n-th and last sub-game your capital is:

$$fk^n \Pi_{i=1}^n P(x_i|x^{i-1}) = fk^n \frac{P(x_1)}{1} \frac{P(x_1x_2)}{P(x_1)} \frac{P(x_1x_2x_3)}{P(x_1x_2)} \cdots \frac{P(x^n)}{P(x^{n-1})} = fk^n P(x^n) \tag{2.17}$$

This coincides with (2.16) and we see that both ways of playing the game are equivalent! As shown below, this allows us to interpret the MDL Principle in an alternative way. We can identify 'predicting that x will happen with probability p' with 'placing a bet on x proportional to p'.

Gambling Interpretation of Stochastic Complexity Let $\mathcal{M}$ be a class of probabilistic models over a discrete space $\mathbf{E}$. Consider the ML estimator $\hat{\theta}(x^n)$. Suppose we predict the x_i sequentially by placing a bet on x_1 proportional to $P(x_1|\hat{\theta}(x^n))$, a bet on x_2 proportional to $P(x_2|x_1, \hat{\theta}(x^n))$, etc. By Equation (2.17) our end capital is proportional to $P(x^n|\hat{\theta}(x^n))$. Had we played the same game using any other $\theta' \neq \hat{\theta}(x^n)$, our end capital would have been proportional to $P(x^n|\theta')$ which by definition is lower than $P(x^n|\hat{\theta}(x^n))$. Hence if we had predicted all the x_i on the basis of the ML estimator $\hat{\theta}(x^n)$, our end capital would have been maximized.

Unfortunately, we can only know the optimal predictor $\hat{\theta}(x^n)$ after the fact: we only know what *would have been* the optimal predictions after the data has been made available to us. From a 'minimax' viewpoint, the best way to predict is then to use a predictor P_{opt} which we can construct a priori (before seeing any data) and which is always as close as possible to the predictor that turns out to be optimal a posteriori. We proceed to show that P_{opt} is identical to the stochastic complexity distribution $P_{sc} = P_{sc}(\cdot|\mathcal{M})$ as defined on page 37. Using predictor (probability distribution) $\mathcal{P}$, the worst possible data that may appear for that predictor is the data which maximizes the proportion between the end capital arrived at using the a-posteriori optimal $P(\cdot|\hat{\theta}(x^n))$ and the end capital arrived at using $\mathcal{P}$, given by

$$\max_{x^n} \frac{P(x^n|\hat{\theta}(x^n))}{\mathcal{P}(x^n)} \tag{2.18}$$

P_{opt} is the distribution which minimizes this worst-case discrepancy. Comparing Equation (2.18) to Equation (2.7) and rewriting probabilities as code lengths, we see that P_{opt} and P_{sc} coincide.

Logarithmic Loss Let $\mathbf{E} = \{e_1, \ldots, e_k\}$ and let P be a probability distribution over $\mathbf{E}$. Suppose we predict that e_1 happens with probability $P(e_1)$, e_2 happens with probability $P(e_2)$ etc. Then some $x \in \mathbf{E}$ arrives and we are charged with a *loss* that measures the discrepancy between our predictions $P(e_1), P(e_2), \ldots$ and the actual outcome x. The *logarithmic loss* is defined as $\text{LOSS}_{lg}(x) = -\log P(x)$. Note that the higher the probability assigned to the actual outcome, the lower the logarithmic loss. In the statistical literature, this loss function is often used as a 'generic' loss function for probabilistic prediction [127]. By the arguments above, we see that we can regard the logarithmic loss both as the code length needed to encode x or as the logarithm of the factor by which our capital is multiplied if we bet proportionally on the e_i according to P.

Four Ways of Looking at MDL Summarizing, we see that we can alternatively interpret the goal of MDL as minimizing description length, maximizing probability, maximizing end capital in proportional betting, or, finally, minimizing logarithmic loss.

Predictive MDL We have seen that the stochastic complexity can be approximated by the two-part code length and by a 'mixture' approximation. At this point we should mention that there is a very important third way of approximating it which has its roots in the gambling interpretation. It is called 'predictive MDL' and was introduced independently with somewhat different motivations by Rissanen [127] and Dawid[7] [36, 37]. For details we refer to Rissanen [128].

2.8 MDL and Bayesian Statistics

Some authors use this approach so that they can use Bayesian methods in disguise without being ridiculed by their anti-Bayesian colleagues.
Wray Buntine on MDL

At this point we have discussed all main ingredients of MDL: the two-part codes, the stochastic complexity and the idea of viewing all model classes as 'probabilistic'. It is now instructive to compare MDL to Bayesian Statistics [17, 38, 18, 77]. Bayesian Statistics and MDL are closely related. Some people have even – wrongly – claimed that, philosophical differences notwithstanding, the two are really the same for all practical purposes. This claim probably stems from the fact that the mathematical formulas used in applications of MDL and Bayesian statistics coincide in two important cases:

[7] Strictly speaking, it can be seen as a special case of Dawid's 'prequential assessment' when used with the logarithmic loss.

Evidence and Stochastic Complexity In Bayesian statistics we always use a prior distribution w for all elements in the chosen model class $\mathcal{M}$. We can then simply calculate the *conditional* probability of the data D given model class $\mathcal{M}$ as

$$P(D|\mathcal{M}) = \int_{\theta \in \Gamma} P(D|\theta) w(\theta) d\theta$$

which coincides with our P_{av} as given by Equation (2.13). Bayesian model class selection [163] works as follows: if we are given some data D and we have to decide whether it is better explained by model class $\mathcal{M}_1$ or by model class $\mathcal{M}_2$, then we choose the class $\mathcal{M}_i$ for which the *evidence* $P(D|\mathcal{M}_i)$ is highest.

If we are willing to use P_{av} with prior w as the 'mixture approximation' to stochastic complexity (see Section 2.5) then, because of the monotonicity of '$-$' and log, this is equivalent to choosing the class with the lowest stochastic complexity.

MAP and 2-part MDL Bayesians often approximate the evidence $P(D|\mathcal{M})$ by $P(D|\breve{\theta})$, $\breve{\theta}$ being the *maximum a posteriori (MAP) model* for D. This is the model that has 'maximal probability in light of the data':

$$\breve{\theta} = \arg\max_{\theta \in \mathcal{M}} P(\theta|D) = \arg\max_{\theta \in \mathcal{M}} \frac{P(D|\theta)w(\theta)}{P(D)} = \arg\max_{\theta} P(D|\theta)w(\theta) \tag{2.19}$$

where the second equality follows from Bayes' rule [17] and the third follows from the fact that $P(D)$ does not change if we vary θ.

We know that for probabilistic model classes, the MDL estimator θ_{mdl} is given by minimizing $-\log P(D|\theta) + L_{C_1}(\theta)$. By the arguments of Section 1.2.3 in Chapter 1, there exists a probability distribution w such that for all D, $-\log w(D) = L_{C_1}(\theta)$. For this w we have:

$$\theta_{\text{mdl}} = \arg\min_{\theta \in \mathcal{M}} \{-\log P(D|\theta) + [-\log w(\theta)]\} \tag{2.20}$$

Comparing (2.19) and (2.20) we see that they are equivalent!

What, then, is the essential difference between the two approaches? Though it remains somewhat hidden in the two examples above, the aim of Bayesian statistics is usually to make inferences and decisions based on data D that *maximize expected utility.* Here the 'utility' of an inference or decision is some function depending on the problem domain. According to MDL, rather than maximizing expected utility we should base inferences on minimizing code length (how to deal with 'decisions' will be explained in Chapter 4). Summarizing:

> Roughly speaking, the aim of MDL is to compress data as much as possible; the aim of Bayesian statistics is to maximize expected utility.

MDL's focus on compression reveals itself in several ways. First, the definition of stochastic complexity $L_{sc}(\cdot|\mathcal{M})$ does not depend on any particular prior distribution. This is how it should be, since it is defined with respect to D and $\mathcal{M}$ and should thus depend only on D and $\mathcal{M}$ and nothing else. If we approximate $L_{sc}(\cdot|\mathcal{M})$ using $L_{av,w}(\cdot|\mathcal{M})$, we should pick the prior with which we obtain code lengths that are as short as possible in the worst-case. Under suitable conditions, this will be Jeffreys' prior as introduced in Section 2.6.

In the same vein, Rissanen [129] shows that the two-part code length given by (1.5) is suboptimal only: there exists a somewhat more efficient version of the two-part code length that *cannot* be reinterpreted in MAP terms (see Chapter 7). There are different sub-schools of Bayesian Statistics and the proponents of each propose different priors for different reasons [17]; usually however, the purpose of such priors does not seem to be to compress as much as possible.

Since the influence of the prior distribution is typically almost negligible except for very small sample sizes, MDL and Bayesian methods will often lead to similar results in practice. However, 'non-typical' cases where the prior has a large influence even for relatively large sample sizes do occur in practice [129]. In such cases MDL and Bayesian methods may lead to different results.

No Expectations Most Bayesians do not seem to hesitate to take *expectations* over priors (see Chapter 7, Section 7.2 for an example). According to MDL, a prior should be used only as a tool to arrive at short descriptions. Therefore it is *not* what we would call a 'generating' probability: the hypothesis θ can certainly not be viewed as 'being drawn according to prior w'. MDL holds the view that taking expectations over non-generating probabilities should be done with care and is meaningful only in certain special cases (see Chapter 4).

MML and Prequential Analysis indexMinimum Message Length Principle o complicate the situation further, there are two branches of Bayesian statistics (or at least, branches of statistics that have their roots in the Bayesian view), which are particularly closely related to MDL. These are (1) Dawid's 'Prequential Analysis' [36, 37] and (2) Wallace's Minimum *Message* Length (MML) Principle [161, 162]. The MML Principle combines Bayesian ideas with data compression; it is discussed at length in Chapter 7. Prequential analysis and MDL share the view that probability distributions are to be seen as *models* of data and not as something 'according to which data are actually distributed'. The relationship between Prequential analysis and MDL is investigated by Dawid in [37].

The differences between Bayesian methods and MDL become more pronounced in the idealized setting where we allow the coding to be done by a universal programming language. This situation has been analyzed in detail by Vitányi and Li [160].

2.9 Conclusion and Outlook

In chapters 1 and 2 we have provided a general overview of the MDL Principle. Summarizing, we can state that the following views are fundamental to MDL:

1. Inductive inference should be done on the basis of a code, or, more generally, a description method.

2. The shorter such a code length assigns to the given data D, the better we have captured the regularities in D.

3. Models should always be viewed as, or be associated with, codes.

4. Classes of models $\mathcal{M}$ can be 'summarized' by a single code: the *stochastic complexity code* based on $\mathcal{M}$.

We discussed several ways of approximating the stochastic complexity; we provided a reinterpretation of 'description methods' in terms of 'betting strategies' and we compared MDL to Bayesian statistics.

However, there are still two unresolved issues we mentioned in passing but did not go into further: in Chapter 1, we argued that overly simple models, or, more generally, models that allow for compression but are nevertheless not in any sense 'true', can nevertheless be useful in predicting or making decisions with respect to future data. This claim still has to be verified. The other issue concerns Section 2.2 in the present chapter, where we connected non-probabilistic model classes to probabilistic ones in a somewhat arbitrary manner (Equation 2.3, page 32). Indeed, as we will show in Chapter 5, Example 5.1, there is something inherently arbitrary about the procedure used in Section 2.2. In Chapter 5 we propose a new idea to deal with both issues raised here. But in order to do so, we need to introduce the concept of *Maximum Entropy* and relate it to MDL. This will be the subject of the next chapter.

Chapter 3

Introduction to Maximum Entropy

This chapter discusses the Maximum Entropy Principle and shows how it is related to MDL. In sections 3-3.6 we give a brief and basic introduction to maximum entropy. We start by informally introducing maximum entropy, giving several examples of its use and relevance. This is followed in sections 3.2-3.4 by formal definitions of entropy, relative entropy, maximum entropy distributions and classes of these distributions. Section 3.5 discusses the 'concentration phenomenon' which lies at the heart of the maximum entropy method. Ever since their introduction, the appropriateness of maximum entropy principles has been fiercely debated. Section 3.6 reviews some of the issues in this debate. In Section 3.7 we show that Maximum Entropy can be seen as a special case of the MDL Principle. The chapter ends with a brief conclusion. sectionMotivation It is a dream of Artificial Intelligence to build intelligent robots. These should be capable of making rational decisions on the basis of incomplete and partially unreliable information coming from their sensors and their database. If a robot's knowledge were represented by a probability distribution (where the probability of e indicates a robot's degree of belief in proposition e), then it could use *decision theory* [17] to arrive at rational decisions. This is only possible if the probability distribution is complete, assigning a probability to each conceivable event e. Where do we get these probabilities from? Some probabilities may be calculated from past experience; but for most of them there is not enough past experience to determine them with any degree of correspondence to the 'real' probabilities. The Maximum Entropy Principle provides a means to arrive at probabilities in the presence of partial knowledge: given certain constraints on a probability distribution, it constructs a *single* probability distribution over all possible outcomes. Once such a single distribution has been constructed, all the tools of decision theory and probability theory become available to make decisions and predictions.

Such a means of arriving at a single distribution from partial knowledge has applications in many other fields. Within the field of AI, we should mention *expert systems* [114] and automated natural language processing [16]. Some successful applications of Maximum Entropy outside of AI are in protein modeling [89], psychological modeling [113], stock market analysis [32] and analysis of genetic algorithms [140]; for a much longer list, see [144]. We will now first discuss maximum entropy in an informal manner.

3.1 Informal Introduction to Maximum Entropy

Let $\mathbf{E}$ be a sample space and let $\phi_1, \ldots, \phi_n$ be functions on $\mathbf{E}$ with ranges $\mathbf{U}_1, \ldots, \mathbf{U}_n$. For simplicity we assume that all $\mathbf{U}_i$'s are intervals in $\mathbf{R}^1$. We are given a finite (possibly empty) set $\mathcal{C}$ of constraints regarding the *expected values* of the functions ϕ_i:

$$\mathcal{C} = \{E[\phi_1(X)] = t_1, \ldots, E[\phi_m(X)] = t_m\} \tag{3.1}$$

where the t_i are values in the interior of $\mathbf{U}_i$.

We ask the following question: if the only knowledge we have about a probability distribution P are the constraints given by (3.1), what is then our best guess for P?

It is of course not well-defined what a 'best guess' really is. A possibility is to look for the P that allows for the best possible prediction of future data. Another option is to search for the P that is in some sense, 'the most likely', given the constraints. A third option is to assume that, at several stages, new information about P will be made available. The aim would then be to pick a P according to a scheme that converges as quickly as possible to the 'true' P according to which the data is actually drawn.

According to some authors, all these aims coincide, and are realized by adopting the *P that maximizes the entropy subject to the given constraints.* The entropy of a distribution P is the expectation of $-\log P(X)$. It is a measure of the amount of randomness or 'disorder' inherent in P. Since $\log P(x)$ determines the goodness-of-fit of data x under model P, the entropy of P may be interpreted as minus the expected goodness-of-fit under P. A distribution with high entropy has a low expected goodness-of-fit. Adopting the probability distribution with the maximum entropy is called the Maximum Entropy Principle. It has been introduced by Jaynes in 1957 [73] but has its roots in the work of Boltzmann and Gibbs on statistical mechanics [77]. Our first example shows that Maximum Entropy is a generalization of Laplace's Principle of Insufficient Reason [91] which we introduced on page xi.

Example 3.1 Let $\mathbf{E} = \{1, \ldots, k\}$. As will be verified shortly, the distribution over $\mathbf{E}$ without any constraints (except $\sum_{i \in \mathbf{E}} P(i) = 1$) with maximum entropy is given by $P(i) = 1/k$ for all $i \in \mathbf{E}$; the entropy of this distribution is

$$E[-\log P(X)] = \sum_{i=1}^{k} P(i)[-\log P(i)] = -k \cdot \frac{1}{k} \log \frac{1}{k} = \log k$$

The expected log-likelihood $E[\log P(X)]$ is $-\log k$. Intuitively, data drawn according to a more skewed distribution over $\mathbf{E}$ will, with high probability, be more regular than data drawn according to P. Indeed the entropy of P decreases (and the expected log-likelihood increases) as $P(i)$ drifts away from $1/k$, and in the limit for $P(i) \to 1, P(j) \to 0$ $(j \neq i)$ the entropy becomes 0.

Before moving on to formal definitions, we give a few more examples of 'MaxEnt', as it is usually abbreviated:

Example 3.2 Let $\mathbf{E} = \{a, b, c\}$. We are given the constraint $P(a) = 1/2$. To see that it is of the form demanded by Equation 3.1, note that $P(a) = p$ is equivalent to $E[\mathcal{I}(X = a)] = p$, where $\mathcal{I}$ is the *indicator function* defined as follows:

$$\mathcal{I}(x = e) = \begin{cases} 1 & \text{if } x = e \\ 0 & \text{if } x \neq e \end{cases} \tag{3.2}$$

For fixed e, $\mathcal{I}(x = e)$ is a function of x only. In such a case we write $\mathcal{I}_e(x)$ instead of $\mathcal{I}(x = e)$ whenever this does not give rise to any confusion.

In the case of constraint $P(a) = 1/2$, the maximum entropy distribution is given by $P(a) = 1/2, P(b) = P(c) = 1/4$.

Example 3.3 [**independence**] Let the bias of a coin be p and the bias of a second coin be q. Let 1 stand for 'heads' and 0 for 'tails'. The maximum entropy distribution over the joint space $\mathbf{E} = \{0, 1\}^2$ is given by $P(11) = pq, P(10) = p(1-q), P(01) = (1-p)q, P(00) = (1-p)(1-q)$. We see that this is the distribution corresponding to two independent coins.

Example 3.4 [**uniform and normal distribution**] Let $\mathbf{E} = [a, b]$. Otherwise no constraints are given. The maximum entropy distribution is the uniform distribution over $\mathbf{E}$. As another example, let $\mathbf{E} = (-\infty, \infty)$. Let the constraints $E[X] = \mu$ and $\mathrm{VAR}[X] = \sigma^2$ be given. The maximum entropy distribution under these constraints is the normal distribution with mean μ and variance σ^2.

Summarizing the previous examples, *complete ignorance* about the relative likelihood of k outcomes is translated into a *uniform* distribution on these outcomes. Complete ignorance about dependencies between random variables X and Y is translated into a distribution that renders X and Y independent.

3.2 Entropy and Relative Entropy

Preliminaries and Notation Let P be a probability distribution over a sample space $\mathbf{E}$. In the discrete case, the expectation $E_P[\phi(X)]$ of a function of the data ϕ is given by $\sum_{x \in \mathbf{E}} P(x)\phi(x)$. If $\mathbf{E}$ is continuous, then $E_P[\phi(X)] = \int_{x \in \mathbf{E}} f(x)\phi(x)dx$ where $f(x)$ is the density function of P. Whenever we use an indexed probability distribution like, say, P_i, we use $E_i[\phi(X)]$ as short for $E_{P_i}[\phi(X)]$. We define the *support* of a distribution P over sample space $\mathbf{E}$ as the set $\{x \in \mathbf{E} \mid P(x) > 0\}$. We say that P has *full* support if the support coincides with $\mathbf{E}$.

It will be convenient to change our unit of measurement and work in 'nats' in stead of 'bits' throughout the remainder of this chapter. Hence in this chapter, we use $L_P(x)$ as an abbreviation of $-\ln P(x)$, the code length (measured in nats) of x obtained by using the code based on P. In case $\mathbf{E}$ is continuous, we adopt the same conventions as in Chapter 1, Section 1.2. Specifically, $-\ln P(X)$ should then be read as $-\ln f(x)$ where f is the density function of P.

Whenever we write $P(x_1, \dots, x_n)$ for $x_i \in \mathbf{E}$, we are referring to the product distribution: $P(x_1, \dots, x_n) = \Pi_{i=1}^{n} P(x_i)$.

Definition 3.5 *The* entropy *of a distribution P is*

$$\mathcal{H}(P) = E_P[L_P(x)] \tag{3.3}$$

The relative entropy *or* Kullback-Leibler (KL) Divergence *between distributions P and Q is given by*

$$D(P||Q) = E_P[L_Q(x) - L_P(x)] = \sum_{x \in \mathbf{E}} P(x) \ln \frac{P(x)}{Q(x)}$$

The entropy has a direct interpretation as the expected number of nats needed to encode an outcome that is distributed according to P, using the code L_P based on P.

If we code the data using the code based on a distribution Q rather than on P, then the expected number of nats needed to encode an outcome drawn according to P will be $E_P[-\ln Q(x)]$. Hence the KL divergence has a direct interpretation as the expected difference in the number of nats needed to encode such an outcome using the code based on P versus the code based on Q. The following result is of fundamental importance (see [30] for a proof):

Theorem 3.6 (Information Inequality) *For any two probability distributions P and Q defined over the same space* $\mathbf{E}$,

$$D(P||Q) \geq 0 \tag{3.4}$$

with equality holding if and only if $P = Q$.

We see that if data are actually distributed according to P, then the optimal (in the sense of minimizing expected code length) code for the data is the code based on P: every other code will yield a larger expected code length. The difference in the expected number of nats needed can be interpreted as a kind of 'distance' between the distributions P and Q. Note however that in general, $D(P||Q) \neq D(Q||P)$ so the KL divergence is not really a distance in the usual mathematical sense of the word.

3.3 Maximum Entropy Distributions

Preliminaries We are given the sample space $\mathbf{E}$ and a set of constraints $\mathcal{C}$ pertaining to an otherwise unknown probability distribution P. We assume $\mathcal{C}$ to be of the form given by Equation 3.1 on page 50. We often abbreviate Equation 3.1 to

$$E[\phi(X)] = t$$

where $\phi(X)$ stands for the vector $(\phi_1(X), \dots, \phi_m(X))$ and t stands for the vector $(t_1, \dots, t_m)$. In this way, the set of constraints $\mathcal{C}$ can be equivalently seen as a single constraint on the vector-valued function ϕ. We write $\mathcal{C}(\phi, t)$ to denote the specific constraint $E[\phi(X)] = t$. We say a probability distribution *satisfies* constraint $\mathcal{C}(\phi, t)$

iff $E_P[\phi(X)] = t$. The set of probability distributions P over $\mathbf{E}$ satisfying $\mathcal{C}(\phi, t)$ is denoted by $\mathbf{M}(\phi, t)$: $\mathbf{M}(\phi, t) = \{P \mid E_P[\phi(X)] = t\}$.

In most practical applications, we are not really given expected values but rather measured *average* values of functions over large sets of data. Let $\overline{\phi(x)}^n$ be the average value of ϕ over n observations:

$$\overline{\phi(x)}^n = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)$$

For a given constraint $\mathcal{C}(\phi, t)$, we define the corresponding *empirical constraint* $\mathcal{C}_e(\phi, t)$ as follows:

$$\mathcal{C}_e(\phi, t) = \{\overline{\phi_1(x)}^n = t_1 \wedge \ldots \wedge \overline{\phi_m(x)}^n = t_n\} = \{\overline{\phi(x)}^n = t\} \tag{3.5}$$

In this case $\mathbf{C}^n(\phi, t)$ denotes the set of data of length n for which the constraint $\mathcal{C}(\phi, t)$ holds:

$$\mathbf{C}^n(\phi, t) = \{x^n \mid \overline{\phi(x)}^n = t\}.$$

3.3.1 Explicit Expression for the MaxEnt Distribution

We are given a constraint $\mathcal{C}(\phi, t)$ for some function $\phi = (\phi_1, \ldots, \phi_m)$ with range $\mathbf{U} = \mathbf{U}_1 \times \ldots \times \mathbf{U}_m$. We look for the distribution P_{me} that maximizes the entropy $\mathcal{H}$ as given in Equation 3.3 under the constraint $\mathcal{C}(\phi, t)$:

$$P_{me} = \arg \max_{P \in \mathbf{M}(\phi, t)} \mathcal{H}(P) \tag{3.6}$$

We verify below that in the discrete case the maximum entropy distribution P_{me} is given by

$$P_{me}(x) = \frac{1}{Z(\beta)} e^{-\beta^T \cdot \phi(x)} \tag{3.7}$$

where $\beta = (\beta_1, \ldots, \beta_m) \in \mathbf{R}^m$ is a set of m parameters; β^T is the transpose of β (and hence $\beta^T \cdot \phi(x)$ denotes the inner product $\sum_{i=1}^{m} (\beta_i \cdot \phi_i(x))$. The quantity $Z(\beta)$ is used as a normalization factor:

$$Z(\beta) = \sum_{x \in \mathbf{E}} e^{-\beta^T \cdot \phi(x)} \tag{3.8}$$

For all n, $x^n \in \mathbf{E}^n$, $P_{me}(x^n)$ is given by the product distribution of P_{me}: $P_{me}(x^n) = \Pi_{i=1}^{n} P_{me}(x_i)$.

(3.7) only holds under certain mild regularity conditions. In particular, it must be the case that the sum (integral) in (3.8) converges and that, in the integral case, when differentiating (3.8) we can swap the order of differentiation and integration. It is straightforward to show that the following conditions are sufficient to ensure that there exists a unique value of β such that (3.7) defines a probability distribution for which the constraint $E[\phi(X)] = t$ holds.

Conditions for Existence of Maximum Entropy Distributions
Let $\phi = (\phi_1, \ldots, \phi_m)$ be a function with domain $\mathbf{E}$ and range $\mathbf{U} = \mathbf{U}_1 \times \ldots \times \mathbf{U}_m$. Let $t = (t_1, \ldots, t_m) \in \mathbf{U}$. We require the constraint $E[\phi(X)] = t$ to be such that for $i = 1, \ldots, m$:

C1 $\mathbf{U}_i$ is the smallest interval in $\mathbf{R}$ such that $\forall x \in \mathbf{E} : \phi_i(x) \in \mathbf{U}_i$.

C2 If $\mathbf{E}$ is continuous, then ϕ_i is continuous. More precisely, if $\mathbf{E} \subseteq \mathbf{R}^k$, then ϕ_i is the restriction to domain $\mathbf{E}$ of some continuous function $\psi : \mathbf{R}^k \to \mathbf{R}$.

C3 In the discrete case $\mathbf{E}$ contains a finite number of elements. In the continuous case, $\mathbf{E}$ can be written as $\mathbf{E}_1 \times \ldots \mathbf{E}_l$ for some $l \geq 1$, where for each $\mathbf{E}_j$ with $1 \leq j \leq l$:

1. $\mathbf{E}_j$ is a closed interval in $\mathbf{R}$
   or
2. $\mathbf{E}_j = \mathbf{R}$ and there exist $\alpha > 0$ and $C \in \mathbf{R}$ such that

$$\forall x_1, \ldots, x_l : \phi_i((x_1, \ldots, x_j, \ldots, x_l)) \geq |x_j|^\alpha - C.$$

C4 t_i lies in the interior of $\mathbf{U}_i$.

These conditions are sufficient but by no means necessary to ensure the existence of a maximum entropy distribution. However, they cover all the cases needed in this thesis (condition **C3**.2 may seem a bit counterintuitive; it is added to make sure that, if in the evaluation of (3.8) we have to integrate over $\mathbf{R}$, the integral converges). Henceforth, whenever we speak of a maximum entropy distribution for function $\phi : \mathbf{E} \to \mathbf{U}$ and constraint $E[\phi(X)] = t$ we tacitly assume **C1**-**C4** to hold.

To see that P_{me} indeed maximizes the entropy subject to the constraint $\phi(x) = t$, let Q be any distribution other than P_{me} satisfying the constraint and notice that:

$$\begin{aligned} \mathcal{H}(Q) = E_Q[-\ln Q(X)] < E_Q[-\ln P_{me}(X)] = E_Q[\beta^T \phi(X) + \ln Z(\beta)] \\ = \beta^T E_Q[\phi(X)] + \ln Z(\beta) \stackrel{(1)}{=} \beta^T t + \ln Z(\beta) \end{aligned} \quad (3.9)$$

where the inequality follows from the information inequality (3.4) and (1) follows from the fact that we defined Q to satisfy the constraint and hence $E_Q[\phi(X)] = t$. On the other hand,

$$\begin{aligned} \mathcal{H}(P_{me}) = E_{me}[-\ln P(X)] = E_{me}[\beta^T \phi(x) + \ln Z(\beta)] \\ = \beta^T E_{me}[\phi(X)] + \ln Z(\beta) \stackrel{(1)}{=} \beta^T t + \ln Z(\beta) \end{aligned} \quad (3.10)$$

where (1) follows from the fact that, by definition, the maximum entropy distribution satisfies the constraint and hence $E_{me}[\phi(X)] = t$.

Together, (3.9) and (3.10) give that $\mathcal{H}(P_{me}) > \mathcal{H}(Q)$ for all $Q \neq P_{me}$ satisfying the constraint, which is what we had to prove.

The proper value of β can be obtained from the equality:

$$t_i = E_{me}[\phi_i(X)] = -\frac{\partial}{\partial \beta_i} \ln Z(\beta)$$

which is easy to check (see Proposition 3.9).

It can now easily be verified that the distributions mentioned in examples (3.1)-(3.4) are indeed the maximum entropy distributions for the respective constraints; see [30] for details. As an example, the 'empty' constraint, leading to the uniform distribution $P(x) = 1/k$, can always be written as $E[\mathbf{0}(X)] = 0$ where $\mathbf{0}(x)$ is the function that maps each x to 0. The optimal β for this constraint turns out to be 0 and (3.7) becomes the uniform distribution.

3.4 Maximum Entropy Model Classes

As will be seen in the next chapter, it is very useful to consider the class of all Maximum Entropy distributions that satisfy constraints of the same functional form. These classes of distributions play a fundamental role in statistics, where they are known as the 'exponential families'. Many of the model classes that are typically used in statistical practice are of this form; a few examples are the Bernoulli, multinomial, Normal, Beta- and Gamma-distributions.

The maximum entropy model class for a function $\phi : \mathbf{E} \rightarrow \mathbf{U}$ contains exactly the maximum entropy distributions for the (vector of) constraints $E[\phi(X)] = t$ for all t in the interior of $\mathbf{U}$. Formally,

Definition 3.7 *Let a function $\phi : \mathbf{E} \rightarrow \mathbf{U}$ be given. Let $P_{me,\phi}(\cdot|t)$ be the maximum entropy model for constraint $E[\phi(X)] = t$. The* maximum entropy model class $\mathcal{M}_{me}$ *for function ϕ is given by*

$$\mathcal{M}_{me} = \{P_{me,\phi}(\cdot|t) \mid t \in \text{int}(\mathbf{U})\}$$

where $\text{int}(\mathbf{U})$ *stands for the interior of* $\mathbf{U}$.

