# The Power and Perils of MDL

Pieter Adriaans and Paul Vitányi

**Abstract**

Practical use of MDL is full of pitfalls in which the practitioners tend to fall head over heels. We analyse the power and the perils in the use of MDL. Generally, the classical approach is inadequate to express the goodness-of-fit of individual models for individual data sets. In practice however, this is precisely what we are interested in: both to express the goodness of a procedure and where and how it can fail. To achieve this practical goal, we paradoxically have to use the, supposedly impractical, vehicle of Kolmogorov complexity.

## I. Introduction

In machine learning pure applications of MDL are rare, partially because of the difficulties one encounters trying to define an adequate model code and data-to-model code, and partially because of the operational difficulties that are poorly understood. Based on new theoretical insights [16] we elucidate aspects of both the power and the perils of the practical use of MDL precisely and formally. We first resurrect a familiar problem from our childhood to illustrate some of the issues involved.

The process of solving a jigsaw puzzle involves an *incremental reduction of entropy*, and this serves to illustrate the analogous features of the learning problems which are the main issues of this work. Initially, when the pieces come out of the box they have a completely random ordering. Gradually we combine pieces, thus reducing the entropy and increasing the order until the puzzle is solved. In this last stage we have found a maximal ordering. Suppose that Alice and Bob both start to solve two versions of the same puzzle, but that they follow different strategies. In the first stage Alice sorts all pieces according to color. Bob starts to sort the pieces according to shape.[1] The crucial insight, shared by experienced puzzle aficionados, is that Alice's strategy is efficient whereas Bob's is not and is in fact even worse than a random strategy. Alice's strategy is efficient, since the probability that pieces with about the same color match is much greater than the unconditional probability of a match. On the other hand the information about the shape of the pieces can only be used in a relatively late stage of the puzzle process. Bob's effort in the beginning is a waste of time, because he must reorder the pieces before he can proceed to solve the puzzle. This example shows that if the solution of a problem depends on finding a *maximal* reduction of

Supported in part by Perrot Enterprises Netherlands and Syllogic. Address: Department of Computer Science, University of Amsterdam, Plantage Muidergracht 24, 1018TV Amsterdam Pieter.Adriaans@ps.net.

Supported in part by the EU fifth framework project QAIP, IST–1999–11234, the NoE QUIPROCONE IST–1999–29064, the ESF QiT Programme, and the EU Fourth Framework BRA NeuroCOLT II Working Group EP 27150. Address: CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: Paul.Vitanyi@cwi.nl.

[1]For the sake of argument we suppose that the puzzle has no recognizable edge pieces.

entropy this does not mean that *every* reduction of entropy brings us closer to the solution. Consequently reduction of entropy is not in all cases a good strategy.

## *A. Entropy Versus Kolmogorov Complexity*

Above we use “entropy” in the often used, but inaccurate, sense of “measure of unorderedness of an individual arrangement.” However, entropy is a measure of uncertainty associated with a random variable, here a set of arrangements each of which has a certain probability of occurring. The entropy of every individual arrangement is by definition 0. To circumvent this problem, often the notion of “empirical entropy” is used, where certain features like letter frequencies of the individual object are analysed, and the entropy is taken with respect to the set of all objects having the same features. The result obviously depends on the choice of what features to use: no features gives maximal entropy and all features (determining the individual object uniquely) gives entropy 0 again. Unless one has knowledge of the characteristics of a definite random variable producing the object as a typical outcome, this procedure gives arbitrary and presumably meaningless, results. This conondrum arises since classical information theory deals with random variables and the communication of information. It does not deal with the information (and the complexity thereof) in an individual object independent of an existing, or non-existing, random variable producing it. To capture the latter notion precisely one has to use “Kolmogorov complexity” instead of “entropy,” and we will do so in our treatment. For now, the “Kolmogorov complexity” of a file is the number of bits in the ultimately compressed version of the file from which the original can still be losslessly extracted by a fixed general purpose decompression program.

## *B. Learning by MDL*

Transferring the jigsaw puzzling insights to the general case of learning algorithms using the minimal description length principle (MDL), we observe that although it may be true that the maximal compression yields the best solution, it may still not be true that every incremental compression brings us closer to the solution. Moreover, in the case of most MDL problems there is a complicating issue in the fact that the maximal compression cannot be computed.

More formally, in constrained model selection the model is taken from a given model class. Using two-part MDL codes for given data, we assume that the shortest two-part code for the data, consisting of the model code and the data-to-model code, yields the best model for the data. To obtain the shortest code, the natural way is to approximate it by a process of finding ever shorter candidate two-part codes. Since we start with a finite two-part code, and with every new candidate two-part code we decrease the code length, eventually we must achieve the shortest two-part code. Unfortunately, there are two problems: (i) the computation to find the next shorter two-part code may be very long, and we may not know how long; and (ii) we may not know when we have reached the shortest two-part code: with each candidate two-part code there is the possibility that further computation may yield a still shorter one. But because of item (i) we cannot a priori bound the length of that computation.

### *C. A Common Misconception*

Therefore, in practice, we look for ever shorter two-part codes, and if our available time runs out we make do with the last candidate we found. It is commonly thought that this candidate yields a better model than any of its predecessors in the search process: they yielded a longer two-part code and therefore a worse model, didn't they? The underlying assumption is that a shorter two-part code for the data yields a better model than a longer two-part code.

It is the purpose of this paper to debunk this myth: While a sequence of ever shorter two-part codes for the data converges in a finite number of steps to the best model, it is not the case that this convergence is necessarily monotonic. In fact, in the sequence of candidate two-part codes converging to the shortest, it is possible that the models involved oscillate from being good to bad, only to converge at the (unknown) very end to the best model. Convergence is only monotone if the model-codes in the successive two-part codes are always the shortest (most compressed) codes for the models involved. But this property cannot be guarantied by any effective method.

It is very difficult, if not impossible, to formalize the goodness-of-fit of an individual model for individual data in the classic statistics setting, which is probabilistic. Therefore, it is impossible to express the practically important issue above in those terms, which is no doubt one of the reasons why this confusion is rampant. Fortunately, new developments in the theory of Kolmogorov complexity make it possible to rigorously analyse the questions involved, and exhibit the phenomena, in certain model classes, as in [16]. Here we elaborate on that treatment, make it more accessible and probe its implications for MDL. This then is illustrative for what happens in practical situations.

### *D. Statistics and Modeling*

At first glance, a central task of statistics is to identify the true source that produced the data at hand. But suppose the true source is 100,000 fair coin flips and our data is the outcome $00\ldots0$. A method that identifies flipping a fair coin as the cause of this outcome is surely a bad method, even though the source of the data it came up with happens to be the true cause. Thus, for a good statistical method to work well we assume that the data are "typical" for the source that produced the data, so that the source "fits" the data. The situation is more subtle for data like $0101\ldots01$. Here the outcome of the source has an equal frequency of 0s and 1s, just like we would expect from a fair coin. But again, it is virtually impossible that such data are produced by a fair coin flip, or indeed, independent flips of a coin of any particular bias. In real-world phenomena we cannot be sure that the true source of the data is in the class of sources considered, or, worse, we are virtually certain that the true source is not in that class. Therefore, the real question is not to find the true cause of the data, but to model the data as well as possible. In recognition of this, we often talk about "models" instead of "sources," and the contemplated "set of sources" is called the contemplated "model class." In traditional statistics "typicality" and "fitness' are probabilistic notions tied to sets of data and models of large measure. In the Kolmogorov complexity setting we can express and quantify "typicality" of individual data with respect to a single model, and express and quantify the "fitness" of an individual model for the given data.

## E. Strings and Numbers

Let $x, y, z \in \mathcal{N}$, where $\mathcal{N}$ denotes the natural numbers and we identify $\mathcal{N}$ and $\{0,1\}^*$ according to the correspondence

$$(0, \epsilon), (1, 0), (2, 1), (3, 00), (4, 01), \ldots$$

Here $\epsilon$ denotes the *empty word*. The *length* $|x|$ of $x$ is the number of bits in the binary string $x$, not to be confused with the *cardinality* $|S|$ of a finite set $S$. For example, $|010| = 3$ and $|\epsilon| = 0$, while $|\{0,1\}^n| = 2^n$ and $|\emptyset| = 0$. The emphasis is on binary sequences only for convenience; observations in any alphabet can be so encoded in a way that is ‘theory neutral’. Below we will use the natural numbers and the binary strings interchangeably. Definitions, notations, and facts we use about prefix codes, self-delimiting codes, and Kolmogorov complexity, can be found in [15] and are briefly reviewed in the Appendix A.

## F. Meaningful Information

The information contained in an individual finite object (a finite binary string) is measured by its Kolmogorov complexity—the length of the shortest binary program that computes the object. Such a shortest program contains no redundancy: every bit is information; but is it meaningful information? If we flip a fair coin to obtain a finite binary string, then with overwhelming probability that string constitutes its own shortest program. However, also with overwhelming probability all the bits in the string are meaningless information, random noise. On the other hand, let an object $x$ be a sequence of observations of heavenly bodies. Then $x$ can be described by the binary string $pd$, where $p$ is the description of the laws of gravity, and $d$ the observational parameter setting: we can divide the information in $x$ into meaningful information $p$ and accidental information $d$. The main task for statistical inference and learning theory is to distil the meaningful information present in the data. The question arises whether it is possible to separate meaningful information from accidental information, and if so, how. The essence of the solution to this problem is revealed when we rewrite the definition of Kolmogorov complexity (9) in Appendix A as follows:

$$K(x) = \min_{p,i}\{K(i) + |p| : T_i(p) = x\} \tag{1}$$

This expression emphasizes the two-part code nature of Kolmogorov complexity. For example

$$x = 10101010101010101010101010$$

we can encode $x$ by a small Turing machine printing copies of the pattern “01,” which computes $x$ from the program “13.” This way, $K(x)$ is viewed as the shortest length of a two-part code for $x$, one part describing a Turing machine, or *model*, for the *regular* aspects of $x$, and the second part describing the *irregular* aspects of $x$ in the form of a program to be interpreted by $T$. The regular, or “valuable,” information in $x$ is constituted by the bits in the “model” while the random or “useless” information of $x$ constitutes the remainder.

## II. Data and Model

Because all finite discrete data can be binary coded, we consider only *data* $x \in \{0,1\}^*$. In a typical statistical inference situation we are given a set $D \subseteq \{0,1\}^*$ of such data, the *data sample*, and are required to infer a *model* for the data sample. Our models will be sets $M \subseteq \{0,1\}^*$, and for $M$ to be a model for $D$ we require that $D \subseteq M$. Instead of $\{0,1\}^*$ we will consider $\{0,1\}^{\leq n} = \bigcup_{i=0}^{n} \{0,1\}^{\leq i}$ in the sequel.

