Scaling behaviour of entropy estimates

2002 J. Phys. A: Math. Gen. 35 1589

(http://iopscience.iop.org/0305-4470/35/7/308)

INSTITUTE OF PHYSICS PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND GENERAL

J. Phys. A: Math. Gen. **35** (2002) 1589–1596 PII: S0305-4470(02)29641-9

Scaling behaviour of entropy estimates

Thomas Schürmann

Research, WGZ-Bank Düsseldorf, Ludwig-Erhard-Allee 20, 40227 Düsseldorf, Germany

E-mail: thomas.schuermann@vr.wgz-bank.de

Received 4 October 2001, in final form 11 January 2002
Published 8 February 2002
Online at stacks.iop.org/JPhysA/35/1589

Abstract

Entropy estimation of information sources is highly non-trivial for symbol sequences with strong long-range correlations. The rabbit sequence, related to the symbolic dynamics of the nonlinear circle map at the critical point as well as the logistic map at the Feigenbaum point, is known to produce long memory tails. For both dynamical systems the scaling behaviour of the block entropy of order n has been shown to increase $\propto \log n$. In contrast to such probabilistic concepts, we investigate the scaling behaviour of certain non-probabilistic entropy estimation schemes suggested by Lempel and Ziv (LZ) in the context of algorithmic complexity and data compression. These are applied in a sequential manner with the scaling variable being the length N of the sequence. We determine the scaling law for the LZ entropy estimate applied to the case of the critical circle map and the logistic map at the Feigenbaum point in a binary partition.

PACS numbers: 75.40.Gb, 05.45.-a, 03.67.-a, 89.75.Da

1. Introduction

Partially random chains of symbols $s_1, s_2, \ldots, s_n$, drawn from some finite alphabet, appear in practically all sciences, e.g. written texts by humans, DNA sequences, spins in one-dimensional magnets, cellular automaton or bits in the storage and transmission of digital data like music scores and pictures. In this context it would be interesting to know to what degree these sequences can be 'compressed' without losing any information. This question was first posed from a probabilistic point of view by Shannon [1], who showed that in this context the relevant measure is the entropy (or average information content) h. With respect to statistical physics, h is related to the thermodynamic (Boltzmann) entropy of the underlying system. One reason for the significance of Shannon's framework is its association with the achievable compression ratio in the limit of very long sequences.

Formally, an information source is, by definition, a mechanism that produces messages over a finite alphabet $A = \{0, 1, \ldots, d-1\}$, i.e. a set of symbols of size d. Let a message of length n be denoted by $s_1^n = s_1, s_2, \ldots, s_n$, with $s_i \in A$. Moreover, let a code C be a translation mechanism (or algorithm) which for each n takes a message from A^n as input and transforms it into a binary sequence of variable length. Such a type of transformation is called a fixed- to variable-length encoding. If the message is produced by a stationary and ergodic source, the quantity of importance in determining the entropy rate h of the source is the block entropy

$$H_n = -\sum_{s_1,\ldots,s_n} p(s_1^n)\ \log p(s_1^n) \tag{1}$$

where $p(s_1^n)$ are the probabilities of subsequences or 'words' of fixed length n while the logarithm is, by definition, of base two. Then the Shannon entropy (or the entropy of the source) is formally given by

$$h = \lim_{n\to\infty} \frac{H_n}{n}. \tag{2}$$

One of the most fundamental theorems of information theory, Shannon's noiseless coding theorem, states that *any code has an expected length per symbol that is at least as large as the entropy rate* (2) *of the source* [1–3]. For the numerical determination of the entropy, simple word counting is commonly used to estimate the relative frequencies of subsequences or words of fixed length n. In the limit of large data sets, the relative frequency distribution becomes the underlying probability distribution.

However, the estimation of entropy can be highly non-trivial depending on the complexity of the source. This is especially true in the case of strong long-range correlations. In this case such correlations may be used to achieve higher compression rates as they reduce the uncertainty of yet unseen symbols. To detect them and take them into account may prove extremely difficult due to the exponential increase of the number of different blocks or 'words' with the length n. Several studies exist on both the estimation of h and the finite sample corrections for the block entropy H_n [4–6]. Empirical investigations lead to power-law scaling, $H_n \approx a\, n^{\mu} + b$, $\mu \approx 0.5$, in the case of several samples of texts and $\mu \approx 0.25$ for classical music [5,7]. For Markov processes of order m, one easily obtains the result that the entropy h of the source is reached for blocks of length $n = m$. Spectral analytical studies of deterministic sequences defined by a substitution process can be found in [8].