Note that if conditions **C1-C3** of page 54 hold for $\mathbf{E}$, $\phi(x) = (\phi_1(x), \ldots, \phi_m(x))$ and $\mathbf{U} = \mathbf{U}_1 \times \ldots \mathbf{U}_m$, then the class $\mathcal{M}_{me}$ for function ϕ is guaranteed to exist. Henceforth, whenever we speak of a maximum entropy class for a function ϕ we tacitly assume $\mathbf{E}$, ϕ and $\mathbf{U}$ to be such that **C1-C3** are satisfied.

Example 3.8 Let $\mathbf{E} = \{0, 1\}$. The maximum entropy model for constraint $P(X = 1) = \theta$ is the corresponding Bernoulli model $P(X = 1) = \theta$ (Chapter 1, Definition 1.7). This can be seen by substituting $\beta = \ln(1 - \theta) - \ln \theta$ in Equation 3.7. Hence the Bernoulli model class restricted to $0 < \theta < 1$ is the maximum entropy model class for $\mathbf{E} = \{0, 1\}$ and constraint $P(X = 1) = \theta$. Note that $\beta = 0$ corresponds to the uniform distribution $\theta = 1/2$. As $\beta \rightarrow \infty$, $\theta \rightarrow 0$ and as $\beta \rightarrow -\infty$, $\theta \rightarrow 1$: the smaller $|\beta|$, the closer the distribution is to uniform and hence the higher the entropy. This relation can be shown to hold for maximum entropy model classes in general, see Proposition 3.9.

How to parameterize $\mathcal{M}_{me}$ We can parameterize maximum entropy model classes in two useful ways. Note that the maximum entropy class $\mathcal{M}_{me}$ for function ϕ contains exactly one element for every $t \in \text{int}(\mathbf{U})$. Therefore we can identify the set of parameters Γ with $\text{int}(\mathbf{U})$. This way of characterizing models is called *mean-value parameterization* [82]. Alternatively, as we know from Section 3.3.1, each element of $\mathcal{M}_{me}$ may be written as in (3.7) for a particular value of β. Since, again by (3.7), $E_\beta[\phi(X)]$ is a continuous function of β, it follows that the class $\mathcal{M}_{me}$ can also be characterized as follows:

$$\mathcal{M}_{me} = \{P(\cdot|\beta) \mid E_\beta[\phi(X)] \in \text{int}(\mathbf{U})\}$$

where $P(\cdot|\beta)$ is given by (3.7) and extended to $x^n \in \mathbf{E}^n$ by taking the product distribution. This indexing of elements of $\mathcal{M}_{me}$ by the corresponding β is called the 'natural' parameterization [82].

Exponential Families and Maximum Entropy Model Classes A k-parameter *exponential family* is a family of probability distributions or densities that can be written in the form

$$P(x|\beta) = \frac{1}{Z(\beta)} e^{-\beta^T \cdot \phi(x)} h(x) \tag{3.11}$$

where $\phi(x) = (\phi_1(x), \ldots, \phi_k(x))$, $\beta \in \mathbf{R}^k$ and $Z(\beta) = \sum_{x \in \mathbf{E}} \exp(-\beta^T \phi(x)) h(x)$. $h(x)$ and $\phi(x)$ are functions defined for all $x \in \mathbf{E}$. The *natural parameter space* of an exponential family is given by

$$\Gamma_{nat} = \{\beta \in \mathbf{R}^k \mid Z(\beta) < \infty\}$$

An exponential family is said to be *full* if it contains a model for every $\beta \in \Gamma_{nat}$. The *dimension* of an exponential family is the dimension of its associated Γ_{nat}. An exponential family is said to be of *irreducible dimension* if there is no $(k-1)$-parameter exponential family expressing the same class of probability distributions.

From the point of view of measure theory, the function $h(x)$ may be absorbed in a dominating measure [82]. One can then drop the factor $h(x)$ from (3.11). We see that maximum entropy model classes and exponential families essentially coincide.

Some Useful Facts about Maximum Entropy Model Classes The following proposition lists some useful (and well-known) facts about maximum entropy/exponential model classes that will be used several times in the chapters to come.

Proposition 3.9 *Let $\mathcal{M}_{me}$ be a maximum entropy class for function $\phi(x) = (\phi_1(x), \ldots, \phi_m(x))$ with range $\mathbf{U} = \mathbf{U}_1 \times \ldots \times \mathbf{U}_m$. Let $\beta = (\beta_1, \ldots, \beta_m) \in \Gamma_{nat}$, where Γ_{nat} is the space of parameters in the natural parameterization of $\mathcal{M}_{me}$. Let $1 \leq i, j \leq m$. Then*

1. *The first two (central) moments of $P(\cdot|\beta)$ are determined by the first two derivatives of $Z(\beta)$:*

$$\begin{aligned}\frac{\partial}{\partial \beta_i} \ln Z(\beta) &= -E_\beta[\phi_i(X)] \\ \frac{\partial^2}{\partial \beta_i \partial \beta_j} \ln Z(\beta) &= \operatorname{cov}[\phi_i(X), \phi_j(X)] = \\ &= E[(\phi_i(X) - E[\phi_i(X)])(\phi_j(X) - E[\phi_j(X)])].\end{aligned}$$

2. *Let $\beta_1, \ldots, \beta_m$ all be fixed except β_i. Then*
   (a) *$E_\beta[\phi_i(X)]$ as a function of β_i is strictly decreasing.*
   (b) *If $\beta_i > 0$, then $\mathcal{H}(\beta)$ is a strictly decreasing function of β_i. If $\beta_i < 0$, then $\mathcal{H}(\beta)$ is a strictly increasing function of β_i.*

3. *The log-likelihood* $\ln P(x^n|\beta)$ *as a function of β_i is* concave, *reaching its maximum at the point where $E_\beta[\phi_i(X)] = \overline{\phi_i(x)}^n$. More generally:*

4. *Let E_θ stand for the expectation under the model $P(\cdot|\theta) \in \mathcal{M}_{me}$ defined by the mean-value parameterization (i.e. $E_\theta[\phi(X)] = \theta$). Assume that $\overline{\phi(x)}^n$ lies in the interior of* $\mathbf{U}$. *Then:*

$$E_{\hat{\beta}(x^n)}[\phi(X)] = E_{\hat{\theta}(x^n)}[\phi(X)] = \overline{\phi(x)}^n = \hat{\theta}(x^n) \tag{3.12}$$

5. *$\theta(\beta) := E_\beta[\phi(X)]$ as a function of β is a continuous bijection from Γ_{nat} to* $\operatorname{int}(\mathbf{U})$.

Proof: All of these properties are straightforward to verify by differentiation, and realizing that when we take derivatives of $Z(\beta)$ we are allowed to swap the order of differentiation and integration by our regularity conditions on ϕ. Otherwise, see [82], Chapter 1. □

3.5 The Concentration Phenomenon

The Maximum Entropy distributions[1] have a very special property: *almost all outcomes that satisfy a given empirical constraint have frequencies extremely close to the maximum entropy probabilities.* Jaynes [75] has termed this the 'concentration phenomenon'. It shows that, among all distributions that one may consider as candidates for having generated the actual outcomes, the maximum entropy distribution has a unique status. This is used by some as a justification of the maximum entropy principle (though others reject it, as will be discussed in the next section).

Below we show how the concentration phenomenon arises. It will allow us to prove some of the claims we have made in earlier chapters, and it will play an important

[1] At least in the case of discrete sample spaces.

role in Chapter 4. We only consider finite sample spaces $\mathbf{E} = \{1, \dots, k\}$. Let P_{me} be the maximum entropy distribution for the distributional constraint $E[\phi(X)] = t$ that corresponds to the empirical constraint $\overline{\phi(x)}^n = t$. For simplicity, we make the following assumption: we assume that for each n, there exist x^n such that the empirical constraint $\mathcal{C}_e(\phi, t)$ holds. The arguments can be easily extended to the situation where this is not the case.

Notation Concerning Frequencies We first introduce some notation and terminology regarding frequencies. A tuple of positive rational numbers $\gamma = (\gamma_1, \dots, \gamma_k)$ is called a *frequency distribution* if $\sum_{i=1}^n \gamma_i = 1$. For a given sequence x^n, we let n_i (where $1 \leq i \leq k$) stand for the number of times that outcome i occurs in x^n. Hence $\gamma_i = n_i/n$. We write $\gamma(x^n) = (\gamma_1(x^n), \dots, \gamma_n(x^n)) = (n_1/n, \dots, n_k/n)$ to denote the tuple of frequencies belonging to the set x^n.

Clearly, a sequence x^n satisfies constraint $\overline{\phi(x)}^n$ if and only if

$$\gamma_1(x^n)\phi(1) + \ldots + \gamma_k(x^n)\phi(k) = t. \tag{3.13}$$

Hence whether or not a sequence satisfies a constraint is completely determined by the frequency distribution γ. We say that γ *satisfies* constraint $\overline{\phi(x)}^n$ iff (3.13) holds.

The Concentration Phenomenon Let us define

$$\mathbf{G}^n(\gamma) = \mathbf{G}^n(\gamma_1, \dots, \gamma_k) = \{x^n | \gamma(x^n) = \gamma\}.$$

In words, $\mathbf{G}^n(\gamma)$ stands for the set of sequences in $\mathbf{E}^n$ with frequencies γ. Observe that

$$|\mathbf{G}^n(\gamma)| = \binom{n}{n_1, \dots, n_k} = \frac{n!}{n_1! \cdots n_k!} \tag{3.14}$$

Straightforward application of Stirling's approximation,

$$\ln n! = n \ln n - n + \ln \sqrt{2\pi n} + O(\frac{1}{n}), \tag{3.15}$$

together with (3.14) gives:

$$|\mathbf{G}^n(\gamma)| = e^{n\mathcal{H}(\gamma_1, \dots, \gamma_k)} \cdot c(n) \tag{3.16}$$

Here $\mathcal{H}(\gamma_1, \dots, \gamma_k)$ stands for the empirical entropy[2] of γ; $c(n) = e^{C/n}$ for some constant C and hence goes to 1 with increasing n.

Now define $\mathbf{C}^n = \{x^n \mid \overline{\phi(x)}^n = t\}$ as the set of data x^n for which the constraint $\overline{\phi(x)}^n$ holds. Let $\gamma = (\gamma_1, \dots, \gamma_k)$ be two sets of frequencies that satisfy the given constraint (so that (3.13) holds for both γ and γ'). Note that both $\mathbf{G}^n(\gamma)$ and $\mathbf{G}^n(\gamma')$ are subsets of $\mathbf{C}^n$. Suppose $\mathcal{H}(\gamma) < \mathcal{H}(\gamma')$. By (3.16), the ratio between the number of

[2]The *empirical entropy* $\mathcal{H}(\gamma) = \mathcal{H}(\gamma_1, \dots, \gamma_k)$ is defined analogously to the entropy for probability distributions: $\mathcal{H}(\gamma_1, \dots, \gamma_k) = -\sum_{i=1}^k \gamma_i \ln \gamma_i$.

elements in $\mathbf{C}^n$ with frequencies γ and the number of elements in $\mathbf{C}^n$ with frequencies γ' decreases exponentially with n:

$$\frac{|\mathbf{G}^n(\gamma)|}{|\mathbf{G}^n(\gamma')|} \rightarrow \frac{\exp(n\mathcal{H}(\gamma))}{\exp(n\mathcal{H}(\gamma'))} = e^{-n\alpha} \text{ for some } \alpha > 0. \tag{3.17}$$

We denote by γ_{me} the tuple of frequencies that maximizes the empirical entropy $\mathcal{H}(\gamma)$ subject to the constraint $\overline{\phi(x)}^n = t$. By (3.17), as n increases, the number of elements in $\mathbf{C}^n$ with frequencies γ_{me} becomes exponentially larger than the number of elements in $\mathbf{C}^n$ with frequencies γ for every fixed $\gamma \neq \gamma_{me}$. This is the 'concentration phenomenon' [75]. It says that nearly all outcomes x^n satisfying constraint $\overline{\phi(x)}^n = t$ will have empirical frequencies $(n_1/n, \ldots, n_k/n)$ extremely close to the maximum entropy frequencies $\gamma_{me} = (\gamma_{me,1}, \ldots, \gamma_{me,k})$. These in turn are equal to the maximum entropy *probabilities* $(P_{me}(1), \ldots, P_{me}(k))$ for constraint $E[\phi(X)] = t$.

The concentration phenomenon enables us to prove some claims we made earlier: (3.16) shows that Equation 1.9 of Chapter 1, page 18 holds. Also, (3.17) shows the validity of the claim that the ratio of $\binom{n}{\gamma n}$ to 2^n goes to 0 exponentially fast for all γ except $\gamma = 1/2$ (page 7).

In the next section, we show how the concentration phenomenon may be used to justify the Maximum Entropy Principle.

3.6 Pros and Cons of MaxEnt

Jaynes first published about maximum entropy in 1957. The idea as a whole and its range of applicability has been a topic of fierce debate ever since. Perhaps the most impressing argument in favor of MaxEnt is the simple fact that it works very well in many applications [113, 16, 77]. However, we would like to know why. There are lots of arguments in favor of MaxEnt; but there are just as many against it. We will sketch some of the most well-known of these arguments. Before presenting them, we give Jaynes' example of the 'Brandeis Dice'. This is the standard example of a 'toy' MaxEnt application. Opponents and proponents of the principle alike use it to illustrate their arguments.

Example 3.10 [Brandeis Dice] Let a six-sided die be given, together with the additional information that the expected number of spots coming up in a throw is 4.5 rather than 3.5. We let p_i stand for $P_{me}(X = i)$. In other words, $\mathbf{E} = \{1, 2, 3, 4, 5, 6\}$ and $E[X] = \sum_{i=1}^{6} ip_i = 4.5$. Jaynes [74, 76] calculated the MaxEnt probabilities as follows:

$$(p_1, \ldots, p_6) = (0.05435, 0.07877, 0.11416, 0.16545, 0.23977, 0.34749) \tag{3.18}$$

The question is now, of course, how to interpret these probabilities.

3.6.1 What's good about MaxEnt

One can justify MaxEnt in many ways [77, 114, 115, 144, 153, 152]. Most of these justifications fall in either of two categories. We discuss these in turn.

Almost all outcomes are typical for the MaxEnt distribution The approaches belonging to the first category are all based on the concentration phenomenon we discussed in the previous section. We first show how the concentration phenomenon can be used as a direct justification of maximum entropy. The justifications work only for the case of finite sample spaces $\mathbf{E} = \{1, \ldots, k\}$. Let x^n be some sequence. Suppose that we are given a constraint $\overline{\phi(x)}^n = t$ for some large n. By the concentration phenomenon we know that the overwhelming majority of x^n that satisfy the constraint will have frequencies of occurrence $(\gamma_1, \ldots, \gamma_k)$ (where γ_i is the number of times outcome i occurs in x^n) extremely close to the maximum entropy probabilities $(P_{me}(1), \ldots, P_{me}(k))$. For lack of any more specific knowledge, it may now seem reasonable to infer as a 'best guess' that the frequencies really will be very close to the maximum entropy probabilities. Going one step further, one may also infer that the maximum entropy distribution is a 'best guess' as a model for the data.

This argument consists of two steps, both of which may be subject to criticism: the first step is the 'inductive principle' that the frequencies that can be realized in the greatest number of ways are a best guess of the actual frequencies. The second is that these frequencies may then be identified with probabilities. The first step can be seen as a straightforward extension of Laplace's Principle of Insufficient Reason. Hence *if* one accepts this principle, *then* one may be inclined to accept the first step. Namely, let us assume, in line with the Principle of Insufficient Reason, that all sequences x^n satisfying the given constraint are equally likely (hence uniformly distributed). Then the concentration phenomenon tells us that with overwhelming *probability*, the frequencies will indeed be close to the maximum entropy probabilities.

Rationality Requirements In the other category of MaxEnt justifications, one postulates a number of axioms that should be satisfied by any rational inference procedure. A typical example would be an axiom expression 'permutation invariance': giving different names to outcomes should not affect the results of the inference procedure.

Shore and Johnson [144] formulated this 'permutation invariance' and several other, similarly innocuous-seeming 'rationality requirements'. They were able to show that the only procedure satisfying the requirements is to adopt the maximum entropy distribution for the given constraints. These results were later extended by other writers [115, 153] who gave even simpler and even more intuitive axioms and showed that only maximum entropy satisfies them. It should be noted though that Uffink [157, 158] demonstrates that some of the requirements may not necessarily be so 'rational' as it seems, and that in some of the approaches, there are hidden additional assumptions used in proving the unique status of MaxEnt.

3.6.2 A Problem with MaxEnt: Ex Nihilo Nihil

MaxEnt has been criticized on lots of different grounds by many different people [130, 49, 141, 116, 157, 158]. It would much too ambitious to try and list all objections here. Instead, we focus on a single, 'classic' one, that stands at the basis of many (though by no means all) of the other ones. Here it is:

If you are only given the information that P satisfies constraint $E_P[\phi(X)] = t$ for some ϕ and t, how then, in principle, can you conclude anything more than what follows logically from the constraint? By a 'logical consequence' we mean everything that can be derived from $E_P[\phi(X)] = t$ in conjunction with the axioms of probability theory (e.g. Kolmogorov's axioms [85]).

In the Brandeis dice example, how can you decide upon distribution (3.18)? The distribution with $p_4 = p_5 = 1/2$ and all other probabilities zero will satisfy the constraints just as well, but it will lead to completely different predictions! Never mind – from this point of view – that MaxEnt is the only inference procedure satisfying some rationality requirements, or that the frequencies corresponding to the p_i's can be realized in the greatest number of ways: the very *attempt* to arrive at a complete probability distribution from partial knowledge is flawed!

This argument goes back to Leslie Ellis ([44]; see also [157, 158]) who objected on the same grounds to Laplace's Principle of Insufficient Reason. In 1850 he coined the phrase 'ex nihilo nihil' (literally, 'nothing out of nothing'). We quote from [44]: 'Mere ignorance is no ground for any inference whatsoever. *Ex nihilo nihil.* It cannot be that because we are ignorant of the matter we know something about it.'

In Chapter 4, Section 4.2 we propose a 'weakened' version of Maximum Entropy that partially answers the 'Ex Nihilo Nihil' argument given above. But in order to do this, we first have to establish the relation between MDL and Maximum Entropy. This is done in the following section.

3.7 Maximum Entropy as a Special Case of MDL

In the previous sections we have introduced the Maximum Entropy Principle. We now relate it to the Minimum Description Length Principle. The connection between the two principles can be understood from two points of view. The first point of view is to assume that the given constraints pertain to some otherwise unknown distribution generating the data. In this setting, the code based on the maximum entropy distribution turns out to yield the shortest worst-case expected code length, where 'worst-case' is defined with respect to all generating distributions that satisfy the constraint. The second point of view is to make no distributional assumptions and to assume that one is given an empirical constraint concerning actual data; it turns out that in this case too, a close connection between MDL and Maximum Entropy exists. Here, we focus on the first point of view, essentially following [128]. The second point of view (no distributional assumptions) is explored in [46, 93].

3.7.1 The Maximum Entropy Distribution Minimizes the Maximum Expected Codelength

Let us assume then that data are actually generated by repeated sampling of some 'true' distribution P^* satisfying the constraint $E[\phi(X)] = t$. Let $\mathbf{M}(\phi, t)$ be the class of all probability distributions over $\mathbf{E}$ that satisfy this constraint. If we code the data

Let Q and R be two distributions over $\mathbf{E}$ satisfying $E[\phi(X)] = t$. Let P_{me} be the maximum entropy distribution for this constraint. We have:

$$\begin{aligned} \mathcal{H}(Q) \overset{(1)}{=} E_Q[-\ln Q(X)] &\overset{(2)}{\leq} E_Q[-\ln P_{me}(X)] \\ &\overset{(3)}{=} E_{me}[-\ln P_{me}(X)] \\ &\overset{(4)}{=} \mathcal{H}(P_{me}) \overset{(5)}{\leq} E_{me}[-\ln R(X)] \end{aligned} \tag{3.19}$$

If $Q \neq P_{me}$, inequality (2) becomes strict; if $R \neq P_{me}$, inequality (5) becomes strict.

(1) and (4) follow from the definition of entropy. (2) and (5) follow from the information inequality (3.4) on page 52. (3) follows from equations (3.9) and (3.10) on page 54. (1)-(5) imply $\mathcal{H}(Q) \leq \mathcal{H}(P_{me})$ which expresses the fact that P_{me} maximizes entropy. It is also implied that $E_Q[-\ln P_{me}(X)] \leq E_{me}[-\ln R(X)]$ which expresses the fact that P_{me} minimizes worst-case description length.

Figure 3.1: Maximum Entropy and Minimum Description Length

using the code based on P_{me}, the maximum entropy distribution for this constraint, then we see from Figure 3.1 (which summarizes equations (3.9) and (3.10) of page 54) that the expected code length of $x^n = (x_1, \ldots, x_n)$ becomes

$$E_{P^*}[-\ln P_{me}(X_1, \ldots, X_n)] = n\mathcal{H}(P_{me}). \tag{3.20}$$

This quantity is independent of the 'true' distribution P^*.

Let us now consider the special case in which the 'true' distribution is P_{me}. For every $R \neq P_{me}$, the code based on R satisfies (by Figure 3.1, Inequality (5)):

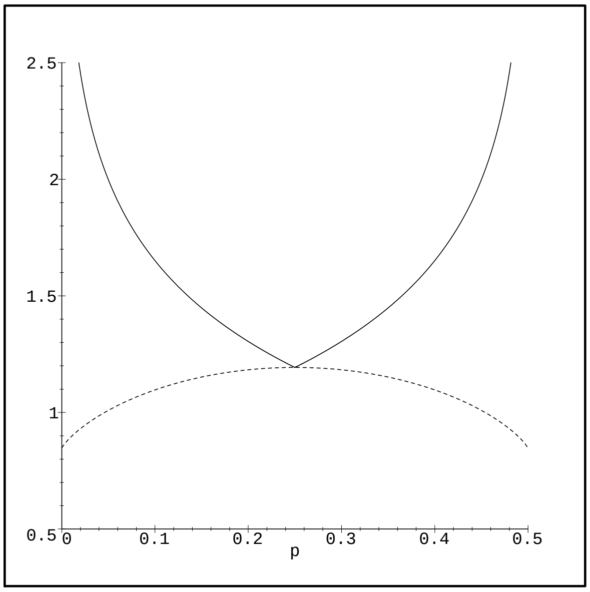
$$E_{me}[-\ln R(X_1, \ldots, X_n)] = n(\mathcal{H}(P_{me}) + \epsilon) \text{ for some } \epsilon > 0,. \tag{3.21}$$

and hence the extra code length needed to encode n outcomes due to using R rather than P_{me} is on the order of n. It follows that when the only information that is given is that the data are distributed according to a distribution P^* satisfying the given constraint, then P_{me}, the maximum entropy distribution for this constraint, minimizes *the worst-case expected code length* among all possible probability distributions over the same sample space:

$$P_{me} = \arg\inf_P \sup_{P^* \in \mathbf{M}(\phi,t)} E_{P^*}[-\ln P(X)] \tag{3.22}$$

This is shown graphically in Figure 3.2, where the specific example of the probability distributions over $\mathbf{E} = \{a, b, c\}$ are used to illustrate the general phenomenon.

Connection to MDL What is the relation between 'minimizing worst-case expected code length' and MDL? The strong law of large numbers [47] says that if data are



The figure depicts both the entropy and the worst-case expected description length of distributions P over $\{a, b, c\}$ satisfying the constraint $P(a) = 1/2$. The x-axis depicts $P(b)$ and ranges over all distributions for which the constraint holds. The dashed curve depicts $\mathcal{H}(P)$, the solid curve depicts $\max_{P^* \in \mathbf{M}(\phi,t)} E_{P^*}[-\ln P(X)]$. The curves touch at $P = P_{me}$ which gives both the maximum entropy and the minimum worst-case expected description length. According to MDL we should pick P_{me} because it minimizes the upper curve, *not* because it maximizes the lower curve. Note that

$$E_{P^*}[-\ln P(X)] = -P^*(a) \ln P(a) - P^*(b) \ln P(b) - P^*(c) \ln P(c).$$

The term $-P^*(a) \ln P(a)$ must be equal to $-1/2 \ln(1/2)$ for all P^* and P that satisfy the constraint. For $P(b) < 1/4$, the worst-case description length is reached for generating distribution $P^*(b) = 1/2; P^*(c) = 0$. For $P(b) > 1/4$, it is given by $P^*(b) = 0; P^*(c) = 1/2$. For $P(b) = P_{me}(b) = 1/4$, every P^* satisfying the constraints yields the same expected description length. The worst-case description length $\max_{P^* \in \mathbf{M}(\phi,t)} E_{P^*}[-\ln Q(X)]$ for distributions Q *not* satisfying the constraint is not depicted, but by equations 3.20 and 3.21 it is easily seen to be larger than $\max_{P^* \in \mathbf{M}(\phi,t)} E_{P^*}[-\ln P_{me}(X)] = E_{me}[-\ln P_{me}(X)]$.

Figure 3.2: The entropy vs. the worst-case expected description length.

drawn by repeated sampling according to a fixed distribution P^*, then, for every coding distribution P, with probability 1 the average code length $n^{-1} \sum_{i=1}^{n} - \ln P(x_i)$ will converge to its expected value $E_{P^*}[-\ln P(X)]$. Hence if we are willing to assume that the data are drawn according to some unknown distribution P^* satisfying the given constraint, and we have no idea as to whether there are any other constraints underlying the data, then modeling our data using (the code based on) P_{me} is in the spirit of the MDL Principle.

It would go too far to say though that the MDL Principle *tells* us to use this code. The reason is that the code based on P_{me} is not *necessarily* the best one to use in this scenario: there are many distributions P' for which, for all n, all $x^n \in \mathbf{E}^n$, $-\ln P'(x^n) \leq -\ln P_{me}(x^n) + c$ for an arbitrarily small constant c. If data are generated by independent sampling from some P^* satisfying the constraints, then the average code length of the code based on P' will converge to some value $L \leq E_{P^*}[-\ln P_{me}(X)]$. As an example, consider the distribution P' with $P'(x^n) = (1/2)P_{me}(x^n) + (1/2)P''(x^n)$ where $P''(x^n)$ is some completely different distribution. Then $-\ln P'(x^n) \leq -\ln P_{me}(x^n) + \ln 2$. If the data happens to be generated by a P^* such that $E_{P^*}[-\ln P''(X)] < E_{P^*}[-\ln P_{me}(X)]$ then the code based on P' will give (with P^*-probability 1) shorter average code lengths than the code based on P_{me}; if P^* is such that $E_{P^*}[-\ln P''(X)] \geq E_{P^*}[-\ln P_{me}(X)]$ then the average code length based on P' will (with P^*-probability 1) converge to the length based on P_{me}.

3.8 Conclusion and Outlook

We have introduced the Maximum Entropy Principle and shown its relation to MDL. In the next chapter we will discuss an important property of maximum entropy model classes; in Chapter 5 the connection between MDL and Maximum Entropy will be exploited to optimize MDL's trade-off between hypothesis complexity and goodness-of-fit.

Chapter 4

Safe Statistics

This Chapter introduces the notion of *reliable* inferences, leading to 'safe statistics'. Section 4.1 introduces reliable inferences in the context of maximum entropy model classes. Reliable inferences allow one to make good predictions and decisions regarding future data under a much wider variety of assumptions than do 'unreliable' inferences. This is extended in Section 4.2 where two forms of applying maximum entropy are distinguished: *safe* and *risky* maximum entropy. The distinction applies not only to maximum entropy, but to statistical procedures in general.

All this is done in preparation of Chapter 5, where 'safe statistics' will be put to use. Essentially, it will allow us to establish in what way we can and in what way we cannot use overly simple models (or, more generally, models that achieve compression but are not in any way related to the 'true' model generating the data). In order to keep the treatment as general as possible, in this chapter we abstract away from the specific statistical procedure used and do not focus specifically on MDL. In the next chapter 'safe statistics' will be connected to MDL.

4.1 You can trust Maximum Entropy models (in some respects)

This section introduces the central ideas of this chapter. We start by noting an extremely important property shared by all Maximum Entropy model classes. Let $\mathcal{M} = \{P_{me,\phi}(\cdot|\theta) \mid \theta \in \mathbf{U}\}$ be a class of maximum entropy distributions for function $\phi : \mathbf{E} \rightarrow \mathbf{U}$ with $\phi(x) = (\phi_1(x), \ldots, \phi_m(x))$. Let $x^n \in \mathbf{E}^*$ be an arbitrary data sequence such that $\overline{\phi(x)}^n$ lies in the interior of $\mathbf{U}$. The following proposition says that the expected value under $\hat{\theta} = \hat{\theta}(x^n)$ of linear transformations of ϕ is equal to their average value over x^n. In particular, this implies that the expected code length according to the maximum likelihood model $\hat{\theta}(x^n)$ is equal to the actual average code length achieved by $\hat{\theta}(x^n)$. More precisely,

Proposition 4.1 *Let $\mathcal{M}$, ϕ, x^n and $\hat{\theta}(x^n)$ be as above.*

For arbitrary $\alpha = (\alpha_0, \alpha_1, \dots, \alpha_m) \in \mathbf{R}^{m+1}$*, let* $\psi(x) = \sum_{i=1}^{m} \alpha_i \phi_i(x) + \alpha_0$*. We have:*

$$\begin{aligned} E_{\hat{\theta}}[\psi(X)] &= \overline{\psi(x)}^n & (4.1) \\ E_{\theta}[\psi(X)] &\neq \overline{\psi(x)}^n \textit{ for all } \theta \in \mathbf{U} \textit{ with } \theta \neq \hat{\theta}. & (4.2) \end{aligned}$$

In particular this implies:

$$E_{\hat{\theta}}[-\ln P(X|\hat{\theta})] = -\frac{1}{n} \ln P(x^n|\hat{\theta}) \tag{4.3}$$

Proof: (4.1) and (4.2) are immediate from Proposition 3.9 (page 56) and the fact that both expectation and averaging are linear operations. To prove (4.3), observe that, from the natural parameterization (Equation 3.7, page 53) of a maximum entropy model $P(\cdot|\beta)$ one can see that there exists a β such that for all $x \in \mathbf{E}$, it holds that $-\ln P(x|\theta) = \beta\phi(x) + Z(\beta)$. Hence there exist α_0, α_1 such that for all $x \in \mathbf{E}$, $-\ln P(x|\theta) = \alpha_0\phi(x) + \alpha_1$, which, by (4.1) and independence ($\sum_{i=1}^{n} \ln P(x_i) = \ln P(x^n)$) implies (4.3). □

This proposition has a very important consequence which is best introduced by means of an example.