### A. What Does It Mean That A Model Fits Given Data

Denote the *complexity of the finite set* $A$ by $K(A)$—the length (number of bits) of the shortest binary program $p$ from which the reference universal prefix machine $U$ computes a listing of the elements of $S$ and then halts. That is, if $A = \{x_1, \ldots, x_d\}$, then $U(p) = \langle x_1, \langle x_2, \ldots, \langle x_{d-1}, x_d \rangle \ldots \rangle\rangle$. The shortest program $p$, or, if there is more than one such shortest program, then the first one that halts in a standard dovetailed running of all programs, is denoted by $A^*$. Consider a data sample $D$ and a model $M$, such that $D \subseteq M \subseteq \{0,1\}^{\leq n}$. Denote the cardinalities by lower case letters:

$$d = |D|, m = |M|.$$

The *conditional complexity* $K(D \mid M, d))$ of $D$ given $M$ and $d$ is the length (number of bits) in the shortest binary program $p$ from which the reference universal prefix machine $U$ from input $M$ (given as a list of elements) and the number of elements $d$, outputs $D$ as a list of elements and halts. We elaborate on the approach of [16]. If $D \subseteq M \subseteq \{0,1\}^{\leq n}$ we have

$$K(D \mid M, d)) \leq \log \binom{m}{d} + O(1). \tag{2}$$

Indeed, consider the selfdelimiting code of $D$, given $M$ and the number $d$ of elements in $D$ followed by the $\lceil \log \binom{m}{d} \rceil$ bit long index of $D$ in the lexicographical ordering of the number of ways to choose $d$ elements from $M$. This code is called the *data-to-model code*. Its length quantifies the maximal "typicality," or "randomness," any data sample of $|D|$ elements can have with respect to model $M$.

Definition 1: The lack of typicality of $D$ with respect to $M$ is measured by the amount by which $K(D \mid M, d)$ falls short of the length of the data-to-model code. The *randomness deficiency* of a data sample $D$, of known cardinality $d = |D|$, in model $M$, is defined by

$$\delta(D \mid M, d) = \log \binom{m}{d} - K(D \mid M, d), \tag{3}$$

for $D \subseteq M$, and $\infty$ otherwise.

If the randomness deficiency is close to 0, then there are no simple special properties that single $D$ out from the majority of data samples to be drawn from $M$. This is not just terminology: If $\delta(D \mid M, d)$ is small enough, then $D$ satisfies *all* properties of low Kolmogorov complexity that hold for the majority of subsets of $M$. To be precise: A *property* $P$ represented by $M$ is a subset of $M$, and we say that $D$ satisfies property $P$ if $D \subseteq P$.

Lemma 1: Let $d, m, n$ be natural numbers, and let $D \subseteq M \subseteq \{0,1\}^{\leq n}$, $|D| = d, |M| = m$, and let $\delta$ be a simple function of the natural numbers to the reals, like $\log$ or $\sqrt{}$.

(i) If $P$ is a property satisfied by all $D \subseteq M$ with $\delta(D \mid M, d) \leq \delta(n)$, then $P$ holds for a fraction of at least $1 - 1/2^{\delta(n)}$ of the subsets of cardinality $d$ of $M$.

(ii) Let $n$ and $M$ be fixed, and let $P$ be any property that holds for a fraction of at least $1 - 1/2^{\delta(n)}$ of the subsets of cardinality $d$ of $M$. There is a constant $c$, such that every such $P$ holds simultaneously for every $D \subseteq M$ with $|D| = d$ and $\delta(D \mid M, d) \leq \delta(n) - K(P \mid M) - c$.

*Proof:* (i) By assumption, all subsets $D \subseteq M$ of cardinality $d$ with

$$K(D|M,d) \geq \log \binom{m}{d} - \delta(n) \tag{4}$$

satisfy $P$. There are only

$$\sum_{i=0}^{\log \binom{m}{d} - \delta(n) - 1} 2^i = \binom{m}{d} 2^{-\delta(n)} - 1$$

programs of length smaller than $\log \binom{m}{d} - \delta(n)$, so there are at most that many subsets $D$ of cardinality $d$ that do not satisfy (4). There are $\binom{m}{d}$ subsets of cardinality $d$ in $M$, and hence a fraction of at least $1 - 1/2^{\delta(n)}$ of them satisfy (4).

(ii) Suppose $P$ does not hold for a data sample $D \subseteq M$ and the randomness deficiency satisfies $\delta(D|M,d) \leq \delta(n) - K(P|M) - c$. Then we can reconstruct $D$ from a description of $P$, which can use $M$, and $D$'s index $j$ in an effective enumeration of all subsets of cardinality $d$ in $M$ for which $P$ doesn't hold. There are at most $\binom{m}{d}/2^{\delta(n)}$ such data samples by assumption, and therefore there are constants $c_1, c_2$ such that

$$K(D \mid M, d) \leq \log j + c_1 \leq \log \binom{m}{d} - \delta(n) + c_2.$$

Hence, by the assumption on the randomness deficiency of $D$, we find $K(P|M) \leq c_2 - c$, which contradicts the necesssary nonnegativity of $K(P|M)$ if we choose $c > c_2$. ■

The *minimal randomness deficiency* function (with known cardinality of the data sample) is

$$\beta_D(\alpha) = \min_M \{\delta(D \mid M, d) : M \supseteq D,\ K(M|d) \leq \alpha\}, \tag{5}$$

where we set $\min \emptyset = \infty$. The smaller $\delta(D \mid M, d)$ is, the more $D$ can be considered as a *typical* data sample from $M$. This means that a set $M$ for which $D$ incurs minimal randomness deficiency, in the model class of contemplated sets of given maximal Kolmogorov complexity, is a "best fitting" model for $D$ in that model class—a most likely explanation, and $\beta_D(\alpha)$ can be viewed as a *constrained best fit estimator*.

REMARK 1 (KNOWN OR UNKNOWN CARDINALITY): In [16] only singleton data samples are considered, therefore always $d = 1$. In the current setting we have multiple data, and we have assumed that the cardinality $d$ of the data sample is known. Formally, this is represented by putting $d$ in the conditional everywhere, as in $K(D \mid M, d)$ and $\delta(D|M,d)$. In some cases it may be more natural to assume that the cardinality of the data sample is unknown. This results in the same theory with minor changes, as follows:

$$K(D \mid M)) \leq m + O(1)$$

$$\delta'(D \mid M) = m - K(D \mid M),$$

and changing $\binom{m}{d}$ to $2^m$, and "subsets of cardinality $d$" into "subsets" in the statement of the lemma above and its proof. The definition of $\beta_D$ in (5) changes to the unconditional (on $d$)

$$\beta'_D(\alpha) = \min_M \{\delta(D \mid M) : M \supseteq D,\ K(M,d) \leq \alpha\},$$

It is a trivial exercise to similarly adapt all of this paper to the case of data samples of unknown cardinality.

REMARK 2 (LOSSY COMPRESSION): The function $\beta_D(\alpha)$ is relevant to lossy compression (used, for instance, to compress images) as was observed in [16]. Assume we need to compress $D$ to $\alpha$ bits where $\alpha \ll K(D)$. Of course this implies some loss of information present in $D$. One way to select redundant information to discard is as follows: Find a set $M \supseteq D$ with $K(M) \leq \alpha$ and with small $\delta(D|M,d)$, and consider a compressed version $M^*$ of $M$. Instead of reconstructing $D$ precisely, we are satisfied to reconstruct an object that cannot be distinguished from $D$ using both $M$ and simple properties. To this purpose, a decompresser uncompresses $M^*$ to $M$ and selects uniformly at random a data sample $D'$ from $M$. Since with high probability the randomness deficiency of $D'$ in $M$ is small, $D'$ serves the purpose of the data sample $D$ as well as does $D$ itself. Let us look at an example. To transmit a picture of "rain" through a channel with limited capacity $\alpha$, one can transmit the indication that this is a picture of the rain and the particular drops may be chosen by the receiver at random. In this interpretation, $\beta_D(\alpha)$ indicates how "random" or "typical" $D$ is with respect to the best model at complexity level $\alpha$—and hence how "indistinguishable" from the original $D$ the randomly reconstructed $D'$ can be expected to be. The relation of the structure function to lossy compression and rate-distortion theory is the subject of an upcoming paper [12].

## III. MINIMUM DESCRIPTION LENGTH ESTIMATOR

The length of the minimal two-part code for $D$, with known cardinality $d = |D|$ consisting of the model cost $K(M)$ ($|M| = m$) and the length of the index of $D$ in the enumeration of choices of $d$ elements out of $m$, in the model class of sets $M$ of given maximal Kolmogorov complexity $\alpha$, the complexity of $M$ upper bounded by $\alpha$, is given by the *MDL* function or *constrained MDL estimator*:

$$\lambda_D(\alpha) = \min_M \{\Lambda(M) : M \supseteq D,\ K(M|d) \leq \alpha\}, \tag{6}$$

where $\Lambda(M) = K(M|d) + \log \binom{m}{d} \geq K(D|d) - O(1)$ is the total length of two-part code of $D$ with help of model $M$ and the cardinality $d$. This function $\lambda_D(\alpha)$ is the celebrated two-part Minimum Description Length code length as a function of $\alpha$, with the model class restricted to models of code length at most $\alpha$. Indeed, consider the following *two-part code* for $D$ when we know its cardinality $d$: the first part is a shortest self-delimiting program $p$ for $M$ and the second part is $\lceil \log \binom{m}{d} \rceil$ bit long index of $D$ in the lexicographical ordering of all choices of $d$ elements from $M$. Since $M, d$ determines $\log \binom{m}{d}$ this code is self-delimiting and we obtain the two-part code, where the constant $O(1)$ is the length of the program to reconstruct $D$ from its two-part code and known cardinality

$d$. [2] For those $\alpha$'s that have $\lambda_D(\alpha) = K(D|d) + O(1)$, the associated model $M$ (witness for $\lambda_D(\alpha)$) for $D$, or the description of $M|d$ of $\leq \alpha$ bits, is called a *sufficient statistic*.

LEMMA 2: If $M$ is a sufficient statistic for $D$, then the randomness deficiency of $D$ in $M$ is $O(1)$, $D$ is a typical data sample for $M$, and $M$ is a model of best fit for $D$.

