Journal of Mathematical Psychology **44**, 133–152 (2000)
doi:10.1006/jmps.1999.1280, available online at http://www.idealibrary.com on IDEAL®

Model Selection Based on Minimum Description Length

P. Grünwald

CWI

We introduce the minimum description length (MDL) principle, a general principle for inductive inference based on the idea that regularities (laws) underlying data can always be used to compress data. We introduce the fundamental concept of MDL, called the *stochastic complexity*, and we show how it can be used for model selection. We briefly compare MDL-based model selection to other approaches and we informally explain why we may expect MDL to give good results in practical applications. © 2000 Academic Press

It is vain to do with more what can be done with fewer.
William of Occam

The task of *inductive inference* is to find laws or regularities underlying some given set of data. These laws are then used to gain insight into the data or to classify or predict future data. The fundamental idea behind the *minimum description length* (*MDL*) *principle* is that any 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. The more regularities there are, the more we are able to compress the data and the more we have *learned* about the data. Formalizing this idea—which is just a version of Occam's famous razor—leads to a theory that is applicable to all kinds of reasoning under uncertainty, including inductive inference. In contrast to most other approaches in this field, it has its roots in theoretical computer science rather than statistics. The MDL principle has mainly been developed by Risannen (1978, 1987, 1989, 1996), having important precursors in the works of Solomonoff (1964) and Wallace and Boulton (1968)[1]. In this introductory and informal paper, we concentrate on the application of MDL to *model selection*, the task of deciding which of several—possibly completely unrelated—classes of models best describes the data at hand. In Sections 2 and 3 we provide the necessary intuitions and mathematics which we then—in Section 4—use to introduce MDL and its central concept, *stochastic complexity* (SC). We show how it can be used for model selection and compare it to the *Bayesian* and *cross-validation* approaches to this problem. Some of the material in this paper is,

[1] Walace and his co-workers extended their original Walace and Boulton (1968) work to the *MML* principle which is closely related to MDL; see Comparisons to Other Approaches.

of necessity, quite technical. To get a first but much gentler glimpse, we advise the reader to just read the following introductory section and the last section, which discusses in what sense we may expect *Occam's razor to actually work*.

THE FUNDAMENTAL IDEA

EXAMPLE 1. Suppose, for the moment, that our data are a finite sequence of bits (zeroes and ones), and consider the following three sequences. We assume that each sequence is 10,000 bits long, and we just list the beginning and the end of each sequence.

```
00010001000100010001.......0001000100010001000100010001 (1)
01110100110100001010101010.......1010111010111011000101100010 (2)
11100111111010011011111 1.......0111101111101111100111101111 (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 and it might make sense to conjecture that future data will also be subject 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. The third sequence contains exactly four times as many 1s as it contains 0s. In contrast to sequence (2), there is some discernible regularity in this data, but of a statistical rather than of a deterministic kind. Again, it seems sensible to note that such a regularity is there and to predict that future data will behave according to it.

The fundamental insight which leads to the MDL principle is that any regularity in the data can be used to *compress* the data, i.e., to describe it in a short manner. Such a description should always completely determine 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 on the basis of D'. Descriptions are always relative to some *description method* which maps descriptions D' in a unique manner to data sets D. A particularly versatile description method is a general-purpose computer language like *C* or Pascal. A description of D is then any computer program that prints D and then halts. Let us see whether our insight works for the three sequences above. Using a language similar to Pascal, we can write a program

```
for i=1 to 2500; print '0001'; next; halt
```

which prints sequence (1) but is clearly a lot shorter than it. The shortest program printing sequence (1) is at least as short as the program above, which means that sequence (1) is indeed highly compressible. On the other hand, we will show in the next section that, if one generates a sequence like (2) by tosses of a fair coin, then with extremely high probability, the shortest program that prints (2) and then halts will look something like this:

```
print
'01110100110100001010101O.......10101110101110110001011000010';
halt
```

This program has size about equal to the length of the sequence. Clearly, it is does nothing more than repeat the sequence. It is easy to show that there exists quite a short program generating sequence (3) too, as we will make plausible in the next section.

Kolmogorov complexity. We now define the *Kolmogorov complexity* (Solomonoff, 1964; Kolmogorov, 1965; Chaitin, 1969) of a sequence as the length of the shortest program that prints the sequence and then halts. The lower the Kolmogorov complexity of a sequence, the *more regular* or, equivalently, the *less random* or, yet again 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 1964, in a pioneering paper that marked the start of all MDL-related research, Solomonoff (1964) proved the *invariance theorem*, which roughly states that it does not matter so much exactly what programming language one uses, as long as it is general purpose[2]: more precisely, for any two general-purpose programming languages A and B, there exists a constant C such that, for every data sequence D, the length of the shortest program for D written in language A and the length of the shortest program for D written in language B differ by no more than C. Here C may depend on the languages A on B, but it is constant in that it does not depend on the size of D: if the data set D is large enough, then the difference in length of the shortest programs according to A and B is very small as compared to the length of the data set. This result, proved independently by Solomonoff, Kolmogorov, and Chaitin, has generated a large body of research on Kolmogorov complexity [for details we refer to (Li & Vitányi, 1997)], which has found its way into psychology before in a context very different from model selection (Chater, 1996).

Unfortunately, the Kolmogorov complexity as such cannot be computed—there can be no computer program that, for every set of data D, when given D as input, returns the length of the shortest program that prints D: assuming such a program exists leads to a contradiction (Li & Vitányi, 1997). Another problem is that in many realistic settings, we are confronted with very small data sets for which the invariance theorem does not say much.

The idea behind the MDL principle is to scale down Solomonoff's approach so that it does become applicable: instead of using a code based on a universal computer language, we should use description methods C which still allow us to compress many of the intuitively regular sequences but which are nevertheless such that for any data sequence D, we can always compute the length of the shortest description of D that is attainable using method 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 compress. But we already know that there can be *no* method for

[2] By this we mean that a universal Turing machine can be implemented in it (Li & Vitáyi, 1997).

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, it is possible to pick a description method C that is based on the class of all polynomials $\mathcal{M}$, such that with the help of C we can compress all data sets which can meaningfully be seen as points on some polynomial; this will be shown below.

Stochastic complexity and model selection. The *Kolmogorov* complexity, of data D, defined relative to some universal computer language L, is the length of the shortest encoding D possible when the encoding is done with the help of L. Rissanen (1987) introduced the *stochastic* complexity as a scaled-down analogue, where the encoding is done with the help of an arbitrary class of models $\mathcal{M}$ rather than a language L (why it is called stochastic will become clear later):

> The stochastic complexity of the data set D with respect to the model class $\mathcal{M}$ is the shortest code length of D obtainable when the encoding is done with the help of class $\mathcal{M}$ (Rissanen, 1987, 1996).