Example 4.2 [**Brandeis once more**] As in Chapter 3, Example 3.10, consider a six-sided die. This time, let the sequence of throws x^n consist of $n/2$ 4's and $n/2$ 5's, but suppose we are only given the partial information that $\overline{x}^n = 4.5$. Let $\mathcal{M}_{me}$ be the maximum entropy model class for constraints $E[X] = t$ where $1 < t < 6$. By (4.1) and (4.2) the maximum likelihood model $P(\cdot|\hat{\theta})$ in the model class $\mathcal{M}_{me}$ is equal to the maximum entropy model P_{me} for the constraint $E[X] = 4.5$. The probabilities $P_{me}(X = i)$ were given on page 59; we repeat them for convenience:

$$(p_1, \dots, p_6) = (0.05435, 0.07877, 0.11416, 0.16545, 0.23977, 0.34749)$$

Clearly, these probabilities do not coincide with the frequencies of $1, \dots, 6$ in the sequence x^n. Still, it holds that

$$-\ln P(x^n|\hat{\theta}) = -\frac{n}{2} \ln p_4 - \frac{n}{2} \ln p_5 = E_{\hat{\theta}}[-\ln P(X^n|\hat{\theta})] = n\mathcal{H}(\hat{\theta})$$

where the second equality follows from (4.3) in the proposition above but can also be checked numerically. If we had used the model $\hat{\theta}$ all the time for prediction (for example, because we already knew from some previous data that $E[X] = 4.5$), and we had predicted using the logarithmic loss $\text{LOSS}_{lg}(e; \hat{\theta}) = -\ln P(e|\hat{\theta})$ (Chapter 2, Section 2.7), then our prediction error would have been *exactly equal to what we expected it to be* (namely $E_{\hat{\theta}}[-\ln P(X^n|\hat{\theta})]$) even though the *frequencies* of actual outcomes $(\gamma_1, \dots, \gamma_6) = (0, 0, 0, 1/2, 1/2, 0)$ would have been very different from their expectations $E_{\hat{\theta}}[\mathcal{I}(X = 1)] = p_1, \dots,$ $E_{\hat{\theta}}[\mathcal{I}(X = 6)] = p_6$. (here $\mathcal{I}$ denotes the indicator function defined as in Equation 3.2 on page 51).

From Chapter 2, Section 2.7 we know that the logarithmic loss with respect to a probability distribution P can be interpreted as a generic kind of loss function, but also as the number of bits needed to encode an outcome using the code based on P and as the accumulated loss in capital when betting proportionally on the basis of P.

It follows that, if we had placed bets on $x_1, \dots, x_n$ proportionally on the basis of $\hat{\theta}$, the (logarithm of the) accumulated loss we would have made would have been exactly as large as we would have expected it to be on the basis of $\hat{\theta}$. This holds even though other aspects of x^n (like the frequencies) may be completely different from what we would expect them to be on the basis of $\hat{\theta}$. Note that the equality of expected and actual loss does not just hold for our particular x^n with $n/2$ 4's and $n/2$ 5's; rather, by (4.3) in Proposition 4.1 above it holds for *every* x^n with $\overline{x}^n = 4.5$. Hence it also holds for many less 'suspicious' samples for which $\gamma_i > 0$ for each $1 \leq i \leq 6$.

Reliable Prediction Apparently, some aspects of the data (code length, prediction loss in proportional betting, the average of x) can be *reliably* predicted on the basis of the maximum entropy model while others (for example, the frequencies of the outcomes) cannot. This is the central point of this chapter.

In general, we will be interested in what can be reliably predicted and what not from a model that is only partially correct. In the present example, we assumed that we were only given the information that $\overline{x}^n = 4.5$; therefore, we could not simply estimate the underlying probability distribution by setting the probabilities equal to the frequencies and we were forced to use a model that is potentially incorrect. In other examples, the whole sequence x^n may be available, but our model class does not contain a model that captures all of the regularity in the data. For example, in Chapter 1, Section 1.5, we showed the case of data being generated by a Markov chain with stationary distribution $1/3$ that was modeled by a Bernoulli distribution. In such a situation, we would like to be able to infer that the optimal Bernoulli model for the data, $\hat{\theta} \approx 1/3$ can be reliably used to predict the frequency of 1s in future data but not to predict the frequency of 11 (two 1's in a row) (page 24).

We will now put the central idea of the example in more abstract terms. Suppose we are trying to learn the parameters of a probability distribution from a sample $x_1, \dots, x_k$. We then use the values we found to predict future data, say $x_{k+1}, \dots, x_n$. Clearly, the model in the class that fits best the future data is $\hat{\theta}(x_{k+1}, \dots, x_n)$. By the arguments in Chapter 2, Section 2.2 this is also the model that, once it is given, allows us to code $x_{k+1}, \dots, x_n$ using a minimal number of bits; finally, had we predicted $x_{k+1}, \dots, x_n$ on the basis of $\hat{\theta}(x_{k+1}, \dots, x_n)$ we would have obtained the minimal possible loss in the game of proportional betting described in Chapter 2, Section 2.7. We see that this model is optimal for data $x_{k+1}, \dots, x_n$ in several different ways. We will call this 'model that will turn out to be the best' the *future-optimal model* and denote it by $\hat{\theta}_{\text{fut}}$.

Suppose now that for some reason or other, our guess $\ddot{\theta}$ based on $x_1, \dots, x_k$ is equal to $\hat{\theta}_{\text{fut}} = \hat{\theta}(x_{k+1}, \dots, x_n)$. In other words, by some coincidence we hit upon the model that will turn out to yield the best predictions of future data. Now if our model class is a maximum entropy model class for function $\phi(x) = (\phi_1(x), \dots, \phi_m(x))$ and we use $\ddot{\theta} = \hat{\theta}_{\text{fut}}$ then not only will our prediction error (as measured in logarithmic loss) be minimal, the error will also be *exactly* equal to what we expect it to be: according to $\hat{\theta}_{\text{fut}}$, our expected error will be $E_{\hat{\theta}_{\text{fut}}}[-\ln P(X_{k+1}, \dots, X_n | \hat{\theta}_{\text{fut}})]$. By Proposition 4.1

this coincides with the error $-\ln P(x_{k+1}, \ldots, x_n|\hat{\theta}_{\text{fut}})$ which we will actually make. The proposition further implies that if we estimate the averages $\overline{\phi_1(x)}, \ldots, \overline{\phi_m(x)}$, by $E_{\hat{\theta}_{\text{fut}}}[\phi_1(X)], \ldots, E_{\hat{\theta}_{\text{fut}}}[\phi_m(X)]$ respectively, then our estimates will be perfect.

Reasonable Inference Procedures In a more realistic situation, if given enough initial data $x_1, \ldots, x_k$ and a good procedure for estimating $\ddot{\theta}$ (for example, two-part code MDL, Chapter 1, Section 1.3), then $\ddot{\theta}$ may be quite close, but not equal to $\hat{\theta}_{\text{fut}}$. In Chapter 5.2, Section 5.3 (Lemma 5.14) it will be shown formally that, if $\ddot{\theta}$ is based on a large enough sample and data are generated by independent sampling according to some arbitrary distribution P^*, then under fairly general conditions on the model class $\mathcal{M}$, exactly this situation will occur with probability 1.

It is this situation that we will study in this section and the next: the results we present will only be useful if the assumption that the estimator $\ddot{\theta}$ will be close to $\hat{\theta}_{\text{fut}}$ is justified. To keep the discussion general, we will not restrict ourselves to MDL estimators. Rather, we will assume that we use some reasonable inference procedure that, for those sequences $x_1, x_2, \ldots$ that are such that there really is 'something to infer', is guaranteed to work well for large enough samples:

Definition 4.3 *Let $\mathcal{M}$ be a model class that is finitely parameterized by some Γ. Let $\mathcal{L}_{\mathcal{M}}$ be an estimation procedure that, for each n, $x^n \in \mathbf{E}^n$ outputs an estimator $\ddot{\theta}(x^n) \in \Gamma$. We call $\mathcal{L}_{\mathcal{M}}$* reasonable *if for every sequence $x_1, x_2, \ldots$ for which $\hat{\theta}(x^n)$ converges to some value, $\ddot{\theta}(x^n)$ converges to that same value; that is*

> if $\lim_{n\to\infty} \hat{\theta}(x^n)$ *exists and is equal to $\hat{\theta}_{fut}$ for some $\hat{\theta}_{fut} \in \Gamma$,*
> then $\lim_{n\to\infty} \ddot{\theta}(x^n)$ *must also exist and be equal to $\hat{\theta}_{fut}$.*

Hence if the model in $\mathcal{M}$ that maximizes the likelihood converges to some value $\hat{\theta}_{\text{fut}}$ then a reasonable estimator should converge to that same value. Note that we assume $\mathcal{M}$ to be finitely parameterized. It can then be seen by the argument in Chapter 1, Section 1.4.1 that the 2-part code MDL estimator and the ML estimator will converge to the same value, hence the MDL estimator is reasonable. Similarly, one can show that (if a reasonable prior is used) the Bayesian MAP estimator and of course, the ML estimator itself are 'reasonable'. Our definition of 'reasonable estimator' does *not* mean that we think the ML estimator is superior to the other estimators; we do not rule out the possibility that for finite n, a different estimator $\ddot{\theta}(x^n)$ may in some cases with high probability be closer to $\hat{\theta}_{\text{fut}}$ than $\hat{\theta}(x^n)$. The point is that *if* there exists a value $\hat{\theta}_{\text{fut}}$ to which $\hat{\theta}(x^n)$ converges *then* this value will in the limit for large n fit the data better than any other model in $\mathcal{M}$. For this reason we demand 'reasonable' estimators to converge to $\hat{\theta}_{\text{fut}}$.

Guaranteed Performance Since for every maximum entropy model class $\mathcal{M}_{me}$, the expectation $E_\theta[\phi(X)]$ is a continuous function of θ (Proposition 3.9), it follows by Proposition 4.1 that in a situation in which the estimate $\ddot{\theta}(x^k)$ based on x^k is close enough to the future-optimal $\hat{\theta}(x_{k+1}, \ldots, x_n)$:

1. The expected value of (every linear combination of) the functions $\phi_1, \ldots, \phi_m$ under the estimator $\ddot{\theta}(x^k)$ will be close to the actual average value they will get on future data $x_{k+1}, \ldots, x_n$.

2. In particular, since the logarithmic loss is a linear combination of the functions $\phi_1, \ldots, \phi_m$, the expected value of logarithmic loss under $\ddot{\theta}$ will be close to the actual average logarithmic loss that will be achieved when predicting future data.

In order to get such a guaranteed performance, it is essential that

1. We use a maximum entropy model class, and

2. We only consider functions ψ that can be written as linear combinations of the ϕ_i.

In Chapter 5.2, Example 5.1 we shall see an example of a non-maximum entropy class for which it can happen that even though $\ddot{\theta}$ is a very good estimate of $\hat{\theta}_{\text{fut}}$ (in the sense that $\ddot{\theta}$ is very close or even equal to $\hat{\theta}_{\text{fut}}$), still $E_{\ddot{\theta}}[\psi(X)]$ is quite different from $\lim_{n\rightarrow\infty} \overline{\psi(x)}^n$ for some function $\psi(x)$ that we would like to estimate: the expectation of $\psi(X)$ based on $\ddot{\theta}$ may be quite different from the average value this function will attain on future data. (we have no proof though that this can happen for all functions ψ and all model classes $\mathcal{M}$ that are not maximum entropy classes; Rissanen ([128], page 112) claims that this is the case but provides no proof). Nevertheless, we shall see in the next chapter that if the functions to be predicted depend *themselves* on the model $\ddot{\theta}$ (in other words, ψ is a function of both x and $\ddot{\theta}$) then we can get around the maximum entropy requirement after all.

If we do use a maximum entropy model class $\mathcal{M}_{me}$ for ϕ but consider a function ψ that is not a linear combination of the ϕ_i, then we may get the same disagreement between expectations and averages once more (again, we have no formal proof that this can happen for *all* non-linear ψ and all $\mathcal{M}_{me}$). We have seen an instance of this in Example 4.2 at the beginning of this section. However, at least for discrete maximum entropy model classes, the disagreement between expectations and averages will only take place for an exponentially small fraction of the data: as we saw in the previous chapter (Section 3.5), for large enough n, nearly all outcomes for which $\overline{\phi(x)}^n = t$ will have frequencies (almost) equal to the probabilities of the MaxEnt distribution for the constraint $E[\phi(X)] = t$, which (Proposition 4.1) coincides with the maximum likelihood model in $\mathcal{M}_{me}$ for constraint $\overline{\phi(x)}^n = t$. One verifies immediately that for such sequences, we have $E_{\hat{\theta}}[\psi(X)] \approx \overline{\psi(x)}^n$ for *every* function $\psi : \mathbf{E} \rightarrow \mathbf{U}$, and not just for linear combinations of the ϕ_i. Nevertheless in many practical situations the actual sequence x^n will indeed be such that the frequencies are very different from the maximum entropy probabilities (see the next chapter; for a concrete example, see Example 5.24 in the Epilogue to part I, page 120).

In the next section we will construct a framework to formally deal with the notion of 'guaranteed performance'.

4.2 Reliable Decisions

It was shown in the previous section that if the estimate $\ddot{\theta}$ of the parameters of a distribution in a maximum entropy model class $\mathcal{M}_{me}$ is close to the future-optimal model $\hat{\theta}_{\text{fut}}$, then we can reliably estimate averages of some functions over $\mathbf{E}$ while we cannot reliably estimate averages over some other functions. In realistic settings, we would not only like to estimate averages of functions but also, more generally, to arrive at optimal *decisions* concerning future data. How to make such decisions is the subject of decision theory [17, 39]. Below we first review some basic notions of decision theory. We will then define the notions of 'reliable' and 'unreliable' estimates and decisions.

4.2.1 Decision Theory

A standard concept in decision theory is the *loss function.* We will restrict ourselves here to loss functions $\text{LOSS} : \mathbf{E} \times \mathcal{D} \to \mathbf{R} \cup \{\infty\}$. Here $\mathbf{E}$ is a space of possible outcomes and $\mathcal{D}$ is a space of possible decisions. $\text{LOSS}(e;\delta)$

stands for the loss that is incurred if, in a situation where, after the *decision* δ has been made, the outcome e occurs. Decisions have to be made before the data occurs; hence we cannot simply make the decision that minimizes the loss for the actual outcome. Instead, we should make some decision that is likely to lead to a small loss. Let us assume that we model our data using a probability distribution P. P will often be arrived at on the basis of previously seen outcomes (data). We may now pick the δ^* that minimizes our *expected loss*:

$$\delta^* = \delta^*(\text{LOSS}, P) = \arg\min_{\delta} E_P[\text{LOSS}(X;\delta)] \tag{4.4}$$

We call $\delta^*(\text{LOSS}, P)$ the *optimal decision*[1] based on P and $\text{LOSS}(\cdot;\cdot)$.

Predicting is a Special Case of Deciding A 'prediction' can be seen as a special kind of a 'decision'. We consider two kinds of prediction in this thesis: *set-wise* and *point-wise* prediction. We have already encountered set-wise prediction in Chapter 2, Section 2.7. It amounts to dividing a unit investment over the space $\mathbf{E}$ of possible outcomes. Hence you have to assign each $e \in \mathbf{E}$ a weight $w(e)$ such that $\sum_{e \in \mathbf{E}} w(e) = 1$. The loss you incur if the actual outcome is e then depends on both e and the function w. In this case the decision space is the space of all possible weight assignments, which corresponds to the class of all possible probability distributions over $\mathbf{E}$. In Chapter 2, Section 2.7 we considered the logarithmic loss where $\text{LOSS}(e, w) = -\log w(e)$ (we interpreted w as a probability distribution there).

Pointwise prediction works somewhat differently: here we have to specify a single value $\hat{e} \in \mathbf{E}$ and we are then charged with a loss $\text{LOSS}(e, \hat{e})$ where e is the actual outcome. A loss function $\text{LOSS} : \mathbf{E} \times \mathbf{E} \to \mathbf{R} \cup \{\infty\}$ is also called a *distortion function.*

[1] The δ^* optimizing (4.4) is usually called the 'Bayes action' [17]. Strictly speaking, it is a concept belonging to Bayesian, not frequentist decision theory. However, it is also used outside of Bayesian statistics; see for example [166].

An example of a distortion function is the squared error function for a fixed value of x. It takes the form $\text{ER}(y|H,x) = (y - \hat{y})^2$ where $\hat{y} = H(x)$. Hence different values of H lead to different predictions $\hat{y}$ of y.

4.2.2 Reliable Decisions

We will now formally define the notions of reliable estimations and decisions. The following definition captures the idea of reliable estimation as explained on page 67. In the definition, $\text{int}(\mathbf{U})$ stands for the interior of the set $\mathbf{U}$.

Definition 4.4 *Let $\mathcal{M}$ be a class of probabilistic models over $\mathbf{E}$ parameterized by some Γ. Let $\psi : \mathbf{E} \to \mathbf{U}$ be a given function. If, for all n, all $x^n \in \mathbf{E}^n$ with $\hat{\theta}(x^n) \in \text{int}(\Gamma)$, we have*

$$E_{\hat{\theta}(x^n)}[\psi(X)] = \overline{\psi(x)}^n \tag{4.5}$$

and, moreover, $E_\theta[\psi(X)]$ is a continuous function of θ, then we say that averages of ψ can be reliably estimated *on the basis of $\mathcal{M}$. Otherwise we say that averages of ψ cannot be reliably estimated on the basis of $\mathcal{M}$.*

Note that we require $E_{\hat{\theta}}[\psi(X)]$ to be a continuous function of θ. As discussed on page 68, in realistic situations, we will not be able to find an estimator $\ddot{\theta}$ that is equal to the future optimal $\hat{\theta}_{\text{fut}}$; rather, we will have $|\ddot{\theta} - \hat{\theta}_{\text{fut}}| = \epsilon$ for some small ϵ; the continuity requirement makes sure that the estimates $E_{\ddot{\theta}(x^n)}[\psi(X)]$ will still be reasonably close to the average of ψ over future data.

The definition becomes slightly more involved in the decision-theoretic case. We first note that when the loss function LOSS is known from the context, we will write $\delta^*(x^n)$ as short for $\delta^*(\text{LOSS}, P(\cdot|\hat{\theta}(x^n)))$. Hence $\delta^*(x^n)$ is the decision that minimizes the expected loss where the expectation is under the maximum likelihood model for data x^n.

Definition 4.5 *Let $\mathcal{M}$ be as above. Let LOSS be a given loss function with domain $\mathbf{E} \times \mathcal{D}$. Let, for data $x^n \in \mathbf{E}^n$, $\delta^*(x^n)$ be the optimal decision based on Equation 4.4 for loss function LOSS and probability distribution $P(\cdot|\hat{\theta}(x^n))$. If, for all n, all $x^n \in \mathbf{E}^n$ with $\hat{\theta}(x^n) \in \text{int}(\Gamma)$, we have*

$$E_{\hat{\theta}(x^n)}[\text{LOSS}(X; \delta^*(x^n))] = \frac{1}{n} \sum_{i=1}^{n} \text{LOSS}(x_i; \delta^*(x^n)) \tag{4.6}$$

and, moreover, for all $\delta \in \mathcal{D}$, $E_\theta[\text{LOSS}(X;\delta)]$ is a continuous function of θ, then we say that LOSS can be reliably used on the basis of $\mathcal{M}$. Otherwise we say that LOSS cannot be reliably used on the basis of $\mathcal{M}$.

In other words: let x^n be arbitrary but fixed. Then the expected loss under $\hat{\theta}(x^n)$ on the basis of the decision $\delta^*(x^n)$ that minimizes this expected loss is equal to the average loss of $\delta^*(x^n)$ over x^n. So, if we base our decisions to predict $x_1, \ldots, x_n$ on an estimate $\ddot{\theta}$ that turns out to be such that $\ddot{\theta} \approx \hat{\theta}(x^n)$, and we always opt for the decision that minimizes expected loss, then our average loss over x^n will be close to the loss we expect to make based on our decisions (this will be illustrated in Example 4.11 on page 74).

Having defined what functions and loss functions can be reliably estimated (used), we proceed to define reliable *estimates* and *decisions*:

Definition 4.6 *Let $\mathcal{L}_{\mathcal{M}}$ be a reasonable estimation procedure based on a class $\mathcal{M}$ as in Definition 4.3. Let $\ddot{\theta}(x^n)$ be the estimator that is output by $\mathcal{L}_{\mathcal{M}}$ for given data x^n. The estimates $E_{\ddot{\theta}(x^n)}[\psi(X)]$ of $\overline{\psi(x)}$ are called reliable if and only if the average of ψ can be reliably estimated on the basis of $\mathcal{M}$. The decisions $\delta^*(\text{LOSS}, \ddot{\theta}(x^n))$ are called reliable iff* LOSS *can be reliably used on the basis of $\mathcal{M}$.*

Suppose you issue the estimate $\ddot{\theta}(x^k)$ for data $x_1, \dots, x_k$. As explained on page 67, the essence of 'reliability' is that *if* future data $x_{k+1}, \dots, x_n$ is such that $\ddot{\theta}(x^k) = \hat{\theta}(x_{k+1}, \dots, x_n)$, *then* your estimate $E_{\ddot{\theta}(x^k)}[\psi(X)]$ is equal to $\overline{\psi(x)}$, the average being taken over $x_{k+1}, \dots, x_n$. This explains the important point that, while on the one hand there is no reason that $\ddot{\theta}(x^k)$ should itself be the maximum likelihood estimator, on the other hand the definition of 'a function ψ that can be reliably estimated' does make use of the maximum likelihood estimator. For *present* data we allow every estimator that has a reasonable possibility of being close to the maximum likelihood estimator over *future* data.

The following two propositions are now straightforward:

Proposition 4.7 *Let $\mathcal{M}_{me}$ be the maximum entropy model class for a function $\phi : \mathbf{E} \rightarrow \mathbf{U}$ with $\phi(x) = (\phi_1(x), \dots, \phi_m(x))$. Let ψ be a function that can be written as a linear combination of the ϕ_i. Then*

1. *Averages of the function ψ can be reliably estimated on the basis of $\mathcal{M}_{me}$.*
2. *Let P_{me} be the maximum entropy distribution for given constraint $\overline{\phi(x)}^n = t$. The estimates $E_{me}[\psi(X)]$ of $\overline{\psi(x)}$ are reliable.*

Proof: Item (1) is immediate from Proposition 4.1. By item (1). and Definition 4.6 the estimates $E_{\hat{\theta}(x^n)}[\psi(X)]$ of $\overline{\psi(x)}$ are reliable. Since the maximum entropy distribution P_{me} for given constraint $\overline{\phi(x)}^n = t$ coincides with the maximum likelihood model $P(\cdot|\hat{\theta}(x^n))$ (Proposition 3.9), the estimates $E_{me}[\psi(X)]$ of $\overline{\psi(x)}$ are also reliable. □

Proposition 4.8 *Let $\mathcal{M}_{me}$ and ϕ be as above. Let $\text{LOSS}_{lg}(x, \theta) := -\log P(x|\theta)$ be the logarithmic loss function. Then:*

1. *$\delta^*(x^n) = \delta^*(\text{LOSS}_{lg}, P(\cdot|\hat{\theta}(x^n))) = \hat{\theta}(x^n)$ ('the optimal weight assignment over* **E** *when expectation is taken over $\hat{\theta}(x^n)$ coincides with $\hat{\theta}(x^n)$').*
2. *LOSS_{lg} can be reliably used for prediction on the basis of $\mathcal{M}_{me}$.*

Proof: We first prove (1). We have to find the probability distribution w over **E** minimizing $E_{\hat{\theta}(x^n)}[-\log P(X|w)]$. By the Information Inequality 3.4 (Chapter 3, page 52), this expression is minimized for $w = \hat{\theta}(x^n)$. Hence $\delta^*(x^n) = w = \hat{\theta}(x^n)$. This proves (1). Now for (2). By (1). and Proposition 4.1 we have $\text{LOSS}_{lg}(y; \delta^*(x^n)) = \text{LOSS}_{lg}(y; \hat{\theta}(x^n)) = -\log P(y|\hat{\theta}(x^n)) = \sum_{i=1}^{m} \alpha_i \phi_i(y) + \alpha_0$ for some $(\alpha_0, \ldots, \alpha_m)$. This is a linear combination of the ϕ_i. Hence, by Proposition 4.7, averages of $\text{LOSS}_{lg}(y; \delta^*(x^n))$ can be reliably estimated. Then by definitions 4.4 and 4.6 LOSS_{lg} can be reliably used on the basis of $\mathcal{M}_{me}$. □

Summary We summarize the meaning of 'reliable': *if* $\ddot{\theta}$ is a *good* estimate in the sense that it will turn out to be close to $\hat{\theta}_{\text{fut}}$, then decisions which are 'reliable' according to Definition 4.4 are guaranteed to give a loss over future data that is close to the $\ddot{\theta}$- expected loss over future data; for decisions which are 'unreliable' we have no such guarantee.

4.2.3 What we can and cannot do

It seems that, if we restrict ourselves to making only provably reliable decisions, then we can hardly conclude *anything* about what decisions to take. However, there are at least three situations in which we can still do something useful:

Proportional Betting In the case of finite $\mathbf{E}$, we can place bets on outcome e_i proportionally to $P(e_i|\ddot{\theta})$. In Chapter 2, Section 2.7 we showed this to be equivalent to predicting with the logarithmic loss function. Hence by Proposition 4.8 proportional betting is reliable: the logarithm of our accumulated capital after n bets will be close to our expectation of the logarithm of our accumulated capital after n bets.

Indicator Functions Consider the case of finite $\mathbf{E} = \{1, \ldots, k\}$ where the function ϕ is the vector of all indicator functions

$$\phi(x) = (\mathcal{I}(x = 1), \ldots, \mathcal{I}(x = k)).$$

In this situation there are no strong restrictions, since we can reliably predict future values of all functions ψ with domain $\mathbf{E}$. The reason is simply that *all* functions of discrete data can be written as linear combinations of indicator functions. In the Brandeis example, $\phi(x) = x = \sum_{i=1}^{6} i\mathcal{I}(x = i)$ which is clearly a linear combination of indicator functions. From this example we immediately see how to write an arbitrary function on a finite domain as a linear combination of indicator functions.

Moment Constraints The first two moments ($E[X]$ and $E[X^2]$) determine some interesting aspects of a probability distribution. If $\mathbf{E}$ is continuous and $\phi = (x, x^2)$, then we can reliably estimate $\phi_1(x) = x$ and $\phi_2(x) = x^2$. A good estimate $\ddot{\theta}$ is then guaranteed to have $E_{\ddot{\theta}}[X]$ and $E_{\ddot{\theta}}[X^2]$ close to $E_{\hat{\theta}_{\text{fut}}}[X]$ and $E_{\hat{\theta}_{\text{fut}}}[X^2]$ respectively. Using *only* these first two moments, we may use for example Chebyshev's inequality to get bounds on the probability that an outcome is more than k standard deviations away from the mean $E_{\ddot{\theta}}[X] \approx E_{\hat{\theta}_{\text{fut}}}[X]$.

There are also many things we *cannot* do if we want to make only reliable estimates and decisions. Roughly stated, *we cannot add and multiply probabilities any more in all the situations where we used to.*

Below we give several examples of what is possible and what is not with reliable estimates and decisions.

Example 4.9 [some probabilities can be added, some cannot] For a six-sided die, suppose we are given that $\gamma_1 = \overline{\mathcal{I}(x=1)}^n$ and $\gamma_2 = \overline{\mathcal{I}(x=2)}^n$. These are the observed frequencies of 1s and 2s in n throws. In this case P_{me} is given by $P_{me}(1) = \gamma_1, P_{me}(2) = \gamma_2, P_{me}(3) = \ldots = P_{me}(6) = (1-\gamma_1-\gamma_2)/4$. P_{me} is also the ML distribution for the sample x^n with respect to the maximum entropy model class $\mathcal{M}_{me}$ of function $\phi(x) = (\phi_1(x), \phi_2(x)) = (\mathcal{I}(x=1), \mathcal{I}(x=2))$. Let us regard P_{me} as an estimate to be used for predicting future data. As $\mathcal{I}(x = 1 \vee x = 2) = \mathcal{I}(x = 1) + \mathcal{I}(x=2)$, we see that $\mathcal{I}(x = 1 \vee x = 2)$ is a linear combination of the functions ϕ_i. By Proposition 4.7 we can reliably use P_{me} to predict that the frequency of the event of throwing a number of spots less than 3 will be $E_{me}[\mathcal{I}(X = 1 \vee X = 2)] = \gamma_1 + \gamma_2$. Since $\mathcal{I}(x=3)$ cannot be written as a linear combination of ϕ_1 and ϕ_2, we cannot use this proposition to prove that it can be reliably predicted. Indeed, it cannot: the sequence x^n may of course be such that $\gamma_3 \neq P_{me}(3)$.

Example 4.10 [if $P(A)$ can be reliably predicted, then so can $P(\neg A)$] Suppose the indicator function $\mathcal{I}(A)$, where A is some proposition regarding the data, is a linear function of ϕ and hence can be reliably predicted if we use the maximum entropy model class $\mathcal{M}_{me}$ for ϕ. Then $\neg A$, the negation of A, has as indicator function $1 - \mathcal{I}(A)$ and can therefore also be reliably predicted.