*Proof:* If $M$ is a sufficient statistic for $D$, then $K(M|d) + \log \binom{m}{d} = K(D|d) + O(1)$. That is, the two-part description of $D$ using the model $M, d$ and as data-to-model code the index of $D$ in the enumeration of the number of choices of $d$ elements from $M$ in $\log \binom{m}{d}$ bits, is as concise as the shortest one-part code of $D$, given its cardinality $d$, in $K(D|d)$ bits. Therefore,

$$\begin{aligned} K(D|d) &\leq K(D, M|d) + O(1) \\ &\leq K(M|d) + K(D \mid M, d) + O(1) \\ &\leq K(M|d) + \log \binom{m}{d} + O(1) = K(D|d) + O(1). \end{aligned}$$

To see this, use straightforward inequalities (for example, given $M \supseteq D$ and $D$, we can describe $D$ self-delimitingly in $\log \binom{m}{d} + O(1)$ bits), and the sufficiency property. This sequence of inequalities implies that $K(D \mid M, d) = \log \binom{m}{d} + O(1)$. ■

REMARK 3 (SUFFICIENT BUT NOT TYPICAL): Note that the data sample $D$ can have randomness deficiency about 0, and hence be a typical element for models $M$, while $M$ is not a sufficient statistic. A sufficient statistic $M$ for $D$ has the additional property, apart from being a model of best fit, that $K(D, M \mid d) = K(D \mid d) + O(1)$ and therefore by (11) in Appendix A we have $K(M|D^*) = O(1)$: the sufficient statistic $M$ is a model of best fit that is almost completely determined by $D^*$, the shortest program for $D$.

REMARK 4 (MINIMAL SUFFICIENT STATISTIC): The sufficient statistic associated with $\lambda_D(\alpha)$ with the least $\alpha$ is called the *minimal sufficient statistic*.

REMARK 5 (KNOWN OR UNKNOWN CARDINALITY): We continue Remark 1. If, instead of assuming that the cardinality $d$ of the data sample $D$ is known, we assume it is unknown, then the definition of $\lambda_D$ of (6) changes to the unconditional (on $d$):

$$\lambda'_D(\alpha) = \min_M \{\Lambda'(M) : M \supseteq D,\ K(M, d) \leq \alpha\},$$

where $\Lambda'(M) = K(M, d) + \log \binom{m}{d} \geq K(D) - O(1)$ is the total length of two-part code of $D$ with help of model $M$ and the cardinality $d$.

REMARK 6 (PROBABILITY MODELS): Following Kolmogorov we analyzed a canonical setting where the models are finite sets (and here we have generalized Kolmogorov's singleton data samples into multiple-item data samples). As Kolmogorov himself pointed out, the finite sets model class is equivalent, up to a logarithmic additive term, to

[2]The original Kolmogorov *structure* function $h_D$ [8], [16] was defined for a singleton data sample $D = \{x\}$ ($d = 1$) as $h_x(\alpha) = \min_M\{\lceil \log |M| \rceil : M \ni x,\ K(M) \leq \alpha\}$, and it was shown in [16] that the graph of $h_x(\alpha) + \alpha$ is situated within a strip of logarithmic (in $n$) width centered on the graph of $\lambda_x(\alpha)$. Thus, $h_x(\alpha)$ can be viewed as the minimal data-to-model code length associated with models of maximal description length $\alpha$.

the model class of probability density functions, as studied in [11]. Formal proofs of this equivalence are given (for the singleton data-sample case) in [16]. In this case, the models are computable probability mass functions, and $\lambda_P(\alpha) = \min_P\{K(P \mid d) + \log 1/P(D) : P(D) > 0$ and $P$ is a computable probability mass function with $K(P|d) \leq \alpha\}$.

## IV. Essence of Model Selection

The first parameter we are interested in is the *simplicity* $K(M \mid d)$ of the model $M$ explaining the data sample $D$ when we know cardinality $d = |D|$. The second parameter is *how typical* the data is with respect to $M$, expressed by the randomness deficiency $\delta(D \mid M, d) = \log \binom{m}{d} - K(D \mid M, d)$. The third parameter is how *short the two part code* $\Lambda(M) = K(M \mid d) + \log \binom{m}{d}$ of the data sample $D$, of cardinality $d$, using theory $M$ is. The second part consists of the full-length index, ignoring savings in code length using possible non-typicality of $D$ in $M$ (like being the first $d$ elements in the enumeration of $M$). These parameters induce a partial order on the contemplated set of models. We write $M_0 \leq M_1$, if the $M_0$ scores equal or less than $M_1$ in all three parameters. If this is the case, then we may say that $M_0$ is at least as good as $M_1$ as an explanation for $D$ (although the converse need not necessarily hold, in the sense that it is possible that $M_0$ is at least as good a model for $D$ as $M_1$ without scoring better than $M_1$ in all three parameters simultaneously.).

The algorithmic statistical properties of a data sample $D$ are fully represented by the set $A_x$ of all triples

$$\langle K(M \mid d), \delta(D \mid M, d), \Lambda(M)\rangle,$$

such that $M \supseteq D$, together with a component wise order relation on the elements of those triples. The complete characterization of how this set may look like for the singleton data sample case, $d = 1$, is now known by the results in [16]. These results are obtained by a proof of, and analysis, of the equality (writing $D = \{x\}$ as $x$):

$$\beta_x(\alpha) = \lambda_x(\alpha) - K(x) \tag{7}$$

which holds within negligible additive logarithmic (in $n$) terms, in argument and value. Every set $M$ that witnesses the value $\lambda_x(\alpha)$, also witnesses the value $\beta_x(\alpha)$ (but not vice versa). The functions $\lambda_x$ and $\beta_x$ can assume all possible shapes over their full domain of definition (up to additive logarithmic precision in both argument and value). We summarize the relevant formal statements in Appendix B

## V. Power and Pitfalls

The previous analysis of MDL allows us to show justify its application and to identify problems in its application that may not be apparent at first glance.

### A. Computability

How difficult is it to compute the functions $\lambda_D, \beta_D$, and the minimal sufficient statistic? To express the properties appropriately we require the notion of functions that are not computable, but can be approximated monotonically by a computable function.

DEFINITION 2: A function $f : \mathcal{N} \rightarrow \mathcal{R}$ is *upper semi-computable* if there is a Turing machine $T$ computing a total function $\phi$ such that $\phi(x, t+1) \leq \phi(x,t)$ and $\lim_{t \rightarrow \infty} \phi(x,t) = f(x)$. This means that $f$ can be computably approximated from above. If $-f$ is upper semi-computable, then $f$ is lower semi-computable. A function is called *semi-computable* if it is either upper semi-computable or lower semi-computable. If $f$ is both upper semi-computable and lower semi-computable, then we call $f$ *computable* (or recursive if the domain is integer or rational).

To put matters in perspective: even if a function is computable, the most feasible type identified above, this doesn't mean much in practice. Functions like $f(x)$ of which the computation terminates in computation time of, say measured in flops, of $t(x) = x^x$ are in among the easily computable ones. But for $x = 30$, even a computer performing an unrealistic Teraflop per second, requires $30^{30}/10^{12} > 10^{28}$ seconds. This is more than $3 \cdot 10^{20}$ years. It is out of the question to perform such computations. Thus, the fact that a function or problem solution is computable gives no insight in how *feasible* it is. But there are worse functions and problems possible: For example, the ones that are semi-computable but not computable. Or still worse, functions that are not even semi-computable.

Semi-computability gives no knowledge-of-convergence guaranties: even though the limit value is monotonically approximated we know at no stage in the process how close we are to the limit value. In Section V-C, the "indirect method" shows that the function $\lambda_D$ (the MDL-estimator) can be monotonically approximated in the upper semi-computable sense. But in [16] it was shown for singleton data samples $D$, and therefore *a fortiori* for multiple data samples $D$, the fitness function $\beta_D$ (the "direct method" in Section V-C) cannot be monotonically approximated in that sense, nor in the lower semi-computable sense, in both cases not even up to any relevant precision. Let us formulate this a little more precisely:

The functions $\lambda_D(\alpha), \beta_D(\alpha)$ have finite domain for given $D$ and hence can be given as a table—so formally speaking they are computable. But this evades the issue: there is no algorithm that computes these functions for given $D$ and $\alpha$. Considering them as two-argument functions it was shown (and the claimed precision quantified):

- The function $\lambda_D(\alpha)$ is upper semi-computable but not computable up to any reasonable precision.
- Moreover, there is no algorithm that given $D^*$ and $\alpha$ finds $\lambda_D(\alpha)$.
- The function $\beta_D(\alpha)$ is not upper- or lower semi-computable, not even to any reasonable precision, but we can compute it given an oracle for the halting problem. [3]
- There is no algorithm that given $D$ and $K(D)$ finds a minimal sufficient statistic for $D$ up to any reasonable precision.

## B. Invariance under Recoding of Data

In what sense are the functions invariant under recoding of the data? If the functions $\beta_D, \lambda_D$ give us the stochastic properties of the data $D$, then we would not expect those properties to change under recoding of the data

[3]The problem of whether an arbitrary given Turing machine started on an initially all-0 tape will eventually terminate or compute forever. This problem was shown to be undecidable by A.M. Turing in 1937, see for example [15]. An oracle for the halting problem will, when asked, tell whether a given Turing machine computation will or will not terminate. Such a device is assumed in order to determine theoretical degrees of noncomputability, and is deemed not to exist.

into another format. Yet, if we recode the elements of $D = \{x_1, \ldots, x_d\}$ by a mapping $c$ of $\{0,1\}^{\leq n}$ to obtain $c(D) = \{c(x_1), \ldots, c(x_d)\}$ such that $c(x_i) = x_i^*$ with $|x_i^*| = K(x_i)$ ($1 \leq i \leq d$), then we are in trouble. We can choose the $x_i$'s such that $K(c(D) \mid \{0,1\}^{\leq \mu}, d) \approx \binom{2^{\mu+1}-1}{d}$, with $\mu = \max\{K(c(x_i)) : 1 \leq i \leq d\}$. Then,

$$\delta(c(D) \mid \{0,1\}^{\leq \mu}, d) \approx 0.$$

That is, $c(D)$ is a typical $d$-element subset of $\{0,1\}^{\leq \mu}$, and the latter in turn is the best fitting model for $c(D)$. Therefore $\lambda_{c(D)}(\alpha)$ drops to the Kolmogorov complexity $K(c(D))$ already for some $\alpha \leq K(\mu) + O(1) = O(\log n)$, since $\lambda_{c(D)}(K(\mu)+O(1)) = K(\{0,1\}^{\leq \mu} \mid d) + \log \binom{2^{\mu+1}-1}{d} \approx K(c(D))$, so almost immediately (and it stays within logarithmic distance of that line henceforth). That is, $\lambda_{c(D)}(\alpha) = K(x_1^*, \ldots, x_d^*)$ for every $\alpha$, up to logarithmic additive terms in argument and value, irrespective of the (possibly quite different) shape of $\lambda_D$. It is clear that a coding $c$ that achieves this is not a recursive function, and neither is the inverse. However, it is not the non-recursiveness alone, but also the necessary partiality of the inverse function (not all data samples contain data of maximal Kolmogorov complexity) that causes the collapse of the structure function. Nonetheless, the structure function is invariant under "proper" recoding of the data, as follows:

THEOREM 1: Let $f$ be a recursive permutation of the set of finite binary strings in $\{0,1\}^n$ (one-one, total, and onto), and extend $f$ to subsets $D \subseteq \{0,1\}^n$. Then, $\lambda_{f(D)}$ is "close" to $\lambda_D$ in the sense that the graph of $\lambda_{f(D)}$ is situated within a strip of width $K(f) + O(1)$ around the graph of $\lambda_D$.

