On Hierarchical Compression and Power Laws in Nature

Arthur Franz(✉)

Independent Researcher, Odessa, Ukraine
franz@fias.uni-frankfurt.de

Abstract. Since compressing data incrementally by a non-branching hierarchy has resulted in substantial efficiency gains for performing induction in previous work, we now explore branching hierarchical compression as a means for solving induction problems for generally intelligent systems. Even though assuming the compositionality of data generation and the locality of information may result in a loss of the universality of induction, it has still the potential to be general in the sense of reflecting the inherent structure of real world data imposed by the laws of physics. We derive a proof that branching compression hierarchies (BCHs) create power law functions of mutual algorithmic information between two strings as a function of their distance – a ubiquitous characteristic of natural data, which opens the possibility of efficient natural data compression by BCHs. Further, we show that such hierarchies guarantee the existence of short features in the data which in turn increases the efficiency of induction even more.

Keywords: Hierarchical compression · Incremental compression · Algorithmic complexity · Universal induction · Power laws · Scale free structure

1 Introduction

The question how humans succeed in deriving theories and explanations from sensory data – the problem of induction – has long remained a mystery of human cognition and philosophy of science. Because it is so central to human thinking, it is essential to solve this problem for any attempt to build a generally intelligent system. Fortunately, Solomonoff's theory of universal induction [1,2] presents a formidable mathematical solution to this thorny problem. However, it is incomputable and tractable approximations have remained elusive.

Nevertheless, for practical purposes, it seems sufficiently satisfactory if we solve the problem of induction "merely" for data presented to us by the actual physical world that we inhabit. For this purpose it is instructive to ask, why for example are deep learning classifiers so successful although it can be shown [3] that classifying arbitrary binary images with n pixels requires at least 2^n parameters in the neural network? And why is it hard for human subjects to

T. Everitt et al. (Eds.): AGI 2017, LNAI 10414, pp. 77–86, 2017.
DOI: 10.1007/978-3-319-63703-7_8

find an algorithm that prints the digits of π given a sequence of its digits even though the algorithm is fairly short compared to other gigabyte heavy software programs written by humans? The world seems to present us with a small subset of all possible data – a circumstance that can be exploited in order to increase the efficiency of induction algorithms.

Lin and Tegmark [3] argue that properties like symmetry, locality and compositionality of real world data are key restrictions for that purpose and show by "no-flattening theorems" that deep networks achieve their efficiency by exploiting the compositionality of data. Further, as they argue, the polynomial structure of the Hamiltonians in the fundamental laws of physics and the compositional way that those laws are expressed when they generate real world data seems to support the generality of this observation.

Indeed, as proven in our previous work [4], exploiting the compositionality of data leads to an efficient incremental way of performing induction. In short[1], if a bit string of data x is representable by a composition of computable functions (Turing machines), $x = f_1 \circ \cdots \circ f_m(\epsilon)$, then these so-called features f_i can be found in a greedy fashion (without backtracking), if we always look for the shortest ones while compressing at least a little (which excludes identity functions). The algorithmic entropy $K(x)$ can then be obtained by

$$K(x) = \sum_{i=1}^{m} l(f_i) + O(1) \tag{1.1}$$

where all features are pairwise algorithmically orthogonal: $I(f_i : f_j) = 0$ for all $i \neq j$. The features can be found by searching through pairs of programs (f, f') such that $f(f'(x)) = x$ and $l(f) + l(f'(x)) < l(x)$ where the length of the shortest descriptive map f' is found to obey a fairly low bound $l(f') \leq \log K(x) + 2 \log\log K(x) + O(1)$ as will be shown in Sect. 5. In spite of this success, the length of the shortest features is not bounded in any way, leaving us with limited theoretical guarantees for the bound on the time complexity of search. Further assumptions about real world data seem necessary in order to obtain such a bound, which would be very helpful in order to boost the efficiency of induction.

A remarkable property of our world seems to be that it has structure on all scales. No matter how much we zoom in toward the microscopic world or zoom out to the macroscopic world, we never seem to arrive at emptiness or a structureless distribution of matter. Typically, this scale invariance can be expressed by power law correlation functions which are found to be ubiquitous in nature. From avalanche distributions, noise spectra, letter sequences in natural language, earthquake and solar flare frequency distributions, species extinction rates, traffic jams, natural images and many more, power law correlation functions are found virtually everywhere in natural data [5–7]. Further, we seem to possess a theoretical justification of the multitude of power laws through the process of self-organized criticality [5].