Furthermore, the border between periodicity and chaos is of special interest when the dynamics of the logistic map take on chaotic characteristics by period doubling. At the Feigenbaum point, Grassberger [9,10] finds the explicit expression

$$H_n \propto \log(n) \tag{3}$$

for blocks of length $n = 2^k$ i.e. the entropy per symbol approaches as $\propto \log(n)/n$ goes to zero. Another example of long-range order is associated with the sine circle map at the critical point [11,12]. As in the case of the logistic map the scaling behaviour of the block entropy is $\propto \log(n)$ [13].

However, a second theorem of Shannon states that *the lower bound* (2) *is asymptotically achievable* [1–3]. This leaves plenty of room for algorithmic design. As a matter of fact, coding algorithms can be split into two groups: codes that are designed for a specific (known) probability distribution over the input string, as well as universal codes that do not require any probabilistic distribution to be known *a priori.* The latter approach is also the best at coming close to the optimum over an entire class of sources. The first group includes Huffmann– and Shannon–Fano codes while the most commonly known algorithms of the second group are the ones by Lempel and Ziv (LZ) [14,15,19]. Originally constructed to provide a complexity

measure for individual finite sequences, the LZ algorithm is similar in spirit to the algorithmic complexity of Kolmogorov, Solomonoff and Chaitin [16–18]. Yet it has been shown that in the case of statistically stationary strings, the LZ-algorithm converges to the Shannon entropy if the length of the string tends to infinity [19].

2. Lempel–Ziv coding

The attempt to eliminate probabilistic ideas, i.e. the method with the best chances of taking long-range correlations into account, has been the coding scheme of LZ [14,15]. In this scheme the sequence of length N is parsed into words of variable length so that the next word is the shortest word not seen in the past. The corresponding code consists of pairs of numbers: each pair being a pointer to the previous occurrence of the prefix of the phrase and the last bit of the phrase.

Formally, the sequence is split into words $w_1, w_2, \ldots$ with $w_1 = s_1$, and w_{k+1} is the shortest new word immediately following w_k. For instance, the sequence $S = 1011010110110\ldots$ is split into segments

$$(1)(0)(11)(01)(011)(0110)(\ldots.$$

Thus each word w_k with $k > 1$ is an extension of some w_j with $j < k$ and a single symbol $s' \in A$. Any element of such partition is then simply encoded by the pair (j, s'). So far, we have only considered a binary alphabet, however an extension to any finite alphabet is obvious. It is evident that this is a good encoding in the sense that the string can be clearly decoded from the encoded sequence. Both the encoder and the decoder built up the same dictionary of words, and therefore the decoder can always find and add the new word. The encoding is efficient because for sequences of low entropy there are frequent repetitions, so that the average length of the words w_k increase faster, and the number of needed pairs (j, s') increase slower than for sources of higher entropy.

This Ziv–Lempel (ZL) coding is indeed a simplification of an earlier scheme by LZ [14], called LZ coding. Here the string is also split into a chain of words $w_1, w_2, \ldots$, however a word w_k is not necessarily an extension of a previous word w_j. Instead, it can be an extension of any substring of S which starts before w_k and might be overlapping. In the above-mentioned example we obtain a different parsing

$$(1)(0)(11)(010)(11011)(0\ldots.$$

This seems more efficient than ZL-parsing in the sense that the average word length extends faster and therefore can make better use of long-range correlation. Furthermore, the code length per word is slightly larger, so that it is not clear whether its compression performance is indeed better than ZL for small N. In any case the convergence of the compression rate with N is theoretically not well understood in either scheme. More precisely, for both the LZ and the ZL schemes, the entropy of stationary and ergodic sequences is related to the length l_N as follows

$$h = \lim_{N\to\infty} \frac{l_N}{N}. \tag{4}$$

Thus, both parsing schemes are universal in the sense that they reach asymptotically the entropy of the source. As in case (2), the convergence of the code length (4) versus the length N of the sequence is from above. In the case of block entropies, the larger the blocks the more correlations are taken into account when the length of the blocks become large, so that the information per symbol decreases with n. On the other hand, in the case of LZ, in addition to the specific sequence, the information about the probability distribution also has to be encoded implicitly. This mostly affects the beginning, when the information per symbol is the highest.