*Proof:* Let $M \supseteq D$ be a witness of $\lambda_D(\alpha)$. Then, $M_f = \{f(y) : y \in M\}$ satisfies $K(M_f) \leq \alpha + K(f) + O(1)$ and $|M_f| = |M|$. Hence, $\lambda_{f(D)}(\alpha + K(f) + O(1)) \leq \lambda_D(\alpha)$. Let $M' \supseteq f(D)$ be a witness of $\lambda_{f(D)}(\alpha)$. Then, $M'_{f^{-1}} = \{f^{-1}(y) : y \in M'\}$ satisfies $K(M'_{f^{-1}}) \leq \alpha + K(f) + O(1)$ and $|M'_{f^{-1}}| = |M'|$. Hence, $\lambda_D(\alpha + K(f) + O(1)) \leq \lambda_{f(D)}(\alpha)$ (since $K(f^{-1}) = K(f) + O(1)$). ■

## C. Finding the MDL Code

Given $D \subseteq \{0,1\}^n$, the data to explain, and the model class consting of all models (sets) $M \subseteq \{0,1\}^n$ that have complexity at most $K(M) \leq \alpha$. Here, $\alpha$ is the maximum complexity of an explanation we allow. As usual, we denote $m = |M|$ and $d = |D|$. We search for programs $p$ of length at most $\alpha$ that print a finite set $M \supseteq D$. Such pairs $(p, M)$ are possible explanations. The *best explanation* is defined to be the $(p, M)$ for which $\delta(D \mid M, d)$ is minimal. Since the function $\delta(D \mid M, d)$ is not computable, there is no algorithm that halts with the best explanation. The programs use unknown computation time and thus we can never be certain that we have found all possible explanations. Following [16], we can overcome this problem by using an indirect method. Initially, we are given data sample $D$.

*Indirect Method:* We minimize the randomness deficiency by minimizing the MDL code length, justified by (7), and thus maximizing the fitness of the model for this data sample. To this end, run all programs dovetailed fashion. If a program, say $p$, halts, then check its output to see whether it is a subset, say $M$, of $\{0,1\}^n$ in agreed-upon standard notation. If so, then check whether that subset contains all elements in the data sample $D$. At every computation step $t$ consider all pairs $(p, M)$ such that program $p$ has printed the set $M$ containing $D$ by time $t$.

Let $(p_t, L_t)$ stand for the pair $(p, M)$ such that $|p| + \log \binom{m}{d}$ is minimal among all these pairs $(p, M)$. The best hypothesis $L_t$ changes from time to time due to the appearance of a better hypothesis. Since no hypothesis is selected as the best one twice, from some moment onwards the explanation $(p_t, L_t)$ which is best does not change anymore. Compare this indirect method with the direct one:

*Direct Method:* This time, we directly minimize the randomness deficiency, as follows: Run all programs dovetailed fashion. After step $t$ of the dovetailing process, select $(p, M)$ for which $\log \binom{m}{d} - K^t(D|M, d)$ is minimal among all programs $p$ that up to this time have printed a set $M$ containing $D$. Here $K^t(D|M, d)$ is the approximation of $K(D|M, d)$ obtained after $t$ steps of dovetailing, that is, $K^t(D|M, d) = \min\{|q| :$ $U$ on input $\langle q, \langle M, d\rangle\rangle$ prints $D$ in at most $t$ steps$\}$. Let $(q_t, B_t)$ stand for that model. This time the same hypothesis can be selected as best hypothesis twice. However from some moment onwards the explanation $(q_t, B_t)$ which is best does not change anymore.

The reason why we prefer the indirect method over the direct one in the theoretical setting is similar to a comparable situation in the practice of the real-world MDL, in the analogous process of finding the MDL code in the real setting. There, we often deal with $t$ that are much less than the time of stabilization of either $L_t$ or $B_t$, both the indirect method and the direct method. And for these small $t$, the model $L_t$ is better than $B_t$ in the following respect: $L_t$ has some guarantee of goodness, since we know that

$$\delta(D \mid L_t, d) + K(D|d) \leq |p_t| + \log \binom{|L_t|}{d} + O(1), \tag{8}$$

since $K(D|d) - K(D \mid L_t, d) \leq K(L_t \mid d) \leq |p_t|$ (ignoring additive constants). That is, we know that the sum of the randomness deficiency of $D$ in $L_t$ and $K(D|d)$ is less than some known value. This idea will be expanded in Theorem 2 below. In contrast, the model $B_t$ has no guarantee of goodness at all: we do not know any upper bound neither for $\delta(D \mid B_t, d)$, nor for $\delta(D \mid B_t, d) + K(D)$.

The results in [16] we have quoted above, and expand in Appendix B, imply that the indirect method of MDL gives not only some garantee of goodness but also that, in the limit, that guarantee approaches the value it upper bounds, that is, approaches $\delta(D \mid L_t, d) + K(D)$, and $\delta(D \mid L_t, d)$ itself is not much greater than $\delta(D \mid B_t, d)$.[4] That is, in the limit, the method of MDL will yield an explanation that is only a little worse than the best explanation.

## VI. Does Shorter MDL Code Mean Better Model?

Thus, if we continue to approximate the MDL code then we will eventually reach the optimal code which is approximately the best explanation at the given model complexity. During this process we have some guarantee of goodness as in (8). That is the good news. The bad news is, that we do not know when we have reached this optimal solution, and the noncomputability of computing $\lambda_D$ to a given precision assures us that there simply does not exist a convergence criterion we could use to terminate the approximation somewhere close to the optimum. The known upper bound (8) on the randomness deficiency of each approximation is noncomputable. Thus, in practice we must

[4]Assuming that $\alpha$ is not critical as in Appendix B.

terminate the search prematurely. A natural assumption is that the longer we approximate the optimal MDL code the better the resulting model explains the data. Thus, many practitioners simply assume that if one approximates the MDL code, than every next shorter MDL code also yields a better model. Alas, this is not true. To give an example that shows where things go wrong it is easiest to first give the conditions under which premature search termination is all right, slightly correcting an idea first given in [16].

Assume that in the indirect MDL algorithm, as described in Section V-C, we change the currently best explanation $(p_1, M_1)$ for data $D$ ($|D| = d$) to the explanation $(p_2, M_2)$ only if $|p_2| + \log \binom{|M_2|}{d}$ is much less than $|p_1| + \log \binom{|M_1|}{d}$, say $|p_2| + \log \binom{|M_2|}{d} \leq |p_1| + \log \binom{|M_1|}{d} - c' \log\log \binom{2^n}{d}$ for a constant $c'$. We show: if $c'$ is large enough and $p_1$ is a shortest program of $M_1, d$, then $\delta(D \mid M_2, d)$ is less than $\delta(D \mid M_1, d)$. That is, every time we change the explanation we improve its goodness unless the change is just caused by the fact that we have not yet found the minimum length program for the current model.

THEOREM 2: Let $(p_1, M_1)$ and $(p_2, M_2)$ be two consecutive candidate best explanations in the search process for the best model for data sample $D$ ($|D| = d$, $0 < d < 2^n$) above. There is a constant $c$ such that if $|p_2| + \log \binom{|M_2|}{d} \leq |p_1| + \log \binom{|M_1|}{d} - (|p_1| - K(M_1)) - 2c \log\log \binom{2^n}{d}$ then $\delta(D \mid M_2, d) \leq \delta(D \mid M_1, d) - c \log\log \binom{2^n}{d} + O(1)$.

*Proof:* For every pair of sets $M_1, M_2 \supseteq D$ we have

$$\delta(D \mid M_2, d) - \delta(D \mid M_1, d) = \Lambda(M_2) - \Lambda(M_1) + \Delta,$$

with

$$\begin{aligned}\Delta &= K(M_2 \mid d) + K(D \mid M_2, d) - K(M_1 \mid d) + K(D \mid M_1, d)\\ &\leq K(M_1, D \mid d) - K(M_2, D \mid d) + O(1) \leq K(M_1 \mid M_2, D) + O(1).\end{aligned}$$

Since

$$\begin{aligned}\Lambda(M_2) - \Lambda(M_1) &\leq |p_2| + \log \binom{|M_2|}{d} - \Lambda(M_1)\\ &= |p_2| + \log \binom{|M_2|}{d} - (|p_1| + \log \binom{|M_1|}{d}) + (|p_1| - K(M_1))\\ &\leq -2c \log\log \binom{2^n}{d},\end{aligned}$$

we need to prove that $K(M_2 \mid M_1, D) \leq c \log\log \binom{2^n}{d} + O(1)$. Note that $(p_1, M_1)$, $(p_2, M_2)$ are consecutive explanations in the algorithm and every explanation may appear only once. Hence to identify $M_1$ we only need to know $p_2, M_2, \alpha$ and $D$. Since $p_2$ may be found from $M_2$ and length $|p_2|$ as the first program computing $M_2$ of length $|p_2|$, obtained by running all programs of length at most $\alpha$ dovetailed style, we have $K(M_2 \mid M_1, D) \leq 2 \log |p_2| + 2 \log |\alpha| + O(1) \leq 4 \log\log \binom{2^n}{d} + O(1)$. Hence we can choose $c = 4$. ■

Thus, to be sure that in the sequence $(p_1, M_1), (p_2, M_2), \ldots$ of candidate explanations of ever shorter MDL codes the explanation $(p_{i+1}, M_{i+1})$ is actually a better explanation for the data than the preceding $(p_i, M_i)$, it suffices that $|p_{i+1}| + \log \binom{|M_{i+1}|}{d} \leq |p_i| + \log \binom{|M_i|}{d} - (|p_i| - K(M_i)) - 2c \log\log \binom{2^n}{d}$. The unknown, and in general noncomputable, quantification of the required improvement in MDL code length is $|p_i| - K(M_i)$. If we have an hypothesis $M_i$ encoded by a program $p_i$ that is far from optimal, then the slack in model code length given by $|p_i| - K(M_i)$ is large, and it is possible that we improve the MDL code length by giving a worse hypothesis $M_{i+1}$ using, however, an encoding $p_{i+1}$ that is shorter than the encoding $p_i$ of the previous candidate $M_i$. Thus,

COROLLARY 1: (i) On the one hand, if $|p_{i+1}| + \log \binom{|M_{i+1}|}{d} \leq |p_i| + \log \binom{|M_i|}{d} - 2c \log\log \binom{2^n}{d}$ and $|p_i| = K(M_i) + O(1)$, then $M_{i+1}$ is a better explanation for data $D$ than is $M_i$, in the sense that $\delta(D \mid M_{i+1}, d) \leq \delta(D \mid M_i, d) - 4 \log\log \binom{2^n}{d} + O(1)$.