[1] For notation and definitions please consult the Preliminaries section below.

In this paper, we show that by extending the present theory of non-branching incremental compression to a branching compression hierarchy (BCH), the mutual algorithmic information between two substrings decays like a power law function of the distance between the substrings. We proceed to show that this circumstance leads to a bound on the feature lengths, which increases the efficiency of incremental compression.

2 Preliminaries

Consider a universal prefix Turing machine U. Strings are defined on a finite alphabet $\mathcal{A} = \{0, 1\}$ with ϵ denoting the empty string. Logarithms are taken on the basis 2. $\mathcal{A}^*$ denotes the set of finite strings made up of the elements of $\mathcal{A}$. Since there is a one-to-one map $\mathcal{A}^* \leftrightarrow N$ of finite strings on natural numbers, strings and natural numbers are used interchangeably. For example, the length $l(n)$ of an integer n denotes the number of symbols of the string that it corresponds to. The map $\langle \cdot, \cdot \rangle$ denotes a one-to-one map of two strings on natural numbers: $\mathcal{A}^* \times \mathcal{A}^* \leftrightarrow N$. The corresponding map for more than two variables is defined recursively: $\langle x, y, z \rangle \equiv \langle \langle x, y \rangle, z \rangle$. In particular, $\langle z, \epsilon \rangle = z$. Since all Turing machines can be enumerated, the universal machine U operates on a number/string $\langle n, p \rangle$ by executing p on the Turing machine T_n: $U(\langle n, p \rangle) = T_n(p)$. Similarly, a string y is applied to another string x by applying the yth Turing machine: $y(x) \equiv T_y(x) = U(\langle y, x \rangle)$. When we speak about the length of a function/feature f, we mean the length of the binary representation of the index y of the respective Turing machine $T_y = f$ in the enumeration. The prefix complexity $K(x|y)$ of x given y is defined by $K(x|y) \equiv \min\{l(z) : U(\langle z, y \rangle) = x\}$ and $K(x) \equiv K(x|\epsilon)$. The complexity of several variables is defined as $K(x, y) \equiv K(\langle x, y \rangle)$. The "+" sign above equality or inequality signs denotes that the relation is valid up to a constant that is independent of the involved variables. The information contained in x about y is defined as $I(x : y) \equiv K(y) - K(y|x)$. However, sometimes we will refer to it as mutual information for the sake of brevity although it is not symmetric. When a string is a concatenation of substrings, $x = x_1 x_2 \cdots x_n$, by distance between substring x_i and x_j we mean the index distance $d_{ij} \equiv |i - j|$.

3 Branching Compression Hierarchies Create Power Laws

Consider a binary string x that can be computed by a hierarchy of functions, Fig. 1. In this section we show that if the functions are the shortest features of their respective substrings, then the information of a substring x_i about another substring x_j of x will be bounded by power law function of their distance:

$$I(x_i : x_j) \lesssim d(x_i, x_j)^{-const}$$

The main idea is the following. Assume that a fraction of the information in a string is lost as we go along an edge in a graph (information dissipation),

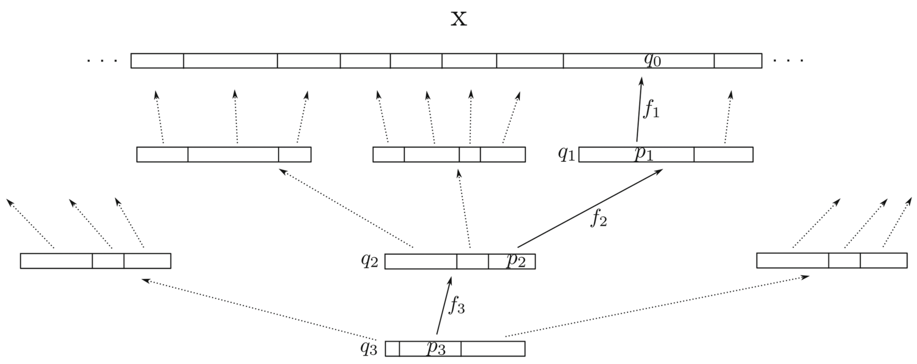

Fig. 1. A branching compression hierarchy (BCH). The path from the root to some leaf is displayed by solid arrows. A bit string x (whole upper chain, only the part computed from q_2 is shown) is computed by concatenating substrings, one of which is shown as q_0. Those substrings are computed by their shortest features f_l and respective parameters, $q_{l-1} = f_l(p_l)$, that are found along the path. Since only a fraction $\alpha_l = K(p_l)/K(q_l)$ of the information in q_l remains at each level l, the information contained in string q_l about q_0 decays exponentially with its height l. Note that each arrow (both dashed and solid) corresponds to a different feature, each q_1 in the first level computes different substrings of x.