In other words, the LZ encoding is self-learning, and its efficiency increases with the length of the sequence.

Slight modifications of LZ's coding (i.e. prefix trees, dictionary extension etc) have been applied to entropy estimation of English texts by Grassberger [20] and others [21, 22]. To obtain a deeper insight into the finite sample behaviour, we compute the rate of convergence associated with the estimate as follows

$$h_N = \frac{l_N}{N}. \tag{5}$$

Let us consider the case of memoryless (Bernoulli) sources. Symbol '0' appears with the probability $p > 0$ and '1' appears with the probability $q = 1 - p$. Then the average excess of h_N with respect to the entropy of the source associated with the ZL scheme is of order $\propto 1/\log N$ [23]. On the other hand, the corresponding expression for the LZ coding is known to be of order $\propto \log\log N/\log N$ and this is conjectured to be the right order. Assuming that this conjecture is correct, the ZL algorithm, based on 'shorter' words in the parsing, is more efficient in the case of Bernoulli sources. Nevertheless, Shields [24] proved that such redundancy rates cannot be achieved for general stationary and ergodic sources. Furthermore, one should keep in mind that in most practical cases the above redundancy rates are not achievable.

3. Parsing the rabbit sequence

Let us take a look at the critical circle map which is represented by partitioning the time series on a binary (generating) partition. Denoting the pieces of the partition by symbols '0' and '1', the dynamical system is mapped clearly on an infinite binary string. This string, also called rabbit sequence [12], is generated by the grammatical rule [13]

$$\begin{aligned} b_0 &= 0 \\ b_1 &= 1 \\ b_{r+1} &= b_r b_{r-1} \qquad \text{for } r \geqslant 1. \end{aligned} \tag{6}$$

The right-hand side of the last equation formally represents the concatenation of the two finite predecessors b_{r-2} and b_{r-1} of b_{r+1}, called Fibonacci words. Thus, the rabbit sequence is equal to the infinite Fibonacci word. The first few Fibonacci words are

$$\begin{aligned} b_1 &= 1 \\ b_2 &= 10 \\ b_3 &= 101 \\ b_4 &= 10110 \\ b_5 &= 10110101 \\ b_6 &= 1.0.11.010.11011.0 \end{aligned}$$

and the dots in the last word indicate the first words of the LZ incremental parsing scheme. Many interesting characteristics of Fibonacci words concerning their symmetry, divisibility or properties of self similarity can be found in [12, 25–27] and corresponding references. One property is that the length (i.e. the number of bits) of the rth Fibonacci word, b_r, equals the $(r+1)$th Fibonacci number F_{r+1}. The Fibonacci numbers are defined by the recursive relation

$$\begin{aligned} F_0 &= 0 \qquad F_1 = 1 \\ F_{r+1} &= F_r + F_{r-1} \end{aligned} \tag{7}$$

i.e. the first are 0, 1, 1, 2, 3, 5, 8, 13, ...

To determine h_N, we construct the incremental parsing associated with the LZ algorithm. In the following let s_i^j be the substring of S starting at position i and ending at j, $s_i^j = s_i, s_{i+1}, \ldots, s_j$. Then we formulate the following:

Theorem 1 (parsing). *Let S be the rabbit sequence defined by* (6)*. Then, for all $k \geqslant 1$, the incremental parsing of S, associated with the LZ-coding scheme is given by the words*

$$w_k = S_{F_{k+1}}^{F_{k+2}-1}. \tag{8}$$

Proof. First note that for all $k \geqslant 1$, the length of the kth word w_k is equal to F_k, and the length of the concatenation of the first M words, $w_1, \ldots, w_M$, is $\sum_{k=1}^{M} F_k = F_{M+2} - 1$. Hence, the number of words in any Fibonacci word b_r is $r - 1$, and as a consequence, the length of the concatenation $w_1, \ldots, w_{r-1}$ is just one symbol shorter than b_r. Let u_r be the last symbol (or word of length 1) of b_r, then we receive the following expression for the rth Fibonacci word

$$b_r = w_1 w_2 \ldots w_{r-1} u_r. \tag{9}$$

Lemma 1. *For $r \geqslant 2$, the last symbol of b_r is*

$$u_r = \frac{1 - (-1)^r}{2}. \tag{10}$$

Proof (by induction).