(ii) On the other hand, if $|p_i| - K(M_i)$ is large, then $M_{i+1}$ may be a much worse explanation than $M_i$ as we show with some examples below.

## VII. SINGLETON DATA SAMPLE

Assume that we want to infer a language, given a single positive example (element of the language). The positive example is $D = \{x\}$ with $x = x_1 x_2 \ldots x_n$, $x_i \in \{0,1\}$ for $1 \leq i \leq n$. For convenience we restrict the question to inferring a language consisting of a set of elements of the same length as the positive example, that is, we infer a subset of $\{0,1\}^n$. We can view this as inferring the slice $L^n$ of the (possibly infinite) target language $L$ consisting of all words of length $n$ in the target language. We also identify the singleton data sample $D$ with the constituent data string $x$.

Each $M \subseteq \{0,1\}^n$ can be represented by its characteristic sequence $\chi = \chi_1 \ldots \chi_{2^n}$ with $\chi_i = 1$ if the $i$th element of $\{0,1\}^n$ is in $M$, and 0 otherwise. Conversely, every string of $2^n$ bits is the characteristic sequence of a subset of $\{0,1\}^n$. Most of these subsets are 'random' in the sense that they cannot be represented concisely: their characteristic sequence is incompressible. Now choose some integer $\delta$. Simple counting tells us that there are only $2^{2^n - \delta} - 1$ binary strings of length $< 2^n - \delta$. Thus, the number of possible binary programs of length $< 2^n - \delta$ is at most $2^{2^n - \delta} - 1$. This in turn implies (since every program describes at best one such set) that the number of subsets $M \subseteq \{0,1\}^n$ with $K(M) < 2^n - \delta$ is at most $2^{2^n - \delta} - 1$. This implies that the number of subsets $M \subseteq \{0,1\}^n$ with

$$K(M) \geq 2^n - \delta$$

is greater than

$$(1 - 1/2^{\delta}) 2^{2^n}.$$

Now if $K(M)$ is significantly greater than $K(x)$, then it is impossible to learn $M$ from $x$. This follows already from the fact that $K(M|x) \geq K(M|x^*) + O(1) = K(M) - K(x) + K(x|M^*) + O(1)$ by (11) (note that $K(x|M^*) > 0$). That is, we need more than $K(M) - K(x)$ extra bits of dedicated information to deduce $M$ from $x$. We can conclude that almost all sets in $\{0,1\}^n$ have such great complexity that no effective procedure can infer this set from a single example. This holds in particular for every (even moderately) random set.

Thus, to infer such a set $M \subseteq \{0,1\}^n$, given a sample datum $x \in M$, using the MDL principle is clearly out of the question. The datum $x$ can be literally described in alread $n$ bits by the trivial MDL code $M = \{x\}$ with $x$ literal at model cost about $n$ bits and data-to-model cost $\log |M| = 0$, which we take to be $O(1)$ bits. It can be concluded that the only sets $M$ that can possibly be inferred from $x$ (using MDL or another effective deterministic procedure) are those that have $K(M) \leq K(x) \leq n + O(\log n)$. Such sets are extremely rare: only a $1/2^{n-\log n}$ fraction of all subsets of $\{0,1\}^n$ has that small complexity. This negligible fraction of possibly learnable subsets are very non-random, simple in the sense that their characteristic sequences have great regularity (otherwise the Kolmogorov complexity could not be this small). But this is all right: we do not want to learn random, meaningless, languages, but only languages that have meaning which is necessarily expressed in terms of regularity. We now show that even if we can learn the target model by an MDL algorithm in the limit, selecting a sequence of models that decrease the MDL code with each next model, a later model in this sequence can be a worse model than a preceding one. Theorem 2 showed conditions that prevent this from happening. We now show that if those conditions are not satisfied, it can indeed happen.

THEOREM 3: Given datum $x$ and target model $M \ni x$, such that the $x$ has minimal randomness deficiency in $M$ in the sense that $\delta(x|M) = \min_{M'}\{\delta(x|M') : M' \subseteq \{0,1\}^n, M' \ni x\}$. Assume that we have an MDL algorithm using an arbitrarily large but fixed constant $c$ as below, and that, on input datum $x$, in the limit finds $M$. Assume that in this MDL algorithm we change the currently best explanation $(p_1, M_1)$ to the explanation $(p_2, M_2)$ only if $|p_2| + \log|M_2|$ is much less than $|p_1| + \log|M_1|$, say $|p_2| + \log|M_2| \leq |p_1| + \log|M_1| - c\log n$. Under these conditions it is still possuble that a later model, say $M_2$, is much worse fitting than an earlier model, say $M_1$, in the sense that $\delta(x|M_2) \gg \delta(x|M_1)$. (Even though we find the best-fitting model in the limit).

*Proof:* We exhibit an MDL estimator that produces shorter and shorter two-part codes and yet periodically worse-fitting models. Fix datum $x^n = uvw$ with $u, v, w$ of equal length (say $n$ is a multiple of 3) with $K(x^n) = K(u) + K(v) + K(w) = 2n/3$, $K(u) = n/9$, $K(v) = 4n/9$, and $K(w) = n/9$. We ignore $O(\log n)$ additive terms in this proof.

Consider an MDL process that produces the sequence of models $M_0, M_1, \ldots, M_n$ with $M_i = \{x_1 \ldots x_i\}\{0,1\}^{n-i}$ $(0 \leq i \leq n)$. The process starts with candidate model $M_0$ and switches to candidate $M_i$, $i = n/3, 2n/3, n$, if that model gives a shorter MDL code than the previous candidate.

Now $K(M_i) = K(x^i)$ and $\log|M_i| = n - i$. However, our MDL estimator uses a compressor that does not grasp the hidden regularities involved, and codes the model for $M_i$ at $0.9i$ bits, that is, it compresses $x^i$ by 10%. Thus, the MDL code length $0.9i + \log|M_i| \leq 0.9i + n - i \leq n$ for every contemplated model $M_i$.

- In particular, the MDL code length of the initial candidate model $M_0$ is $n$. The randomness deficiency $\delta(x^n|M_0) = n - K(x^n|M_0) = n/3$. The last equality holds since clearly $K(x^n|M_0) = K(x^n|n)$.
- For the contemplated model $M_{n/3}$ we find: The MDL code length for model $M_{n/3}$ is $n - n/30$. The randomness deficiency $\delta(x|M_{n/3}) = 2n/3 - K(v|n) - K(w|n) = n/9$.
- For contemplated model $M_{2n/3}$ we find: The MDL code length is $n - 2n/30$. The randomness deficiency is $\delta(x|M_{2n/3}) = n/3 - K(w|n) = 2n/9$.

Thus, our MDL process initializes with candidate model $M_0$, then switches to candidate $M_{n/3}$ since this model decreases the MDL code length by $n/30$. Indeed, $M_{n/3}$ is a much better model than $M_0$, since it decreases the randomness deficiency by a whopping $2n/9$. Subsequently, however, the MDL process switches to candidate model

$M_{2n/3}$ since it decreases the MDL code length greatly again, by $n/30$. Oops, $M_{2n/3}$ is a much worse model than the previous candidate $M_{n/3}$, since it increases the randomness deficiency again by a whopping $n/9$. ■

REMARK 7: By Theorem 2 we know that if, in the process of MDL estimation by a sequence of decreasing MDL codes, a candidate model is represented by its shortest program, then the following candidate model which improves the MDL estimation is actually a model of at least as good fit as the preceding one. Thus, if in the example used in the proof above we encode the models at shortest code length, we obtain MDL code lengths $n$ for $M_0$, $K(u) + 2n/3 = 7n/9$ for $M_{n/3}$, and $K(u) + K(v) + n/3 = 8n/9$ for $M_{2n/3}$. Hence the MDL estimator using shortest model code length changes candidate model $M_0$ for $M_{n/3}$, improving the MDL code length by $n/9$ and the randomness deficiency by $2n/9$. However, and correctly, it does not change candidate model $M_{n/3}$ for $M_{2n/3}$, since that would increase the MDL code length by $n/9$. It so prevents, correctly, to increase the randomness deficiency by $n/9$. Thus, by the cited theorem, the oscillating randomness deficiency in the MDL estimation process in the proof above can only arise in cases where the consecutive candidate models are not coded at minimum cost while the corresponding two-part MDL code lengths are decreasing.

## VIII. MULTIPLE DATA SAMPLE

Assume that we want to infer a language, given a set of positive examples (elements of the language) $D$. For convenience we restrict the question to inferring a language $M \subseteq \{0,1\}^n$. We can view this as inferring the slice $L^n$ of the target language $L$ consisting of all words of length $n$ in the target language. Since $D$ consists of a subset of positive examples of $M$ we have $D \subseteq M$. To infer a language $M$ from a set of positive examples $D \subseteq M$ is, of course, a much more natural situation than to infer a language from a single example element $x \in M$ as in the previous example. Note that while the complexity $K(x)$ of a single example element $x$ in the language cannot exceed $n + O(\log n)$, while the complexity of the language itself can rise to $2^n + O(n)$, in the current setting $K(D)$ can rise to $2^n + O(n)$, just as $K(M)$ can. So contrary to the singleton datum case, in principle sets $M$ of every complexity can be inferred depending on the data $D$ at hand. An obvious example is $D = M$. Note that the cardinality of $D$ plays a role here, since the complexity $K(D|n) \leq \log \binom{2^n}{|D|} + O(\log |D|)$ with equality for certain $D$. A traditional and well-studied problem in this setting is to infer a grammar from language examples, which is the subject of the next section.