since only a part of the string serves as input to the function at the edge. Then the information of a string about another will decrease exponentially with the length of the path between the strings. If the mutual information of two leaf strings is mediated only via the earliest common ancestor in a tree, then it will drop exponentially with the height of that ancestor. Further, the distance between two leaves increases exponentially with the height of their common ancestor. Inserting two exponential functions into each other leads to the power law. The idea that branching hierarchies create power laws is not new [7], but this is to the best of our knowledge the first time that it is shown in all generality for algorithmic information and arbitrary data.

Definition 1 (Branching compression hierarchy). *Let T be a perfect tree with a varying branching factor, $F = \{f_i\}$ a set of computable functions for each edge in the tree and q_h a binary string at the root node. For any path from the root to a leaf, index the functions as $f_h, \dots, f_1$ and compute $f_l(p_l) = q_{l-1}$ for each $l = h, \dots, 1$, where p_l is some substring of q_l. Let further f_l be the shortest feature and p_l its respective parameters of q_{l-1} (see [4] for definitions). Then, the triple $H = \{T, F, q_h\}$ is called a* ***branching compression hierarchy****. The fraction*

$$\alpha_l \equiv \frac{K(p_l)}{K(q_l)} \tag{3.1}$$

shall be called ***information dissipation rate****.*

An Example. Consider a binary image of a triangle. In the first compression level the features could encode lines while the parameters encode the coordinates of the line ends. Then $f_1 = \mathit{line}$ and $p_1 = x_1 y_1 x_2 y_2$ encode one of the sides, other features $f_1^{(2)}$ and $f_1^{(3)}$ at the same level encode the other sides with their respective parameters $p_1^{(2)} = x_2 y_2 x_3 y_3$ and $p_1^{(3)} = x_3 y_3 x_1 y_1$. All those parameters are concatenated to $q_1 = p_1 p_1^{(2)} p_1^{(3)} = x_1 y_1 x_2 y_2 x_2 y_2 x_3 y_3 x_3 y_3 x_1 y_1$. This string is further compressed by $f_2 = \mathit{copy}2$ which could be a function that copies consecutive entries, such that $q_1 = f_2(p_2)$, where $p_2 = x_1 y_1 x_2 y_2 x_3 y_3$. This concludes a two-level branching hierarchical compression of a triangle leaving us with the coordinates of the corners as a concise representation of a triangle.

Lemma 1 (Exponential information decay). *Let H be a BCH according to the above definition. We assume:*

1. *The information content in the root node q_h is uniformly distributed through q_l for each l:*
$$\frac{K(p_l|q_h)}{K(q_l|q_h)} = \frac{K(p_l)}{K(q_l)} = \alpha_l \tag{3.2}$$
2. *The root q_h does not contain any information about any feature below in the path $I(q_h : f_l) = 0$ for all $l \le h$.*

Then the information content in q_h about a leaf q_0 is given by

$$I(q_h : q_0) \overset{+}{=} K(q_h) \cdot \prod_{l=1}^{h} \alpha_l \tag{3.3}$$

Of course, there is no guarantee that assumption (1) holds, but it can be expected to hold on average, since after all, the information in q_h has to go somewhere, since x is ultimately computed from it. If it doesn't go into p_l then into some p_l on another path to x. Assumption (2) is discussed below.

Since $0 < \alpha_l \le 1$, Eq. (3.3) constitutes an exponential decay of information in a string q_h about a leaf q_0 as its height h increases. In the special case of $\alpha \equiv \alpha_l = \text{const}$ the decay $I(q_h : q_0) \overset{+}{=} K(q_h) \cdot \alpha^h$ becomes apparent. What can we say about the information two arbitrary substrings y and z about each other?