Example 4.11 [decision theory] Let $\mathbf{E} = \{0, 1\}$ and let $x^n \in \mathbf{E}^n$. We model x^n by the Bernoulli model class, which (Example 3.8, page 55) is equivalent to the maximum entropy model class for the function $\phi(x) = x$. Let n_0 be the number of 0s and n_1 be the number of 1s in x^n. Let $\hat{\theta}(x^n) = \gamma = n_1/n$, i.e. the frequency of 1s in x^n is γ. Suppose for some reason or other we are in possession of a perfect estimate $\ddot{\theta} = \hat{\theta}(x^n)$ which we will use to predict the x^n. Note that every function ψ over $\mathbf{E}$ can be written as $\psi(x) = ax + b$ which is a linear combination of $\phi(x)$. Every loss function $\textsc{loss}(x; \delta)$ can, for each fixed δ, be thought of as a function over $\mathbf{E}$. It follows, by Proposition 4.7 that we can make reliable predictions against every loss function. We can also directly show this as follows: Let $\textsc{loss}$ be an arbitrary loss function such that the optimal decision $\delta^* = \arg\min_{\delta} E_{\ddot{\theta}}[\textsc{loss}(X; \delta)]$ exists. At each time step $1 \leq i \leq n$ we take decision δ^*. Our total loss after n decisions will be

$$\begin{aligned}
\sum_{i=1}^{n} \textsc{loss}(x_i; \delta^*) &= n_0 \textsc{loss}(0; \delta^*) + n_1 \textsc{loss}(1; \delta^*) \\
&= n((1-\ddot{\theta})\textsc{loss}(0; \delta^*) + \ddot{\theta}\textsc{loss}(1; \delta^*)) = n E_{\ddot{\theta}}[\textsc{loss}(X; \delta^*)],
\end{aligned}$$

where the second equality follows from the fact that $\ddot{\theta}$ was defined as $\ddot{\theta} = \hat{\theta}(x^n) = n_1/n$. Hence the loss we expect to make based on our optimized decision will be equal to the loss we will actually make.

This result generalizes to $\mathbf{E} = \{1, \ldots, k\}$ and the class $\mathcal{M}$ of multinomial distributions over $\mathbf{E}$. Why then don't we always use this class in case there are k possible outcomes per trial? The answer is given once again by the MDL principle: $\mathcal{M}$ contains $k-1$ parameters - if k is large, we would need an awful lot of data to obtain accurate estimates for all of these (note that what we call 'reliable' only means that *if* the estimate of θ is close to $\hat{\theta}_{\text{fut}}$, *then* predictions of some functions of the data will be reliable; if we have too many parameters, then we need a lot of data in order for the premise of this implication to be true. It may then often be more reasonable to use a simpler (in the sense of less parameters) model class; the price we pay is that not all our decisions are reliable any more. Obviously, on the basis of a small data sample we cannot reliably predict a complex phenomenon.

Example 4.12 [non-i.i.d. data] We continue the example above. While all functions of *single* outcomes could be reliably estimated based on $\ddot{\theta} = \hat{\theta}(x^n) = \gamma$, we *cannot* reliably estimate properties of the data like 'the frequency of two ones in a row'. We may be tempted to say that this will be $P(X_{i+1} = 1 | X_i = 1, \ddot{\theta}) \cdot P(X_i = 1 | \ddot{\theta}) = \gamma^2$. But this is *un*-reliable. For example, the data may be generated by a Markov Chain P^* with stationary distribution $P(\cdot|\ddot{\theta})$, while $P^*(X_{i+1} = 1 | X_i = 1)$ is highly different from $P^*(X_{i+1} = 1 | X_i = 0)$. In such a case, for large n, with high probability $\hat{\theta}(x^n) \approx 1/3$ while the frequency of two one's in a row is, with high probability, different from 1/9. We saw an example of this already in Chapter 1, Section 1.5.

If we look at the indicator function for two ones in a row: $\phi(x_{i+1}, x_i) = \mathcal{I}(x_{i+1} = x_i = 1)$, we see that it has domain $\mathbf{E}^2$ rather than $\mathbf{E}$. Therefore Proposition 4.7 does not apply and, as expected, we cannot prove that the function can be reliably estimated. One could extend the whole formalism by allowing constraints and functions pertaining to more than one outcome, but we will not pursue that idea further.

The previous examples notwithstanding, it seems clear that the requirement of making only reliable decision and predictions will often restrict us too much. In the next chapter we shall see how to partially get around this problem.

4.3 Safe and Risky Statistics

In this section we explore what happens if, based on an estimate $\ddot{\theta}(x^k)$, one restricts oneself to making only reliable predictions and decisions for future data $x_{k+1}, \ldots, x_n$. This leads to a distinction between 'safe' and 'risky' statistics. In safe statistics, only provably reliable predictions and decisions are allowed on the basis of $\ddot{\theta}(x^n)$). In risky statistics the estimate $\ddot{\theta}(x^n)$ is used as if data were actually generated according to it.

We start by treating the special case of maximum entropy. Restricting this principle so that only reliable inferences can be drawn makes it very weak but it nicely illustrates the general idea. Afterwards, we discuss safe statistics in general.

4.3.1 Safe and Risky Maximum Entropy

Let a constraint $\overline{\phi(x)}^n = t$ be given and let P_{me} be the maximum entropy distribution for this constraint. In almost all applications of maximum entropy, one deals with

Let $\overline{\phi(x)}^n = t$ be a given constraint. Let P_{me} be the maximum entropy distribution for this constraint and let $\mathcal{M}_{me}$ be the maximum entropy model class for function ϕ.
We distinguish between two forms of applying the Maximum Entropy formalism:

Safe MaxEnt We use the MaxEnt distribution P_{me} *only* to make provably reliable decisions and predictions, where 'reliable' is defined with respect to the class $\mathcal{M}_{me}$ (Definition 4.4; Proposition 4.7).

Risky MaxEnt We use the MaxEnt distribution as if data were actually distributed according to it.

Figure 4.1: Safe vs. Risky Maximum Entropy

empirical constraints of this form rather than constraints regarding expected values of unknown probability distributions. We can then view the adoption of P_{me} as a model for data x^n as being equivalent to maximum likelihood estimation of x^n using the maximum entropy model class $\mathcal{M}_{me}$ for ϕ as the model class. Since n will usually be large, P_{me} will also be hardly distinguishable from other reasonable (Definition 4.3) estimators based on $\mathcal{M}_{me}$.

While the examples of the previous section show that 'reliable' decisions do give us the possibility to predict *some* aspects of future data using this P_{me}, we may still want more; sometimes, we can also do more. For example, we may have additional reasons for assuming that all sequences of data satisfying the constraints are about equally likely to occur; in that case, by the concentration phenomenon (Chapter 3, sections 3.5 and 3.6), the frequencies $(\gamma_1, \dots, \gamma_k)$ will with very high probability almost coincide with the maximum entropy probabilities $(P_{me}(1), \dots, P_{me}(k))$. Since the frequencies are the averages of the indicator functions $(\mathcal{I}(x = 1), \dots, \mathcal{I}(x = k))$, in this 'typical' case the average over x^n of every function ψ over $\mathbf{E}$ will be close to its expected value under P_{me} and P_{me} will serve as a good basis for all predictions and/or decisions concerning the data. In cases where we know that the data are subject to additional constraints, but we just do not know which, no data sequences satisfying the known constraint but not the unknown constraints can arise. In that case, the concentration phenomenon is not applicable and we cannot guarantee that $\overline{\psi(x)}^n = E_{me}[\psi(X)]$ except for functions ψ that are linear combinations of the ϕ. If we are interested in estimating the average of a function $\psi(x)$ that cannot be written as a linear combination of the $\phi(x)$, it may still be a reasonable *guess* to predict that $\overline{\psi(x)}^n = E_{me}[\psi(X)]$ – but this prediction has a fundamentally different status from a 'reliable' prediction.

This consideration leads us to identify two forms of maximum entropy: a *safe* and a *risky* form. They are defined in Figure 4.1. Several people, including Jaynes, have hinted at this difference [77, 24] but, as far as we know, nobody has ever formalized it before.

Ex Nihilo Nihil (almost) Restricting oneself to 'safe' maximum entropy takes the sting out of the principle, as we now show. The price to pay is that with 'safe' maximum entropy we can hardly make any non-trivial inferences any more.

Assume that the constraint $\overline{\phi(x)}^k = t$ is given and that we use the maximum entropy distribution P_{me} for this constraint only to make 'safe' estimates regarding future data $x_{k+1}, \ldots, x_n$.

By Definition 4.4, we are then only allowed to estimate the average value of functions ψ that provably satisfy the following property:

$$\textit{if } \frac{1}{k}\sum_{i=k}^{n} \phi(x_i) \approx \frac{1}{n-k}\sum_{i=k+1}^{n} \phi(x_i) \text{ then } E_{me}[\psi(X)] \approx \frac{1}{n-k}\sum_{i=k+1}^{n} \psi(x_i) \tag{4.7}$$

In words, *if* future data (almost) behaves in accordance with the given constraint $\overline{\phi(x)}^k$ over present data, *then* the average of $\psi(x)$ must be (almost) equal to our estimate of it.

The only thing that is needed for the estimates of averages of functions ψ satisfying (4.7) to be accurate is that the average of $\phi(x)$ over future data will be approximately equal to t, its average over present data. This is still an assumption, or perhaps more aptly 'inductive principle' of course. However, for large k and large $n-k$ it is an assumption that is fundamental to make predictions based on i.i.d. model classes work at all: 'the future should behave similarly to the past'. This inductive principle will be acceptable to many statisticians who have difficulties accepting the maximum entropy principle in general. That is the good news. The bad news is that we can hardly make any non-trivial inferences at all: the only functions ψ that can be provably reliably predicted are linear combinations of the ϕ. The only non-obvious linear combination of the ϕ is the codelength function $-\log P(\cdot)$, which allows us to reliably use P_{me} in proportional betting (Section 4.2.3).

4.3.2 Safe and Risky Statistics

The notions of 'safe' and 'risky' can be extended to parametric statistical estimation in general - whether two-part code MDL, Bayesian MAP estimation or any other sufficiently reasonable procedure is used. Let $\mathcal{L}_{\mathcal{M}}$ be some reasonable statistical inference procedure (Definition 4.3), that on input x^n outputs $\ddot{\theta}(x^n)$. Following Figure 4.1, we may call estimates, decisions and predictions based on $\ddot{\theta}(x^n)$ that are provably reliable 'safe statistics'; we may call them 'risky' otherwise. The crucial difference between safe and risky statistics is that the assumptions that have to be made about 'how the world works' are much weaker in the case of safe statistics than in the case of risky statistics.

To get a clearer idea of 'safe statistics', let us look at a function ψ over $\mathbf{E}$ that cannot be reliably estimated. In that case, the following holds: there exist x^n and y^n in $\mathbf{E}^n$ such that

$$\hat{\theta}(x^n) = \hat{\theta}(y^n) \text{ but } \frac{1}{n}\sum_{i=1}^{n} \psi(x_i) \neq \frac{1}{n}\sum_{i=1}^{n} \psi(y_i)$$

Hence, while x^n and y^n share the same optimal model in the class, using the function ψ one is able to 'detect' that x^n and y^n are really different. Therefore, the value $\hat{\theta} = \hat{\theta}(x^n) = \hat{\theta}(y^n)$ simply does not give sufficient information to determine the average of ψ (it is, in other words, not a sufficient statistic for ψ). This means that we can informally rephrase 'safe statistics' as follows:

Safe Statistics Use an estimator $\ddot{\theta} = \ddot{\theta}(x^k)$ for given data x^k only to predict those properties of future data $x_{k+1}, \ldots, x_n$ that are shared by *all* $x_{k+1}, \ldots, x_n$ for which $\ddot{\theta} \approx \hat{\theta}(x_{k+1}, \ldots, x_n)$.

We end this section by analyzing 'safe' statistics from two more points of view:

Safe Statistics and Data Compression As argued in Chapter 1, Section 1.1.3, there can be no practically applicable statistical inference procedure which, for every data sequence D, outputs a model that captures *all* regularities in D (in the sense that it maximally compresses D using a universal computer language as a code). This means that it will always be possible that the model for data D as found by a practical instantiation of two-part MDL (or any other reasonable inference procedure) does not capture *all* (computable) regularity in the data. We already discussed this in Chapter 1, Section 1.5 where we modeled non-i.i.d. data using the Bernoulli model class. We saw that this leads to unreliable predictions as soon as one tries to estimate aspects of the data like the frequency of '11'. 'Safe statistics' allows us to use an overly simple model with less danger. The only thing that is still required from the data generating process is that the estimate $\ddot{\theta}(x^n)$ is close to $\hat{\theta}_{\text{fut}}$, the maximum likelihood model for future data. We emphasize at this point that this is still a requirement that may not hold in practice. Nevertheless, safe statistics can be seen as a safeguard against the conclusion that an estimate captures *all* the regularity in the data: it ensures that we predict only those aspects of the data on the basis of estimate $\ddot{\theta}$ that are shared by *all* x^n for which $\ddot{\theta} = \hat{\theta}(x^n)$, even 'special' x^n that can be compressed further on the basis of another, more complex model.

Classical Statistics is Risky Let ϕ be a given function over sample space $\mathbf{E}$. It is useful to consider the case in which data are generated by repeated sampling of some distribution P^*. We will assume that $E_{P^*}[\phi(X)] = t$ for some t in the range of ϕ. Let $\mathcal{M}$ be a model class that is finitely parameterized by some Γ. Under fairly general conditions (as we shall prove in Chapter 5, Section 5.3), there exists a model $\tilde{\theta} \in \Gamma$ such that

$$\tilde{\theta} = \arg\min_{\theta \in \Gamma} E_{P^*}[-\log P(X|\theta)] \tag{4.8}$$

Hence $\tilde{\theta}$ maximizes the expected log-likelihood. By the law of large numbers, as $n \to \infty$, we obtain $\overline{\phi(x)}^n \to t$ and, if $\mathcal{M}$ is regular enough, also $\hat{\theta}(x^n) \to \tilde{\theta}$ with probability 1 (the latter to be shown in Chapter 5, Lemma 5.14). But it may very well be possible that $E_{\tilde{\theta}}[\phi(x)] \neq E_{P^*}[\phi(x)]$ and hence $E_{\hat{\theta}(x^n)}[\phi(X)] \not\rightarrow \overline{\phi(x)}^n$. In such a case, with high probability we end up with a model that gives a wrong idea of the average of ϕ. There are two ways to avoid this: the first is to adhere to 'safe statistics' and to use as functions ϕ only those functions whose averages we can reliably predict on the basis of our model class $\mathcal{M}$; it is clear that the problem can then not occur. The second way is a kind of cheating: simply declare that the discrepancy mentioned will never happen by making the additional *assumption* that $P^* = P(\cdot|\theta^*)$ for some $\theta^* \in \Gamma$ ('the true model is in the class'). Then by the Information Inequality (Equation 3.4, page 52), $\tilde{\theta} = \theta^*$. In this case, we can apply the law of large numbers to obtain that for all functions ϕ over $\mathbf{E}$, $E_{\hat{\theta}(x^n)}[\phi(X)] \rightarrow \overline{\phi(x)}^n$ with probability 1.

The assumption that data are actually generated by a model $P^* = P(\cdot|\theta^*)$ in the class $\mathcal{M}$ under consideration is fundamental to much of classical statistics (consider, for example, classical hypothesis testing [6]). Unfortunately, it can hardly ever be justified [128]. In our view, it is a form of 'risky' statistics. The alternative assumption that data are generated by i.i.d. sampling from *some* distribution - not necessarily contained in one's model class - is, of course, still an assumption that is difficult to justify, but it is nevertheless much weaker. Whereas in classical statistics, one freely draws conclusions that strictly speaking only follow (with high probability) if one's model class contains the true distribution, in safe statistics one only draws conclusions that already follow (with high probability) under the much weaker assumption that the data is generated by i.i.d. sampling from *some* distribution such that the minimizing $\tilde{\theta}$ in (4.8) exists.

What to use: safe, risky, or in between? We are not saying that one should only use 'safe' statistics. There is nothing wrong in using risky statistics, as long as one is aware that one is then implicitly making some additional assumptions. It can also be the case that one has additional knowledge about the 'data generating machinery' that make some of the 'risky' predictions less risky. For example, if the data consists of repeated trials of some physical experiment (like dice throwing), it is reasonable to assume that the outcomes are truely (almost) independent. Let $\mathcal{M}_{me}$ be once more the maximum entropy class for $\phi(x) = x$ and let P_{me} be the maximum entropy model for $\overline{x}^n = t$. Our knowledge of physics allows us to predict that the event $\{X_i = a, X_{i+1} = b\}$ will occur with probability $P_{me}(X_i = a, X_{i+1} = b) = P_{me}(X_i = a)P_{me}(X_{i+1} = b)$ even though this is, strictly speaking, not a safe inference (see Example 4.12).

4.4 Conclusion and Outlook

We have introduced a distinction between 'reliable' and 'unreliable' inferences, leading to 'safe' and 'risky' statistics. In the next chapter we will put the distinction to use.

Chapter 5

How to Make Predictions Reliable

According to the MDL Principle, models of data are always probabilistic: if a class of non-probabilistic models is used to model the data at hand, it is first mapped to a corresponding probabilistic class (Chapter 2, Section 2.2). On the other hand, as was repeatedly stressed in chapters 1 and 2, these models are to be interpreted as *codes* for the data - not as traditional probability distributions according to which data are generated. This raises the question of what conclusions (predictions) about future data can and what conclusions cannot be drawn on the basis of such 'probabilistic' models. The question becomes all the more difficult if we acknowledge, in line with the MDL philosophy, that our models will always be partially wrong – even if they allow us to substantially compress the data.

Main Concepts and Results of this Chapter In this chapter, we identify conditions under which a probabilistic model, inferred from the data, can be used to reliably predict future data *even if that model is really a probabilistic representation of a non-probabilistic model and/or if the model is wrong*. We show that given a model class $\mathcal{M}$ with a fixed number of parameters ($\mathcal{M}$ is not necessarily probabilistic) and an error function ER, we can turn $\mathcal{M}$ into a probabilistic version $\langle \mathcal{M} \rangle_{\text{ER}}$ that is essentially equivalent to $\mathcal{M}$ except that it leads to 'reliable' estimates of the error function (where 'reliable' is defined as in Definition 4.4 of the previous chapter). We call $\langle \mathcal{M} \rangle_{\text{ER}}$ the *entropification* of $\mathcal{M}$. Entropification stands at the basis of the main results of this chapter (theorems 5.16-5.18) which can be summarized as follows:

1. Under the assumption that the data are i.i.d. according to an essentially arbitrary P^*, we can infer from a large enough data set D, with high probability, a model $\tilde{\theta}$ in $\langle \mathcal{M} \rangle_{\text{ER}}$ that is
   (a) the optimal model in $\langle \mathcal{M} \rangle_{\text{ER}}$ for predicting future data against error function ER: among the models in $\langle \mathcal{M} \rangle_{\text{ER}}$, $\tilde{\theta}$ minimizes the 'true' expected error $E_{P^*}[\text{ER}(Y|\tilde{\theta}, X)]$.

(b) can be 'reliably' used, since it gives a truthful impression of its own performance in the sense that $E_{\tilde{\theta}}[\text{ER}(Y|\tilde{\theta}, X)] = E_{P^*}[\text{ER}(Y|\tilde{\theta}, X)]$).

Essentially, this means that whenever the assumption that data are i.i.d. can be justified and the function ER according to which errors will be measured is known, the model $\tilde{\theta}$ can be used (1) to arrive at optimal predictions (relative to the model class $\langle \mathcal{M} \rangle_{\text{ER}}$) of future data against ER and (2) as an accurate estimator of how good these predictions will be - even if $\mathcal{M}$ is a wrong ('misspecified') model class that does not contain any model that is similar to the 'true' P^*.

While $\tilde{\theta}$ can be inferred from data by many statistical inference procedures (not necessarily MDL), the 'entropification' of $\mathcal{M}$ turns out to yield additional results when combined with MDL, leading to the other two important results of this chapter:

2. Entropification removes an inherent arbitrariness in MDL's trade-off between error and model complexity that occurs if non-probabilistic model classes are used (Section 5.4)

3. Entropification allows us to associate codes with non-probabilistic model classes in an optimal manner, in the sense that the 'worst-case expected codelength' is minimized (Proposition 5.22).

Preliminary Status The material presented in this chapter has a very preliminary character; it could certainly be rephrased in a simpler, clearer and more compact fashion. Nevertheless, we decided to include it in this thesis, since it does serve to justify our view of probability distributions as codes, which, on one hand, cannot be used to predict just any aspect of the data, but which, on the other hand, can be used to reasonably predict some aspects of the data even if they are very different from the 'true' distribution generating the data.

Organization of the Chapter We will start (Section 5.1) with a motivating example. In Section 5.2 we formally introduce the concept of 'entropification', present some of its basic properties and give some examples of its use. Section 5.3 presents our main results (item 1 in the listing above). In Section 5.4 we show how entropification can be used in the context of MDL.

Attached to the chapter is an Epilogue to Part I of this thesis in which we argue that 'entropification' resolves the problematic issues concerning MDL that were mentioned at various places in Chapters 1–3.

5.1 An Introductory Example

In the next section we will formally introduce the concept of 'entropification'. In this section we present an introductory example.

Example 5.1 Consider once more the fitting of polynomials. Let data $D = (x^n, y^n)$ be given. In a non-probabilistic approach to this problem, we would use some algorithm that, for each D, when input D, outputs a polynomial $\ddot{H}$ that it regards as an optimal

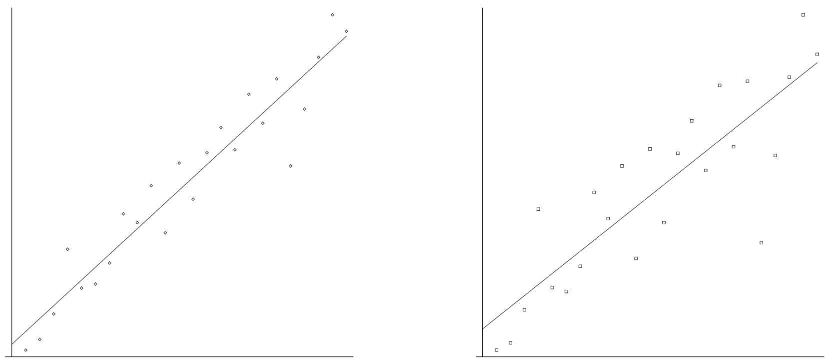


Figure 5.1: Two sets of points with the same least-squares first-degree polynomial. The distances between $f(x_i)$ and y_i in the picture on the right are exactly twice as large as those in the picture on the left.

hypothesis for D. Such a polynomial $\ddot{H}$ in itself does not give any information on how good it will be on future data, and this can be problematic. For example, imagine that a company uses some sophisticated tool to infer $\ddot{H}$ from lots of data, and then sells $\ddot{H}$ to a client so that the client can use it to predict future data ($\ddot{H}$ may, for example, be a model for some data from the stock exchange and the client may use it as a guideline for future investments). If the company only gives $\ddot{H}$ to the client, then the client has no means of knowing how well $\ddot{H}$ actually *will* predict future data. This can be most easily demonstrated if we imagine that the model class $\mathcal{M}$ is restricted to the class of first-degree polynomials. In Figure 5.1 we show two sets of points that have the same optimal (in the sense of minimizing squared error) first degree polynomial $\widehat{H}$. Let us denote by D_1 the set of points depicted on the left of Figure 5.1 and by D_2 the set of points depicted on the right. Assuming that the company uses a reasonable (see Chapter 4, Definition 4.3) method to infer the best polynomial, it will infer a polynomial reasonably close to $\widehat{H}$ for both data sets depicted. However, if future data behaves like present data, then in the case of D_1, $\widehat{H}$ will be a much better predictor than in the case of D_2. The client (who has not seen the 'training' data) would probably like to know how good the hypothesis $\ddot{H} \approx \widehat{H}$ is before he decides whether to buy it or not; but $\ddot{H}$ does not reveal this information. Therefore, the client may rather want the company to sell a tuple $(\ddot{H}, \ddot{\sigma}^2)$ where $\ddot{\sigma}^2$ is some reasonable estimate of the error $\ddot{H}$ will make on future data. In this way, he will get a *reliable* impression of the performance of $\ddot{H}$.

When using two-part code MDL, we do not estimate a polynomial for data D directly. Rather, we turn each polynomial $H \in \mathcal{M}$ into a probability distribution $P(\cdot|H)$ and we look for the polynomial H_{mdl} minimizing $-\log P(D|H) + L(H)$. Specifically, in

Chapter 1, Section 1.4 and Chapter 2, Section 2.2 we showed how to change $\mathcal{M}$ to a corresponding class of (conditional) probability distributions $\mathcal{M}_{pr}$ such that each polynomial H is mapped to a corresponding $P(\cdot|H,\cdot)$ with for all $D = (x^n, y^n)$:

$$-\log P(y^n|H, x^n) = \text{ER}_{sq}(y^n|H, x^n) + K \tag{5.1}$$

We showed that the $P(\cdot|H,\cdot)$ achieving this for us is a product distribution of Gaussians with density given by:

$$\begin{aligned} P(y^n|H, x^n) &= \prod_{i=1}^{n} P(y_i|H, x_i) \text{ where} \\ P(y|H, x) &= \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y - H(x))^2}{2\sigma^2}). \end{aligned} \tag{5.2}$$

with the variance σ^2 given by $\sigma^2 = (2\ln 2)^{-1}$. The advantage of turning each H into a probability distribution is that we can express the error H makes on D in terms of bits and can therefore add to it a complexity term which is also expressed in bits.

Now, as we saw above, the optimal *polynomial* H_{mdl} for data D tells us nothing on how good it will be in predicting future data. However, the probabilistic version $P(\cdot|H_{\text{mdl}},\cdot)$ does make such a statement, for it defines an expected error over future data $\text{ER}_1 = E_{P(\cdot|H_{\text{mdl}})}[\text{ER}_{sq}(Y|H_{\text{mdl}}, X)]$. We may be tempted to use ER_1 as an estimate of the error we will make when predicting future data. But this cannot work, since we have for all $H \in \mathcal{M}$:

$$E_{P(\cdot|H,\cdot)}[\text{ER}_{sq}(Y|H, X)] = \sigma^2 \tag{5.3}$$

(this is a standard fact about regression; see [6]). Returning to Figure 5.1, if we are asked to use $P(\cdot|H_{\text{mdl}},\cdot)$ to make an estimate of the average squared error we will make on future data, we would answer $\sigma^2 = (2\ln 2)^{-1}$ both in the case where H_{mdl} is based on D_1 and in the case where H_{mdl} is based on D_2. Clearly, the answer will necessarily be false for at least one of the two sets D_1 or D_2 (and in general, for almost all D for which H_{mdl} is the optimal two-part code model).

We could solve this problem by *never* using the optimal polynomial H_{mdl} in its probabilistic form $P(\cdot|H_{\text{mdl}},\cdot)$ and instead supply it, just as the non-MDL estimate $\ddot{H}$ we referred to above, by a separately obtained estimate $\ddot{\sigma}^2$ of the error that H_{mdl} will make on future data. But this is not in line with the general MDL Principle (Chapter 2, Section 2.1), which says that we should hunt for the model that gives optimal compression of our data, and hence best captures the regular features of the data. The reason is that the estimate $\ddot{\sigma}^2$ can be used to further compress the data: it gives additional information about the data over and above the information contained in H_{mdl}. Therefore, we should rather try to find a single model for the data which combines the regularities expressed by H_{mdl} and those expressed by $\ddot{\sigma}^2$. This is the first central idea of this chapter. It can be achieved as follows: instead of mapping each $H \in \mathcal{M}$ to a *single* probability distribution $P(\cdot|H,\cdot)$ with a fixed value of σ^2, we will map it to a whole class of distributions $P(\cdot|H,\sigma^2,\cdot)$ (one for each $\sigma^2 > 0$) where $P(\cdot|H,\sigma^2,\cdot)$ is the conditional normal distribution given by (5.2) with variance

σ^2. The resulting class of models contains, for each $H \in \mathcal{M}$, the probabilistic models $P(\cdot|H, \sigma^2, \cdot)$ for all $\sigma^2 > 0$. By (5.3), $E_{P(\cdot|H,\sigma^2,\cdot)}[\text{ER}_{sq}(Y|H, X)] = \sigma^2$. Hence, the models $P(\cdot|H, \sigma^2, \cdot)$ restricted to fixed H all correspond to a different expected squared error of H. We can now use two-part code MDL (but, if we like, also another method) to find the optimal probabilistic model $(H_{\text{mdl}}, \sigma^2_{\text{mdl}})$ for the data. This probabilistic model *does* give a reliable idea of its own performance over future data. Formally, let $H \in \mathcal{M}$ be arbitrary and let $\hat{\sigma}^2_H$ stand for the maximum likelihood estimator of D for this H. That is, among all models (H, σ^2) with this fixed H it maximizes the likelihood of D. The following is a standard result about regression (also easy to verify): for all n, all $D = (x^n, y^n)$ and for every $x \in \mathbf{E}_x$,

$$E_{(H,\hat{\sigma}^2_H)}[\text{ER}_{sq}(Y|H, X)|X = x] = \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} \text{ER}_{sq}(y_i|H, x_i)$$

In words, for every $H \in \mathcal{M}$, according to the maximum likelihood estimate $\hat{\sigma}^2_H$, the expected squared error of predicting Y given that $X = x$ is independent of x and equal to the average squared error over the training data. This implies that for each fixed H, the average error of H can be *reliably* estimated. Here 'reliable' is defined as in Definition 4.4 of Chapter 4. The advantage of reliable estimation has been amply discussed in the previous chapter (see page 73). Roughly, it implies that *if* future data is similar to the training data, *then* the expected value of the error according to an estimate based on the training data will be close to the actual average error obtained when this estimate is used to predict future data. In our example, it implies that, if $(\ddot{H}, \ddot{\sigma}^2)$ is estimated from data D_1 by a reasonable (Chapter 4, Definition 4.3) method, then $\ddot{\sigma}^2$ will be approximately equal to the average error of $\ddot{H}$ over D_1; if it is estimated from D_2, then $\ddot{\sigma}^2$ will be approximately equal to the average error of $\ddot{H}$ over D_2.