More precisely, the class $\mathcal{M}$ uniquely determines a specific code $C_{\mathcal{M}}$. The stochastic complexity of D with respect to $\mathcal{M}$ is then the codelength of D when it is encoded using code $C_{\mathcal{M}}$. The code $C_{\mathcal{M}}$ should be defined such that it embodies the idea of coding with the help of $\mathcal{M}$: *intuitively*, this means that if there exists a model in $\mathcal{M}$ which captures the regularities in D well, or equivalently *gives a good fit* to D, then the code length of D should be short. It turns out to be quite hard to give a *formal* definition of stochastic complexity (that is, of the code $C_{\mathcal{M}}$). Indeed, a completely satisfactory formal definition has only been found very recently (Rissanen, 1996); we will present it in the section after the next. The most important uses of SC are *prediction* of future data and *model* (*class*[3]) *selection*. Here, we concentrate on the latter application, for which SC can be used in the following intuitive way: suppose we are given a set of data D, and we want to choose between model class $\mathcal{M}_a$ and model class $\mathcal{M}_b$. Then we pick the class for which the stochastic complexity of data D is lower: if D can be compressed more with the help of $\mathcal{M}_a$ than with the help of $\mathcal{M}_b$, $\mathcal{M}_a$ apparently captures better the regularities in D and thus should be preferred. We give a little example of the kind of problem we would like to solve using stochastic complexity in Fig. 1.

Example 2 [under- and overfitting]. Suppose we are given a sequence of data consisting of pairs (x, y) where x and y are real numbers. We are interested in finding a function H such that $H(x)$ predicts y reasonably well. The classical statistical solution to this problem is to do a standard regression: we select the linear function

[3] In our terminology, the problem described here should actually be called model class selection since we are interested in selecting the best class of models for our data. We will nevertheless refer to it using the more common model selection, since (1) this is the usual terminology and (2) as will be seen in the section on reinterpretations of SC, SC allows us to view a class of models $\mathcal{M}$ as if it were a single model summarizing all models in $\mathcal{M}$.

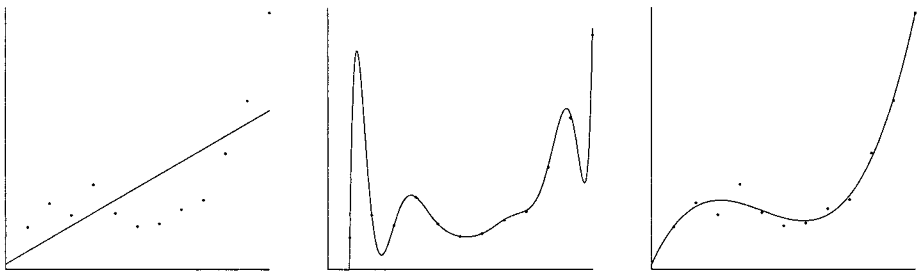

FIG. 1. A simple (a), complex (b) and a trade-off (3rd degree) (c) polynomial.

that is optimal in the least-squares sense, i.e. the linear function H for which the sum of the squared errors $\sum(y_i - H(x_i))^2$ is as small as possible (Fig.1.a). We end up with a line that seems to capture *some* of the regularity in the data, but definitely not too 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, like, for example, the set of *polynomials*. A naive extension of the classical statistical solution will now typically pick a polynomial of degree $n-1$ that completely covers the data, i.e., that goes *exactly* through all the points (x_i, y_i) (Fig. 1.b). Intuitively, this may not be the polynomial we are looking for—we run a large risk of *overfitting*. Instead, we might prefer a third-degree polynomial (Fig. 1.c): one that has small (but not 0) error and is still relatively simple (i.e., has few parameters). In other words, we are looking for an optimal trade-off between model complexity and goodness-of-fit. We will see later that this is indeed what the MDL principle will give us: the stochastic complexity will, in the present example, be lowest for the class of polynomials of the third degree. However, it may always be the case that a completely different class of models—such as, for example, the class of all backpropagation neural networks—is even "better" for the data at hand than the class of *all* polynomials. One can define stochastic complexity even for such broad classes and select the one which allows for more compression of the data at hand (Rissanen, 1989).

CODES, PROBABILITY DISTRIBUTIONS, AND HYPOTHESES

In the next section we give a formal definition of stochastic complexity. We prepare this by first making precise the notion of *description method* or, equivalently, *code* and then showing that both probabilistic and nonprobabilistic models can be expressed as codes. For details we refer to (Cover & Thomas, 1991) and/or (Li & Vitáni, 1997).

Throughout this paper, we assume that our data D always consist of a sequence of symbols from some finite *data* or *source alphabet*. Each such sequence will be encoded as another sequence of symbols coming from some finite *coding alphabet*. An *alphabet* is simply a finite set of distinct symbols. An example of an alphabet is the binary alphabet $\mathbf{B} = \{0, 1\}$; the three data sequences of Example 1 are sequences from the binary alphabet. Sometimes our data will consist of real numbers rather than 0s and 1s. If we assume that all these numbers are truncated to a given finite precision (as is always the case in practical applications), then we can again model them as symbols coming from a finite data alphabet. For alphabet $\mathbf{A}$, we let $\mathbf{A}^*$ denote the set of all finite sequences of symbols in $\mathbf{A}$. In our setting we will always be given a sequence $D = (x_1, ..., x_n) \in \mathbf{A}^*$ where each $x_i \in \mathbf{A}$. We will sometimes use the notation x^i for $(x_1, ..., x_i)$. Similarly, whenever the length n is not clear from the context, we write x^n instead of D. It is easy to see (Rissanen, 1989) that without loss of generality we may always describe our data sequences as binary sequences:

DEFINITION 1. Let $\mathbf{A}$ be some data alphabet. A *code* or *description method* C is a one-to-one map from $\mathbf{A}^*$ into $\mathbf{B}^*$, the set of binary strings.

Here is a simple example (taken from Rissanen, 1989) 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 $x, y \in \mathbf{A}^*$, $C_1(xy) = C_1(x)\, C_1(y)$. 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 show that there is a simple algorithm which for any encoded sequence $C_1(x)$ uniquely retrieves the original sequence x. Moreover, in the encodings $C_1(x)$, no commas are needed to separate the constituent codewords. Codes with this property are called uniquely decodable. For various reasons, all codes considered in any application of MDL are uniquely decodable; henceforth whenever we speak of a code, we actually mean a uniquely decodable one [actually, the situation is slightly more complicated than this; for details see (Grünwald, 1998, Chap. 1)].

Compression and small subsets. We can now argue why it is impossible to substantially compress randomly generated sequences like sequence (2) of Example 1. Let us take some arbitrary but fixed code C over the binary data alphabet $\mathbf{B}$. Suppose we are given a data sequence of length n (in our example, $n = 10{,}000$). Clearly, there are 2^n possible binary data sequences of length n. We see that C can map only two of these to a description of length 1 (the reason being that (a) there are only two binary sequences of length 1: "0" and "1" and (b) if data sequence $D_1 \neq D_2$, then, by the definition of a code, $C(D_1) \neq C(D_2)$.). Similarly, only a subset of at most 2^m sequences can have a description of length m. This implies that the fraction of data sequences of length n that C compresses by more than k bits must decrease exponentially in k. It follows (we will not show this in a rigorous manner) that the probability that we can compress a randomly generated sequence by k bits is smaller than 2^{-k}, which even for small k will already be an extremely small number. Seen in this light, finding a short code length for the data is equivalent to identifying the data as belonging to a tiny, very *special* subset out of all a priori possible data sequences. This is also the reason why we can compress sequence (3): the number of all sequences of length n with four times as many 1s as 0s is *extremely* (exponentially) small compared to 2^n, the total number of sequences of length n (Cover & Thomas, 1991).