Lemma 2 (Information of strings about each other). *Let two strings y and z be conditionally independent given string a:*

$$K(y, z|a) = K(y|a) + K(z|a) \tag{3.4}$$

Then the information of y about z is bounded by the information of a about z:

$$I(y : z) \overset{+}{\le} I(a : z) \tag{3.5}$$

It should be noted that the assumptions of conditional independence in Lemma 2 and assumption (2) in Lemma 1 merely reflect that the only link between the substrings is the common ancestor. Otherwise, two arbitrary substrings of an arbitrary string certainly can carry more information about each other.

We can now state our main theorem.

Theorem 1 (Power law information decay in BCHs). *Let H be a BCH with the assumptions of Lemma 1 and each pair of leaves conditionally independent given their common ancestor q_h at height h. Let further $d_{ij} \equiv |i-j|$ be the index distance between to substrings x_i and x_j. Then the information contained in x_i about x_j is bounded by a power law function of the distance:*

$$I(x_i : x_j) \overset{+}{<} K(q_h) \cdot d_{ij}^{-\langle \nu \rangle} \tag{3.6}$$

where $\nu_l \equiv \log_{\bar{b}}(1/\alpha_l) > 0$, $\bar{b}$ the average branching factor of H and α_l the information dissipation rate at level l.

Indeed we observe that the algorithmic information carried by one substring about another decays according to a power law function of their distance. This circumstance makes information storage local in the sense that mutual information between substrings exists mostly for nearby substrings. In the following section we will show that the locality of information entails the existence of short features in the whole data string.

4 Information Locality Implies Short Features

Since nearby substrings contain most information about each other in a BCH, we now prove that this implies compressibility of the concatenated string, which in turn implies the existence of a feature of the whole string.

Lemma 3 (Information of a string about another implies compressibility). *Let x and y be two strings with x carrying information about y: $I(x : y) > 0$. Then the composite string is compressible: $K(xy) \overset{+}{<} l(xy)$.*

Theorem 2 (Compressibility of substring implies existence of feature). *Let x be a string partitioned into y and p. Let further q be the shortest program with $U(q) = y$ and λ the program that computes y from q and specifies where to insert it into x: $\lambda(q, p) = x$. If y is $l(\lambda)$-compressible, $K(y) + l(\lambda) < l(y)$, then x is compressible as well and there is a feature f and a corresponding descriptive map f' of x such that*

$$f(f'(x)) = x$$

and $l(f) < l(y)$.

The gist of the results in this and the previous section can be summarized as follows. A BCH computes several strings from a single one at each level of the hierarchy which creates mutual information between the substrings at the leaves. In this section we have shown that it implies the existence of a feature of the whole string (concatenation of all leaves) such that the length of the feature is bounded by the length of just two neighboring leaves. Since the length of the leaves is generally much smaller than the length of the whole string, we have effectively derived a low bound on feature length. In other words, if our data breaks down into many small pieces, the size of the contained regularities is limited by the length of those little pieces.

5 A Tighter Bound on the Length of the Shortest Descriptive Map

This section is independent from the previous ones but aims at the same goal: the reduction of the time complexity of compression, which is directly tied to the lengths of features and descriptive maps. While in the previous sections we were occupied with the length of the features, this section goes back to deriving a tighter bound on descriptive maps than in our previous work [4, Theorem 6]. There, we have derived the bound $l(f') \overset{+}{\leq} 2\log K(x) + 4\log\log K(x)$ for the shortest descriptive map f'. Here, we managed to get rid of the factor two.

Theorem 3 (Bound on the length of descriptive map). *Let f' be the shortest descriptive map of a finite string x. Then the following bound holds on $l(f')$:*

$$l(f') \overset{+}{\leq} \log K(x) + 2\log\log K(x) \tag{5.1}$$

and the number of high values of $l(f')$ is low in the sense

$$P\{x : l(f') \geq s\} \leq s^3 2^{-s} \tag{5.2}$$

for any s and for all computable and semicomputable distributions P.

Note that this result is valid not only for BCHs, but for incremental compression in general.

6 Discussion

We have shown that data generated by a hierarchy of functions create power law distributed algorithmic information between two pieces of the data. Since correlation is a simple form of algorithmic information, the ubiquity of power law correlation functions in nature constitutes evidence that natural data could successfully be represented by hierarchies.

We have further shown that such structures imply that the data generated like this possesses simple features, i.e. that structure can be found at the smallest scale. This is crucial for the efficiency of induction algorithms since it allows us to find features whose description is bounded by the size of that scale. In other words, we can look at small pieces of the available data where we can find structure, derive a feature from that and continue to incrementally compress the data greedily.