One can use the reliability of error estimates to prove (as we will in Section 5.3) that if the data are *independently* generated according to an essentially arbitrary distribution P^*, then every reasonable inference procedure will, for large enough D, output a model $P(\cdot|\ddot{H}, \ddot{\sigma}^2, \cdot)$ such that the following holds: let, among all $H \in \mathcal{M}$, $\widetilde{H}$ be the model that minimizes the 'true' expected error $E_{P^*}[\text{ER}_{sq}(Y|H, X)]$ and let $\tilde{\sigma}^2 = E_{P^*}[\text{ER}_{sq}(Y|\widetilde{H}, X)]$ be the 'true' expected error of this optimal $\widetilde{H}$. Then, with probability 1, (1) $\ddot{H}$ is arbitrarily close to the optimal model $\widetilde{H}$ and (2) $\ddot{\sigma}^2$ itself is arbitrarily close to $\tilde{\sigma}^2$.

What is new?

It is well-known that modeling errors using the normal distribution with varying σ^2 works even when the errors are not truely normally distributed [20], so this far, there is nothing really new here. Our own contribution lies in the fact that we consider the general case of (almost) *arbitrary* error functions ER and model classes $\mathcal{M}$. In the next section, we will give a recipe of how, given $\mathcal{M}$ and ER, one can define a new probabilistic model class $\langle \mathcal{M} \rangle_{\text{ER}}$ that has some special properties. We call the model class $\langle \mathcal{M} \rangle_{\text{ER}}$ the *entropification* of $\mathcal{M}$ with respect to ER. $\langle \mathcal{M} \rangle_{\text{ER}}$ is constructed from $\mathcal{M}$ by adding a single extra real-valued parameter β as part of the hypotheses: models in $\langle \mathcal{M} \rangle_{\text{ER}}$ are indexed by parameters $\theta = (H, \beta)$ for some $H \in \mathcal{M}$ and $\beta \in \mathbf{R}$. If

(H, β) is inferred from data D, then the β associated with H can be interpreted as a reliable estimate of the error H will make on future data. β will also determine the entropy of the model (H, β), hence the name 'entropification'. We give three examples corresponding to three often used error functions.

If $\mathcal{M}$ is a class of continuous functions and ER is the squared error, then $\langle \mathcal{M} \rangle_{\text{ER}}$ turns out to be equivalent to the class $\{P(\cdot|H, \sigma^2, \cdot) \mid H \in \mathcal{M}; \sigma^2 > 0\}$ where $P(\cdot|H, \sigma^2, \cdot)$ is as given by (5.2). This coincides with the case considered above: if (H, σ^2) is inferred from D, then σ^2 can be interpreted as an estimate of the squared error H will make on future data.

If $\mathcal{M}$ is a class of concepts (functions from $\mathbf{E}_x$ to $\mathbf{E}_y = \{0, 1\}$) and ER is the 0/1-error (see Chapter 2, Example 2.4), then, as we will see in Example 5.3, $\langle \mathcal{M} \rangle_{\text{ER}}$ is equivalent to a class of distributions $\{P(\cdot|H, \theta, \cdot) \mid H \in \mathcal{M}; 0 < \theta < 1\}$ where

$$E_{P(\cdot|H,\theta,\cdot)}[\text{ER}(Y|H, X)] = \theta$$

θ can be interpreted as the probability that $H(X) \neq Y$. If (H, θ) is inferred from D, then θ can be interpreted as an estimate of the 0/1-error that H will make on future data.

If $\mathcal{M}$ is a class of probabilistic models $\{P(\cdot|\eta) \mid \eta \in \Gamma\}$, and ER is the logarithmic error (Chapter 2, Section 2.7), then (roughly) $\langle \mathcal{M} \rangle_{\text{ER}}$ will turn out to be equivalent to a class $\{P(\cdot|\eta, \beta) \mid \eta \in \Gamma; \beta \in \mathbf{R}\}$; an explicit formula for the distributions $P(\cdot|\eta, \beta)$ will be given later (Example 5.4). If (η, β) is inferred from D, then β can be interpreted as an estimate of the logarithmic error that η will make on future data.

Other Error Functions The possibility of making estimates of errors ER reliable by using the model class $\langle \mathcal{M} \rangle_{\text{ER}}$ raises the following interesting question: what happens if we use a model (H, β) in the class $\langle \mathcal{M} \rangle_{\text{ER}}$ to predict against other error functions $\text{ER}' \neq \text{ER}$? The models (H, β) we infer from data D are still, in the first place, to be interpreted as codes rather than classical probability distributions. Therefore, when $\text{ER}' \neq \text{ER}$, it is not a priori clear whether it makes any sense to use $E_{P(\cdot|H,\beta,\cdot)}[\text{ER}'(Y|H, X)]$ as an estimate of the average error H will make over future data. It turns out that such an estimate (of ER') can still be seen as a 'reasonable guess', but it has a status fundamentally different (weaker) from the estimate of ER. This will be seen in the Epilogue to this chapter (page 119).

The Goals of Entropification: reliable estimates of errors + applications in MDL Summarizing, 'entropification' will serve as a generic means to make estimates of errors reliable. This fact will be used in Section 5.3 to prove our main results, which we already summarized at the beginning of this chapter (items 1(a) and 1(b), page 82). It turns out that, in connection with MDL, entropification serves several other purposes besides making estimates of errors reliable; these were also already summarized on page 82. They will be treated in Section 5.4.

Now that we have given a first idea of 'entropification' and have mentioned its main goals, we proceed to define the concept formally.

5.2 Entropification of a Model Class

In this section we formally introduce the concept of 'entropification' and give some examples of its use. This will be done in (sub-) sections 5.2.1 and 5.2.2. Section 5.2.3 presents some of the basic properties of entropification and Section 5.2.4 summarizes the essentials of these properties.

5.2.1 Preliminaries

We will assume throughout the remainder of this chapter that all error functions ER considered are sufficiently regular. More precisely, let $\mathbf{E} = \mathbf{E}_x \times \mathbf{E}_y$, and let $\text{ER} : \mathbf{E} \times \mathcal{M} \rightarrow \mathbf{R}$ be an error function defined as in Chapter 2 on page 31. Our assumption is that, for all fixed $H \in \mathcal{M}$, $\text{ER}(y|H,x)$ considered as a function of x and y can be used to define a maximum entropy model class. Specifically, we assume that for all $H \in \mathcal{M}$, the function ϕ_H over $\mathbf{E}_x \times \mathbf{E}_y$ defined by $\phi_H(x,y) := \text{ER}(y|H,x)$ satisfies conditions **C1**-**C3** of page 54. This ensures that the maximum entropy model class for $\phi_H(x,y)$ exists. But we need something stronger than merely the guaranteed existence of this class, as we will now explain.

Conditional and Unconditional Case Throughout this chapter, we consider two cases. In the first case, the hypothesis class $\mathcal{M}$ contains models relating to $\mathbf{E}_y$ and not $\mathbf{E}_x$ (for example, each $H \in \mathcal{M}$ is itself a probabilistic model over $\mathbf{E}_y$ or each H is a relation over $\mathbf{E}_y$ not involving $\mathbf{E}_x$). In such a case $\mathbf{E}_x$ does not really play any role and we could have equally well set $\mathbf{E} = \mathbf{E}_y$. In this situation, the entropification of a model class $\mathcal{M}$ with respect to error function $\text{ER} : \mathbf{E} \times \mathcal{M} \rightarrow \mathbf{R}$ is the class of probabilistic models containing, for each $H \in \mathcal{M}$, the class of distributions $P(\cdot|H,\beta)$ defined by

$$P(y|H,\beta) = \frac{1}{Z_H(\beta)} \exp(-\beta \text{ER}(y|H)) \tag{5.4}$$

where $Z_H(\beta)$ is a normalizing factor:

$$Z_H(\beta) := \sum_{y \in \mathbf{E}_y} \exp(-\beta \text{ER}(y|H)) \tag{5.5}$$

and β ranges over all $\beta \in \mathbf{R}$ for which $P(y|H,\beta)$ is well-defined. As can be immediately seen from equations 3.7 and 3.8 (Chapter 3, page 53), the distributions (5.4) are formally equivalent to maximum entropy distributions.

The formal definition of entropification we give below unifies the unconditional case with the more complicated *conditional* or *supervised* case. In this case, we assume that the outcomes $x \in \mathbf{E}_x$ do play a role, and we are interested in the conditional version of (5.4), $P(y|H,\beta,x) = Z_{H,x}^{-1}(\beta) \exp(-\beta \text{ER}(y|H,x))$ with $Z_{H,x}(\beta) := \sum_{y \in \mathbf{E}_y} \exp(-\beta \text{ER}(y|H,x))$. However, all our results will only hold if the resulting distributions are still 'essentially' maximum entropy distributions. For this reason, we must additionally assume that the following conditions hold:

C4 $\mathbf{E}_x$ is either finite or compact.

C5 ER is such that for each fixed H and each fixed $\beta \in \mathbf{R}$, $Z_{H,x}(\beta)$ is either equal for all $x \in \mathbf{E}_x$ or it diverges for all $x \in \mathbf{E}_x$.

C5 turns out to hold for most error functions ER of interest. These include (as will be shown formally in Proposition 5.11) error functions like the squared and 0/1-error. **C5** allows us to drop the subscript x in $Z_{H,x}(\beta)$ and write, for arbitrary $x \in \mathbf{E}_x$:

$$Z_H(\beta) = \sum_{y \in \mathbf{E}_y} \exp(-\beta \mathrm{ER}(y|H,x)). \tag{5.6}$$

In the remainder of this chapter, we tacitly assume ER to satisfy conditions **C1**-**C3** of page 54 and conditions **C4** and **C5** listed here.

5.2.2 Formal Definition of Entropification

We are now ready to define entropification formally.

Definition 5.2 (Entropification) *Let $\mathbf{E} = \mathbf{E}_x \times \mathbf{E}_y$. The entropification of a model class $\mathcal{M}$ with respect to error function* ER : $\mathbf{E} \times \mathcal{M} \rightarrow \mathbf{R}$ *is the class of (conditional) probabilistic models*

$$\langle \mathcal{M} \rangle_{\mathrm{ER}} = \{P(\cdot|\theta, \cdot) \mid \theta = (H, \beta); H \in \mathcal{M}; \beta \in \Gamma_{nat}(H)\} \tag{5.7}$$

Here $P(\cdot|\theta,\cdot) = P(\cdot|H,\beta,\cdot)$ is a conditional probabilistic model defined as follows:

1. *For each $x \in \mathbf{E}_x$, $P(\cdot|\theta, x) = P(\cdot|H,\beta,x)$ is a probability distribution over $\mathbf{E}_y$ defined by*

$$P(y|H,\beta,x) = \frac{1}{Z_H(\beta)} \exp(-\beta \mathrm{ER}(y|H,x)) \text{ for all } y \in \mathbf{E}_y. \tag{5.8}$$

   Here $Z_H(\beta)$ is as in (5.6).

2. *For all $(x^n, y^n) \in \mathbf{E}^n$, $P(y^n|H,\beta,x^n) = \prod_{i=1}^n P(y_i|H,\beta,x_i)$.*

For each H, the set of β such that $P(\cdot|H,\beta,\cdot) \in \langle \mathcal{M} \rangle_{\mathrm{ER}}$ is given by

$$\Gamma_{nat}(H) = \{\beta \mid Z_H(\beta) < \infty\}.$$

If $\mathbf{E}$ is continuous, the sum in $Z_H(\beta)$ gets replaced by the corresponding integral. $Z_H(\beta)$ acts as normalizing constant.

Codelengths based on models in $\langle \mathcal{M} \rangle$; first interpretation of β The code corresponding to $P(\cdot|H,\beta,.)$ leads to the following code lengths (expressed in nats):

$$L(y^n|H,\beta,x^n) = -\ln P(y^n|H,\beta,x^n) = \beta \sum_{i=1}^n \mathrm{ER}(y^n|H,x^n) + n \ln Z_H(\beta) \tag{5.9}$$

We see that the code length of y^n given H, β, x^n contains an error term and a 'uniform' term $n \ln Z_H(\beta)$ that grows linearly in n and is equal for all y^n. This shows that β

can be interpreted as determining how strongly the error should be weighed in the code length corresponding to hypothesis (H, β). The extreme case $\beta = 0$ corresponds, for each x_i, to the uniform distribution over all outcomes in $\mathbf{E}_y$. For fixed H, in the limit for $\beta \to \infty$, the probability under (H, β) of an outcome y^n given x^n with $\mathrm{ER}(y^n|H, x^n) > 0$ becomes 0.

Obtaining β; second interpretation of β We can estimate β by any statistical means, for example, by two-part code MDL (Chapter 1). In that case, we would pick as optimal hypothesis for given data D the $\theta_{\mathrm{mdl}} = (H_{\mathrm{mdl}}, \beta_{\mathrm{mdl}})$ that minimizes the sum of (5.9) and $L_C(H_{\mathrm{mdl}}, \beta_{\mathrm{mdl}})$, the description length of the (extended) hypothesis. In general, if the tuple (H, β) is estimated from data D, then β can be interpreted as (a monotonic transform of) an estimate of the error H will make on future data, as will be made clear in the examples below. This gives a second interpretation of β; we will encounter even a third. These interpretations will be summarized in Section 5.2.4.

Example 5.1, continued Let $\mathcal{M}$ be a class of continuous functions $H : \mathbf{E}_x \to \mathbf{E}_y$. From the definition of entropification (Definition 5.2) we can see (by substituting $\beta = (1/2\sigma^2)$) that $\langle \mathcal{M} \rangle_{\mathrm{ER}_{sq}}$, the entropification of $\mathcal{M}$ with respect to the squared error, is equivalent to the model class that supplies $\mathcal{M}$ with a normal error distribution of arbitrary variance $\sigma^2 > 0$. Formally,

$$\langle \mathcal{M} \rangle_{\mathrm{ER}_{sq}} = \{P(\cdot|H, \sigma^2, \cdot) \mid H \in \mathcal{M}; \sigma^2 > 0\} \tag{5.10}$$

with $P(\cdot|H, \sigma^2, \cdot)$ as given by (5.2) (page 84). In this case, the value of $Z_H(\beta)$ is independent of H and finite for all $\beta > 0$. The parameter space will be $\Gamma_{nat}(H) = \{\beta|\beta > 0\}$ independently of H, corresponding to all variances $\sigma^2 = 1/(2\beta) > 0$. Note that, by Equation 5.3, $E_{P(\cdot|H,\sigma^2,\cdot)}[\mathrm{ER}_{sq}(Y|H, X)] = \sigma^2$. Hence when the tuple (H, β) is inferred from data, then β, which determines σ^2, can be seen as an estimate of the expected squared error of H.

Example 5.3 [concept learning and Bernoulli parameters] Let $\mathcal{M}$ be a class of concepts over $\mathbf{E} = \mathbf{E}_x \times \{0, 1\}$ and let ER_{01} be the 0/1-error function (Chapter 2, Example 2.4). Let the observational data $D = (x^n, y^n)$. Let $\langle \mathcal{M} \rangle_{\mathrm{ER}_{01}}$ be the entropification of $\mathcal{M}$ with respect to error function ER_{01}. Let $\langle H \rangle_{\mathrm{ER}_{01}} = \{P(\cdot|H, \beta) \mid \beta \in \Gamma_{nat}(H)\}$ be the restriction of $\langle \mathcal{M} \rangle_{\mathrm{ER}_{01}}$ to models with fixed $H \in \mathcal{M}$. Substituting $\beta := \ln(1-\theta) - \ln \theta$ in Definition 5.2, we find that $\langle H \rangle_{\mathrm{ER}_{01}}$ is just the class of Bernoulli models containing one model for each possible probability of error (see also Chapter 3, Example 1.1):

$$E_\beta[\mathrm{ER}_{01}(Y|H, X)] = P_\beta\{\mathrm{ER}_{01}(Y|H, X) = 1\} = P_\beta\{H(X) \neq Y\} = \\ = \frac{1}{Z(\beta)} e^{-\beta \cdot 1} = \theta \tag{5.11}$$

and

$$E_\beta[|1 - \mathrm{ER}_{01}(Y|H, X)|] = P_\beta\{\mathrm{ER}_{01}(Y|H, X) = 0\} = P_\beta\{H(X) = Y\} = \\ = \frac{1}{Z(\beta)} e^{-\beta \cdot 0} = 1 - \theta \tag{5.12}$$

It follows that the class $\langle \mathcal{M} \rangle_{\text{ER}_{01}}$ can equivalently be parameterized as $\langle \mathcal{M} \rangle_{\text{ER}_{01}} = \{P(\cdot|H,\theta,\cdot) \mid H \in \mathcal{M}, 0 < \theta < 1\}$, such that, if $\text{ER}_{01}(y^n|H,x^n) = k$, then

$$P(y^n|H,\theta,x^n) = \theta^k(1-\theta)^{n-k} \tag{5.13}$$

This expresses that the probability of error of H is equal to θ for each observation, independently of any other observations. (5.11) shows that, if (H,β) is inferred from data D, then β (which determines θ) can be interpreted as an estimate of the expected 0/1-error of H, which is just the probability that H misclassifies D. Just as in the previous example, β serves to estimate the expected (in this case, 0/1-) error.

MDL is usually applied to concept classes in a way that does not involve entropification [122, 84]. In Section 5.4, Example 5.21 we show that the 'traditional' way of applying MDL to a concept class $\mathcal{M}$ is essentially equivalent to applying MDL to the probabilistic class $\langle \mathcal{M} \rangle_{\text{ER}}$, thus reconciling the two views.

Example 5.4 [Entropification of probabilistic model classes] What happens if we try to entropify a probabilistic model class $\mathcal{M}$? For simplicity, we only consider the case where $\mathcal{M} = \{P(\cdot|\eta) \mid \eta \in \Gamma_{\mathcal{M}}\}$ is a class of i.i.d. probabilistic models over $\mathbf{E}_y$. Similarly, we consider only error functions $\text{ER} : \mathbf{E}_y \times \mathcal{M} \to \mathbf{R}$. The values of x_i are therefore irrelevant and $\mathcal{M}$ consists of full rather than conditional probability distributions. Definition 5.2 is seen to simplify in this case to

$$\begin{aligned} \langle \mathcal{M} \rangle_{\text{ER}} &= \{P(\cdot|(\eta,\beta)); \eta \in \Gamma_{\mathcal{M}}; \beta \in \Gamma_{nat}(\eta)\} \text{ where} \\ P(y|(\eta,\beta)) &= \frac{1}{Z_\eta(\beta)} \exp(-\beta \text{ER}(y|\eta)), \\ P(y^n|(\eta,\beta)) &= \prod_{i=1}^{n} P(y_i|(\eta,\beta)) \end{aligned} \tag{5.14}$$

A natural error function for probabilistic models is the logarithmic error $\text{ER}_{lg}(y^n|\eta) = -\sum_{i=1}^{n} \ln P(y_i|\eta)$; see Chapter 2, Section 2.7. Using the logarithmic error, we obtain:

$$P(y|\eta,\beta) = \frac{1}{Z_\eta(\beta)} \exp(\beta \ln P(y|\eta)) = \frac{P(y|\eta)^\beta}{\sum_{y \in \mathbf{E}_y} P(y|\eta)^\beta} \tag{5.15}$$

We now consider two cases:

$\mathcal{M}$ is an exponential family If $\mathcal{M}$ is a full exponential family (Chapter 3, Section 3.4) then $\langle \mathcal{M} \rangle_{\text{ER}_{lg}} = \mathcal{M}$ as can easily be seen from substituting Equation 3.11 on page 56 into (5.15). If $\mathcal{M}$ is an exponential family that is not full and that contains a model for some $\beta \neq 0$, then entropification serves to make it full (to see this, check Equation 3.11 once more).

We see that full exponential families, and hence 'full' maximum entropy model classes, are closed under entropification.

$\mathcal{M}$ is not an exponential family This case is more interesting. Many useful probabilistic model classes are not of the exponential form; as a simple example, consider hidden Markov Models [123, 125]. For such model classes, entropification can nevertheless be useful, for two reasons: (1) it leads to reliable estimates of the logarithmic error in the sense of Definition 4.4 (this fact will be put to use in Section 5.3.4); and (2), using $\langle \mathcal{M} \rangle_{\text{ER}_{lg}}$ instead of $\mathcal{M}$ can often lead to additional compression of the data when data is encoded using the MDL 2-part code (as will be discussed in Section 5.4.2).

5.2.3 Properties of Entropification

In this subsection we present some useful properties of entropified model classes $\langle \mathcal{M} \rangle_{\text{ER}}$. These properties will be used in the proofs of our main results in Section 5.3.

The key to proving all the properties is that for each fixed $H \in \mathcal{M}$, the subclass of models $\langle H \rangle_{\text{ER}}$ containing (H, β) for all $\beta \in \Gamma_{nat}(H)$ (that is, $\langle H \rangle_{\text{ER}} := \{(H, \beta) \mid (H, \beta) \in \langle \mathcal{M} \rangle_{\text{ER}}\}$) is essentially (though not strictly) a maximum entropy model class (compare Definition 5.2 with Definition 3.7 on page 55). The reason that the correspondence is not strict is that $\langle H \rangle_{\text{ER}}$ is a class of *conditional* models. This leads to some technical, but not essential complications in proving the properties. In order to focus on what is really essential, we moved some of the proofs of these properties to an appendix to this chapter (Appendix 5.6, page 111).

The Properties

Throughout the remainder of this subsection, we assume that we are given a sample space $\mathbf{E} = \mathbf{E}_x \times \mathbf{E}_y$, a model class $\mathcal{M}$ and an error function ER.

Briefly, we will show that (1) even though the distributions indexed by (H, β) are conditional, one can define entropy and expectation of error with respect to these distributions; (2) entropification leads to 'reliable' estimates of error (in the sense of Definition 4.4); (3) for fixed H, the models (H, β) are all, in an important sense, equivalent, but they differ in that they all have different entropy; and (4) there exists a particularly well-behaved class of error functions which we will call 'simple'. Below we show all properties in detail. In stating them, we need to use the maximum likelihood estimator for *fixed hypothesis* H and similarly, for *fixed* β, which we now define:

Definition 5.5 *Let $D = (x^n, y^n) \in \mathbf{E}^n$. The maximum likelihood estimator of D for* fixed H *with respect to $\langle \mathcal{M} \rangle_{\text{ER}}$, denoted by $\hat{\beta}(D|H)$, is (if it exists) given by*

$$\hat{\beta}(D|H) = \arg\max_{\beta \in \Gamma_{nat}(H)} \{P(y^n | H, \beta, x^n)\} \tag{5.16}$$

The maximum likelihood estimator of D for fixed β *with respect to $\langle \mathcal{M} \rangle_{\text{ER}}$, denoted by $\widehat{H}(D|\beta)$, is (if it exists) given by*

$$\widehat{H}(D|\beta) = \arg\max_{H \in \mathcal{M}} \{P(y^n | H, \beta, x^n)\} \tag{5.17}$$

Notational Conventions

The first property we show allows us to simplify notation. Recall that we assume throughout this chapter that $Z_H(\beta)$ does not depend on x (page 88). As will be shown in Appendix 5.6 this implies that for fixed H and β, the expectation of the error under (H, β) is independent of the given x. The same holds for the entropy. Formally, we let $E_{(H,\beta)}$ denote expectation under the model $P(\cdot|H, \beta, \cdot) \in \langle\mathcal{M}\rangle_{\text{ER}}$. Then for all $(H, \beta) \in \langle\mathcal{M}\rangle_{\text{ER}}$ and all $x_1, x_2 \in \mathbf{E}_x$, we have

$$E_{(H,\beta)}[\text{ER}(Y|H, X)|X = x_1] = E_{(H,\beta)}[\text{ER}(Y|H, X)|X = x_2] \tag{5.18}$$

Also, for all $x_1, x_2 \in \mathbf{E}_x$, the entropy $\mathcal{H}(P(\cdot|H, \beta, \cdot))$ satisfies:

$$\mathcal{H}(P(\cdot|H, \beta, x_1)) = \mathcal{H}(P(\cdot|H, \beta, x_2)) \tag{5.19}$$

(5.18) and (5.19) will be proven formally in Appendix 5.6. They imply that the expectation $E_{(H,\beta)}[\text{ER}(Y|H, X)]$ over the conditional model (H, β) supplied with an arbitrary distribution P_x over $\mathbf{E}_x$ does not depend on P_x. This allows us to write $E_{(H,\beta)}[\text{ER}(Y|H, X)]$ instead of $E_{(H,\beta)}[\text{ER}(Y|H, X)|X = x]$. Similarly, we will write $\mathcal{H}(H, \beta)$ instead of $\mathcal{H}(P(\cdot|H, \beta, x))$.

Entropification and Reliable Estimates of Errors

We proceed to show that entropification leads to reliable estimates of error. This will be the key to proving the theorems on entropification which we prove in the next section. In Example 5.1 we discussed why 'reliability' is a desirable property.

Proposition 5.6 [reliability] *Let $D = (x^n, y^n)$. For each $H \in \mathcal{M}$, $E_{(H,\beta)}[\text{ER}(Y|H, X)]$ as a function of β is continuous. Moreover,*

$$E_{(H,\hat{\beta}(D|H))}[\text{ER}(Y|H, X)] = \frac{1}{n}\sum_{i=1}^{n} \text{ER}(y_i|H, x_i)$$

This proposition will be proven in Appendix 5.6. It shows that, for each model $(H, \hat{\beta}(D|H))$, its expected error over future data is equal to its average error over the given data. By Definition 4.4 (Chapter 4), this implies that for each $H \in \mathcal{M}$, the average error $\text{ER}(y|H, x)$ can be reliably estimated on the basis of the restriction of the class $\langle\mathcal{M}\rangle_{\text{ER}}$ to models containing this specific H.

The Name 'Entropification'

We now state two properties that explain why we have chosen the name 'entropification': let $H \in \mathcal{M}$ be arbitrary but fixed. The models (H, β) in $\langle\mathcal{M}\rangle$ are, for all $\beta \in \Gamma_{nat}(H)$ except $\beta = 0$, partially equivalent to H as stand-alone in the sense that they leave the ordering (in terms of goodness-of-fit) that they impose on the data unchanged: the ordering with respect to the original error function equals the new ordering with respect to the logarithmic error. Yet the models (H, β) are all different in the sense that they all have different entropies. We show both properties below.

Proposition 5.7 *Let $H \in \mathcal{M}$ and let x^n, y^n and z^n be such that* $\text{ER}(y^n|H, x^n) > \text{ER}(z^n|H, x^n)$. *Then for all $\beta \in \Gamma_{nat}(H)$ with $\beta > 0$,*

$$-\ln P(y^n|H, \beta, x^n) > -\ln P(z^n|H, \beta, x^n).$$

while for all $\beta < 0$, $-\ln P(y^n|H, \beta, x^n) < -\ln P(z^n|H, \beta, x^n)$.

Proof: Immediate from Definition 5.2. □

Hence for each H, entropification either leaves unchanged or reverses the ordering in terms of goodness-of-fit that H imposes on the data: for every β, the ordering with respect to $\text{ER}(\cdot|H, x^n)$ is identical or reversed to the ordering with respect to the code length (or 'logarithmic error') $-\ln P(\cdot|H, \beta)$.

For the second property mentioned above, recall that $\mathcal{H}(H, \beta)$ stands for the entropy of the model $P(\cdot|H, \beta, x)$ (for arbitrary x) restricted to single outcomes in $\mathbf{E}_y$.

Proposition 5.8 *For all $\beta \in \Gamma_{nat}(H)$ with $\beta > 0$, the entropy $\mathcal{H}(H, \beta)$ is a strictly decreasing function of β. For all $\beta < 0$, $\mathcal{H}(H, \beta)$ is a strictly increasing function of β.*

The proposition is proven in Appendix 5.6. Together with Proposition 5.7 it tells us that for fixed H and varying β, the compound models (H, β) can all be seen as 'versions' of H with different entropies.

Entropy and Expected Error

Suppose $\beta > 0$. As β increases, the entropy $\mathcal{H}(H, \beta)$ decreases. One may expect that, with decreasing entropy (and thus decreasing 'inherent disorder'), the expected error $E_{(H,\beta)}[\text{ER}(Y|H, X)]$ also decreases. This relation indeed holds, but only if $\beta > 0$ (the case $\beta < 0$ will be explored briefly in Section 5.3.3): if $\beta < 0$, then (as we saw above), the entropy is an increasing function of β while the expected error remains a decreasing function of β. In general, let, for fixed H, $\mathbf{U}_H$ be the smallest (possibly unbounded) interval in $\mathbf{R}$ such that $\forall (x, y) \in \mathbf{E} : \text{ER}(y|H, x) \in \mathbf{U}_H$. Then

Proposition 5.9 $E_{(H,\beta)}[\text{ER}(Y|H, X)]$ *is a strictly decreasing function of β. For each t in the interior of $\mathbf{U}_H$ there exists a unique value of β such that $E_{(H,\beta)}[\text{ER}(Y|H, X)] = t$.*

The proposition is proven in Appendix 5.6.

Simple Error Functions

Some error functions, among which the squared error and the 0/1-error, turn out to have a useful additional property which automatically makes them satisfy our regularity conditions for error functions and which makes sure that entropification leaves the relative ordering of hypotheses in terms of goodness-of-fit for given data unchanged. We call such hypotheses *simple*:

Definition 5.10 *If* ER *is such that for all $H_1, H_2 \in \mathcal{M}$ and all β,*

$$Z_{H_1}(\beta) = Z_{H_2}(\beta),$$

where $Z_H(\beta)$ is defined as in Equation 5.6, then we call ER *a* simple *error function for $\mathcal{M}$.*

The two error functions we have encountered earlier are both simple, as shown by the following proposition:

Proposition 5.11 *Let $\mathbf{E} = \mathbf{E}_x \times \mathbf{E}_y$, let $\mathcal{M}$ be an arbitrary class of functions $H : \mathbf{E}_x \to \mathbf{E}_y$. If $\mathbf{E}_y = \mathbf{R}$, then the squared error function ER_{sq} is simple for $\mathcal{M}$. If $\mathbf{E}_y = \{0,1\}$, then the 0/1-error function ER_{01} is simple for $\mathcal{M}$.*

Proof: In the squared error case, $Z_{H,x}(\beta) = \int_{y \in \mathbf{E}_y} \exp(-\beta \mathrm{ER}_{sq}(y|H,x))dy = \int \exp(-\beta(y - H(x))^2)dy = \sqrt{\pi/\beta}$ which does not depend on either H or x. The case of the 0/1-error is analogous. □

For model classes entropified with simple error functions we can drop the subscript from $Z_H(\beta)$ and simply write $Z(\beta)$. By calculating the entropy $\mathcal{H}(H,\beta)$ of the model (H,β) it follows immediately that this entropy depends only on β and not on H.