Codes "are" probability distributions. Many model classes encountered in practice are *probabilistic*. A very simple example is the class of Bernoulli distributions $\mathcal{M}_B$. $\mathcal{M}_B$ is indexed by a single parameter θ. For any $0 \leqslant \theta \leqslant 1$, θ represents the hypothesis that the data is generated by independent flips of a biased coin with probability θ of coming up heads (we identify "heads" with 1). The probability of data D under model θ is denoted by $P(D \mid \theta)$. In this paragraph, we show that we can map any probabilistic model θ into a code such that, for all D, the code length of D reflects the probability of D. For this, we first introduce some further notation. For any code C, we denote by $L_C(\alpha)$ the length (number of bits) of the description of α when the description is done using code C. Throughout this paper, log stands for logarithm to base two, and for any real number x, $\lceil x \rceil$ stands for the *ceiling* of x, the smallest integer greater than or equal to x.

Codes (actually, only uniquely decodable ones) have a very important property which connects them to probability distributions:

PROPOSITION 1. *For every code C defined on a finite alphabet $\mathbf{A}$ there exists a probability distribution P_C such that for all $x \in \mathbf{A}$ we have $P_C(x) = 2^{-L_C(x)}$. For every probability distribution P defined over a finite set $\mathbf{A}$ there exists a code C_P such that for all $x \in \mathbf{A}$ we have $L_{C_P}(x) = \lceil -\log P(x) \rceil$.*

This proposition follows directly from the famous Kraft–McMillan inequality. For this inequality and its proof, see Cover and Thomas (1991). The key to the proof is the fact we saw above: a code can only give a short code length to very few data sequences. A deeper analysis of this fact reveals that $\sum_x 2^{-L_C(x)}$ must be less than or equal to one for every uniquely decodable code C; it is simply not possible that a large enough number of elements x has a short enough codelength to make $\sum_x 2^{-L_C(x)}$ larger than one.

Henceforth we will drop the integer requirement for codelengths (this will lead to an inaccuracy of at most one bit). Once we have done this, Proposition 1 implies that we can interpret any probability distribution over sequences of a given length as a code and vice versa! This correspondence allows us to *identify* codes and probability distributions: for every probability distribution P there exists a code C with code lengths $L_C(D) = -\log P(D)$ for all D of given length, and for every code C there exists a probability distribution P with for all D, $P(D) = 2^{-L_C(D)}$, or, equivalently, $L_C(D) = -\log P(D)$. We see that a short code length corresponds to a high probability and vice versa: whenever $P(D) > P(D')$, we have $-\log P(D) < -\log P(D')$. We note that 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 are drawn according to the probability distribution corresponding to C. The results of the present section only tell us that the function $P(x) = 2^{-L_C(x)}$ satisfies all the necessary conditions to be a probability distribution. By discretizing to a finite precision we can do the same for continuous data: for any probability density function $f(x)$, $-\log f(x)$ can be interpreted as a code length (Rissanen, 1987).

Maximum likelihood = minimal code length. Suppose we are given a probabilistic (say Bernoulli) model class $\mathcal{M}$ containing models θ. If the class of models is regular enough (as we assume here), then there exists a *maximum likelihood* (*ML*) *estimator* $\hat{\theta}$ for every data set D of every length n. This is the model in $\mathcal{M}$ that maximizes the probability of D. We denote this θ by $\hat{\theta}$. For example, the ML Bernoulli model $\hat{\theta}$ for sequence (1) of Example 1 at the beginning of this paper is $\hat{\theta} = 1/4$. The probability it assigns to sequence (1) is $P(\text{seq.}(1) \mid \hat{\theta}) = (\frac{1}{4})^{2500} (\frac{3}{4})^{7500}$. Sometimes we write $\hat{\theta}(D)$ rather than the usual $\hat{\theta}$ since we want to stress that $\hat{\theta}$ is actually a *function* of D. If we pick our models from model class $\mathcal{M}$, we can write

$$P(D \mid \hat{\theta}(D)) = \max_{\theta \in \mathcal{M}} P(D \mid \theta) = \min_{\theta \in \mathcal{M}} -\log P(D \mid \theta) = \min_{\theta \in \mathcal{M}} L(D \mid \theta) \tag{4}$$

where the third equality follows from the fact that log and—are monotonic transformations and the fourth equality indicates the fact that each θ defines a code such that the code length of D is given by $-\log P(D \mid \theta)$. Since this term can be interpreted as a code length, we abbreviate it to $L(D \mid \theta)$. For any θ_1, θ_2, $P(D \mid \theta_1) > P(D \mid \theta_2)$ if and only if $-\log P(D \mid \theta_1) < -\log P(D \mid \theta_2)$, so the code length of D using any θ precisely measures the probability of D under θ, i.e. it measures *how well θ fits D*. From (4) we then see that $\hat{\theta}$ is also the element in $\mathcal{M}$ that *minimizes the code length of D*.

Codes "are" (non)probabilistic hypotheses. Model classes $\mathcal{M}$ like the class of all polynomials are not probabilistic; instead, they always go together with an *error* function that measures the goodness-of-fit for each combination of data D and hypothesis $H \in \mathcal{M}$. In the polynomial example, this was the squared error: 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 $\mathrm{ER}_{sq}(x^n, y^n \mid H) < \mathrm{ER}_{sq}(x^n, z^n \mid H)$, where ER_{sq} is the total squared error: $\mathrm{ER}_{sq}(x^n, y^n \mid H) = \sum_{i=1}^{n} (y_i - H(x_i))^2$.

Rissanen (1987, 1989) shows that we can turn just about any combination of hypothesis and error function into a code such that the code length of data D *precisely* reflects the error the hypothesis makes on the data: the shorter the code length, the better the fit. We illustrate this for the polynomials and the squared error. It is possible (Rissanen, 1989, p. 18) to construct a code C_2 such that for all $D = ((x_1, y_1), ..., (x_n, y_n))$ and all polynomials H:

$$L_{C_2}(x^n, y^n \mid H) = \mathrm{ER}_{sq}(x^n, y^n \mid H) + K. \tag{5}$$

The notation $L_{C_2}(\cdot \mid H)$ is used to indicate that the code-length may depend on H: for different polynomials, C_2 encodes D in a different way; strictly speaking it is not a code but a *function* that maps each H to a different code $C_2(H)$. The notation $L_{C_2}(\cdot \mid H)$ is then short for $L_{C_2(H)}(\cdot)$. In the equation above K is a constant that may depend on n but *not* on H or any of the x_i or y_i. Using the code with property (5), we have for every two data sets $y_1, ..., y_n$ and $z_1, ..., z_n$ that

$$\Delta L_{C_2} = L_{C_2}(x^n, z^n \mid H) - L_{C_2}(x^n, y^n \mid H) = \Delta \mathrm{ER}_{sq} = \mathrm{ER}_{sq}(x^n, z^n \mid H) - \mathrm{ER}_{sq}(x^n, y^n \mid H) \tag{6}$$

which means that the difference in goodness-of-fit between any two data sets is precisely reflected in the difference in code length. We can perform the same trick for just about any other class of models and any other error measure. It follows that we can turn *any* such class of models into a class of *codes* that is equivalent in that the codelength of data D when encoded using code $C_2(H)$ is equal to the error H makes on D up to a constant that only depends on the size of D: any model, probabilistic or not, may be identified with a code. Because codes can always be mapped to probability distributions and back, we may equivalently regard all model classes as being *probabilistic*. This explains the name *stochastic* complexity: we will define it for probabilistic model classes only and map any nonprobabilistic model class first into an equivalent probabilistic version. For any data D, the model in $\mathscr{M}$ with minimal error for D will be called the *best-fitting* model for D, denoted by $\hat{H}$. By Eq. (5) it is also the model with the shortest code length for D. Remarkably, if we turn $\mathscr{M}$ into an equivalent probabilistic version $\mathscr{M}'$, then by (4) we see that $\hat{H}$ is mapped to the maximum likelihood model $\hat{\theta}(D)$! From now on we regard *any* model class as being probabilistic—including the polynomials. We stress once more that this does *not* mean that these models should be interpreted as probability distributions in the usual sense; rather, codes or, equivalently, probability distributions are used merely as a *universal representation language* for hypotheses.