## IX. INFERRING A GRAMMAR (DFA) FROM POSITIVE EXAMPLES

The field of grammar induction studies a whole class of algorithms that aims at constructing a grammar by means of incremental compression of the data set represented as a digraph representation of a DFA accepting the data set. This digraph can be seen as a model for the data set. Every word in the data set is represented as a path in the digraph with the symbols either on the edges or on the nodes. The learning process takes the form of a guided incremental compression of the data set by means of merging or clustering of the nodes in the graph. None of these algorithms explicitly makes an estimate of the data-to-model code. Instead they use heuristics to guide the model reduction. After a certain time a proposal for a grammar can be constructed from the current state of the compressed graph. Examples of such algorithms are SP [21], [20], EMILE [17], ADIOS [19], and a number of DFA induction algorithms, specifically evidence driven state merging (EDSM), [18], [22]. Our results (above and below) do not imply that compression algorithms improving the MDL code of DFA's can never work on real life data sets. There is a lot of empirical evidence that there are situations in which they work. In those cases specific properties of a restricted class of languages or data sets must be involved. However, our results are quite general, and applicable to the common digraph simplification techniques used in grammar inference. Contrary to what is generally believed, this holds equally for algorithms that use just positive, just negative examples, or both.

DEFINITION 3: A DFA $A = (S, Q, q_0, t, F)$ where $S$ is a finite set of *input symbols*, $Q$ is a finite set of *states*, $t : Q \times S \to Q$ is the *transition function*, $q_0 \in Q$ is the *initial state*, and $F \subseteq Q$ is a set of *final states*.

The DFA $A$ is started in the initial state $q_0$. If it is in state $q \in Q$ and receives input symbol $s \in S$ it changes its state to $q' = t(q, s)$. If the machine after zero or more input symbols, say $s_1, \ldots, s_n$, is driven in a state $q \in F$ then it is said to *accept* the word $w = s_1 \ldots s_n$, otherwise it *rejects* the word $w$. The *language accepted* by $A$ is $L(A) = \{w : w \text{ is accepted by } A\}$. We denote $L^n(A) = L(A) \bigcap \{0,1\}^n$.

We can effectively enumerate the DFAs as $A_1, A_2, \ldots$ in lexicographic length-increasing order. This enumeration we call the *standard enumeration*.

The first thing we need to do is to show that all laws that hold for finite-set models also hold for DFA models, so all theorems, lemmas, and remarks above, both about the power and about the perils, apply. To do so, we show that for every data sample $D \subseteq \{0,1\}^n$ and a contemplated finite set model for it, there is an almost equivalent DFA.

LEMMA 3: For every $D \subseteq M \subseteq \{0,1\}^n$ there is a DFA $A$ with $L^n(A) = M$ such that $K(A) \leq K(M) + O(1)$ and $\delta(D \mid A, d, n) = \delta(D \mid M, d) + O(1)$.

*Proof:* Since $M$ is a finite set of strings, there is a DFA that accepts it, by elementary formal language theory. Define DFA $A$ such that $A$ is the first DFA in the standard enumeration for which $L^n(A) = M$—note that we can infer $n$ from $M$. Hence, $K(A) \leq K(M) + O(1)$ Trivially, $\log \binom{|M|}{d} = \log \binom{|L^n(A)|}{d}$ and $K(D \mid A, d, n) = K(D \mid M, d) + O(1)$, so that $\delta(D \mid A, d, n) = \delta(D \mid M, d) + O(1)$. ■

The converse of Lemma 3 is: for every data sample $D$ and a contemplated DFA model for it, there is a finite set model for $D$ that has no worse complexity, randomness deficiency, and worst-case data-to-model code for $D$, up to additive logarithmic precision.

LEMMA 4: Let $A$ be a DFA. For every $D \subseteq L^n(A) \subseteq \{0,1\}^n$, there is a finite set $M \supseteq D$ such that $\log \binom{|M|}{d} = \log \binom{|L^n(A)|}{d}$, $\delta(D \mid M, d) \leq \delta(D \mid A, d, n) + O(1)$ and $K(M) \leq K(A, n) + O(1)$.

*Proof:* Choose $M = L^n(A)$. Then, $\log \binom{|M|}{d} = \log \binom{|L^n(A)|}{d}$ and $K(M) \leq K(A, n) + O(1)$. Since also $K(D \mid A, d, n) \leq K(D \mid M, d) + O(1)$, we have $\delta(D \mid A, d, n) \geq \delta(D \mid M, d) + O(1)$. ■

## A. MDL Estimation

To analyse the MDL estimation for DFAs, given a data sample, we first fix details of the code. For the model code, the coding of the DFA, we encode as follows. Let $A = (Q, S, t, q_0, F)$ with $q = |Q|$, $s = |S|$. Then there $q$

possibilities for $F$, by renaming of the states we can always take care that $F \subseteq Q$ are the last $f$ states of $Q$. There are $q^{sq}$ different possibilities for $t$, and $q$ possibilities for $q_0$. Altogether, for every choice of $q, s$ there are $\leq q^{qs+2}$ distinct DFAs, some of which may accept the same languages.

**Small Model Cost but Difficult to Decode:** We can enumerate the DFAs by setting $i := 2, 3, \ldots$, for every $i$ considering $q + s = i$ for all partitions of $i$ in positive integer summands, and for every particular choice of $q, s$ considering every choice of final states, transition function, and initial state. This way we obtain a standard enumeration $A_1, A_2, \ldots$ of all DFAs, and, given the index $j$ of a DFA $A_j$ we can retrieve the particular DFA concerned, and for every $n$ we can find $L^n(A_j)$.

**Larger Model Cost but Easy to Decode:** We encode a DFA $A$ with $q$ states and $s$ symbols by

- The encoding of the number of symbols $s$ in self-delimiting format in $\lceil \log s \rceil + \lceil \log \log s \rceil$ bits;
- The encoding of the number of states $q$ in self-delimiting format in $\lceil \log q \rceil + \lceil \log \log q \rceil$ bits;
- The encoding of the set of final states $F$ by indicating that all states numbered $q - f, q - f + 1, q$ are final states, by just giving $q - f$ in $\lceil \log q \rceil$ bits;
- The encoding of the initial state $q_0$ by giving its index in the states $1, \ldots, q$, in $\lceil \log q \rceil$ bits; and
- The encoding of the transition function $t$ in lexicographical order of $Q \times S$ in $\lceil \log q \rceil$ bits per transition, which takes $qs\lceil \log q \rceil$ bits altogether.

Altogether, this encodes $A$ in a self-delimiting format in $(qs+3)\lceil \log q \rceil + 2\lceil \log \log q \rceil + \lceil \log s \rceil + 2\lceil \log \log s \rceil \approx (qs+4)\log q + 2\log s$ bits. Thus, we reckon the model cost of a $(q,s)$-DFA as $m(q,s) = (qs+4)\log q + 2\log s$ bits. This cost has the advantage that it is easy to decode and that $m(q,s)$ is an easy function of $q, s$. We will fix on this model cost.

**Data-to-model cost:** Given a DFA model $A$, the word length $n$, in $\log n + 2\log\log n$ bits which we simplify to $2\log n$ bits, and the size $d$ of the data sample $D \subseteq \{0,1\}^n$, we can describe $D$ by its index $j$ in the set of $d$ choices out of $l = L^n(A)$ items, that is, up to rounding upwards, $\log \binom{l}{d}$ bits. For $0 < d \leq l/2$ this can be estimated by $lH(d/l) - \log l/2 + O(1) \leq \log \binom{l}{d} \leq lH(d/l)$ where $H(p) = p \log 1/p + (1-p)\log 1/(1-p)$ $(0 < p < 1)$ is Shannon's entropy function. For $d = 1$ or $d = l$ we set the data-to-model cost to $1 + 2\log n$, for $1 < d \leq l/2$ we set it to $2\log n + lH(d/l)$ (ignoring the possible savings of $\log l/2$ term), and for $l/2 < d < l$ we set it to the cost of $l - d$. This reasoning brings us to the following MDL cost of a data sample $D$ for DFA model $A$:

DEFINITION 4: The *MDL code length* of a data sample $D$ of $d$ strings of length $n$, for a DFA model $A$ such that $D \subseteq L^n(A)$, denoting $l = |L^n(A)|$, is given by

$$MDL(D, A) = (qs+4)\log q + 2\log s + 2\log n + lH(d/l).$$

## B. Randomness Deficiency Estimation

Given data sample $D$ and DFA $A$ with $D \subseteq L^n(A) \subseteq \{0,1\}^n$, we can estimate the randomness deficiency. Again, use $l = L^n(A)$ and $d = |D|$. By (3), the randomness deficiency is

$$\delta(D \mid A, d, n) = \log \binom{l}{d} - K(D \mid A, d, n).$$

Then, substituting the estimate for $\log \binom{l}{d}$ from the previous section, up to logarithmic additive terms,

$$\delta(D \mid A, d, n) = lH(d/l) - K(D \mid A, d, n).$$

Thus, by finding a computable upper bound for $K(D \mid A, d, n)$, we can obtain a computable lower bound on the randomness deficiency $\delta(D \mid A, d, n)$ that expresses the fittness of DFA model $A$ with respect to data sample $D$.

## *C. Less MDL Code Length Doesn't Mean Better Model*

The task of finding the smallest DFA consistent with a set of positive examples is trivial. This is the universal DFA. Yet the universal DFA will in most cases have a poor generalization error and randomness deficiency. We show that the randomness deficiency behaves independently of the MDL code: the randomness deficiency can either grow or shrink with a reduction of the length of the MDL code.

We show this by example. Let the set $D$ be a sample set consisting of 50% of all binary strings of length $n$ with an even number of 1's. Note, that the number of strings with an even number of 1's equals the number of strings with an odd number of ones, so $d = |D| = 2^n/4$. Initialize with a DFA $A$ such that $L^n(A) = D$. We can obtain $D$ directly from $A, n$, so we have $K(D \mid A, n) = O(1)$, and since $d = l$ we have $\log \binom{l}{d} = 0$, so that altogether $\delta(D \mid A, d, n) = -O(1)$, while $MDL(D, A) = (qs+4)\log q + 2\log s + 2\log n + O(1) = (2q+4)\log q + 2\log n + O(1)$, since $s = 2$. Without loss of generality we can assume that the MDL algorithm involved works by splitting or merging nodes of the digraphs of the produced sequence of candidate DFA's. But the argument works for every MDL algorithm, whatever technique it uses.