One may wonder, if a BCH has troubles solving tasks like the induction of a program for π given a sequence of its digits, why is it that human programmers don't have particular difficulties in specifying such a program? However, such a program is never *induced* from the digits of π, but rather *deduced* from background knowledge about geometry and other, i.e. from other sources of information. Hence, such a task would be an unfair test for both algorithmic and human induction capabilities.

In future work, it would be interesting to derive an actual compression algorithm for hierarchically structured data and compare its time complexity to non-branching incremental compression and to non-incremental Levin search. Even more interesting would be to try to implement such an algorithm which is current work in progress.

A Proofs

Proof (Lemma 1). Recall that f_l and p_l are the shortest feature and parameter of q_{l-1} and therefore independent, $K(q_{l-1}) \overset{\pm}{=} l(f_l) + K(p_l)$, as was proven in [4, Corrolary 2]. From Eq. (3.1) we obtain

$$\begin{aligned} K(q_0) &\overset{\pm}{=} l(f_1) + K(p_1) \overset{\pm}{=} l(f_1) + \alpha_1 K(q_1) \overset{\pm}{=} l(f_1) + \alpha_1 \left(l(f_2) + \alpha_2 K(q_2)\right) \\ &\overset{\pm}{=} K(q_h) \prod_{l=1}^{h} \alpha_l + \sum_{m=1}^{h} l(f_m) \prod_{l=1}^{m-1} \alpha_l \end{aligned} \tag{A.1}$$

Since f_l and p_l cannot be made dependent by conditioning, we get $K(q_{l-1}|q_h) \overset{\pm}{=} K(f_l|q_h) + K(p_l|q_h)$. Due to assumption (2), the first term becomes $K(f_l|q_h) = K(f_l) \overset{\pm}{=} l(f_l)$. Therefore, the conditional version can be computed analogously to Eq. (A.1):

$$K(q_0|q_h) \overset{\pm}{=} K(q_h|q_h) \prod_{l=1}^{h} \alpha_l + \sum_{m=1}^{h} l(f_m) \prod_{l=1}^{m-1} \alpha_l \tag{A.2}$$

However, since $K(q_h|q_h) = O(1)$ we obtain for the information in q_h about q_0:

$$I(q_h : q_0) \equiv K(q_0) - K(q_0|q_h) \overset{\pm}{=} K(q_h) \prod_{l=1}^{h} \alpha_l$$

□

Proof (Lemma 2). We can in general expand [8, Theorem 3.9.1, p. 247]

$$K(y, z|a) \overset{\pm}{=} K(y|a) + K(z|y, K(y), a)$$

and insert it into the independence relation Eq. 3.4. This leads to

$$K(z|a) \overset{+}{=} K(z|y, K(y), a) \overset{+}{\leq} K(z|y)$$

where the last inequality follows from the fact that conditioning can only reduce the description length of z [8, Theorem 2.1.2, p. 108]. Subtracting this inequality from $K(z)$ yields $K(z) - K(z|a) \overset{+}{\geq} K(z) - K(z|y)$. Now we insert the definition of mutual information $I(a : z) \equiv K(z) - K(z|a)$ on both sides from which the claim follows. □

Proof (Theorem 1). First, from the result in Eq. (3.3) and Lemma 2 it follows that $I(x_i : x_j)$ decays exponentially with the height h of their common ancestor q_h

$$I(x_i : x_j) \overset{+}{\leq} K(q_h) \cdot \prod_{l=1}^{h} \alpha_l \tag{A.3}$$

under our assumptions. Consider that the maximal index distance between leaves in a perfect tree increases exponentially with the height h of the common ancestor:

$$d_{ij} < \prod_{l=1}^{h} \hat{b}_l \tag{A.4}$$

where $\hat{b}_l$ is the average branching factor at level l of the tree. By defining the total average branching factor $\bar{b} \equiv \left(\prod_{l=1}^{h} \hat{b}_l\right)^{1/h} > d_{ij}^{1/h}$, we can solve for $h > \log_{\bar{b}}(d_{ij})$ and compute:

$$\log_{\bar{b}}\left(\prod_{l=1}^{h} \alpha_l\right) < \sum_{l=1}^{\log_{\bar{b}}(d_{ij})} \log_{\bar{b}}(\alpha_l) = -\sum_{l=1}^{\log_{\bar{b}}(d_{ij})} \nu_l = -\langle v \rangle \log_{\bar{b}}(d_{ij}) = \log_{\bar{b}}\left(d_{ij}^{-\langle \nu \rangle}\right)$$

where $\nu_l \equiv \log_{\bar{b}}(1/\alpha_l) > 0$. Inserting this into Eq. A.3 concludes the proof. □