STOCHASTIC COMPLEXITY AND MODEL SELECTION

We will now introduce the notion of stochastic complexity. The MDL principle tells us to look for the shortest encoding of D. In the context of stochastic complexity, we assume a model class $\mathscr{M}$ and we ask for the shortest encoding of D obtainable with the help of $\mathscr{M}$ (see p. 136). By (4) the element in $\mathscr{M}$ that allows for the shortest encoding of D is given by (the code corresponding to) $\hat{\theta}(D)$. It *seems* we should code our data D using the ML model $\hat{\theta}(D)$, 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—MDL says that we must code our data using some *fixed* code, which compresses *all* data sets that contain a model in $\mathscr{M}$ that fits them well (this was pointed out on page 136. But the code corresponding to $\hat{\theta}(D)$, i.e., the code that encodes any D' using $L(D' \mid \hat{\theta}(D)) = -\log P(D' \mid \hat{\theta}(D))$ bits, only gives optimal compression for *some* of these data sets (among which D). For most other data sets $D' \neq D$, $\hat{\theta}(D)$ will definitely not be optimal: if we had been given such a different data set 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 $L(D' \mid \hat{\theta}(D))$ bits] may be very inefficient. Since MDL tells us to code our data using some *fixed* code, we are not allowed to use the code based on $\hat{\theta}(D)$ if our data happen to be D and the code based on $\hat{\theta}(D')$ if our data happens to be D'; we would then encode D using a different code than when encoding D'. It would therefore be very desirable if we could come up with a code that compressed each possible D as well as the best-fitting or, equivalently, most compressing element in $\mathscr{M}$ for that specific D! In other words, we would like to have a single code C_1 such that $L_{C_1}(D) = L(D \mid \hat{\theta}(D))$ for all possible D. However, such a code does not exist as soon as our model class contains more than one element—intuitively, the reason being a fact we discussed earlier: whatever code we use, only very few data sets can receive short code lengths. Formally, suppose our model class contains at least the two models θ_1 and

θ_2 where $P(\cdot\,|\,\theta_1) \neq P(\cdot\,|\,\theta_2)$ and suppose, by means of contradiction, that a code C_0 exists such that $L_{C_0}(D) = L(D\,|\,\hat{\theta}(D))$ for all D of fixed length. By Proposition 1 there exists a probability distribution P_0 with $P_0(D) = 2^{-L_{C_0}(D)} = 2^{-L(D\,|\,\hat{\theta}(D))} = P(D\,|\,\hat{\theta}(D))$ for all D of fixed length (the last equality follows by definition of $L(D\,|\,\hat{\theta}(D))$). Since for all D, $P(D\,|\,\hat{\theta}(D)) \geqslant P(D\,|\,\theta_1)$, and for at least one D, $P(D\,|\,\hat{\theta}(D)) > P(D\,|\,\theta_1)$, we have $\sum_D P_0(D) > 1$ which contradicts the fact that P_0 is a probability distribution.

We see that the code C_1 we referred to above does not exist in general. Nevertheless[4], it *is* possible to construct a code C_2 such that

$$L_{C_2}(D) = -\log P(D\,|\,\hat{\theta}(D)) + K = L(D\,|\,\hat{\theta}(D)) + K \tag{7}$$

for all D of length n. Here K is a constant that may depend on n but is equal for all D of length n. In this way, the code length obtained using C_2 precisely reflects for each D how good the best-fitting model in the class for D is—compare this to Eqs. (5) and (6); we give a clarifying example below. Picking C_2 such that the constant K is as small as possible yields the most efficient code that satisfies (7). We call the resulting code the *stochastic complexity code* and denote it by C^*. The corresponding minimal K is denoted by K^* and is called the *model cost* of $\mathscr{M}$. We define the code length of D when encoded using this code to be the *stochastic complexity of D with respect to model class* $\mathscr{M}$ which we write as $I(D\,|\,\mathscr{M})$:

$$I(D\,|\,\mathscr{M}) = L_{C^*}(D) = L(D\,|\,\hat{\theta}(D)) + K^* \text{ where } \hat{\theta}(D) \in \mathscr{M}. \tag{8}$$

The trade-off. $I(D\,|\,\mathscr{M})$ is a sum of a *goodness-of-fit* term $L(D\,|\,\hat{\theta}(D))$ and a *complexity term* K^* and as such embodies the trade-off between fit and complexity referred to in Fig. 1. To see this for the polynomial example, let us compare the polynomials in Fig. 1.a: let $\mathscr{M}_i$ be the class of all polynomials of the ith degree (e.g., $\mathscr{M}_{12}$ is the class of 12th degree polynomials); similarly, let $\hat{\theta}_i(D)$ be the maximum likelihood estimator for data D within class $\mathscr{M}_i$. We note that, by Eqs. (5) and (6), the difference in the first terms of $I(D\,|\,\mathscr{M}_1)$ and $I(D\,|\,\mathscr{M}_{12})$ is *equal* to the difference in the squared errors on D of the best-fitting models in these classes! We have

$$\begin{aligned} I(D\,|\,\mathscr{M}_1) - I(D\,|\,\mathscr{M}_{12}) &= L(D\,|\,\hat{\theta}_1(D)) - L(D\,|\,\hat{\theta}_{12}(D)) \\ &\quad + K_1^* - K_{12}^* = \Delta\mathrm{ER}_{sq} + \Delta\text{complexity}. \end{aligned}$$

Here K_1^* is the complexity term for $\mathscr{M}_1$, which will be much *smaller* than K_{12}^*: $\mathscr{M}_{12}$ contains good-fitting models for *many more* data sets than $\mathscr{M}_1$; therefore, $L(D\,|\,\hat{\theta}_{12}(D))$ will be very short for many more D. Since the code C_i^* corresponding to class $\mathscr{M}_i$ can only give a short code length $I(D\,|\,\mathscr{M}_i) = L(D\,|\,\hat{\theta}_i(D)) + K_i^*$ to very few data sets, K_{12}^* must be much larger than K_1^* (this can be shown more formally using Proposition 1 by an argument similar to the one we used in proving that the

[4] Actually, such a code C_2 can be created only for model classes with a fixed number of parameters; the definition for classes like the class of *all* polynomials is more involved (Rissanen, 1996).

code C_1 referred to above does not exist). A concrete interpretation and an explicit value for K_i^* will be given later on. For the set of points in Fig. 1, it turns out that the sum is minimized for $\mathscr{M}_3$, the class of third-degree polynomials, which brings us to the following

MDL principle for model selection. Given data D and any two model classes $\mathscr{M}_a$ and $\mathscr{M}_b$, we should prefer model class $\mathscr{M}_a$ if and only if the stochastic complexity of D with respect to $\mathscr{M}_a$ is smaller than the stochastic complexity of D with respect to $\mathscr{M}_b$, i.e., if $I(D \mid \mathscr{M}_a) < I(D \mid \mathscr{M}_b)$. The larger the difference, the more confidence we have in our choice.