*Initialize:* Assume that we start our MDL estimation with the trivial DFA $A_0$ that literally encodes all $d$ elements of $D$ as a binary directed tree with $q$ nodes. Then, $2^n/2 - 1 \leq q \leq 2^{n+1} - 1$, which yields

$$MDL(D, A_0) \geq 2^n n/2$$
$$\delta(D \mid A_0, d, n) \approx 0,$$

the latter equation since $d = l$, so $\log \binom{l}{d} = 0$, and $K(D \mid A_0, d, n) = O(1)$. Since the randomness deficiency $\delta(D \mid A_0, d, n) \approx 0$, we have that $A_0$ is a best fitting model for $D$. Indeed, it represents all conceivable properties of $D$ since it literally models $D$. However, $A_0$ doesn't achieve the optimal MDL code.

*Better MDL estimation:* In a later MDL estimation we improve the MDL code by inferring the parity DFA $A_1$ with two states ($q = 2$) that checks the parity of 1's in a sequence. Then,

$$\begin{aligned} MDL(D, A_1) &\leq 8 + 2\log n + \log \binom{2^n/2}{2^n/4} \approx 2^{n-1} - n/4 \\ \delta(D \mid A_1, d, n) &= \log \binom{2^n/2}{2^n/4} - K(D \mid A_1, d, n) \\ &\approx 2^{n-1} - n/4 - K(D \mid A_1, d, n) \end{aligned}$$

We now consider two different instantiations of $D$, denoted as $D_0$ and $D_1$. The first one is regular data, and the second one is random data.

**Case 1, regular data:** Suppose $D = D_0$ consisting of the lexicographical first 50% of all $n$-bit strings with an even number of occurrences of 1's. Then $K(D_0 \mid A_1, d, n) = O(1)$ and

$$\delta(D_0 \mid A_1, d, n) = 2^{n-1} - O(n).$$

In this case, even though DFA $A_1$ has a much better MDL code than DFA $A_0$, it has nonetheless a much worse fit since its randomness deficiency is far greater.

**Case 2, random data:** Suppose, $D = D_1$ where $D_1$ is a random subset consisting of 50% of the $n$-bit strings with even number of occurrences of 1's. Then, $K(D_1 \mid A_1, d, n) = \log \binom{2^n/2}{2^n/4} + O(1) \approx 2^{n-1} - n/4$, and

$$\delta(D_1 \mid A_1, d, n) \approx 0.$$

In this case, DFA $A_1$ has a much better MDL code than DFA $A_0$, and it has equally good fit since both randomness deficiencies are about 0.

REMARK 8: We conclude that improved MDL estimation of DFA's for multiple data samples doesn't necessarily result in better models, but can do so nonetheless.

REMARK 9 (SHORTEST MODEL COST): By Theorem 2 we know that if, in the process of MDL estimation by a sequence of decreasing MDL codes, a candidate DFA is represented by its shortest program, then the following candidate DFA which improves the MDL estimation is actually a model of at least as good fit as the preceding one. Let us look at an Example: Suppose we start with DFA $A_2$ that accepts all strings in $\{0,1\}^*$. In this case we have $q = 1$ and

$$MDL(D_0, A_2) = \log \binom{2^n}{2^n/4} + O(1)$$
$$\delta(D_0 \mid A_2, d, n) = \log \binom{2^n}{2^n/4} - O(1).$$

Here $\log \binom{2^n}{2^n/4} = 2^n H(\frac{1}{4}) - O(n) \approx 3 \cdot 2^{n-2} - O(n)$. ($H(\frac{1}{4}) \approx \frac{2}{3}$) Suppose the subsequent candidate DFA is the parity machine $A_1$. Then,

$$MDL(D_0, A_1) = \log \binom{2^n/2}{2^n/4} + O(1)$$
$$\delta(D_0 \mid A_1, d, n) \approx \log \binom{2^n/2}{2^n/4} - O(1),$$

since $K(D_0 \mid A_1, d, n) = O(1)$. Since $\log \binom{2^n/2}{2^n/4} = 2^{n-1} - O(n)$, we have $MDL(D_0, A_2) \approx \frac{2}{3} MDL(D_0, A_2)$, and $\delta(D_0 \mid A_2, d, n) \approx \frac{2}{3}\delta(D_0 \mid A_1, d, n)$. So the improved MDL cost is accompanied by improved fitness by decreasing randomness deficiency. This indeed is forced by Theorem 2, since both DFA $A_1$ and DFA $A_2$ have $K(A_1), K(A_2) = O(1)$. That is, the DFA's are represented and costed according to their shortest programs (a forteriori of length $O(1)$) and therefore improved MDL estimation increases the fitness of the successive DFA models significantly.

## Appendix

### A. Appendix: Preliminaries

*1) Self-delimiting Code:* A binary string $y$ is a *proper prefix* of a binary string $x$ if we can write $x = yz$ for $z \neq \epsilon$. A set $\{x, y, \ldots\} \subseteq \{0,1\}^*$ is *prefix-free* if for any pair of distinct elements in the set neither is a proper prefix of the other. A prefix-free set is also called a *prefix code* and its elements are called *code words*. An example of a prefix code, that is useful later, encodes the source word $x = x_1 x_2 \ldots x_n$ by the code word

$$\overline{x} = 1^n 0x.$$

This prefix-free code is called *self-delimiting*, because there is fixed computer program associated with this code that can determine where the code word $\bar{x}$ ends by reading it from left to right without backing up. This way a composite code message can be parsed in its constituent code words in one pass, by the computer program. (This desirable property holds for every prefix-free encoding of a finite set of source words, but not for every prefix-free encoding of an infinite set of source words. For a single finite computer program to be able to parse a code message the encoding needs to have a certain uniformity property like the $\overline{x}$ code.) Since we use the natural numbers and the binary strings interchangeably, $|\bar{x}|$ where $x$ is ostensibly an integer, means the length in bits of the self-delimiting code of the binary string with index $x$. On the other hand, $\overline{|x|}$ where $x$ is ostensibly a binary string, means the self-delimiting code of the binary string with index the length $|x|$ of $x$. Using this code we define the standard self-delimiting code for $x$ to be $x' = \overline{|x|}x$. It is easy to check that $|\overline{x}| = 2n + 1$ and $|x'| = n + 2\log n + 1$. Let $\langle \cdot \rangle$ denote a standard invertible effective one-one encoding from $\mathcal{N} \times \mathcal{N}$ to a subset of $\mathcal{N}$. For example, we can set $\langle x, y \rangle = x'y$ or $\langle x, y \rangle = \bar{x}y$. We can iterate this process to define $\langle x, \langle y, z \rangle \rangle$, and so on.

*2) Kolmogorov Complexity:* For precise definitions, notation, and results see the text [15]. Informally, the Kolmogorov complexity, or algorithmic entropy, $K(x)$ of a string $x$ is the length (number of bits) of a shortest binary program (string) to compute $x$ on a fixed reference universal computer (such as a particular universal Turing machine). Intuitively, $K(x)$ represents the minimal amount of information required to generate $x$ by any effective process. The conditional Kolmogorov complexity $K(x|y)$ of $x$ relative to $y$ is defined similarly as the length of a shortest program to compute $x$, if $y$ is furnished as an auxiliary input to the computation. For technical reasons we use a variant of complexity, so-called prefix complexity, which is associated with Turing machines for which the set of programs resulting in a halting computation is prefix free. We realize prefix complexity by considering a special type of Turing machine with a one-way input tape, a separate work tape, and a one-way output tape. Such Turing machines are called *prefix* Turing machines. If a machine $T$ halts with output $x$ after having scanned all of $p$ on the input tape, but not further, then $T(p) = x$ and we call $p$ a *program* for $T$. It is easy to see that $\{p : T(p) = x, x \in \{0,1\}^*\}$ is a *prefix code*. Let $T_1, T_2, \ldots$ be a standard enumeration of all prefix Turing machines with a binary input tape, for example the lexicographical length-increasing ordered syntactic prefix Turing machine descriptions, [15], and let $\phi_1, \phi_2, \ldots$ be the enumeration of corresponding functions that are computed by the respective Turing machines ($T_i$ computes $\phi_i$). These functions are the *partial recursive* functions or *computable* functions (of effectively prefix-free encoded arguments). The Kolmogorov complexity of $x$ is the length of the shortest binary program from which $x$ is computed. For the development of the theory we actually require the Turing machines to use *auxiliary* (also called *conditional*) information, by equipping the machine with a special read-only auxiliary tape containing this information at the outset.

One of the main achievements of the theory of computation is that the enumeration $T_1, T_2, \ldots$ contains a machine, say $U = T_u$, that is computationally universal in that it can simulate the computation of every machine in the enumeration when provided with its index: $U(\langle y, \bar{i}p \rangle) = T_i(\langle y, p \rangle)$ for all $i, p, y$. We fix one such machine and designate it as the *reference universal prefix Turing machine*.

DEFINITION 5: Using this universal machine we define the *Kolmogorov complexity*

$$K(x \mid y) = \min_q \{|q| : U(\langle y, q \rangle) = x\}, \tag{9}$$

the *conditional version* of the prefix Kolmogorov complexity of $x$ given $y$ (as auxiliary information). The unconditional version is set to $K(x) = K(x \mid \epsilon)$.

A prominent property of the prefix-freeness of $K(x)$ is that we can interpret $2^{-K(x)}$ as a probability distribution since $K(x)$ is the length of a shortest prefix-free program for $x$. By the fundamental Kraft's inequality, see for example [2], [15], we know that if $l_1, l_2, \ldots$ are the code-word lengths of a prefix code, then $\sum_x 2^{-l_x} \leq 1$. Hence,

$$\sum_x 2^{-K(x)} \leq 1. \tag{10}$$

This leads to the notion of universal distribution—a rigorous form of Occam's razor—which implicitly plays an important part in the present exposition. The functions $K(\cdot)$ and $K(\cdot \mid \cdot)$, though defined in terms of a particular machine model, are machine-independent up to an additive constant and acquire an asymptotically universal and absolute character through Church's thesis, from the ability of universal machines to simulate one another and execute any effective process. The Kolmogorov complexity of an individual object was introduced by Kolmogorov [7] as an absolute and objective quantification of the amount of information in it. The information theory of Shannon [9], on the other hand, deals with *average* information *to communicate* objects produced by a *random source*. Since the former theory is much more precise, it is surprising that analogs of theorems in information theory hold for Kolmogorov complexity, be it in somewhat weaker form. An example is the remarkable *symmetry of information* property used later. Let $x^*$ denote the shortest prefix-free program $x^*$ for a finite string $x$, or, if there are more than one of these, then $x^*$ is the first one halting in a fixed standard enumeration of all halting programs. Then, by definition, $K(x) = |x^*|$. Denote $K(x, y) = K(\langle x, y \rangle)$. Then,

$$\begin{aligned} K(x, y) &= K(x) + K(y \mid x^*) + O(1) \qquad (11) \\ &= K(y) + K(x \mid y^*) + O(1). \end{aligned}$$

REMARK 10: The information contained in $x^*$ in the conditional above is the same as the information in the pair $(x, K(x))$, up to an additive constant, since there are recursive functions $f$ and $g$ such that for all $x$ we have $f(x^*) = (x, K(x))$ and $g(x, K(x)) = x^*$. On input $x^*$, the function $f$ computes $x = U(x^*)$ and $K(x) = |x^*|$; and on input $x, K(x)$ the function $g$ runs all programs of length $K(x)$ simultaneously, round-robin fashion, until the first program computing $x$ halts—this is by definition $x^*$.