Simple error functions have an additional important property which is dual to the property expressed by Proposition 5.7. Whereas in that proposition, we showed that the ordering in goodness-of-fit imposed on data by a hypothesis (H,β) is identical for all β (up to their sign), the present result shows that in the case of simple error functions, a reverse property holds too: the ordering in goodness-of-fit imposed on hypotheses (H,β) by given data D is identical for all β (up to their sign):

Proposition 5.12 *Let ER be a simple error function for $\mathcal{M}$ and let $D = (x^n, y^n)$. For each $\beta_1, \beta_2 \in \Gamma_{nat}$ with $\beta_1, \beta_2 > 0$ or $\beta_1, \beta_2 < 0$ and each $H_1, H_2 \in \mathcal{M}$ we have:*

$$P(y^n|H_1,\beta_1,x^n) > P(y^n|H_2,\beta_1,x^n) \quad \Rightarrow \quad P(y^n|H_1,\beta_2,x^n) > P(y^n|H_2,\beta_2,x^n)$$

and, in particular, if $\beta_1, \beta_2 > 0$ and there exists a single H that minimizes the empirical error $\mathrm{ER}(y^n|H,x^n)$, then:

$$\widehat{H}(D|\beta_1) = \widehat{H}(D|\beta_2) = \arg\min_{H \in \mathcal{M}} \{\mathrm{ER}(y^n|H,x^n)\} \tag{5.20}$$

Hence the H in the tuple (H,β) that maximizes the likelihood is independent of β (except for the sign of β) and (if $\beta > 0$) is equal to the $H \in \mathcal{M}$ that minimizes the empirical error.

Proof: Immediate from instantiating $P(y^n|H_i,\beta_i,x^n)$ using Definition 5.2 and the fact that $Z(\beta)$ does not depend on H. □

We have now established all basic properties of entropification. We will use these properties to prove the main results of this chapter, concerning its behaviour in the i.i.d. case. Before we do this, we give a summary of the possible interpretations of β.

5.2.4 Interpretations of β

The parameter β as part of the hypothesis (H,β) can be interpreted in the following ways:

1. β determines the expected error $E_{(H,\beta)}[\mathrm{ER}(Y|H,X)]$ (Proposition 5.9); hence ...

2. ... when (H, β) is inferred from data D, β serves as an estimate of the error H will make on future data.

3. β determines the entropy $\mathcal{H}(H, \beta)$ (Proposition 5.8): the closer $|\beta|$ to 0, the larger $\mathcal{H}(H, \beta)$.

4. β determines how strongly the error $\text{ER}(y^n|H, x^n)$ is weighed in the code based on $P(\cdot|H, \beta, \cdot)$, which has lengths $L(y^n|H, x^n) = \beta\text{ER}(y^n|H, x^n) + n \ln Z_H(\beta)$ (Equation 5.9): the closer $|\beta|$ to 0, the closer P is to the uniform distribution.

The last two items show that β can be interpreted as a kind of 'noise' level, measuring for each fixed H *the apparent randomness of the data with respect to hypothesis H* (we use the word 'apparent' because a small value of β does not mean that the data are random in any general sense; it only means that H *does not give very much information about the data*). From the polynomial example we see by the substitution $\beta = 1/(2\sigma^2)$ that small β means large variance σ^2 and hence a large amount of 'noise' with respect to H. From the concept learning example (Example 5.3) we see by the substitution $\beta = \ln(1 - \theta) - \ln(\theta)$ that a hypothesis H together with a small (absolute) value of β expresses the extended hypothesis that the probability that H makes a mistake is θ, where θ is close to 1/2 (uniform). $\beta = 0$ cancels the effect of the hypothesis altogether and makes $P(\cdot|H)$ equal to the uniform distribution.

β and temperature The idea to turn an error function $\text{ER}(\cdot|\cdot)$ into a class of probability distributions $P(\cdot|\cdot, \beta) = Z(\beta)^{-1} \exp(-\beta\text{ER}(\cdot|\cdot))$ is actually not new: it is common practice in Statistical Mechanics [155, 69] where the error function is called 'energy function' and in stead of a parameter β one uses a parameter T (called a 'temperature') satisfying $\beta = 1/T$. Such 'energy functions' and 'temperatures' are frequently used outside of a purely physical context. As far as we know, the role of the temperature T is somewhat different from our β since T is not treated as a parameter to be estimated from data.

5.3 Entropification and Generating Distributions

In this section we present our main results concerning entropification. We study the behaviour of entropified model classes when data are independently distributed according to some unknown 'true' distribution P^*. Roughly, it is shown that with an entropified model class $\langle\mathcal{M}\rangle_{\text{ER}}$, if given enough data, we can find the model in $\langle\mathcal{M}\rangle_{\text{ER}}$ with the smallest expected prediction error under P^*. Additionally, this model will provide a correct estimate of the average prediction error over future data that it will achieve; hence the model gives a good impression of 'how good it really is' when errors are measured by ER. The important thing is that this model that is both optimal and 'reliable' will be found *even if P^* is not contained in $\langle\mathcal{M}\rangle_{\text{ER}}$*. Below we state a technical lemma that is needed in our theorems, which follow in sections 5.3.1-5.3.5. The proof of the lemma is quite involved and is not directly related to entropification. Therefore it has been deferred to an appendix (Appendix 5.7, page 114). The proofs of the theorems are directly related to the main properties of entropification, so they can be found in this section. These theorems concern the general case (Section 5.3.2), the case of simple error functions (Section 5.3.3), the case of the logarithmic error (Section 5.3.4) and the case of the squared error (Section 5.3.5). The lemma and the theorems all assume that the data are i.i.d. according to some unknown but fixed probability distribution P^*. We have to impose some mild conditions on P^*. These amount to there being some 'window' (i.e. a bounded set containing more than one element) within which all data will fall. The reason is that otherwise the required expectation $E_{P^*}[\text{ER}(Y|H,X)]$ may not exist. Here is a formal definition of this condition:

Definition 5.13 (Regularity Condition for the True Distribution)
Let a sample space $\mathbf{E} = \mathbf{E}_1 \times \ldots \times \mathbf{E}_m$ be given for $m \geq 1$. Whenever in the following we speak of a 'true', or 'generating' distribution P^*, we assume P^* to be a distribution over $\mathbf{E}_{P^*} = \mathbf{E}_{P^*,1} \times \ldots \times \mathbf{E}_{P^*,m}$ with full support such that for $1 \leq i \leq m$, (a) $\mathbf{E}_{P^*,i} \subseteq \mathbf{E}_i$ (b) $\mathbf{E}_{P^*,i}$ contains more than one element and (c) if $\mathbf{E}_i$ is continuous, then $\mathbf{E}_{P^*,i}$ is compact.

We now state our technical lemma. Essentially, it says the following: if $\mathcal{M}$ is compactly parameterized, then the average code length of x^n based on the maximum likelihood model for x^n converges (with probability 1) to the expected code length based on the model in the class that minimizes this expected code length. This holds if data are i.i.d. according to some P^* satisfying Definition 5.13. Note that P^* is *not* required to be a member of $\mathcal{M}$.

Lemma 5.14 *Let $\mathcal{M} = \{P(\cdot|\theta) \mid \theta \in \Gamma\}$ be a class of i.i.d. probabilistic models over sample space $\mathbf{E}$ that is finitely parameterized by $\Gamma \subset \mathbf{R}^k$ where Γ is compact. Let the data be i.i.d. according to some P^* satisfying Definition 5.13. Then the following minima exist for all n, all $x^n \in \mathbf{E}^n$:*

$$\begin{aligned} \hat{L}(x^n) &:= \min_{\theta \in \Gamma}\{-\ln P(x^n|\theta)\} \\ \widetilde{L}(P^*) &:= \min_{\theta \in \Gamma}\{E_{P^*}[-\ln P(X|\theta)]\} \end{aligned}$$

We have with P^*-probability 1:

$$\lim_{n\to\infty} \frac{1}{n}\hat{L}(x^n) = \widetilde{L}(P^*)$$

The proof is given in Appendix 5.7. Lemma 5.14 is the key to the theorems which will be proven in the next subsections.

5.3.1 The Theorems

This subsection gives an informal overview of the theorems we are about to prove. We start by defining an analogue of the definition of 'reliable' (Definition 4.4) for the setting where some true (i.i.d.) distribution is assumed to exist.

Definition 5.15 *Let the data be i.i.d. according to some distribution P^*. Let P be some given probabilistic model over $\mathbf{E}$ and let $\psi : \mathbf{E} \to \mathbf{U}$ be some given function. We call P* reliable with respect to ψ under P^* *if*

$$E_P[\psi(X)] = E_{P^*}[\psi(X)].$$

A model P that is reliable with respect to ψ under P^* is (with probability 1) guaranteed to give a correct impression of the average $\overline{\psi(x)}^n$ for large n: by the law of large numbers, $\overline{\psi(x)}^n \rightarrow E_{P^*}[\psi(X)] = E_P[\psi(X)]$ as n increases, with P^*-probability 1.

Informal Discussion of the Theorems Let $\langle \mathcal{M} \rangle_{\text{ER}}$ be a model class entropified with respect to an error function ER. Let the data be i.i.d. according to some arbitrary P^* (not necessarily in $\langle \mathcal{M} \rangle_{\text{ER}}$). The main point of *Theorem 5.16* is that for each $H \in \mathcal{M}$, there exists a unique $\tilde{\beta}_H$ such that (1) $E_{(H,\tilde{\beta}_H)}[\text{ER}(Y|H,X)] = E_{P^*}[\text{ER}(Y|H,X)]$ (hence $(H, \tilde{\beta}_H)$ is reliable with respect to ER under P^*), and (2), $\hat{\beta}(D|H)$, the maximum likelihood estimator for fixed H (Definition 5.5), converges with P^*-probability 1 to $\tilde{\beta}_H$. Hence for each H, a reliable estimate of its performance can, with probability 1, be obtained.

If the error function ER is *simple* (Definition 5.10), then in addition, the stronger *Theorem 5.17* applies. Its essence is (roughly) that the maximum likelihood estimator $(\widehat{H}, \hat{\beta})$ converges with P^*-probability 1 to the model $(\widetilde{H}, \tilde{\beta})$ where $\widetilde{H}$ is the optimal model in $\mathcal{M}$, minimizing the 'true' expected error $E_{P^*}[\text{ER}(Y|\widetilde{H},X)]$, and $\tilde{\beta}$ is such that $E_{(\widetilde{H},\tilde{\beta})}[\text{ER}(Y|\widetilde{H},X)] = E_{P^*}[\text{ER}(Y|\widetilde{H},X)]$, and so $(\widetilde{H}, \tilde{\beta})$ is reliable for $\text{ER}(\cdot|\widetilde{H},\cdot)$ under P^*. Hence the optimal $\widetilde{H}$ and a reliable estimate of its performance can both, with probability 1, be obtained.

If the error function is not simple, then things get more complicated. Nevertheless, *Theorem 5.18* shows that in the special case of the (non-simple) logarithmic error function, an analogue to the above (maximum likelihood estimators converging to an optimal and reliable model) still holds.

The squared error function *is* simple but satisfies an additional interesting property, as will be shown in *Theorem 5.19*.

Use of ML Estimators in the Theorems In the theorems we make use of the maximum likelihood estimator $\hat{\beta}(D|H)$ for fixed H which is defined as in Definition 5.5. It is straightforward to show that, under our conditions for the sample space of the generating distribution P^*, a unique value of $\hat{\beta}(D|H)$ always exists. Sometimes we will also make use of the full maximum likelihood estimator $(\widehat{H}, \hat{\beta})(D)$. In all these cases, it is straightforward to show that there exists at least one maximum of the likelihood. We use the convention that, if several (H, β) maximizing the likelihood exist, then $(\widehat{H}, \hat{\beta})$ denotes the first one according to some prespecified ordering over $\langle \mathcal{M} \rangle_{\text{ER}}$.

5.3.2 General Error Functions

In this subsection we present Theorem 5.16 which is applicable to all regular error functions.

Theorem 5.16 *Let $\mathbf{E} = \mathbf{E}_x \times \mathbf{E}_y$. Let $\mathcal{M}$ be a class of models and let $\text{ER} : \mathbf{E} \times \mathcal{M} \rightarrow \mathbf{R}$ be an error function for $\mathcal{M}$. Assume that data (x, y) are generated by independent sampling from a distribution P^* over $\mathbf{E}_{P^*}$ as in Definition 5.13. Then for all fixed $H \in \mathcal{M}$, $E_{P^*}[\text{ER}(Y|H,X)]$ exists, and*

1. *there exists a unique $\tilde{\beta}_H$ depending on H such that*

$$E_{(H,\tilde{\beta}_H)}[\text{ER}(Y|H,X)] = E_{P^*}[\text{ER}(Y|H,X)]. \tag{5.21}$$

and at the same time, for all $\beta \in \Gamma_{nat}(H)$ with $\beta \neq \tilde{\beta}_H$,

$$E_{P^*}[-\ln P(Y|\beta,H,X)] > E_{P^*}[-\ln P(Y|\tilde{\beta}_H,H,X)] \tag{5.22}$$

2. *with P^*-probability 1,*

$$\lim_{n\to\infty} \hat{\beta}(x^n, y^n|H) = \tilde{\beta}_H \tag{5.23}$$

and hence

$$\lim_{n\to\infty} E_{(H,\hat{\beta}(D|H))}[\text{ER}(Y|H,X)] = E_{P^*}[\text{ER}(Y|H,X)] \tag{5.24}$$

Proof: We only prove the theorem for continuous $\mathbf{E}_x$ and $\mathbf{E}_y$. The case where $\mathbf{E}_x$ or $\mathbf{E}_y$ or both are discrete is completely analogous.

We first prove existence of $E_{P^*}[\text{ER}(Y|H,X)]$. Definition 5.13 tells us that P^* is defined over a compact subspace of $\mathbf{E}$. Conditions **C1**-**C3** on ER (see the remark at the beginning of Section 5.2) make sure that $\text{ER}(y|H,x)$ is continuous at all $(x,y) \in \mathbf{E}$. For continuous $\mathbf{E}$, we only consider P^* with associated continuous density functions (see Chapter 1, Section 1.2). Existence of $E_{P^*}[\text{ER}(Y|H,X)]$ now follows.

Now to prove item 1, note that by Definition 5.13, $E_{P^*}[\text{ER}(Y|H,X)]$ must lie in the interior of $\mathbf{U}_H$. Therefore, by Proposition 5.9 there must be a unique value $\tilde{\beta}_H$ for which (5.21) holds.

We have, for each β, $E_{P^*}[-\ln P(Y|H,\beta,X)] = \beta E_{P^*}[\text{ER}(Y|H,X)] + \ln Z_H(\beta)$. By differentiating with respect to β one verifies that $E_{P^*}[-\ln P(Y|H,\beta,X)]$ as a function of β is convex and reaches its unique minimum at the value $\tilde{\beta}_H$ for which (5.21) holds. This proves (5.22).

Concerning item 2, we will first prove (5.24). Since the data are i.i.d. we can apply the strong law of large numbers [47] which gives that with P^*-probability 1, $n^{-1}\sum_{i=1}^{n} \text{ER}(y_i|H,x_i)$ converges to $E_{P^*}\text{ER}(Y|H,X)$. (5.24) then follows by the reliability of estimates of ER (Proposition 5.6). (5.23) is now immediate by (5.24) and the fact (which we just showed) that $E_{P^*}[-\ln P(Y|\beta,H,X)]$ as a function of β is convex and reaches its single maximum at $\beta = \tilde{\beta}_H$. □

We now proceed to the special case where ER is a simple error function.

5.3.3 Simple Error Functions

For simple error functions Proposition 5.12 applies: if $\beta > 0$, then minimization of logarithmic error $-\ln P(y^n|H,\beta,x^n)$ corresponds to minimization of the error function ER. This allows us to prove Theorem 5.17, which says that the maximum likelihood estimator $(\widehat{H},\hat{\beta})$ for data D converges to a model $(\widetilde{H},\tilde{\beta})$ where, if $\tilde{\beta} > 0$, then $\widetilde{H}$ minimizes the 'true' expected error $E_{P^*}[\text{ER}(Y|H,X)]$ over all $H \in \mathcal{M}$. Let us briefly consider the case $\tilde{\beta} < 0$. In the case of the squared error, $\Gamma_{nat}(H)$ only contains positive parameter values, so then always $\tilde{\beta} > 0$ and the problem does not occur. In the special case of the 0/1-error, something interesting happens which we illustrate with an example. Suppose our concept class $\mathcal{M}$ contains only two models H_1 and H_2. Suppose P^* to be such that $E_{P^*}[\text{ER}(Y|H_1, X)] = 0.3$ and $E_{P^*}[\text{ER}(Y|H_2, X)] = 0.9$. Then the hypothesis minimizing the expected 0/1-error is clearly H_1. However, H_2 can be trivially modified into another 'inverse' hypothesis $\overline{H_2}$ with $E_{P^*}[\text{ER}(Y|\overline{H_2}, X)] = 0.1$: $\overline{H_2}(x)$ predicts 1 if $H_2(x) = 0$ and 0 otherwise. This trivial modification can be achieved by entropification: the entropified model $(\widetilde{H}, \tilde{\beta})$ that leads to the shortest expected code length will in our example be given for $\widetilde{H} = H_2$ and $\tilde{\beta} < 0$; the fact that $\tilde{\beta} < 0$ makes H_2 behave like its inverse $\overline{H_2}$. H_2 will lead to much shorter (expected) code lengths than H_1 (all this can be easily checked using equations 5.11 and 5.12 of Example 5.3, page 89).

Theorem 5.17 *Let* $\mathbf{E}$*, data* (x, y)*,* P^* *and* E_{P^*} *be as in the statement of Theorem 5.16. Let* ER *be a simple error function and assume* $\mathcal{M}$ *to be such that* $\langle\mathcal{M}\rangle_{\text{ER}}$ *is finitely parameterized by* $\Gamma_{\mathcal{M}} \times \Gamma_{nat}$ *where* $\Gamma_{\mathcal{M}}$ *is compact. Then*

1. *The following minima exist:*

$$\widetilde{\text{ER}}(P^*) := \min_{H \in \mathcal{M}} E_{P^*}[\text{ER}(Y|H, X)] \tag{5.25}$$

$$\widetilde{L}(P^*) := \min_{P(\cdot|\theta,\cdot) \in \langle\mathcal{M}\rangle_{\text{ER}}} E_{P^*}[-\ln P(Y|\theta, X)] \tag{5.26}$$

Let $\tilde{\theta}$ *be one of the models for which the minimum in (5.26) is obtained. Then*

2. $\tilde{\theta} = (\widetilde{H}, \tilde{\beta})$ *for some* $\tilde{\beta} \in \Gamma_{nat}$*. If* $\tilde{\beta} > 0$*, then* $\widetilde{H}$ *is (one of) the hypothesis (hypotheses) for which the minimum in (5.25) is obtained (*$\tilde{\beta}$ *is identical for all such* $\widetilde{H}$*).*

Let $(\widehat{H}, \hat{\beta}) := (\widehat{H}, \hat{\beta})(D)$ *denote the maximum likelihood estimator in* $\langle\mathcal{M}\rangle_{\text{ER}}$*.*

3. *We have:*

$$\begin{aligned} &\lim_{n\to\infty} E_{(\widehat{H},\hat{\beta})}[\text{ER}(Y|\widehat{H}, X)] = \\ &\quad E_{(\widetilde{H},\tilde{\beta})}[\text{ER}(Y|\widetilde{H}, X)] = E_{P^*}[\text{ER}(Y|\widetilde{H}, X)] \end{aligned} \tag{5.27}$$

Hence, for each 'true', 'generating' distribution P^* there exists an optimal model $(\widetilde{H}, \tilde{\beta})$ such that the 'true' expectation under P^* of the error $\text{ER}(Y|\widetilde{H}, X)$ is minimal and *equal* to the expectation of this error under $(\widetilde{H}, \tilde{\beta})$: when given enough data, every reasonable (Chapter 4, Definition 4.3) inference procedure will hit upon a model that is optimal in this sense.

Proof: We only prove the theorem for continuous $\mathbf{E}_x$ and $\mathbf{E}_y$. The case where $\mathbf{E}_x$ or $\mathbf{E}_y$ or both are discrete is completely analogous.

Concerning items 1 and 2, existence of $\widetilde{\text{ER}}(P^*)$ is straightforward by compactness of $\Gamma_{\mathcal{M}}$. Existence of $\widetilde{L}(P^*)$ and item 2 will now be proven at the same time. First write

$$E_{P^*}[-\ln P(Y|\theta, X)] = \beta E_{P^*}[\text{ER}(Y|H, X)] + \ln Z(\beta) \tag{5.28}$$

for $(H, \beta) = \theta$. Since we assume ER to be simple here, $Z(\beta)$ does not depend on H. This shows that for each fixed $\beta > 0$, $E_{P^*}[-\ln P(Y|(H, \beta), X)]$ reaches its minimum for the set $\mathcal{H}^+$ of H minimizing $E_{P^*}[\text{ER}(Y|H, X)]$. By differentiating with respect to β and the fact that $Z(\beta)$ does not depend on H, one finds that there exists a single $\tilde{\beta}^+$ which minimizes $E_{P^*}[-\ln P(Y|(H, \beta), X)]$ for all $H \in \mathcal{H}^+$. For fixed $\beta < 0$, $E_{P^*}[-\ln P(Y|(H, \beta), X)]$ reaches its minimum for the set $\mathcal{H}^-$ of H maximizing $E_{P^*}[\text{ER}(Y|H, X)]$ (which exists by compactness of $\Gamma_{\mathcal{M}}$). In this case there exists a single $\tilde{\beta}^-$ which minimizes $E_{P^*}[-\ln P(Y|(H, \beta), X)]$ for all $H \in \mathcal{H}^-$. If $\beta = 0$, then $E_{P^*}[-\ln P(Y|(H, \beta), X)]$ reaches its minimum for all $H \in \mathcal{M}$. From this it easily follows that a $(\tilde{\beta}, \widetilde{H})$ minimizing (5.28) exists, that all minima of (5.28) have the same component $\tilde{\beta}$ and that if $\tilde{\beta} > 0$, then $\widetilde{H} \in \mathcal{H}^+$. This proves both existence of $\widetilde{L}(P^*)$ and item 2 of the theorem.

The key to the proof of item 3 is the result (5.33) below. In order to obtain this result we need to apply Lemma 5.14 of page 96. The lemma cannot be simply applied to model classes $\langle \mathcal{M} \rangle_{\text{ER}}$, since these contain conditional rather then regular probabilistic models. To avoid this problem we change $\langle \mathcal{M} \rangle_{\text{ER}}$ into a class of essentially equivalent regular probabilistic models over $\mathbf{E}_x \times \mathbf{E}_y$ by extending it with the uniform distribution over $\mathbf{E}_x$ (this is possible since $\mathbf{E}_x$ is compact by condition **C4**; see the beginning of Section 5.2) (for motivation, see Lemma 5.23 in Appendix 5.6 where a similar trick is used).

Let P^u be the uniform distribution over $\mathbf{E}_x$ and let

$$P^u(x^n, y^n|H, \beta) = P(y^n|H, \beta, x^n)P^u(x^n) \tag{5.29}$$

be the distribution that extends each conditional distribution $P(\cdot|H, \beta, \cdot)$ to a full distribution over $\mathbf{E}_x \times \mathbf{E}_y$. We have for all $H \in \mathcal{M}$:

$$-\ln P^u(x^n, y^n|H, \tilde{\beta}) = -\ln P(y^n|H, \tilde{\beta}, x^n) + C \cdot n \tag{5.30}$$

for some constant C. Here $\tilde{\beta}$ is as in the statement of the theorem, item 2. Let

$$\widetilde{L}^u(P^*) := \min_{H \in \mathcal{M}} E_{P^*}[-\ln P^u(X, Y|H, \tilde{\beta})] = \widetilde{L}(P^*) + C \tag{5.31}$$

Here C is the same constant as in (5.30). Finally, let $\langle \mathcal{M} \rangle_{\tilde{\beta}} = \{P(\cdot, \cdot|H, \tilde{\beta}) \mid H \in \mathcal{M}\}$ be the class of probabilistic models of form (5.29) for which $\beta = \tilde{\beta}$.

It is straightforward to check that $\langle \mathcal{M} \rangle_{\tilde{\beta}}$ is such that Lemma 5.14 of page 96 applies. Substituting $\hat{L}(x^n) = -n^{-1} \ln P^u(x^n, y^n|\widehat{H}, \tilde{\beta})$, this gives that with P^*-probability 1,

$$\lim_{n \to \infty} -\frac{1}{n} \ln P^u(x^n, y^n|\widehat{H}, \tilde{\beta}) = \widetilde{L}^u(P^*) \tag{5.32}$$

(5.30), (5.31) and (5.32) give us (with P^*-probability 1):

$$\lim_{n\to\infty} -\frac{1}{n} \ln P(y^n|\widehat{H}, \tilde{\beta}, x^n) = \widetilde{L}(P^*) = E_{\tilde{\beta}}[-\ln P(Y|\widetilde{H}, X)] \tag{5.33}$$

for all the $\widetilde{H}$ minimizing (5.25), where the last equality follows from item 2 in the statement of the theorem which we proved already. By the identity $-\ln P(y^n|H, \beta, x^n) = \beta\text{ER}(y^n|H, x^n) + n \ln Z(\beta)$ this gives with P^*-probability 1:

$$\lim_{n\to\infty} -\frac{1}{n}\tilde{\beta}\text{ER}(y^n|\widehat{H}, x^n) + \ln Z(\tilde{\beta}) = \tilde{\beta}E_{\tilde{\beta}}[\text{ER}(Y|\widetilde{H}, X)] + \ln Z(\tilde{\beta}) \tag{5.34}$$

By reliability of estimates of ER (Proposition 5.6) we have $n^{-1}\text{ER}(y^n|\widehat{H}, x^n) = E_{(\widehat{H},\hat{\beta})}[\text{ER}(Y|\widehat{H}, X)]$. Plugging this into (5.34) proves (5.27). □

5.3.4 Non-Simple Error Functions; Logarithmic Error

It is, in general, difficult to analyze for non-simple error functions whether an analogue of Theorem 5.17 holds. The proof of Theorem 5.17 is based on the fact that, for simple error functions, minimization of logarithmic error corresponds to minimization (or maximization) of the error function ER. For non-simple error functions this need not be the case since $Z(\beta)$ varies with H. However, a special case occurs if $\mathcal{M}$ is probabilistic and we entropify with respect to the logarithmic error. In that case, the function $\text{ER} = \text{ER}_{lg}$ measures itself the log-likelihood of the data, while the optimal model in $\langle\mathcal{M}\rangle_{\text{ER}}$ is also optimal with respect to expected log-likelihood. This allows an analogue to Theorem 5.17 to be proven after all; it is embodied in Theorem 5.18 below. Since in this situation, it turns out to be somewhat harder to identify exact conditions under which the required minima exist, we will keep things simple and simply assume $\mathcal{M}$ to be such that they exist. Specifically, let $\mathcal{M}$ be a class of probabilistic models over sample space $\mathbf{E}$ that is finitely parameterized by $\Gamma_{\mathcal{M}}$ and let $\langle\mathcal{M}\rangle_{\text{ER}_{lg}}$ be the entropification of $\mathcal{M}$ under the logarithmic error. $\langle\mathcal{M}\rangle_{\text{ER}_{lg}}$ can be parameterized by $\Gamma_{\mathcal{M}} \times \Gamma_{nat}(H)$. Let data $x_1, x_2, \ldots$ be generated by independent sampling from a distribution P^* over $\mathbf{E}_{P^*}$ where P^* is as in Definition 5.13. We assume that (1) $\mathbf{E}_{P^*}$ is such that for all n, $x^n \in \mathbf{E}^n_{P^*}$, the maximum likelihood estimator of x^n in $\langle\mathcal{M}\rangle_{\text{ER}_{lg}}$, denoted by $\hat{\theta} := \hat{\theta}(x^n)$ exists and falls within a compact subset of $\Gamma_{\mathcal{M}} \times \Gamma_{nat}(H)$, and (2),

$$\widetilde{L}(P^*) := \min_{\theta\in\Gamma_{\mathcal{M}}\times\Gamma_{nat}(H)} E_{P^*}[-\ln P(X|\theta)] \tag{5.35}$$

exists and is obtained by a single model $\tilde{\theta}$.

Theorem 5.18 *Let $\mathcal{M}$, P^* and $\tilde{\theta}$ be as above. Then with P^*-probability 1:*

$$\lim_{n\to\infty} E_{\hat{\theta}(x^n)}[-\ln P(X|\hat{\theta}(x^n))] = E_{\tilde{\theta}}[-\ln P(X|\tilde{\theta})] = E_{P^*}[-\ln P(X|\tilde{\theta})]. \tag{5.36}$$

Proof: In the proof we assume the notation of Example 5.4. Specifically, $P(\cdot|\eta)$ stands for the model in $\mathcal{M}$ indexed by η. $P(\cdot|(\eta,\beta)) = Z_\eta^{-1}(\beta)\exp(\beta \ln P(\cdot|\eta))$ stands for the model in $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$ indexed by (η,β). Note also that $\hat{\theta} = (\hat{\eta},\hat{\beta})$. By reliability of the estimates of ER_{lg} when the class $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$ is used (Proposition 5.6) we find

$$E_{(\hat{\eta},\hat{\beta})}[-\ln P(X|\hat{\eta})] = -\frac{1}{n}\sum_{i=1}^{n} \ln P(x_i|\hat{\eta})$$

By straightforward calculation this gives

$$E_{(\hat{\eta},\hat{\beta})}[-\ln P(X|(\hat{\eta},\hat{\beta}))] = -\frac{1}{n}\sum_{i=1}^{n} \ln P(x_i|(\hat{\eta},\hat{\beta})) \tag{5.37}$$

Let $\langle\mathcal{M}\rangle'_{\mathrm{ER}_{lg}}$ be the restriction of $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$ to models with parameter values in the compact set within which $\hat{\theta}(x^n)$ must fall (we assumed that such a set exists). Clearly, $\tilde{\theta}$ must be a member of this set. Now we can apply Lemma 5.14 (page 96) to $\langle\mathcal{M}\rangle'_{\mathrm{ER}}$. This gives, with P^*-probability 1,

$$\lim_{n\to\infty} -\frac{1}{n}\ln P(x^n|\hat{\theta}(x^n)) = \widetilde{L}(P^*). \tag{5.38}$$

Together, (5.37) and (5.38) show that (5.36) holds. □

5.3.5 Special Status of the Squared Error

In classical statistics, the problem of curve-fitting is cast in the following terms: one assumes data to be independently generated by some unknown distribution P^* and one tries to identify the function H^* (called the 'regression function') that, for each x, gives the expected value (the mean) of Y given that $X = x$. (some would prefer to say: one assumes that data are generated by some function H^* with *errors* distributed according to P^*). Whatever the distribution of the errors, this can be achieved by using the squared error function in the learning phase, as will be shown in Theorem 5.19 below. Such results have been known for a long time [20]. For completeness and since it is not difficult, we have included Theorem 5.19 nevertheless.