Computing $I(D|\mathscr{M})$. For most reasonably regular model classes, the first term in the stochastic complexity is easy to compute (for polynomials it involves finding a least-squares fit, for example). Computing K^*, however, is very difficult. For sufficiently regular model classes (which include, for example, the class of polynomials combined with the squared error), we can, however, approximate K_k^* (i.e., the K^* for a model class with k parameters) very well (Rissanen, 1996) as follows:

$$K_k^* \approx \frac{k}{2} \log n + C_k \tag{9}$$

The first term grows linearly in the number of parameters k and logarithmically in n, the size of the data set. The second term is constant in that it does not grow as n increases, but it does depend on k. Neglecting this term and selecting $\mathscr{M}_k$ by choosing the k which minimizes $-L(D \mid \hat{\theta}(D)) + \frac{k}{2} \log n$ has, confusingly, been called the *MDL model selection criterion* (Rissanen, 1989). One can use it as a first approximation, but it will only work well if one really has a lot of data, since C_k can be very large. For sufficiently regular model classes, C_k can be further evaluated as follows (Rissanen, 1996):

$$C_k = -\frac{k}{2} \log 2\pi + \log \int_{\theta \in \mathscr{M}_k} \sqrt{|\, I(\theta)|}\, d\theta + o(1). \tag{10}$$

Here $o(1)$ is a term that becomes negligible for large enough n ($\lim_{n \to \infty} o(1) = 0$) and $|I(\theta)|$ is the determinant of the *Fisher information matrix* $I(\theta)$ (for a definition, see Berger, 1985 or Rissanen, 1996). It measures effects on the description length that are due to geometrical properties of the model class $\mathscr{M}_k$. To get a rough idea of where these effects come from, consider the function $L(D' \mid \hat{\theta}(D))$ as a function of D'. The slope of this function for D' in a neighborhood of D is different for different D. A precise (and difficult) analysis of this phenomenon leads to (10). Equations (9) and (10) not only describe how the complexity of a model class depends on its size (number of parameters k), but also how it depends on its internal structure: if most models in the class are extremely *similar* to each other (in the sense that they represent almost the same probability distributions) then the model class is not so complex (rich) after all. In this case, the term in (10) involving $|I(\theta)|$ will be small. The more truly different the models in $\mathscr{M}$ are, the larger this term. To summarize the essence of Eqs. (8)–(10), we note that for every given model class $\mathcal{M}$ with a fixed number of parameters, the error term $L(D \mid \hat{\theta}(D))$ grows *linearly* in the sample size n for (nearly) all D. The complexity term that is due to the number of parameters k grows *logarithmically* in n, and the complexity due to the internal structure of the model class does not grow with n.

REINTERPRETATIONS AND PRACTICAL VERSIONS OF SC

Many model classes are such that either (9) does not hold at all or the constant term (10) cannot be easily calculated. Fortunately, there exist at least three good approximations to the SC, which at the same time give alternative, perhaps clearer interpretations to it. We give two now and a third one in the next section.

SC as an average. By (8), $I(D \mid \mathcal{M}) = L_{C^*}(D)$. Of course we can map C^* to a probability distribution P^* such that for all D, $-\log P^*(D) = I(D \mid \mathcal{M})$. Just as C^* can be seen as the code giving the shortest codes possible for data sets with good models in $\mathcal{M}$, P^* can be seen as the probability distribution giving as much probability as possible to those data sets for which there is a good model in $\mathcal{M}$. To make the dependence on $\mathcal{M}$ explicit we will henceforth write $P^*(\cdot \mid \mathcal{M})$ rather than P^*. We call $P^*(\cdot \mid \mathcal{M})$ the stochastic complexity distribution with respect to $\mathcal{M}$. Now suppose, just for the moment, that our (probabilistic) class $\mathcal{M}$ has a finite number of elements. In this case we may define a new probability distribution

$$P_{av}(D \mid \mathcal{M}) = \sum_{\theta \in \mathcal{M}} P(D \mid \theta) \times w(\theta). \tag{11}$$

Here $w(\theta)$ is a normalizing factor that can be viewed as a probability distribution over $\mathcal{M}$. It is usually called a *prior distribution.* If we let w be *uniform* (all terms in the average have the same weight), then $P_{av}(D \mid \mathcal{M})$ is a very good approximation for $P^*(D \mid \mathcal{M})$. 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 $P_{av}(D \mid \mathcal{M})$. For model classes indexed by parameters ranging over a continuous domain—as, for example, the class of all Bernoulli models—the sum gets replaced by an integral. In this case, again for many model classes there exists a prior $w(\theta)$ with which $-\log P_{av}(D \mid \mathcal{M})$ approximates $I(D \mid \mathcal{M})$ extremely well but it is *not* the uniform prior (Rissanen, 1996); rather it is the so-called *Jeffreys' prior* (for a precise definition, see Wasserman, 2000). For most other priors which give probability >0 to all models in $\mathcal{M}$, the approximation to $I(D \mid \mathcal{M})$ will still be reasonable but only for large data sets. We see that $P^*(\cdot \mid \mathcal{M}) \approx P_{av}(\cdot \mid \mathcal{M})$ is a probability distribution that may (roughly) be interpreted as an average over all models in $\mathcal{M}$. Since $P^*(\cdot \mid \mathcal{M})$ is a probability distribution, it may itself be seen as a single model. Hence, the name model selection, the usual term for comparing *classes* of models, makes sense in our terminology after all: according to MDL, selecting between model classes $\mathcal{M}_a$ and $\mathcal{M}_b$ for data D amounts to deciding which of the summarizing models $P^*(\cdot \mid \mathcal{M}_a)$ or $P^*(\cdot \mid \mathcal{M}_b)$ gives a better fit (shorter codelength) to the data.

SC as a two stage description of the data. Another approximation to the SC arises if we code our data in two parts as follows: for any model class $\mathcal{M}$, data set D is encoded by first encoding a hypothesis $\theta \in \mathcal{M}$ and then encoding D using the code corresponding to θ, i.e., using $L(D \mid \theta)$ bits where $L(D \mid \theta) = -\log P(D \mid \theta)$. Each parameter in θ must be encoded up to a certain precision d. If $\mathcal{M}$ contains k-parameter models, we need approximately $L(\theta) \approx k \cdot d$ bits for this, a number that grows linearly in k again. By taking the model $\theta^* \in \mathcal{M}$ and the precision d for which the sum $L(D \mid \theta) + L(\theta)$ is as small as possible, we arrive at a reasonable approximation of stochastic complexity (where we write $\theta^*(D)$ to indicate that the optimal θ^* is actually a function of D): $I(D \mid \mathcal{M}_k) \approx L(D \mid \theta^*(D)) + L(\theta^*(D))$ (Rissanen, 1989).

This can be shown formally if $\mathcal{M}_k$ is regular enough: then the optimal precision for each parameter can be shown to be $\frac{1}{2}\log n$ where n is the size of D; moreover, for such model classes, it holds (under mild additional conditions) that asymptotically for large n, $L(D \mid \hat{\theta}(D)) \leqslant L(D \mid \theta^*(D)) \leqslant L(D \mid \hat{\theta}(D)) + C$ for some constant C (and hence $\hat{\theta}$ and θ^* converge to each other). This gives by (8) and (9)

$$L(D \mid \theta^*) + L(\theta^*) = L(D \mid \theta^*) + \frac{k}{2}\log n \approx L(D \mid \hat{\theta}(D)) + K^*.$$

This equation shows that the complexity term $K^* = k/2 \log n + O(1)$ can be (roughly) interpreted as follows: it can be seen as the number of bits (up to a constant) needed to encode the parameter $\theta^*(D) \approx \hat{\theta}(D)$ that minimizes the total two-part description length of the data.