Proof (Lemma 3). Consider the general expansion [8, Theorem 3.9.1, p. 247]

$$K(xy) \overset{+}{=} K(x) + K(y|x, K(x))$$

I is defined by $I(x : y) \equiv K(y) - K(y|x)$ and is larger than zero by assumption. Since in general $K(y|x, K(x)) \overset{+}{\leq} K(y|x)$ we obtain

$$\begin{aligned} K(xy) &\overset{+}{=} K(x) + K(y) + K(y|x, K(x)) - K(y|x) - I(x : y) \\ &\overset{+}{<} K(x) + K(y) \overset{+}{\leq} l(x) + l(y) = l(xy) \end{aligned}$$

□

Proof (Theorem 2). Since y is $l(\lambda)$-compressible by q, $\lambda(q, p) = U(\langle \lambda, q, p \rangle) = x$ and $l(x) = l(y) + l(p)$, x is compressible as well:

$$K(x) \leq l(\lambda) + l(q) + l(p) = l(\lambda) + K(y) + l(x) - l(y) < l(x)$$

We define $f \equiv \langle \lambda, q \rangle$ and obtain $U(\langle f, p \rangle) = f(p) = x$ – the main feature equation. We can define the descriptive map f' by a function that removes y from x to obtain the remainder p: $f'(x) = p$. It suffices if it does so for that particular x and y, not in general.

From fs definition, we get $l(f) = l(\lambda) + l(q) = l(\lambda) + K(y) < l(y)$ since y is $l(\lambda)$-compressible by assumption. It follows that the (f, p)-pair compresses x at least to some extent, $l(f) + l(p) < l(y) + l(p) = l(x)$. Therefore, f is indeed a feature of x and its length is bounded by $l(y)$. □

Proof (Theorem 3). In general, the relation $K(p) \overset{+}{\leq} K(p|z) + K(z)$ is valid, since if p is computable by a detour via z, its shortest program without the detour can only be shorter. Setting $z = K(x)$ and conditioning on x leads to

$$K(p|x) \overset{+}{\leq} K(p|K(x), x) + K(K(x)|x) \tag{A.5}$$

The conditioning operation is not valid in general, however the detour argument is still valid in this case. Since $K(p|x) = l(f')$ [4, Lemma 1(2)] and $K(p|K(x), x) = O(1)$ [4, Theorem 3(3)], we get

$$l(f') \overset{+}{\leq} K(K(x)|x) \tag{A.6}$$

We now insert the "complexity of the complexity" expression in [8, Lemma 3.9.2, Eq. (3.18)] $K(K(x)|x) \overset{+}{\leq} \log K(x) + 2 \log\log K(x)$ and the first claim follows. The second claim is a property of $K(K(x)|x)$ [8, Eq. (3.13)] and therefore also holds for $l(f')$. □

References

1. Solomonoff, R.J.: A formal theory of inductive inference. Part I. Inf. Control **7**(1), 1–22 (1964)
2. Solomonoff, R.J.: A formal theory of inductive inference. Part II. Inf. Control **7**(2), 224–254 (1964)
3. Lin, H.W., Tegmark, M.: Why does deep and cheap learning work so well? arXiv preprint arXiv:1608.08225 (2016)
4. Franz, A.: Some theorems on incremental compression. In: Steunebrink, B., Wang, P., Goertzel, B. (eds.) AGI -2016. LNCS, vol. 9782, pp. 74–83. Springer, Cham (2016). doi:10.1007/978-3-319-41649-6_8
5. Bak, P.: How Nature Works: The Science of Self-organized Criticality. Copernicus, New York (1996)
6. Saremi, S., Sejnowski, T.J.: Hierarchical model of natural images and the origin of scale invariance. Proc. Natl. Acad. Sci. **110**(8), 3071–3076 (2013)
7. Lin, H.W., Tegmark, M.: Critical behavior from deep dynamics: a hidden dimension in natural language. arXiv preprint arXiv:1606.06737 (2016)
8. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications. Springer, New York (2009)