Let $\mathbf{E} = \mathbf{E}_x \times \mathbf{E}_y$ where $\mathbf{E}_x \subset \mathbf{R}$ and $\mathbf{E}_y = \mathbf{R}$. We assume data to be generated by independent sampling from P^* as in Definition 5.13. Let $H^*(x) = E_{P^*}[Y|X = x]$. That is, $H^*(x)$ gives the mean of Y for each $x \in \mathbf{E}_x$. We will assume that $H^*(x)$ is continuous at all $x \in \mathbf{E}_x$. Let $(\sigma^*)^2 = E_{P^*}[(Y - H^*(X))^2]$. Hence $(\sigma^*)^2$ denotes the 'expected true variance' of Y. We already know that, for a given class $\mathcal{M}$ of functions $\mathbf{E}_x \to \mathbf{E}_y$, $\langle\mathcal{M}\rangle_{\mathrm{ER}_{sq}}$ consists of conditional Gaussian distributions as given by Equation 5.2 on page 84 (Example 5.1). These distributions are obtained from the natural parameterization of $\langle\mathcal{M}\rangle_{\mathrm{ER}_{sq}}$ by substituting $\beta = 1/2\sigma^2$. Under the natural parameterization, theorems 5.16 and 5.17 are applicable. The following theorem extends these theorems for the specific case of $\langle\mathcal{M}\rangle_{\mathrm{ER}_{sq}}$. Briefly, the only non-trivial results that are added are the following: (1) for every P^*, the optimal model

$P(\cdot|\tilde{\sigma}^2, \widetilde{H}) = P(\cdot|\tilde{\beta}, \widetilde{H})$ will be such that $\widetilde{H}$ is the function in $\mathcal{M}$ that is closest (in the mean squared error sense) to the 'true' function H^* and (2) $\tilde{\sigma}^2$ can be interpreted as the mean squared error of $\widetilde{H}$. Since, for every P^*, $(\widehat{H}, \hat{\sigma}^2)$ will converge to $(\widetilde{H}, \tilde{\sigma}^2)$ with P^*-probability 1, this implies that in the special case with $H^* \in \mathcal{M}$, $\widehat{H}$ will converge to the true H^* and $\hat{\sigma}^2$ will converge to the true variance $(\sigma^*)^2$ with P^*-probability 1. This holds *independently* of whether $P^*(\cdot|x)$ is Gaussian or not.

In the following theorem, we assume models in $\langle \mathcal{M} \rangle_{\mathrm{ER}_{sq}}$ to be specified by (H, σ^2) rather than (H, β).

Theorem 5.19 *Let* $\mathbf{E}$, $\mathcal{M}$, $\langle \mathcal{M} \rangle_{\mathrm{ER}_{sq}}$ *and* P^* *be as above. Then:*

1. *For all* $H \in \mathcal{M}$, $E_{P^*}[\mathrm{ER}_{sq}(Y|H,X)]$ *exists, and there exists a* $\tilde{\sigma}^2_H$ *depending on* H *such that*

$$\begin{aligned} &E_{(H,\tilde{\sigma}^2_H)}[(Y - H(X))^2] = \\ &\quad E_{P^*}[(Y - H(X))^2] = (\sigma^*)^2 + E_{P^*}[(H^*(X) - H(X))^2] \end{aligned} \tag{5.39}$$

2. *Further assume* $\mathcal{M}$ *to be such that* $\langle \mathcal{M} \rangle_{\mathrm{ER}_{sq}}$ *is finitely parameterized by* $\Gamma_{\mathcal{M}} \times \Gamma_{nat}$ *where* $\Gamma_{\mathcal{M}}$ *is compact. Then the following minimum exists:*

$$\begin{aligned} \tilde{\sigma}^2 &:= \min_{H \in \mathcal{M}} E_{P^*}[(Y - H(X))^2] = \\ &= (\sigma^*)^2 + \min_{H \in \mathcal{M}} E_{P^*}[(H^*(X) - H(X))^2] \end{aligned} \tag{5.40}$$

Let $\widetilde{H}$ *be one of the models for which the minimum in (5.40) is obtained. We have:*

$$\begin{aligned} &\lim_{n \to \infty} E_{(\widehat{H},\hat{\sigma}^2)}[(Y - \widehat{H}(X))^2] = E_{(\widetilde{H},\tilde{\sigma}^2)}[(Y - \widetilde{H}(X))^2] = \tilde{\sigma}^2 = \\ &\quad = (\sigma^*)^2 + \min\nolimits_{H \in \mathcal{M}} E_{P^*}[(H^*(X) - H(X))^2] \end{aligned} \tag{5.41}$$

In particular, if $H^* \in \mathcal{M}$, *then* $\hat{\sigma}^2$ *converges with probability 1 to the true variance* $(\sigma^*)^2$ *and* $\widehat{H}$ *converges to the true hypothesis* H^*.

Proof: Most of item (1) of the theorem is straightforward from Theorem 5.16; the only thing that still needs to be proven is the fact that

$$E_{P^*}[(Y - H(X))^2] = (\sigma^*)^2 + E_{P^*}[(H^*(X) - H(X))^2] \tag{5.42}$$

Item (2) follows Theorem 5.17. The only thing that is not immediate from this theorem is, once more, (5.42), and additionally

$$E_{(\widetilde{H},\tilde{\sigma}^2)}[(Y - \widetilde{H}(X))^2] = \tilde{\sigma}^2. \tag{5.43}$$

It is a standard fact of regression [6] (also straightforward to verify by calculation) that for all σ^2 and H, we have $E_{(H,\sigma^2)}[(Y - H(X))^2] = \sigma^2$. This shows (5.43). (5.42) is a variation of the well-known bias-variance decomposition [55], also straightforward to prove:

$$
\begin{aligned}
& E_{P^*}[(Y - H(X))^2] - E_{P^*}[(Y - H^*(X))^2] \stackrel{(1)}{=} \\
& E_{P^*(X)}[E_{P^*(Y|X)}[2YH^*(X) - 2YH(X) + H(X)^2 - H^*(X)^2 | X = x]] \stackrel{(2)}{=} \\
& E_{P^*}[(H(X) - H^*(X))^2]
\end{aligned} \tag{5.44}
$$

(1) follows from using the linearity of expectation, working out the squares and conditioning on X. (2) is obtained by using $E_{P^*}[Y|X = x] = H^*(x)$. From (5.44), the equality (5.42) is immediate. □

5.4 Entropification and MDL

Is entropification merely a convenient tool to make predictions reliable or are there additional reasons as to why we should 'entropify' our model classes? In this section we show that if we use the MDL Principle as our statistical inference procedure, then it is often a good idea to use an entropified model class for at least two further reasons: first, entropification can serve to optimize the trade-off between hypothesis complexity and goodness-of-fit as needed in the two-part MDL code and the stochastic complexity. Second, it leads to codes for non-probabilistic model classes that can be justified in terms of minimizing expected code length. These points are discussed in Section 5.4.1 and Section 5.4.2 respectively.

5.4.1 Entropification and Model Selection

There have been different proposals in the literature on how to deal with two-part codes and stochastic complexity codes for non-probabilistic model classes in the MDL framework. For simplicity we will restrict our discussion to the two-part codes. Recall that in the basic, probabilistic case, we select the H minimizing

$$
- \log P(x^n | H) + L_{C_1}(H) \tag{5.45}
$$

where C_1 is some code used for encoding the parameters indexing the hypothesis. Rissanen ([128], page 18/19) proposes to turn a non-probabilistic model class into a probabilistic class $\mathcal{M}_{pr}$ (which essentially corresponds to entropification with $\beta = 1$). This leads to finding the H minimizing

$$
\beta \text{ER}(y^n | H, x^n) + n \ln Z_H(\beta) + L_{C_1}(H) \tag{5.46}
$$

with $\beta = 1$. The problem here is that choosing $\beta = 1$ is essentially arbitrary but can have large consequences: choosing a different value of β we may end up, at least for small n, with H of very different complexity (the closer β to 0, the larger the relative weight of the complexity term in (5.46)).

Barron [13] proposes to select the hypothesis H minimizing the sum of the empirical error $\text{ER}(y^n | H, x^n)$ and the square root of the complexity term times the sample size,

$\sqrt{n \cdot L_{C_1}(H)}$. While this criterion can be shown to have some strong asymptotic properties, it is in a sense not faithful to the MDL principle since the resulting sum does not have a natural interpretation as a code length.

Yamanishi [167] proposes to minimize $\beta\mathrm{ER}(y^n|H, x^n) + L_{C_1}(H)$ for some β whose value is made dependent on the size of the training set. Again, this has strong asymptotic properties, but again, it is not clear how to interpret the resulting sum from a purely coding-theoretic point of view.

Instead, we propose to entropify $\mathcal{M}$ and then use (5.46) (now with an additional term $L_{C_1}(\beta)$ added to account for the number of bits needed to encode β). We think there are several advantages to using entropified model classes. We first note that, at least non-asymptotically, using the entropified class $\langle\mathcal{M}\rangle_{\mathrm{ER}}$ can lead one to choose different models for the same data than when using $\mathcal{M}_{pr}$ for fixed β. We give a little example.

Example 5.20 Consider a class of continuous functions $\mathcal{M}$ entropified with the squared error (as in Example 5.1, page 82). Let data $D = (x^n, y^n)$ and model $H \in \mathcal{M}$ be given. Denote the average squared error H makes on D by $\overline{\mathrm{ER}_{sq}}$. Using the code (5.46) for fixed β, we obtain as total description length of the y^n given the x^n:

$$L(y^n; H|x^n) = n(\overline{\mathrm{ER}_{sq}} + \frac{1}{2}\ln\frac{\pi}{\beta}) + L_{C_1}(H) \tag{5.47}$$

while using (5.46) for the entropified model (H, β) where $\beta = \hat{\beta}(D|H)$ is the parameter that maximizes the likelihood of D given H, we obtain:

$$L'(y^n; H|x^n) = \frac{1}{2}n(1 + \ln 2\pi + \ln\overline{\mathrm{ER}_{sq}}) + L_{C_1}(\beta) + L_{C_1}(H) \tag{5.48}$$

which depends logarithmically rather than linearly on the average error. (both equations can be easily verified by substituting $\beta = 1/2\sigma^2$ as in Example 5.1). When two-part code MDL is used, the MDL-optimal β_{mdl} for given D and fixed H will not be equal to $\hat{\beta}$, but nevertheless it will be reasonably close. $L_{C_1}(\beta)$ will be equal to $1/2\log n + c$ for some constant c (for both facts see Chapter 1, Section 1.4). This implies that there can very well be hypotheses H_1 and H_2 with a different number of parameters (so $L_{C_1}(H_1) \neq L_{C_2}(H_2)$) such that H_1 minimizes (5.47) while H_2 minimizes (5.48). In such a case, two-part code MDL based on the entropified model class leads to a different optimal H.

Using (5.46) with entropified model classes $\mathcal{M}_{\mathrm{ER}}$ allows the sum (5.46) (now with the additional term $L_{C_1}(\beta)$) to be interpreted as a code length. By *learning* the optimal value of β from the data (which is what entropification in the two-part code setting amounts to), we essentially choose the value that allows for the shortest code length of the data, which is in line with the general MDL philosophy. Moreover, since each model in $\langle\mathcal{M}\rangle_{\mathrm{ER}}$ corresponds to a code, we can also define stochastic complexity with respect to such model classes in the usual way and use it as a basis of model class selection; it allows us to compare different model classes based on different error functions for the same data, since the performance of all the classes are measured using the same criterion, namely, the codelength. This is once again in line with the general

MDL philosophy of using code lengths as a 'universal yardstick' [128], to be employed whenever different models or model classes are to be compared for the same data.

Another thing to be said for entropification is that it unifies different instantiations of MDL. In the existing literature on MDL, the question of how to code the data given an hypothesis has been given different answers depending on the category of model class used. For probabilistic model classes, generally the Shannon-Fano code with $L(D) = -\log P(D)$ is used [128, 14]. For *concept* classes (classes consisting of functions $\mathbf{E}_x \to \{0,1\}$; see Chapter 2, Example 2.4), the usual approach (see e.g. [122, 84]) has been to explicitly code the mistakes a hypothesis H makes on data D; see Example 5.21 below for details. This is different from all the versions of the two-part code we have seen before. For the case of non-probabilistic model classes with arbitrary error functions ER, there have been several proposals, as we saw above. Entropification (where data $D = (x^n, y^n)$ given hypothesis (H, β) is encoded using the code with lengths $-\log P(y^n|H, \beta, x^n)$) is an approach to handle non-probabilistic model classes that contains the existing treatments of probabilistic and concept classes as special cases. In the probabilistic case, as long as the model class is a full exponential family, then entropification will not change anything; we showed this in Example 5.4. In the case where $\mathcal{M}$ is a concept class, the code based on entropification with respect to the 0/1-error (Example 5.3), while superficially different, is essentially equivalent to the traditional approach of coding the mistakes H makes on D. We show this formally in Example 5.21 below. This suggests (but does not prove of course) that entropification can serve as the general 'preprocessing' tool to make a single version of two-part code MDL applicable to essentially arbitrary model classes.

Example 5.21 [concept learning and Bernoulli parameters II] Let $\mathcal{M}$ be a class of concepts (Chapter 2, Example 2.4) over $\mathbf{E} = \mathbf{E}_x \times \{0,1\}$ and let the observational data $D = (x^n, y^n)$. Two-part codes for concept classes are traditionally (e.g. in [122] and [84]) based on the following coding scheme: the x_i are regarded as given. The y_i are encoded by first encoding an hypothesis $H \in \mathcal{M}$ and then encoding the *exceptions* to H, which are all the indices i for which $y_i \neq H(x_i)$. We assume that hypotheses are encoded using some fixed code $C_1 : \mathcal{M} \to \mathbf{B}^*$. Clearly, given the x_i, H and the list of exceptions $M = \{i_1, \ldots, i_k\}$ we can fully reconstruct $y_1, \ldots, y_n$ (for x_i with $i \notin M$ we set $y_i = H(x_i)$; for x_i with $i \in M$ we set $y_i = |1 - H(x_i)|$).

If H makes k mistakes on a sample D of length n, there are $\binom{n}{k}$ different exception sets $M = \{i_1, \ldots, i_k\}$. Hence we need $\ln \binom{n}{k} + L(k)$ nats to encode all these mistakes. Here $L(k) = O(\ln k)$ equals the number of nats needed to encode k using some prefix code for the numbers $0, \ldots, n$ (note that k has to be encoded too to allow unique decoding). The total description length of the y_i given the x_i becomes:

$$L(y^n, H|x^n) = \ln \binom{n}{k} + L(k) + L_{C_1}(H) \tag{5.49}$$

Another way to arrive at a two-part code for the data would be to first entropify the concept class $\mathcal{M}$ with respect to the 0/1-error function and then to encode data by first coding some H, then some parameter β (using a fixed code C_1') and then encoding the y_i using the code based on $P(\cdot|H, \beta)$. This would take

$$L(y^n, H, \beta|x^n) = -\ln P(y^n|H, \beta, x^n) + L_{C_1'}(\beta) + L_{C_1}(H) \text{ nats.} \tag{5.50}$$

We proceed to show that (5.49) and (5.50) approximately coincide and hence that both ways of coding the data are essentially equivalent.

We saw in Example 5.3 (page 89) that, by substituting $\beta := \ln(1-\theta) - \ln\theta$, we find that the class $\langle \mathcal{M} \rangle_{\text{ER}_{01}}$ can equivalently be parameterized as follows: $\langle \mathcal{M} \rangle_{\text{ER}_{01}} = \{P(\cdot|H, \theta, \cdot) \mid H \in \mathcal{M}, 0 < \theta < 1\}$, such that, if $\text{ER}_{01}(y^n|H, x^n) = k$, then

$$P(y^n|H, \theta, x^n) = \theta^k (1-\theta)^{n-k} \tag{5.51}$$

Instead of coding β we can also code the θ corresponding to it. We can therefore rewrite (5.50) as follows:

$$L(y^n, H, \theta|x^n) = -\ln P(y^n|H, \theta, x^n) + L_{C_1'}(\theta) + L_{C_1}(H) \text{ nats.} \tag{5.52}$$

where $P(y^n|H, \theta, x^n)$ is given by (5.51). The ML estimator $\hat{\theta}$ maximizing (5.51) for fixed H and (x^n, y^n) is given by $\hat{\theta} = k/n$. Based on H and $\hat{\theta}$, the number of nats $-\ln P(y^n|H, \hat{\theta}, x^n)$ needed to code the data becomes

$$-\ln P(y^n|H, \hat{\theta}, x^n) = -\ln \hat{\theta}^k (1-\hat{\theta})^{n-k} \overset{(1)}{=} n\mathcal{H}(\hat{\theta}) \overset{(2)}{\approx} \ln \binom{n}{k}$$

where (1) follows by straightforward calculation and (2) by Stirling's approximation (3.15) (page 58) (or by applying (3.16) on page 58 with $\gamma = k/n$) (for precise bounds on $|n\mathcal{H}(\hat{\theta}) - \ln \binom{n}{k}|$ see [30], Chapter 12). Since $\hat{\theta} = k/n$ and hence, when n is known (as we assume in this example), can be reconstructed from k only, we need approximately $L(k)$ nats to describe $\hat{\theta}$, where $L(k)$ is defined as above. The total description length of the y_i then becomes:

$$-\ln P(y^n, H, \hat{\theta}|x^n) \approx \ln \binom{n}{k} + L(k) + L_{C_1}(H) \tag{5.53}$$

which is seen to coincide with (5.49). Hence if we code the data based on the entropified model class $\langle \mathcal{M} \rangle_{\text{ER}_{01}}$ and use the optimal β (corresponding to the optimal θ) for given D and H, then the number of bits we need coincides with the number of bits needed to efficiently encode the exceptions.

5.4.2 Worst-Case Optimality of the Shannon-Fano Code

The arguments given in the previous subsection suggest that entropification can serve as a general means to apply two-part code MDL to non-probabilistic model classes. Of course, they do not *prove* that entropification will be as well-behaved as either Barron's or Yamanishi's approach to this problem. Whether it really is or not is a challenging (and difficult) question at the time of writing this thesis. We did prove one small result though that provides some extra justification for entropification. This is discussed in the present subsection.

In Chapter 1 we saw that when a probabilistic model class is used in two-part code MDL, data is encoded by first encoding some model θ and then coding the data based on the *Shannon-Fano code* $L(x^n|\theta) = -\log P(x^n|\theta)$. At page 17 we already asked ourselves why to use this code and not any other one. There are many other possibilities; to give an example, we could map each model θ to the code with lengths $L'(x^n|\theta) = -\log(\sqrt{P(x^n|\theta)}/\sum_{z^n in \mathbf{E}^n} \sqrt{P(z^n|\theta)})$ which, by the Kraft Inequality (Theorem 1.6) also corresponds to a probability distribution over $\mathbf{E}^n$. Why does the Shannon-Fano code have a special status? In Chapter 1 we gave the justification that, using the Shannon-Fano code, the code length of the data precisely reflects the probability: if $P(D_1|\theta) = a \cdot P(D_2|\theta)$, then $L(D_1|\theta) = L(D_2|\theta) + \log a$.

Some authors prefer a different (or at any rate, additional) justification based on the Information Inequality (Equation 3.4 on page 52): if θ turns out to be the 'true' model, i.e. data is generated by repeated sampling from θ, then the *expected* code length $E_\theta[L(X^n)]$ is minimized if we set $L(x^n) := -\log P(x^n|\theta)$. By using the Shannon-Fano code, we map each model θ to the code that will be optimal if θ is actually true; hence it is the code that best 'suits' θ. This justification of the use of the Shannon-Fano code can be found in, for example [37, 162, 160].

We have always had some doubts about this argument, for two reasons: (a) it does not say anything about the (realistic) case where the model class contains models that allow us to compress the data (hence we can learn something about the data) yet none of these models are close to the 'true' one generating the data; (b) it is not clear how to extend this argument to non-probabilistic model classes.

Proposition 5.22 below shows how entropification allows us to extend the Shannon-Fano argument to a more general case which includes non-probabilistic model classes. Whereas we still assume the existence of *some* true probability distribution generating the data, we do not assume any more that it is contained in the model class $\mathcal{M}$ under consideration. For simplicity, we will consider only the unconditional, 'unsupervised' case, where we are interested in coding the complete outcomes (and not just y-values conditioned on x-values). Formally, we consider a class $\mathcal{M}$ of models and an error function $\text{ER} : \mathbf{E} \times \mathcal{M} \rightarrow \mathbf{R}$ (in case $\mathcal{M}$ is probabilistic we take ER to be the logarithmic error). We let $\theta = (H, \beta)$ index a model in $\langle \mathcal{M} \rangle$. Let $\mathcal{G}$ be the class of probability distributions P^* over $\mathbf{E}$ satisfying

$$E_{(H,\beta)}[\text{ER}(X|H)] = E_{P^*}[\text{ER}(X|H)]. \tag{5.54}$$

Let $\mathcal{L}$ be the class of all code length functions $L : \mathbf{E} \rightarrow \mathbf{R} \cup \{\infty\}$ satisfying the Kraft Inequality $\sum_{x \in \mathbf{E}} 2^{-L(x)} \leq 1$.

We are now ready to state our proposition. We discuss its implications after the proof.

Proposition 5.22 *Let $\mathcal{M}$, ER, $\mathcal{G}$, θ and $\mathcal{L}$ be as above. Let $L(\cdot|\theta) \in \mathcal{L}$ be the code length function of the Shannon-Fano code for $\theta = (H, \beta)$, restricted to one outcome $x \in \mathbf{E}$. That is,*

$$L(x|H, \beta) = -\log P(x|H, \beta) = \beta\text{ER}(x|H) + \ln Z_H(\beta)$$

We have:

1.

$$L(\cdot|\theta) = \arg\inf_{L \in \mathcal{L}} \sup_{P^* \in \mathcal{G}} E_{P^*}[L(X|\theta)] \tag{5.55}$$

That is, $L(\cdot|\theta)$ gives the shortest worst-case expected code lengths, the worst-case being taken over all distributions satisfying (5.54).

2. *Let, for given H, $\mathbf{U}_H$ be the smallest interval such that $\forall x : \text{ER}(x|H) \in \mathbf{U}_H$. For every $H \in \mathcal{M}$ and for every $P^* \in \mathcal{G}$ for which $E_{P^*}[\text{ER}(X|H)]$ lies in the interior of $\mathbf{U}_H$, there exists a β such that (5.54) holds.*

Proof: Define $t := E_{(H,\beta)}[\text{ER}(X|H)] = E_{P^*}[\text{ER}(X|H)]$. As is clear from our regularity conditions for error functions (page 87), the probability distribution $P(\cdot|H,\beta)$ is the maximum entropy distribution for constraint $E[\text{ER}(X|H)] = t$. By Figure 3.1 on page 62 we have that $E_{P^*}[L(X|H,\beta)] = E_{(H,\beta)}[L(X|H,\beta)] = \mathcal{H}(H,\beta)$ for every $P^* \in \mathcal{G}$. On the other hand, let $L' \in \mathcal{L}$ be a code length function different from $L(\cdot|H,\beta)$. By Figure 3.1 there exists a $P^* \in \mathcal{G}$ (namely, $P^* = P(\cdot|H,\beta)$) such that $E_{P^*}[L'(X)] > \mathcal{H}(H,\beta)$. This proves (1).

To prove (2), note that the class of maximum entropy models for function $\text{ER}(X|H)$ coincides with the class of models in $\langle\mathcal{M}\rangle_{\text{ER}}$ restricted to fixed H. Let us denote this subclass by $\mathcal{M}_{me}$. By Proposition 3.9 (Chapter 3, page 56), for each t in the interior of $\mathbf{U}$, $\mathcal{M}_{me}$ contains a model satisfying $E[\text{ER}(X|H)] = t$. This proves (2). □

This proposition shows that the Shannon-Fano code for models θ in entropified model classes (a) leads to codes that are worst-case optimal if the probability distribution θ is 'true' only in the sense that its expectation of the error coincides with the true expectation of error, and (b) that entropified model classes *always* (except possibly for P^* with expected errors at the boundaries of the error space) contain a model that is 'true' in this weak respect.

If one uses a non-probabilistic model class $\mathcal{M}$, one usually does not have a clear idea about the distribution generating the data. If one is at all willing to assume that such a distribution nevertheless exists[1], then it seems reasonable to make as few assumptions as possible about it. This directly leads to our worst-case scenario, which really says that *every* i.i.d. distribution is a possible candidate for generating the data. That is why we regard this proposition as justifying the use of the Shannon-Fano code for the entropification (probabilistic version) of $\mathcal{M}$.

We hasten to add though that there do exist codes (based on non-i.i.d. model classes) whose expected code lengths under every P^* are arbitrarily close to that of the Shannon-Fano code. An example is the code based on the universal computer language that was discussed in Chapter 1, Section 1.1.2.

The proposition also has something to say about the case where $\mathcal{M}$ itself is probabilistic and we entropify with respect to the logarithmic error ER_{lg}. If $\mathcal{M}$ is itself an exponential family, this will not change $\mathcal{M}$ (Example 5.4), and the proposition tells us

[1] Of course, one may argue that if one uses a non-probabilistic model class, then the data generating process may be inherently non-probabilistic. But then it becomes very hard to prove anything about it. In any case, the present setting (assuming an underlying distribution while working with non-probabilistic model classes) is taken for granted by many researchers [167, 13, 69].

that the Shannon-Fano code for $\mathcal{M}$ is optimal not only in the case that data is generated by one of the models in $\mathcal{M}$, but also in the case that it is generated by some i.i.d. model not in $\mathcal{M}$. If $\mathcal{M}$ is not an exponential family, then the usual optimality of the Shannon-Fano code holds for the models in $\mathcal{M}$, while the 'worst-case' optimality holds, by Proposition 5.22, for the models in $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$. Whether one should entropify or not then depends on whether one thinks that one of the models in the class will be very close to being 'truely a true model': if one entropifies, one adds an extra dimension to the parameter space. This can lead to logarithmically larger code lengths; if $\mathcal{M}$ contains the true model, then it will lead even with probability 1 to larger codelengths of the data, when data is encoded using either the stochastic complexity or the two-part MDL code. However, if the true model is not in $\mathcal{M}$, then using $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$ instead of $\mathcal{M}$ can sometimes lead to a linear *decrease* in codelengths. We briefly show why.

To see that if $\mathcal{M}$ contains the true model, one will need more bits to encode the data based on $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$ rather than $\mathcal{M}$, we use a result by Clarke and Barron [27]. They proved an analogue of the asymptotic expansion of stochastic complexity (Chapter 2, Section 2.6) for the case where data is distributed according to one of the models in a (probabilistic) model class $\mathcal{M}$. Let $\mathcal{M}$ be a probabilistic model class consisting of i.i.d. models. Clarke and Barron showed that, if the data are generated by one of the models θ^* in $\mathcal{M}$, then under some mild further conditions on $\mathcal{M}$,

$$L_{sc}(x^n|\mathcal{M}) = -\log P(x^n|\theta^*) + \frac{k}{2}\log n + O(1) \tag{5.56}$$

with θ^*-probability 1. Here $L_{sc}(x^n|\mathcal{M})$ is the stochastic complexity of x^n with respect to $\mathcal{M}$ and k is the number of parameters needed for parameterizing $\mathcal{M}$. By the results discussed in Chapter 2, Section 2.6, the two-part code length is within $O(1)$ of the stochastic complexity. Observe that if the true model θ^* is in $\mathcal{M}$, then it is also in $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$; therefore, by (5.56), using $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$, the number of parameters k is increased by 1 which results in a logarithmic increase in codelength (with probability 1).

If θ^* is not in $\mathcal{M}$ then, supposing $\mathcal{M}$ is finitely parameterized by $\Gamma \in \mathbf{R}^k$, the asymptotic expansion of both stochastic complexity and two-part code (Chapter 2, Section 2.6) gives

$$L_{sc}(x^n|\mathcal{M}) = -\log P(x^n|\hat{\theta}(x^n)) + \frac{k}{2}\log n + O(1)$$

By applying Lemma 5.14, one sees that with θ^*-probability 1,

$$\lim_{n\to\infty} -\frac{1}{n}L_{sc}(x^n|\mathcal{M}) = \min_{\theta\in\Gamma} E_{\theta^*}[-\ln P(X|\theta)]$$

This will hold for both the original model class $\mathcal{M}$ and its entropified version $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$ (in the latter case, the values in β have to be restricted to a compact set). By Proposition 5.22, there exist θ^* such that

$$\min_{\theta\in\langle\Gamma\rangle_{\mathrm{ER}_{lg}}} E_{\theta^*}[-\ln P(X|\theta)] < \min_{\theta\in\Gamma} E_{\theta^*}[-\ln P(X|\theta)],$$

where we used $\langle\Gamma\rangle_{\mathrm{ER}_{lg}}$ to denote the parameterization of $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$. In such a case, both the two-part and the stochastic complexity code based on $\langle\mathcal{M}\rangle_{\mathrm{ER}_{lg}}$ will clearly achieve more compression (by a linear amount) than the codes based on $\mathcal{M}$. Since $\mathcal{M} \subset \langle \mathcal{M} \rangle_{\text{ER}_{lg}}$, the opposite event (the code based on $\mathcal{M}$ achieving a linear gain in compression compared to the code based on $\langle M \rangle_{\text{ER}_{lg}}$) has zero probability for any θ^*.