The two-part code version of SC may be of particular interest to psychologists since one can compute it for just about *any* model class one can think of. In psychological applications (Myung & Pitt, 1997), one more often than not uses complicated models for which the other approximations to SC mentioned in this paper are very hard to compute. Examples of strange model classes for which two-part codes have already been constructed are context-free grammars (Grünwald, 1996), various neural networks (Barron, 1990), and several models arising in the analysis of DNA and protein structure (Dowe *et al*., 1996).

COMPARISON TO OTHER APPROACHES

Below we briefly compare MDL to some alternative approaches. We will focus on Bayesian statistics (Wasserman, 2000) and cross-validation (Browne, 2000) since they are both closely related to MDL. A model selection criterion that is quite different from MDL (yet partially served as its inspiration) is Akaike's AIC (Bozdogan, 2000). For a comparison of AIC and MDL in the context of regression, we refer to Speed and Yu (1993). Below we will only consider the use of MDL for model class selection. Outside this context, MDL is still closely related to several other inductive methods. For example, the successful yet controversial principle of maximum entropy can be seen as a special case or, at any rate, version of MDL (Rissanen, 1989; Li & Vitányi, 1997; Grünwald, 1998).

Bayesian evidence and stochastic complexity. In Bayesian statistics (Berger, 1985; Wasserman, 2000) we always use a prior distribution w for all elements in the chosen model class $\mathscr{M}$. We can then simply calculate the *conditional* probability of the data D given model class $\mathscr{M}$ as $P(D \mid \mathscr{M}) = \sum_{\theta \in \mathscr{M}} P(D \mid \theta)\, w(\theta)$ which coincides with our P_{av} as given by (11). If we want to do model selection between classes $\mathscr{M}_a$ and $\mathscr{M}_b$, Bayesian statistics tells us to prefer the class for which $P(D \mid \mathscr{M})$, also called the *evidence*, is highest. We immediately see that if we use the P_{av} approximation to stochastic complexity, then selecting the class with lowest stochastic complexity becomes equivalent to selecting the class with the highest evidence.

What, then, is the difference between the two approaches? From a philosophical point of view, the difference can be summarized as follows: *According to MDL the aim is always to compress our data as much as possible*; *there is no such goal in Bayesian statistics*!

MDL's focus on compression reveals itself in several ways. We mention just one: the real definition of stochastic complexity as given in (8) does not depend on any particular prior distribution. This is how it should be, since it is defined with respect to D and $\mathscr{M}$ and should thus depend on D and $\mathscr{M}$ and nothing else. If we approximate $I(\cdot \mid \mathscr{M})$ using P_{av}, we should always pick a prior with which we obtain code lengths that are as short as possible, i.e., such that we are as close as possible to the real $I(\cdot \mid \mathscr{M})$ (Rissanen (1996) gives a precise definition of "being as close as possible").

Nevertheless, in practical settings MDL and Bayesian techniques will often give similar results. An advantage of MDL is that we always know what we are aiming for and thus, first, are never left with the dilemma what kind of prior distribution we should use, and, second, we may use variations of MDL which do not involve priors at all (like predictive MDL; see below) if that is easier for the problem at hand. A disadvantage of MDL may be that we force ourselves to use particular priors while in some cases some alternative priors may, for some reason or other, give better results.

MDL and MML. The relationship between MDL and Bayesianism presented here is based on the particular viewpoint implicit in Rissanen's work (Rissanen, 1989, 1996); other researchers hold somewhat different views and the precise relationship between the two approaches is still subject to much debate (Wallace & Freeman, 1987; Dawid, 1992; Rissanen, 1996; Vitányi & Li, 1997). In particular, there exists a statistical inference method called the *minimum message length* (*MML*) *principle* which is based on combining a Bayesian view on probability distributions with data compression (Wallace & Boulton, 1968; Wallace & Freeman, 987; Wallace, 1996). Though in practice MML and MDL will usually behave similarly, there are also some subtle differences; in Grünwald (1998) the relationship is discussed in more detail. Another statistical paradigm which has its roots in Bayesian statistics while being actually closely related to MDL is Dawid's *prequential analysis*. Dawid (1992) studied the relation between prequential analysis and MDL.

MDL & cross-validation—SC as accumulated prediction error. For most model classes, yet another good approximation to the stochastic complexity is given by the sum of prediction errors made when predicting future data using past data only.

For example, using the class of k-degree polynomials $\mathcal{M}_k$, we may sequentially predict all y_i by using the best-fitting (maximum likelihood) model $\hat{H}_{i-1}$ for $D_{i-1} = ((x_1, y_1), ..., (x_{i-1}, y_{i-1}))$. We may then measure the error that $\hat{H}_{i-1}$ makes on the y_i that actually arrives as $(y_i - \hat{H}_{i-1}(x_i))^2$ which is just the squared error. Rissanen (1989) shows that for large n, $\sum_{i=1}^{n} (y_i - \hat{H}_{i-1}(x_i))^2 \approx I(D_n \mid \mathcal{M})$. This way of approximating $I(D_n \mid \mathcal{M})$ is called predictive MDL (actually, the way we present it here we avoid some tricky but important details; see Rissanen, 1989).

Suppose now that we compare $I(D \mid \mathcal{M}_a)$ to $I(D \mid \mathcal{M}_b)$ for the purpose of selecting either $\mathcal{M}_a$ or $\mathcal{M}_b$. By the preceding discussion we may interpret $I(D \mid \mathcal{M}_a)$ as indicating how well we were able to sequentially predict each item in D when doing the prediction of the ith item using each time the model in $\mathcal{M}_a$ that best fitted previous data D_{i-1}. $I(D \mid \mathcal{M}_b)$ may be interpreted analogously. This is very similar to another model selection method called cross-validation (Browne, 2000). We briefly describe cross-validation to show the similarities and differences to predictive MDL.

In cross-validation, the data D are partitioned into a *training set* D^a and a *test set* D^b. Then $\hat{H}(D^a)$, the best-fitting model for D^a is determined, and it is tested on D^b; that is, the error it makes on D^b, is determined. In our example of a class of functions with squared error, this amounts to computing $\text{ER}(D^b \mid \hat{H}(D^a)) = \sum_{(x,y) \in D^b} \text{ER}_{sq}(x, y \mid \hat{H}(D^a))$. This is usually repeated for several partitionings of D, say $(D^{a_1}, D^{b_1}), ..., (D^{a_m}, D^{b_m})$ (where $D^{a_j} \cup D^{b_j} = D$ for all j) and the average prediction error $\frac{1}{m} \sum_{j=1}^{m} \text{ER}(D^b \mid \hat{H}(D^a))$ is taken as a measure of how well suited $\mathcal{M}$ is for D (analogously to MDL's stochastic complexity $I(D \mid \mathcal{M})$). The size of the sets D^{a_j} and D^{b_j} is usually taken equally large for all j. This is in contrast to predictive MDL, where $\hat{H}(D_1)$ for $D_1 = (x_1, y_1)$ is used to predict (x_2, y_2), then $\hat{H}(D_2)$ [with $D_2 = ((x_1, y_1), (x_2, y_2))$] is used to predict (x_3, y_3) etc. and the sum of the n resulting errors is used to approximate $I(D \mid \mathcal{M})$.