(i) Let $r = 2k$ be even, $k \geqslant 1$. For $k = 1$ it follows that $b_2 = b_1 0$. Step from k to $k+1$: by induction assumption the last symbol of b_{2k} is '0' and by definition (6) we have $b_{2(k+1)} = b_{2k+1} b_{2k}$. Therefore, the last symbol of $b_{2(k+1)}$ is '0'.

(ii) Let $r = 2k + 1$ be odd, $k \geqslant 1$. For $k = 1$ it follows, $b_3 = b_2 1$, by definition. Step from k to $k + 1$: by assumption the last symbol of b_{2k+1} is '1'. By rule (6) we have $b_{2(k+1)+1} = b_{2k+2} b_{2k+1}$. Therefore, the last symbol of $b_{2(k+1)+1}$ is '1'. □

Lemma 2 (dynamical phrase generation). *For $k \geqslant 1$, let w_k^- be the kth word in* (8)*, but with an inverted last symbol. In this case the following rule of recursion is stated:*

$$\begin{aligned} &w_1 = 1 \qquad w_2 = 0 \\ &w_k = w_{k-2} w_{k-1}^- \qquad \textit{for } k \geqslant 3. \end{aligned} \tag{11}$$

Proof. Inserting expression (9) into the recursion relation of the Fibonacci words (6) and using the identity $u_{k-1} = u_{k+1}$, it follows that

$$w_k = u_k w_1 w_2 \ldots w_{k-2} \qquad \text{for } k \geqslant 1. \tag{12}$$

Then, (11) will be proven by induction:

(i) Let $k = 3$, then it follows $w_3 = 11 = w_1 0^-$. Induction step: assume (11) is true for k fixed. Multiplying (11) by w_{k-1} from the left and using (12) for $k + 1$, the result is $w_{k-1} w_k = w_{k+1}^-$. Multiplying the latter equation by $u_k w_1 w_2 \ldots w_{k-2}$ from the left and using $u_{k+2} = u_k$, the result is $w_{k+2} = w_k w_{k+1}^-$, which is the first part of the proof.

(ii) Let $k = 4$, then $w_4 = 010 = w_2 w_3^-$. Induction step: with similar arguments as in (i), from $w_{k+1} = w_{(k+1)-2} w_{(k+1)-1}^-$, we receive the final expression $w_{k+3} = w_{k+1} w_{k+2}^-$ which leads to the desired result.

Now, since the last symbol of successor w_{k-1}^{-} in (11) is 'flipped', one can be sure that no extension to the right of any word w_k can appear. To conclude the proof of theorem 1, we finally have to check if no substring exists equal to w_k before the word w_{k-2}, i.e. before position F_{k-1} in the rabbit sequence. However this property is given by theorem (a) in [13] by setting $k = s + 2$, i.e. all substrings of length $F_k - 1$, that start at position $1, \ldots, F_k$ are different. □

The compression ratio (5) can be determined by computing the number of bits l_N needed to encode the first N symbols of the rabbit sequence. We know from theorem 1 that the length of the kth word in the LZ-parsing is equal to F_k. From lemma 2 it follows that the reference word in the history of word w_k is at the position equal to F_{k-1}. According to the analytical expression of Fibonacci numbers we get the approximation

$$F_k \approx \frac{\phi^k}{\sqrt{5}} \qquad \phi = \frac{1+\sqrt{5}}{2} = 1.618\ldots$$

which becomes exact when rounded to the nearest integer [29]. Since one needs approximately $\lceil \log i \rceil$ digits to encode an integer i, the code length of the kth word equals $k \log \phi + \mathrm{O}(1)$. Adding up these contributions we find

$$l_N = \frac{\log \phi}{2} M^2 + \mathrm{O}(M). \tag{13}$$

The number of words, M, in the LZ-parsing is related to the number of symbols of the string by

$$N = F_{M+2} - 1. \tag{14}$$

Therefore, the inverse relation is up to rounding errors:

$$M = \frac{1/2 \ \log 5 + \log(N+1)}{\log \phi} - 2. \tag{15}$$

Inserting the last expression into (13), the leading term of the estimate h_N is

$$h_N = \frac{(\log N)^2}{2N \log \phi} + \mathrm{O}\left(\frac{\log N}{N}\right). \tag{16}$$