5.5 Conclusion

We have introduced the concept of 'entropification' and shown how it can be used in the context of estimating prediction error and in the context of MDL. We leave detailed conclusions to the Epilogue to Part I of this thesis (page 119), where we discuss how the results obtained in this chapter can be used to partially resolve the problematic issues concerning MDL that were raised in the Introduction and in Chapters 1-3.

5.6 Appendix: Entropification and Maximum Entropy Model Classes

In this appendix we prove the properties of entropification that we presented in Section 5.2.3 (Equations 5.18 and 5.19; Propositions 5.6-5.9).

Let $\mathcal{M}$ be a class of models, ER be an error function and $\langle \mathcal{M} \rangle_{\text{ER}}$ be the entropification of $\mathcal{M}$. $\langle H \rangle_{\text{ER}}$, the subclass of models from $\mathcal{M}$ restricted to fixed H (i.e. $\langle H \rangle_{\text{ER}} := \{(H, \beta) \mid (H, \beta) \in \langle \mathcal{M} \rangle_{\text{ER}}\}$) is essentially a maximum entropy model class. The properties we want to show would follow immediately if it were really such a class. However, since $\langle H \rangle_{\text{ER}}$ is a class of *conditional* models, we need to use a trick: we extend $\langle \mathcal{M} \rangle_{\text{ER}}$ to a class of distributions over $\mathbf{E}_x \times \mathbf{E}_y$ by supplying it with the uniform distribution over $\mathbf{E}_x$; this distribution exists since we assume $\mathbf{E}_x$ to be either finite or compact. As will be shown below, the resulting model class, which we denote by $\langle \mathcal{M} \rangle^u_{\text{ER}}$, *is* a maximum entropy model class. We then use standard results about such classes to prove certain properties for $\langle \mathcal{M} \rangle^u_{\text{ER}}$, and we then show that if these properties hold for $\langle \mathcal{M} \rangle^u_{\text{ER}}$, they must also hold for the class of conditional distributions $\langle \mathcal{M} \rangle_{\text{ER}}$. This will be done in Lemma 5.23 below. After having proved the lemma, we will show that the properties presented in the main text follow as immediate corollaries from this lemma.

Lemma 5.23 *Let* $\mathbf{E} = \mathbf{E}_x \times \mathbf{E}_y$ *and let* ER : $\mathbf{E} \times \mathcal{M} \rightarrow \mathbf{R} \cup \{\infty\}$ *be an error function. Let* $P^u(\cdot)$ *be the uniform distribution over* $\mathbf{E}_x$. *Let* $\langle \mathcal{M} \rangle^u_{\text{ER}}$ *be the class of probabilistic models* $P^u(\cdot, \cdot | H, \beta)$ *where, for each* (H, β) *in* $\mathcal{M}$, *for each* (x^n, y^n), $P^u(x^n, y^n | H, \beta) = P(y^n | H, \beta, x^n) P^u(x^n)$. *We have:*

1. *There exists a constant* $c \in \mathbf{R}$ *such that for all* n, *all* $(x^n, y^n) \in \mathbf{E}^n$,

$$P(y^n | \beta, H, x^n) \cdot c^n = P^u(x^n, y^n | \beta, H) \tag{5.57}$$

Let, for fixed $H \in \mathcal{M}$, $\langle H \rangle^u_{\text{ER}} = \{(H, \beta) | (H, \beta) \in \langle \mathcal{M} \rangle^u_{\text{ER}}\}$ *be the restriction of* $\langle \mathcal{M} \rangle^u_{\text{ER}}$ *to models with fixed* H.

2. *$\langle H\rangle^u_{\text{ER}}$ is the maximum entropy model class for the function $\phi(x,y) := \text{ER}(y|H,x)$ with range $\mathbf{U}$. Here $\mathbf{U}$ is the smallest (open or closed) interval in $\mathbf{R}$ such that $\forall(x,y) \in \mathbf{E} : \phi(x,y) \in \mathbf{U}$.*

Let x be an arbitrary element of $\mathbf{E}_x$. Let $P(\cdot|H,\beta,x)$ be the distribution over $\mathbf{E}_y$ given by $P(y|H,\beta,x) = Z_H(\beta)^{-1}\exp(-\beta\text{ER}(y|H,x))$ and let $\mathcal{H}(P(\cdot|H,\beta,x))$ stand for the entropy of the distribution $P(\cdot|H,\beta,x))$.

3. *We have for all (H,β):*

$$E_{P^u(\cdot,\cdot|H,\beta)}[\text{ER}(Y|H,X)] = E_{P(\cdot|H,\beta,x)}[\text{ER}(Y|H,X)|X=x] = = -\frac{\partial}{\partial\beta}\ln Z_H(\beta) \quad (5.58)$$

4. *We also have for all (H,β):*

$$\mathcal{H}(P(\cdot|H,\beta,x)) = \mathcal{H}(P^u(\cdot,\cdot|H,\beta)) + \ln c \quad (5.59)$$

Proof: Item (1) is straightforward. Item (2) follows directly by our assumptions on ER (see the remark at the beginning of Section 5.2, page 87) and the definition of $\langle\mathcal{M}\rangle_{\text{ER}}$ (Definition 5.2). To prove item (3), note that for each $P(\cdot|H,\beta,\cdot)$ the corresponding unconditional model $P^u(\cdot,\cdot|H,\beta)$ is given by

$$P^u(x,y|H,\beta) = \frac{c}{Z_{H(\beta)}}\exp(-\beta\text{ER}(y|H,x))$$

Since $\langle\mathcal{M}\rangle^u_{\text{ER}}$ is a maximum entropy model class, we can apply Proposition 3.9 on page 56 to get:

$$E_{P^u(\cdot,\cdot|H,\beta)}[\text{ER}(Y|H,X)] = -\frac{\partial}{\partial\beta}(\ln Z_H(\beta) - \ln c) = -\frac{\partial}{\partial\beta}\ln Z_H(\beta) \quad (5.60)$$

Now choose an arbitrary $x \in \mathbf{E}_x$. Let $\langle H(x)\rangle_{\text{ER}}$ be the class of models containing, for each $\beta \in \Gamma_{nat}(H)$, the distribution $P(\cdot|H,\beta,x)$ defined as in the statement of item (3) of the lemma. $\langle H(x)\rangle$ is a class of maximum entropy models for function $\psi(y) := \text{ER}(y|H,x)$ (note that ψ is a function of y only; x is kept fixed). This is straightforward to verify by our assumptions on ER, see page 87. We can therefore apply Proposition 3.9 on page 56 to give us:

$$E_{P(\cdot|H,\beta,x)}[\text{ER}(Y|H,X)|X=x] = -\frac{\partial}{\partial\beta}\ln Z_H(\beta) \quad (5.61)$$

(5.60) and (5.61) coincide. Since we picked x arbitrarily, (5.58) follows.

Now for item (4). By straightforward calculation we see that the entropy of $P(\cdot|H,\beta,x)$ is equal to

$$\beta E_{P(\cdot|H,\beta,x)}[\text{ER}(Y|H,X)|X=x] + \ln Z(\beta).$$

while the entropy of $P(\cdot,\cdot|H,\beta)$ is given by

$$\beta E_{P(\cdot,\cdot|H,\beta)}[\text{ER}(Y|H,X)] + \ln Z(\beta) - \ln c.$$

Together with (5.58) in item (3) of the proposition, equality (5.59) follows. □

Having proven Lemma 5.23, we can proceed to prove the properties stated in Section 5.2.3.

Proving the Properties

We first note that Equations 5.18 and 5.19 (page 92) follow trivially from items 3 and 4 of Lemma 5.23. This leaves us with propositions 5.6, 5.8 and 5.9, which we now restate and prove.

Proposition 5.6 *Let $D = (x^n, y^n)$. For each $H \in \mathcal{M}$, $E_{(H,\beta)}[\text{ER}(Y|H,X)]$ as a function of β is continuous. Moreover,*

$$E_{(H,\hat{\beta}(D|H))}[\text{ER}(Y|H,X)] = \frac{1}{n}\sum_{i=1}^{n} \text{ER}(y_i|H,x_i) \tag{5.62}$$

Proof: Let $H \in \mathcal{M}$ be fixed. By Lemma 5.23, item (1), the model $P^u(\cdot,\cdot|H,\hat{\beta})$ that maximizes, for fixed H, the likelihood of D within the class of unconditional models $\langle H\rangle^u_{\text{ER}}$ (as defined in Lemma 5.23) is indexed by the same value $\hat{\beta}$ as the model $P(\cdot|H,\hat{\beta},\cdot)$ that maximizes the likelihood of D within the class of conditional models $\langle H\rangle_{\text{ER}}$. Also by Lemma 5.23, $\langle H\rangle^u_{\text{ER}}$ is a maximum entropy model class. Therefore, using Proposition 4.7 about the reliability of estimates for maximum entropy model classes, we have that the expectation under the unconditional model indexed by H and $\hat{\beta}(D|H)$ is equal to the average over data D:

$$E_{P^u(\cdot,\cdot|H,\hat{\beta}(D|H))}[\text{ER}(Y|H,X)] = \frac{1}{n}\sum_{i=1}^{n} \text{ER}(y_i|H,x_i) \tag{5.63}$$

By Lemma 5.23, item 3 this shows (5.62). □

Proposition 5.8 *For all $\beta \in \Gamma_{nat}(H)$ with $\beta > 0$, the entropy $\mathcal{H}(H,\beta)$ is a strictly decreasing function of β. For all $\beta < 0$, $\mathcal{H}(H,\beta)$ is a strictly increasing function of β.*

Proof: We know from Lemma 5.23, item (4) that $\mathcal{H}(H,\beta) = \ln c + \mathcal{H}(P^u(\cdot,\cdot|H,\beta))$ for some constant c. The second term in this expression stands for the entropy of a maximum entropy distribution with parameter β (Lemma 5.23, item (2)). Hence we can apply Proposition 3.9 of page 56, item 2(b) (which says that for maximum entropy distributions, the entropy is a strictly increasing (decreasing) function for $\beta > 0$ ($\beta < 0$)) and the result follows. □

Let, for fixed H, $\mathbf{U}_H$ be the smallest (possibly unbounded) interval in $\mathbf{R}$ such that $\forall (x,y) \in \mathbf{E} : \text{ER}(y|H,x) \in \mathbf{U}_H$.

Proposition 5.9 $E_{(H,\beta)}[\text{ER}(Y|H,X)]$ *is a strictly decreasing function of* β. *For each* t *in the interior of* $\mathbf{U}_H$ *there exists a unique value of* β *such that* $E_{(H,\beta)}[\text{ER}(Y|H,X)] = t$.

Proof: We know from Lemma 5.23, item (3) that for all $\beta \in \Gamma_{nat}(H)$, $E_{(H,\beta)}[\text{ER}(Y|H,X)] = E_{P^u(\cdot,\cdot|H,\beta)}[\text{ER}(Y|H,X)]$. Here (Lemma 5.23, item (2)) $P^u(\cdot,\cdot|H,\beta)$ is a maximum entropy distribution with natural parameter β. Hence we can apply Proposition 3.9 of page 56. The first part of the proposition above follows immediately from item 2(a) of Proposition 3.9 (which says exactly that $E_{P^u(\cdot,\cdot|H,\beta)}[\text{ER}(Y|H,X)]$ is a strictly decreasing function of β). The second part of the proposition above follows immediately from item 5 of Proposition 3.9. □

5.7 Appendix: Proof of Lemma 5.14

In this appendix we prove Lemma 5.14, which is restated below. Essentially, it says the following: if $\mathcal{M}$ is compactly parameterized, then the average code length of x^n based on the maximum likelihood model for x^n converges (with probability 1) to the expected code length based on the model in the class that minimizes this expected code length. This is reminiscent of the 'uniform laws of large numbers' that are central to Vapnik's work [159]. The exact connection needs to be further investigated.

Lemma 5.14 *Let* $\mathcal{M} = \{P(\cdot|\theta) \mid \theta \in \Gamma\}$ *be a class of i.i.d. probabilistic models over sample space* $\mathbf{E}$ *that is finitely parameterized by* $\Gamma \subset \mathbf{R}^k$ *where* Γ *is compact. Let the data be i.i.d. according to some* P^* *satisfying Definition 5.13. Then the following minima exist for all* n*, all* $x^n \in \mathbf{E}^n$*:*

$$\hat{L}(x^n) := \min_{\theta\in\Gamma}\{-\ln P(x^n|\theta)\} \tag{5.64}$$

$$\widetilde{L}(P^*) := \min_{\theta\in\Gamma}\{E_{P^*}[-\ln P(X|\theta)]\} \tag{5.65}$$

We have with P^**-probability 1:*

$$\lim_{n\to\infty}\frac{1}{n}\hat{L}(x^n) = \widetilde{L}(P^*) \tag{5.66}$$

Proof: In the proof we assume that $\mathbf{E}$ is continuous. Adaption to the case of $\mathbf{E}_{P^*} = \mathbf{E}_{P^*,1} \times \ldots \times \mathbf{E}_{P^*,m}$ where some of the $\mathbf{E}_{P^*,i}$ are discrete is completely straightforward.

By compactness of $\mathbf{E}_{P^*}$ and Γ and the fact that we only consider P^* with associated density functions (see Chapter 1, Section 1.2) the minima (5.64) and (5.65) evidently exist. We can cover Γ with a grid of k-dimensional rectangles with side width s. The set Γ is thus partitioned into a finite number, say M, of rectangles R_i. Let, for $1 \leq i \leq M$, θ^i be the model in Γ corresponding to the center of R_i. In this way we obtain a reduced parameter set $\Gamma_s = \{\theta^1, \ldots, \theta^M\}$.

We first consider the simple case where $\mathcal{M}$ is such that the following four minima are all attained by a unique value for each n, $x^n \in \mathbf{E}^n$:

$$\begin{aligned}
\tilde{\theta} &= \arg\min_{\theta\in\Gamma}\{E_{P^*}[-\ln P(X|\theta)]\} \\
\tilde{\theta}_s &= \arg\min_{\theta_s\in\Gamma_s}\{E_{P^*}[-\ln P(X|\theta_s)]\} \\
\hat{\theta}(x^n) &= \arg\min_{\theta\in\Gamma}\{-\ln P(x^n|\theta)\} \\
\hat{\theta}_s(x^n) &= \arg\min_{\theta_s\in\Gamma_s}\{-\ln P(x^n|\theta_s)\}
\end{aligned} \tag{5.67}$$

We now show in two stages that (5.66) holds in the case where these single minimizing values exist.

Stage 1 Let n and $\epsilon > 0$ be given. We claim that if we pick the rectangle side width s small enough both of the following equations will hold:

$$|E_{P*}[-\ln P(X|\tilde{\theta}_s)] - E_{P*}[-\ln P(X|\tilde{\theta})]| < \frac{1}{3}\epsilon \tag{5.68}$$

$$|-\frac{1}{n}\ln P(x^n|\hat{\theta}_s(x^n)) + \frac{1}{n}\ln P(x^n|\hat{\theta}(x^n))| < \frac{1}{3}\epsilon \qquad \text{for all } x^n \in \mathbf{E}^*_{P^*} \tag{5.69}$$

We first show (5.69). Let

$$f_n(\theta, x^n, \theta_0) := \\ -\frac{1}{n}\ln P(x^n|\theta) + \frac{1}{n}\ln P(x^n|\theta_0) = \frac{1}{n}\sum_{i=1}^{n}[-\ln P(x_i|\theta) + \ln P(x_i|\theta_0)] \tag{5.70}$$

$\mathcal{M}$ is finitely parameterized by Γ. Checking Definition 2.7 on page 35, we see that for all $x \in \mathbf{E}_{P^*}$, $f_1(\theta, x, \theta_0)$ regarded as a function of θ must be continuous at all $\theta \in \Gamma$. By compactness of $\mathbf{E}_{P^*}$ it is easy to show that

$$\begin{aligned}
f_{\max}(\theta_0,\theta) &:= \max_{x\in\mathbf{E}_{P^*}} f_1(\theta, x, \theta_0) \quad \text{and} \\
f_{\min}(\theta_0,\theta) &:= \min_{x\in\mathbf{E}_{P^*}} f_1(\theta, x, \theta_0)
\end{aligned}$$

are well-defined continuous functions of θ with $f_{\max}(\theta_0, \theta_0) = f_{\min}(\theta_0, \theta_0) = 0$. By compactness of Γ, it is clear that the following function is well-defined for all $\delta > 0$:

$$g_{\max}(\delta) := \max_{\mathcal{U}(\delta)}\{\ |f_{\max}(\theta_0, \theta)|\ \}, \tag{5.71}$$

where the maximum is taken over the set $\mathcal{U}(\delta) = \{\theta, \theta_0 \in \Gamma \mid |\theta - \theta_0| \leq \delta\}$. Moreover (compactness of Γ) one can easily show that $\lim_{\delta\downarrow 0} g_{\max}(\delta) = g_{\max}(0) = 0$. The same holds for $g_{\min}(\delta)$ which we define analogously to $g_{\max}$. These properties of $g_{\max}$ and

$g_{\min}$ show the following: for every $\epsilon > 0$, we can pick the rectangle width s small enough such that the following implication holds for all $\theta, \theta_0 \in \Gamma$ and all $x^n \in \mathbf{E}^n_{P^*}$:

if θ and θ_0 both fall in the same rectangle R_i *then*

$$f_{\max}(\theta_0, \theta) < \epsilon/3 \text{ and } f_{\min}(\theta_0, \theta) > -\epsilon/3 \quad (5.72)$$

It can be seen from (5.70) that for $n \geq 1$, for all $x^n \in \mathbf{E}_{P^*}$,

$$f_{\min}(\theta_0, \theta) \leq f_n(\theta, x^n, \theta_0) \leq f_{\max}(\theta_0, \theta) \quad (5.73)$$

(5.69) now follows by combining (5.72) and (5.73) and substituting $\hat{\theta}(x^n)$ for θ_0.

A similar but simpler argument shows that (5.68) holds. We omit the details.

Stage 2 By the strong law of large numbers [47], we have with P^*-probability 1 that for all $\theta_s \in \Gamma_s$, for all $\delta > 0$, there exists an n_0 such that $n \geq n_0 \Rightarrow | - n^{-1} \sum \ln P(x^n|\theta_s) - E_{P^*}[-\ln P(X|\theta_s)]| < \delta$. Since Γ_s contains only a finite number of elements, this implies that, for all $\epsilon > 0$, with P^*-probability 1 there exists an n_0 such that *for all* $\theta_s \in \Gamma_s$:

$$n \geq n_0 \quad \Rightarrow \quad | - \frac{1}{n} \ln P(x^n|\theta_s) - E_{P*}[-\ln P(X|\theta_s)]| < \frac{1}{3}\epsilon \quad (5.74)$$

In addition, we also have for all $x^n \in \mathbf{E}^*$:

$$\begin{aligned} E_{P^*}[-\ln P(X|\hat{\theta}_s(x^n))] \quad &\geq \quad E_{P^*}[-\ln P(X|\tilde{\theta}_s)] && (5.75) \\ -\frac{1}{n} \ln P(x^n|\hat{\theta}_s(x^n)) \quad &\leq \quad -\frac{1}{n} \ln P(x^n|\tilde{\theta}_s) && (5.76) \end{aligned}$$

By first applying (5.74) with $\theta_s := \hat{\theta}_s(x^n)$ and then (5.75) we find

$$-\frac{1}{n} \ln P(x^n|\hat{\theta}_s(x^n)) \geq E_{P^*}[-\ln P(X|\tilde{\theta}_s)] - \frac{1}{3}\epsilon. \quad (5.77)$$

Using (5.76) and then applying (5.74) with $\theta_s := \tilde{\theta}_s$ we find

$$-\frac{1}{n} \ln P(x^n|\hat{\theta}_s(x^n)) \leq E_{P^*}[-\ln P(X|\tilde{\theta}_s)] + \frac{1}{3}\epsilon. \quad (5.78)$$

Combining (5.68), (5.69), (5.77) and (5.78) we find that for all $\epsilon > 0$, there exists an n_0 such that with P^*-probability 1, for all $n \geq n_0$,

$$| - \frac{1}{n} \ln P(x^n|\hat{\theta}(x^n)) - E_{P^*}[-\ln P(X|\tilde{\theta})]| < \epsilon \quad (5.79)$$

which is equivalent to (5.66). This proves the lemma for the case that the four minima in (5.67) are all attained by single values in the parameter space.

If this is not the case, we proceed as follows: by compactness all minima exist; the only problem is that they may be attained for several values. This can be handled by defining $\widetilde{\Theta}$ as the set of all θ minimizing $E_{P^*}[-\ln P(X|\theta)]$ (analogously to the first line of (5.67)) and defining $\widetilde{\Theta}_s$, $\widehat{\Theta}$ and $\widehat{\Theta}_s$ similarly. By the same reasoning as in Stage 1 of the proof, we can now prove the following existentially quantified version of (5.68) and (5.69): 'Let n and $\epsilon > 0$ be given. We claim that if we pick the rectangle side width s small enough then there exist $\tilde{\theta}_s \in \widetilde{\Theta}_s$, $\tilde{\theta} \in \widetilde{\Theta}$, $\hat{\theta}_s(x^n) \in \widehat{\Theta}_s$ and $\hat{\theta}(x^n) \in \widehat{\Theta}$ such that (5.68) and (5.69) hold'. By the same reasoning as in Stage 2 of the proof, we can also prove a universally quantified version of (5.68) and (5.69): 'if n is large enough, then for all $\hat{\theta}_s(x^n) \in \widehat{\Theta}_s$ and all $\tilde{\theta}_s \in \widetilde{\Theta}_s$ equations (5.77) and (5.78) hold with P^*-probability 1.' Combining these new versions of (5.68), (5.69), (5.77) and (5.78) we can proceed as above to show that (5.79) holds. The lemma then follows. $\square$

Epilogue: Using Models in a Careful Way

In this Epilogue to Part I of the thesis we study whether we have resolved the problematic issues concerning MDL raised in the Introduction and in Chapters 1-3. Briefly, these were:

1. Can we use models that are partially wrong to give reasonable predictions of future data? If so, what can we do with them and what not? ('What does a model say about the world?' as we wrote on page 28).

2. What to do if we are asked to use our model to predict against various loss functions? (Chapter 2, Section 2.2)

3. When using a probabilistic model class, why should we use the Shannon-Fano code $L(D|\theta) = -\log P(D|\theta)$ to encode the data with the help of model? When using a non-probabilistic model class, why should we use the code for which $L(D|H) = \text{ER}(D|H) + K$ for all D (Chapter 2, Section 2.2)?

Can we use models that are partially wrong? From the point of view of the MDL philosophy, we choose a model class $\mathcal{M}$ because we think it will help us to capture some of the regularity inherent in the data - but we have no hope that it will capture all (except if $\mathcal{M}$ corresponds to the class of all computer programs written in some language- but then the inference process becomes noncomputable). In Chapter 1 and in the Introduction to Part I of this thesis, we argued that this is the situation we will usually be in: all our models will always, to some extent, be wrong; therefore, though the question *came up* in connection with the MDL philosophy, it should be relevant not only to MDL, but to all statistical inference procedures.

Once we accept the fact that our model H for data D will always be partially wrong, we are faced with the question of what can be reliably inferred from such a model and what not. In Chapter 5, we showed that we can always change our model classes in such a way that we can reliably estimate the prediction error over future data. We proved (in Section 5.3) that this will lead, with high probability, to accurate estimates of error over future data even if data are independently drawn according to a distribution that is completely different from our model. Hence, if we are willing to make the i.i.d. assumption, then as long as we measure the error our model makes when predicting future data using the same error function as the one that was used in inferring the model from the data, we can use the model H reliably: even though it is partially wrong, it will give a correct impression of how accurate it is in predicting future data. Note however, that this result was proved *only* under the i.i.d. assumption[2] – so we still have to assume something about 'the truth out there'. Hence, we have not resolved in general whether using an overly simple (that is, partially wrong) model can lead to 'disastrous' results, as was feared by the statistician mentioned in the Introduction (page 3). We have only shown that prediction errors may be accurately estimated under much wider assumptions than the classical assumption that one's model class contains the true model. The question remains of whether this is not too weak; whether we will not always be interested in estimating more aspects about the data than just their prediction error. The example below shows at the same time that sometimes 'reliable' predictions are enough, while 'unreliable' predictions can lead to very misleading results.

Example 5.24 [Classification] Recall that in concept learning the model class consists of functions $H : \mathbf{E}_x \to \{0, 1\}$. Frequently the goal will be to use the concept $\ddot{H}$ learned on the basis of data D to *classify* new data: one is given a value $x \in \mathbf{E}_x$ and one has to predict the corresponding $y \in \{0, 1\}$. Suppose one uses a class of concepts $\mathcal{M}$ entropified with the 0/1-error for this. Suppose further that, for given data D, the estimate inferred is $(\ddot{H}, \ddot{\theta})$. In Example 5.3 on page 89 it was shown that this estimate says that 'if the model $\ddot{H}$ is used, then the probability of making a wrong prediction is $\ddot{\theta}$. Let us assume that data are i.i.d. according to a model P^* (Definition 5.13) so that Theorem 5.17 applies; this theorem tells us that if D is large enough and one uses a reasonable estimation procedure (Definition 4.3), then $\ddot{\theta} \approx P^*\{\ddot{H}(X) \neq Y\}$ (see Example 5.3). This means that if we are only interested in *classifying* future data, the component $\ddot{\theta}$ of our model $(\ddot{H}, \ddot{\theta})$ will give us a good idea on just how well we can do that. Hence if we use our model only for classification, we will have neither an overly optimistic nor an overly pessimistic idea of how good our model is at this task.

Now let us consider a specific example where the model $(\ddot{H}, \ddot{\theta})$ as issued by our estimation procedure on the basis of a large data set $D = (x^n, y^n)$ has $\ddot{\theta} = .95$. By Theorem 5.17 this means that, if D is really large, we can predict future data with 95% accuracy. However, it may be the case that, for the x_i where $\ddot{H}(x_i) = 1$, y_i is always equal to 1 while for the cases where $\ddot{H}(x_i) = 0$, $y_i \neq \ddot{H}(x_i)$ half the time. If $\ddot{H}(x_i) = 1$ in 90% of the cases, then we will have $\ddot{\theta} \approx .95$ while, if a new value x is given such that $\ddot{H}(x) = 0$ and we use $\ddot{H}(x)$ for prediction, we will only be right in about 50% of the cases. Hence our model is very bad for new data with $\ddot{H}(x) = 0$, and if a loss function is used such that predicting a 'false zero' leads to a much higher loss than when predicting a 'false one', then our model will really be quite worthless. Prediction against such a loss function is not reliable - and indeed, if we stick to 'safe statistics', we are not allowed to make such a prediction. We used an extreme example, but similar examples, where there exists a very simple rule that gives accurate predictions for a large subset of the x_i while being quite bad on the remaining x_i do occur in practice.

---

[2] One might conjecture though that the results can in some cases be extended to non-i.i.d. sources. For example, in the case of stationary ergodic non-i.i.d. sources the law of large numbers, which was the key to Theorems 5.16-5.18 remains applicable.

The 'naive Bayes' model class that we will use in the next chapter sometimes yields exactly such models based on real-world data sets.

This automatically brings us to the next question:

Predicting against arbitrary loss functions If we use an entropified model class $\langle\mathcal{M}\rangle_{\text{ER}}$, and we want to use our estimate $(\ddot{H},\ddot{\beta})$ to make predictions or decisions against loss functions LOSS that can*not* be written as a linear combination of ER we see that, at least if the sample space $\mathbf{E}$ is discrete, this will often still work – but it will also often be *unreliable*. That is, because the model class $\langle\mathcal{M}\rangle_{\text{ER}}$ restricted to models with fixed H is essentially a maximum entropy model class, one can apply the concentration phenomenon (Chapter 3, Section 3.5). In this case, it tells us that for an exponentially large majority of those data sets to which $(\ddot{H},\ddot{\beta})$ gives a good fit, the frequencies $(\gamma_1,\ldots,\gamma_k)$ will be approximately equal to the probabilities $P(1|\ddot{H},\ddot{\beta}),\ldots,P(k|\ddot{H},\ddot{\beta})$. If future data indeed belongs to this majority, then the average of every function (hence also every loss function) over future data will be approximately equal to its expectation over $(\ddot{H},\ddot{\beta})$, and the predictions will be accurate. Only in the few cases where the frequencies and the probabilities do not coincide will the predictions not be accurate - nevertheless, as we saw in the example above, these cases may certainly occur.

The Universal Yardstick This leaves us with the question of justifying the use of the Shannon-Fano code and how to associate codes with non-probabilistic models. In Section 5.4.2 we showed that, when using entropified model classes, the Shannon-Fano code can be justified in terms of minimizing worst-case expected code length *for probabilistic and non-probabilistic model classes alike.* This makes 'entropification' a quite general means of turning model classes into codes. As such, it is in line with the general MDL philosophy, in which all models are viewed as 'probabilistic' (Chapter 2.3, Section 2.2), or more properly, as *codes.* Let us consider a quote by Rissanen ([128], page 20):

> "... we then see that the unification obtained by interpreting all models as probabilistic has given us an immutable yardstick, the code length, which we never can reduce to zero by scaling or other devices. The same cannot be said about the usually suggested prediction error measures, which we easily can scale to any size, and which therefore will never be able to serve as a universal yardstick for model selection."

We agree that it is desirable and probably even possible to compare all models and model classes for given data D in terms of the code length they assign to D. But we also think that Rissanen's view leaves open two questions: first, how to base predictions and decisions on a 'probabilistic' model - since the main interpretation of the model is a code rather than a probability distribution according to which data are distributed, it is not a priori clear how this should be done. The second question is how to change model classes that are normally viewed as being 'non-probabilistic' into associated probabilistic model classes in a principled way. As we see it, the concept of 'reliable estimation' is a step towards answering the first question, while 'entropification' is a step towards answering the second question.