WHY IT WORKS: OVERFITTING, UNDERFITTING, AND BIAS

Is there some way or other in which we can prove that MDL works? There have been several studies which indicate that MDL performs well both in theory and in practice (Barron & Cover, 1991; Rissanen, 1989; Kontkanen, Myllymäki, & Trirri, 1996). We will not go into these studies any further but rather appeal to a common-sense argument, which, in a nutshell, says that there is a crucial difference between overfitting and underfitting.

Overfitting and underfitting. Let us illustrate this using our polynomial example once again. Clearly, the first-degree polynomial used in Fig. 1.a is too simple. If we look at the average error per data item (x_i, y_i), we see that it is quite high. It may seem reasonable to assume that if we use the line found as a basis for predicting future data, then we will make approximately the same error on average as we have made on the data D we have seen so far. However, if we look at the complicated polynomial depicted in Fig. 1.b, we see that it makes average error 0 on the data D. It would be very unreasonable to assume that this polynomial will keep making error 0 on any future data: if we assume that the data are subject to a small amount

If you *overfit*, you think you know more than you really know. If you *underfit*, you do not know much but you know that you do not know much. In this sense, *underfitting* is relatively harmless while *overfitting* is dangerous.

FIG. 2. Another way of looking at Occam's Razor.

of random noise, then the overly complex polynomial will just reflect this noise rather than any law underlying the data.

In other words, if we use the model that is too simple, we may get a reliable estimate of how well we will predict future data; if our model is too complex, we will not. Since this observation is so important, it has been repeated in Fig. 2.

It turns out that one can even make this observation (partially) mathematically precise (Grünwald, 1998, chapt. 5). The fact that, in MDL, all inductive inference is done with respect to a coding system leads to a kind of generic avoidance of overfitting. Equations (8) and (9) imply that, when given few data, we will prefer a first-degree polynomial even though a third-degree polynomial will turn out to be better in the end. But the first-degree polynomial will be useful nevertheless, until our data set is large enough so that we have *enough information* to decide with any reliability on a higher degree polynomial.

MDL and Bias. We will now make this idea a bit more precise. In the following, we assume that the data $D=(x_1, ..., x_n, x_{n+1}, ..., x_m)$ can be partitioned into a training set $D^{\mathrm{tr}}=(x_1, ..., x_n)$ and a test set $D^{\mathrm{test}}=(x_{n+1}, ..., x_m)$. By a learning method we mean some automated procedure that, when given as input the training set, outputs a model $\tilde{H}$ as a hypothesis for data D. This model is then used to predict the data in the test set against some error measure ER. We will call the model $\tilde{H}$ a good model for D if predicting the elements in D on the basis of $\tilde{H}$ leads to a small average error $\frac{1}{m}\sum_{i=1}^{m}\mathrm{ER}(x_i \mid \tilde{H})$. We say that the inference of $\tilde{H}$ from training data D^{tr} is *reliable* for test data D^{test} if

$$\frac{1}{n}\sum_{i=1}^{n}\mathrm{ER}(x_i \mid \tilde{H}) \approx \frac{1}{m-n}\sum_{j=n+1}^{m}\mathrm{ER}(x_j \mid \tilde{H})$$

That is, the error $\tilde{H}$ makes on the training set is a reasonable indicator of the error it makes on the test set. Let us make the assumption that the x_i are all independent and are each drawn according to some unknown distribution P. Then, roughly speaking and under some mild additional conditions, we can say that the larger (more complex) the class of candidate models $\mathscr{M}(n)$ considered for a training set of size n is, the larger the probability that the inferred model $\tilde{H} \in \mathscr{M}(n)$ will be *un*reliable. This holds no matter what learning method is used to infer $\tilde{H}$ from the training set. In order to make reliable inferences *at all* possible, we must do something to effectively restrict the size of $\mathscr{M}(n)$. In other words, we must introduce *bias*[5]: out of the (often infinitely many) models that we *might* consider, we choose

[5] We use the word bias in an informal sense here, not in its traditional statistical meaning in terms of mathematical expectation.

one from the small set $\mathcal{M}(n)$. However, this preferred subset of models may be allowed to increase as the size of the sample (data set) increases. With increasing sample size, more information about the data generating process becomes available. This information can be used to reliably infer the best model for the data within a larger set of candidate models. Hence, as the data set increases, the bias may be gradually weakened[6].

In MDL, the bias is determined by the code that is used to encode data. In the context of model class selection and stochastic complexity, this is expressed by the fact that the complexity term K^* in the formula for SC (Eq. (8)) will be larger for more complex model classes. Suppose we are given a complex model class $\mathcal{M}_a$, a simple model class $\mathcal{M}_b$, and training data D^{tr} of size n. Then

$$I(D^{\text{tr}} \mid \mathcal{M}_a) = L(D^{\text{tr}} \mid \hat{H}_a(D^{tr})) + K_a^* \tag{12}$$

and similarly for $I(D^{\text{tr}} \mid \mathcal{M}_b)$. Since $\mathcal{M}_a$ is more complex, $K_a^* > K_b^*$. Suppose that the best fitting model for training set D^{tr} in $\mathcal{M}_a$ makes average error $\frac{1}{n}\text{ER}(D^{\text{tr}} \mid \hat{H}_a(D^{\text{tr}}))$ and the best fitting model in $\mathcal{M}_b$ makes average error $\frac{1}{n}\text{ER}(D^{\text{tr}} \mid \hat{H}_b(D^{\text{tr}}))$ such that $\frac{1}{n}\text{ER}(D^{\text{tr}} \mid \hat{H}_a(D^{\text{tr}})) < \frac{1}{n}\text{ER}(D^{\text{tr}} \mid \hat{H}_b(D^{\text{tr}}))$. Then, since we always construct the codes with lengths $L(\cdot \mid H)$ such that these lengths correspond to the errors $\text{ER}(\cdot \mid H)$ (see Eq. (5) for the example of the squared error), we must have

$$\frac{1}{n} L(D^{\text{tr}} \mid \hat{H}_a(D^{\text{tr}})) = \frac{1}{n} L(D^{\text{tr}} \mid \hat{H}_b(D^{\text{tr}})) - \varepsilon \tag{13}$$

for some $\varepsilon > 0$. From (12) and (13) we see that $\mathcal{M}_a$ will *only* be preferred over $\mathcal{M}_b$ if n is larger than some minimal value n_0: for smaller values of n, the simpler model class will be chosen—even though $\mathcal{M}_a$ contains a model that fits the data better, it is too large a model class to consider for the small training set.

MDL, bias and no free lunch—conclusion. We just saw that bias is a necessary ingredient of every *reasonable* learning method. But in fact, *every* learning method must necessarily be biased[7]. The fact that all methods of inductive inference are necessarily biased is the essence of the so-called no free lunch theorems (Wolpert, 1996). These theorems have sometimes[8] been interpreted as stating that one can never say that method of inductive inference A is in general better than method of inductive inference B. In our view, this conclusion is much too strong, for a learning method is always defined relative to a model class $\mathcal{M}$, and it is possible to set up the bias of the method such that it *reflects* the bias that is already implicit in the choice of $\mathcal{M}$. Of course, if the model class $\mathcal{M}$ does not contain any good model for the data generating machinery to begin with, we cannot expect our learning method

[6] For a discussion of model selection that takes this idea as a starting point, we refer to (Geman *et al.*, 1992).