*3) Precision:* It is customary in this area to use "additive constant $c$" or equivalently "additive $O(1)$ term" to mean a constant, accounting for the length of a fixed binary program, independent from every variable or parameter in the expression in which it occurs. In this paper we use the prefix complexity variant of Kolmogorov complexity for convenience.

## *B. Appendix: Structure Functions and Model Selection*

We summarize a selection of the results in [16]. There, the data sample $D$ is a singleton set $\{x\}$. The results extend to the multiple data sample case in the straightforward way.

THEOREM 4: Let $n$ be a positive integer, data sample $x \subseteq \{0,1\}^n$ and complexity $k = K(x)$, and $K(x \mid n) \leq n + O(1)$. Then:

(i) There is an integer valued non-increasing function $\lambda$ defined on $[0,k]$ such that $\lambda(0) \leq n$, $\lambda(k) = k$ and $\lambda_x$ is close to $\lambda$ in the sense that the graph of $\lambda_x$ is within a strip of width $\log n + O(1)$.

(ii) Conversely, for every $n$ and non-increasing integer valued function $\lambda$ whose domain includes $[0,k]$ and such that $\lambda(0) \leq n$ and $\lambda(k) = k$, there is $x$ of length $n$ and complexity $k \pm (K(k,n,\lambda) + O(1))$, such that $\lambda_x$ is close to $\lambda$ in the sense that the graph of the former is situated within a strip of width $2K(i,n,\lambda) + O(1)$ centered on the graph of the latter.

*Intuition: The MDL code length $\lambda_x$ can assume essentially every possible relevant shape as a function of the contemplated maximal model complexity.*

As a consequence, so-called "non-stochastic" data $x$ for which $\lambda_x(\alpha)$ stabilize on $K(x)$ for large $\alpha$ are common.

THEOREM 5: Let $n$ be a positive integer, data $x \in \{0,1\}^n$ of complexity $k = K(x)$. Therefore $K(x \mid n) \leq n + O(1)$. Then: $\beta_x(\alpha) + k$ is close to $\lambda_x(\alpha)$ in the sense that the graph of the former is situated within a strip of width $O(\log n)$ centered on the graph of the latter.

*Intuition: A model achieving the MDL code length $\lambda_x(\alpha)$, essentially achieves the best possible fit $\beta_x(\alpha)$.*

COROLLARY 2: For every $x \in \{0,1\}^n$ and complexity $k = K(x)$ there is a non-increasing function $\beta$ such that $\beta(0) \leq n - k$, $\beta(k) = 0$ and the graph of $\beta_x$ is situated within a strip of $O(\log n)$ of the graph of $\beta$. Conversely, for every non-increasing function $\beta$ such that $\beta(0) \leq n - k$, $\beta(k) = 0$ there is $x \in \{0,1\}^n$ and complexity $k \pm O(\log n) + 2K(\beta)$ such that $\beta_x$ is situated within a strip of $O(\log n) + 2K(\beta)$ of the graph of $\beta$.

*Intuition: The best-fit function $\beta_x$ can assume essentially every possible relevant shape as a function of the contemplated maximal model complexity.*

In [16] Theorem 5 is proven for singleton data samples. The generalization to multi-element data samples $D$ is straightforward. From the proof we see that, given the data sample $D$, for every finite set $M \supseteq D$, of complexity at most $\alpha + O(\log n)$ and minimizing $\Lambda(M)$, we have $\delta(D \mid M, d) \leq \beta_D(\alpha) + O(\log m)$. Ignoring $O(\log m)$ terms, at every complexity level, every best hypothesis at this level with respect to $\Lambda(M)$ is also a best one with respect to typicality. This explains why it is worthwhile to find shortest two-part descriptions for given data sample $D$: this is the single known way to find an $M \supseteq D$ with respect to which $D$ is as typical as possible at that complexity level. Note that the set $\{\langle D, M, \beta \rangle \mid D \subseteq M,\ \delta(D \mid M, d) < \beta\}$ is not enumerable so we are not able to generate such $M$'s directly, [16].

The converse is not true: not every hypothesis, consisting of a finite set, witnessing $\beta_D(\alpha)$ also witnesses $\lambda_D(\alpha)$. For example, let $D = \{x\}$ with $x$ a string of length $n$ with $K(x) \geq n$. Let $M_1 = \{0,1\}^n \cup \{y0 \ldots 0\}$ where $y$ is a string of length $\frac{n}{2}$ such that $K(x, y) \geq \frac{3n}{2}$ and let $M_2 = \{0,1\}^n$. Then both $M_1, M_2$ witness $\beta_D(\frac{n}{2} + O(\log n)) = O(1)$ but $\Lambda(M_1) = \frac{3n}{2} + O(\log n) \gg \lambda_D(\frac{n}{2} + O(\log n)) = n + O(\log n)$ while $\lambda(M_2) = n + O(\log n)$.

However, for every $\alpha$ such that $\lambda_D(i)$ decreases when $i \to \alpha$ with $i \leq \alpha$, a witness set for $\beta_D(\alpha)$ is also a witness set for $\lambda_D(\alpha)$. We will call such $\alpha$ *critical* (with respect to $D$): these are the model complexities at which the two-part MDL code-length decreases, while it is stable in between such critical points. In [16] it is shown, for singleton sample sets $D$, that for critical $\alpha$, for every $A \supseteq D$ with $K(A) \approx \alpha$ and $\delta(D \mid A, d) \approx \beta_D(\alpha)$, we have $\Lambda(A) \approx \lambda_D(\alpha)$. More specifically, if $K(A) \approx \alpha$ and $\delta(D \mid A, d) \approx \beta_D(\alpha)$ but $\Lambda(A) \gg \lambda_D(\alpha)$ then there is $M \supseteq D$ with $K(M) \ll \alpha$ and $\Lambda(M) \approx \lambda_D(\alpha)$.

## REFERENCES

[1] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[2] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.

[3] D. Donoho, The Kolmogorov sampler, *Annals of Statistics*, submitted.

[4] P. Elias, List decoding for noisy channels. *Wescon Convention Record*, Part 2, Institute for Radio Engineers (now IEEE), 1957, 94–104.

[5] P. Elias, Error-correcting codes for List decoding, *IEEE Trans. Inform. Th.*, 37:1(1991), 5–12.

[6] P.D. Grünwald and P.M.B. Vitányi, Shannon information and Kolmogorov complexity, Manuscript, CWI, December 2003.

[7] A.N. Kolmogorov, Three approaches to the quantitative definition of information, *Problems Inform. Transmission* 1:1 (1965) 1–7.

[8] A.N. Kolmogorov. Complexity of Algorithms and Objective Definition of Randomness. A talk at Moscow Math. Soc. meeting 4/16/1974. An abstract available in *Uspekhi Mat. Nauk* 29:4(1974),155; English translation in [16].

[9] C.E. Shannon. The mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948.

[10] C.E. Shannon. Coding theorems for a discrete source with a fidelity criterion. In *IRE National Convention Record, Part 4*, pages 142–163, 1959.

[11] A.Kh. Shen, The concept of $(\alpha, \beta)$-stochasticity in the Kolmogorov sense, and its properties, *Soviet Math. Dokl.*, 28:1(1983), 295–299.

[12] N.K. Vereshchagin and P.M.B. Vitányi, Algorithmic rate-distortion theory, *IEEE Trans. Inform. Th.*, Submitted. http://arxiv.org/abs/cs/0411014

[13] J.M. Wozencraft, List decoding. *Quarterly Progress Report*, Research Laboratory for Electronics, MIT, Vol. 58(1958), 90–95.

[14] Mitchell T. M., , Machine Learning, McGraw-Hill, New York, (1997)

[15] Li M., Vitányi P.M.B. An Introduction to Kolmogorov Complexity and Its Applications, 2nd ed., Springer-Verlag, New York, (1997)

[16] Vereshchagin N.K., Vitányi P.M.B., Kolmogorov's structure functions and model selection, IEEE Trans. Information Theory, vol. 50, nr. 12, 3265–3290, (2004)

[17] Adriaans P., Vervoort M., The EMILE 4.1 grammar induction toolbox, In: *Grammatical Inference: Algorithms and Applications; 6th International Colloquium, ICGI 2002*, P. Adriaans and H. Fernau and M. van Zaanen eds., LNCS/LNAI 2484, 293–295, 2002.

[18] Lang K. J., Pearlmutter B. A., Price R. A. , Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In: *Grammatical Inference: Algorithms and Applications; 6th International Colloquium, ICGI 2002*, P. Adriaans and H. Fernau and M. van Zaanen eds., LNCS/LNAI 2484, 1–12, 2002.

[19] Solan Z., Horn D., Ruppin E., Edelman S., Unsupervised learning of natural languages, *Proc. Natn'l Academy Sci.*, 102: 33(2005), 11629-11634.

[20] Wolff J.G., Computing As Compression: An Overview of the SP Theory and System, *New Generation Comput.*, 13:2(1995), 187–214.

[21] Wolff, J. G., Information Compression by Multiple Alignment, Unification and Search as a Unifying Principle in Computing and Cognition, *J. Artificial Intelligence Research*, 19:3(2003), 193–230.

[22] *Proc. Workshop and Tutorial on Learning Context-Free Grammars* at the *14th Eur. Conf. Mach. Learn. (ECML)* and the *7th Eur. Conf. Principl. and Pract. Knowl. Disc. Databases (PKDD)*; Dubrovnik, Croatia, de la Higuera, Adriaans, van Zaanen, and Oncina (eds.), 2003.