[7] To get an idea why, consider the case of binary data: if we are given a sample of size n, then there are only 2^n possibilities for this sample. Hence, whatever inference method we use, it will implicitly use a model class $\mathcal{M}(n)$ for data of size n containing maximally 2^n elements: since the learning method outputs at most one hypothesis for every distinct data set of size n, it has to choose between maximally 2^n hypotheses.

[8] The author has heard several people make such claims at several conferences.

to work well. But *if* $\mathscr{M}$ contains good models for the data D (that achieve small average prediction error on both training set and test set), *then* such models should be output by the learning method, at least if the training set is large enough. In our view, one form or other of this *soundness property* should be satisfied by every method of inductive inference. In the context of model (class) selection, the soundness property (roughly) becomes as follows: first, define a *good model class* as a model class that *contains* models that achieve small average prediction error on both training set and test set. Now *if* the set $\mathscr{S} = \{\mathscr{M}_1, \mathscr{M}_2, ...\}$ of considered model classes contains good model classes for data D, *then* one of those good model classes should be output by the model (class) selection method, at least if the training set is large enough.

We think that soundness properties similar to this one are minimum requirements for all methods of model selection. On the other hand, one can imagine several other, less crucial but still desirable, properties for model selection criteria which are mutually exclusive in the sense that no method of inductive inference satisfies all of them (Rissanen, 1989, p. 95). Summarizing (in our view), there *do* exist good and bad model selection criteria. The theoretical results obtained for MDL (Barron & Cover, 1991; Rissanen, 1987, 1996) strongly suggest that it is among the good ones. On the other hand, one *cannot* say that it is always to be preferred over other methods; several other approaches can also be shown to have good theoretical properties. We do think, however, that whether one uses MDL or not, it is always good to keep MDL's philosophy in mind: the goal of model selection is not the hunt for any true model, but rather for a simple model that gives a reasonable fit to the data; both goodness-of-fit and model complexity contribute to the number of bits needed to describe the data; and finally, a model that is much too complex is typically worthless, while a model that is much too simple can still be useful.

ACKNOWLEDGMENTS

The author would like to thank Richard Shiffrin for financial support. Paul Vitányi, Ronald de Wolf, In Jae Myung, Malcolm Forster and the anonymous referees are to be thanked for providing detailed feedback on earlier versions of this paper. Correspondence and reprint requests should be adressed to Dr. Peter Grünwald, robotics Laboratory, Gates Building 1A, Stanford University, 353 Serra Mall, Stanford, CA 94305-9010. E-mail: grunwald@cs.stanford.edu.

REFERENCES

Barron, A. (1990). Complexity regularization with application to artificial neural networks. In G. Roussas (Ed.), *Nonparametric functional estimation and related topics* (pp. 561–576). Dordrecht: Kluwer Academic.

Barron, A., & Cover, T. (1991). Minimum complexity density estimation. *IEEE Transactions on Information Theory*, **37**(4), 1034–1054.

Berger, J. (1985). *Statistical decision theory and Bayesian analysis* (2nd ed). New York: Springer-Verlag.

Bozdogan, H. (2000). Akaike's information criterion and recent developments in informational complexity. *Journal of Mathematical Psychology*, **44**, 62–91.

Browne, M. (2000). Cross-validation methods. *Journal of Mathematical Psychology*, **44**, 108–132.

Chaitin, G. (1969). On the length of programs for computing finite binary sequences: statistical considerations. *Journal of the Association for Computing Machinery*, **16**, 145–159.

Chater, N. (1996). Reconciling simplicity and likelihood principles in perceptual organization. *Psychological Review* **103**, 566–581.

Cover, T., & Thomas, J. (1991). *Elements of information theory*. New York: Wiley Interscience.

Dawid, A. (1992). Prequential analysis, stochastic complexity and Bayesian inference. In J. Bernardo, J. Berger, A. Dawid, & A. Smith (Eds.), *Bayesian statistics* (Vol. 4, pp. 109–125). Oxford University Press. (Proceedings of the Fourth Valencia Meeting)

Dowe, D., Allison, L., Dix, T., Hunter, L., Wallace, C., & Edgoose, T. (1996). Circular clustering of protein dihedral angles by minimum message length. In *Proceedings Pacific symposium on biocomputing '96*. World Scientific.

Geman, S., Bienenstock, E., & Doversat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, **4**, 1–58.

Grünwald, P. (1996). A minimum description length approach to grammar inference. In G. S. S. Wermter, & E. Riloff (Eds.), *Connectionist, statistical and symbolic approaches to learning for natural language processing* (pp. 203–216). Berlin: Springer-Verlag.

Grünwald, P. (1998). *The MDL principle and reasoning under uncertainty*. Ph.D. thesis, University of Amsterdam, available as ILLC Dissertation Series, DS 1998-03.

Kearns, M., Mansour, Y., Ng, A., & Ron, D. (1997). An experimental and theoretical comparison of model selection methods. *Machine Learning*, **27**, 7–50.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems in Information Transmission*, **1**, 1–7.

Kontkanen, P., Myllymäki, P., & Tirri, H. (1996). Comparing Bayesian model class selection criteria by discrete finite mixtures. In D. Dowe, K. Korb, & J. Oliver (Eds.), *Proceedings of the information, statistics and induction in science (ISIS) conference* (pp. 364–374). Singapore: World Scientific.

Li, M., & Vitányi, P. (1997). *An introduction to Kolmogorov complexity and its applications* (2nd ed.). New York: Springer-Verlag.

Myung, I., & Pitt, M. Issues in selecting mathematical models of cognition. In J. Grainger and A. M. Jacobs (Eds.), *Localist Connectionist Approaches to Human Cognition*. (Hillsdale, NJ: Erlbaum in press.)

Rissanen, J. (1978). Modeling by the shortest data description. *Automatica* **14**, 465–471.

Rissanen, J. (1987). Stochastic complexity. *Journal of the Royal Statistical Society, B*, **49**, 223–239. (Discussion: pages 252–265).

Rissanen, J. (1989). *Stochastic complexity in statistical inquiry*. Singapore: World Scientific.

Rissanen, J. (1996). Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, **42**, 40–47.

Solomonoff, R. (1964). A formal theory of inductive inference, part 1 and part 2. *Information and Control*, **7**, 1–22; 224–254.

Speed, T., & Yu, B. (1993). Model selection and prediction: normal regression. *Annals of the Institute of Statistical Mathematics*, **45**, 35–54.

Vitányi, P., & Li, M. (1997). *Minimum description length induction, Bayesianism, and Kolmogorov complexity*, (submitted).

Wallace, C. (1996). False oracles and SMML estimators. In D. Dowe, K. Korb, & J. Oliver (Eds.), *Proceedings of the information, statistics and induction in science (ISIS) conference* (pp. 304–316). Melbourne, Australia: World Scientific.

Wallace, C., & Boulton, D. (1968). An information measure for classification. *Computing Journal*, **11**, 185–195.

Wallace, C., & Freeman, P. (1987). Estimation and inference by compact coding. *Journal of the Royal Statistical Society B*, **49**, 240–251. (Discussion: pp. 252–265)

Wasserman, L. (2000). Bayesian model selection and model averaging. *Journal of Mathematical Psychology*, **44**, 92–107.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, **8**, 1341–1390.

Received: December 8, 1997; revised: November 4, 1998