4. Incremental parsing at the Feigenbaum point

Also of special interest is the border between periodicity and chaos, when the dynamics of the logistic map becomes chaotic in the way of period doubling. Various properties of the associated symbolic dynamics have been discussed in the context of block entropy computations [9, 10, 29]. The recursive grammatical rule at the Feigenbaum point is [29]

$$\begin{aligned} a_0 &= 1 \\ a_1 &= 10 \\ a_{k+1} &= a_k a_{k-1} a_{k-1} \end{aligned} \tag{17}$$

for $k \geqslant 1$. The first few 'Feigenbaum words' are

$$\begin{aligned} a_0 &= 1 \\ a_1 &= 10 \\ a_2 &= 1011 \\ a_3 &= 10111010 \\ a_4 &= 1.0.11.1010.10111011. \end{aligned}$$

The dots in a_4 indicate the first words of the LZ incremental parsing scheme. In contrast to the case of the Fibonacci words, here the dots seem to be positioned at the end of any preceding a_k. This comfortable situation is stated more precisely in:

Theorem 2 (parsing). *Let S be the symbol sequence generated by the Feigenbaum words* (17). *Then the words in the LZ incremental parsing are*

$$\begin{aligned} w_0 &= 1 \\ w_k &= a_{k-1}^- \qquad \text{for } k \geqslant 1. \end{aligned} \tag{18}$$

Proof. By definition of LZ, for $k = 0$, we simply get $w_0 = 1$. For the case $k \geqslant 1$ we state the following:

Lemma 3. *For any* $k \geqslant 1$ *it is*

$$a_{k-1}a_{k-1} = a_k^-. \tag{19}$$

Proof. The proof is given by induction. Start of induction, $k = 1$: it follows by definition that $a_0a_0 = 11 = 10^- = a_1^-$. Induction step from k to $k + 1$: by induction assumption we have $a_k^- = a_{k-1}a_{k-1}$. 'Flipping' the last bit in the equation it simply follows that $a_k = a_{k-1}a_{k-1}^-$. Multiplying by a_k from the left we get by definition (17), $a_ka_k = a_ka_{k-1}a_{k-1}^- = (a_{k+1})^- = a_{k+1}^-$.

By lemma 1, for $k \geqslant 1$, any Feigenbaum word, i.e. the first 2^k bits of S can be expressed as $a_k = a_{k-1}a_{k-1}^-$. Since the suffix a_{k-1}^- is identical to the prefix a_{k-1} except for the last symbol and both are of equal length, it follows that $w_k = a_{k-1}^-$. Then, the preceding reference word of any w_k starts at the beginning of S and stops at the position of w_k in S. □

Thus, for $k \geqslant 1$, the length of w_k is 2^{k-1}, and the number of bits necessary for encoding w_k is $k + \mathrm{O}(1)$. Adding up these contributions we find the code length

$$l_N = \frac{M^2}{2} + \mathrm{O}(M). \tag{20}$$

The number of words, M, in the LZ-parsing are simply related to the number of symbols of the string (up to rounding errors) by

$$M = 1 + \log N. \tag{21}$$

Finally, inserting the last expression into (19), we find

$$h_N = \frac{(\log N)^2}{2N} + \mathrm{O}\left(\frac{\log N}{N}\right). \tag{22}$$

5. Conclusion

In the case of memoryless random sources the expected excess of h_N over the entropy of the source associated with the ZL-scheme is known to be of order $\propto 1/\log N$ [23], whereas for the LZ-coding it is $\propto \log\log N/\log N$. Therefore, in the case of Bernoulli sources, the ZL algorithm—based on 'shorter' words in the parsing—converges faster. On the other hand, in the case of the rabbit sequence or the logistic map at the Feigenbaum point, the convergence of the LZ scheme is faster than for the memoryless case.

The author did not prove similar results for the ZL-parsing, but we conject the convergence of leading order $\propto \log N/\sqrt{N}$ for both dynamical systems, which is noticeably slower compared to LZ. Thus, one would expect a faster convergence of the LZ-scheme for sequences with long-range correlations, whereas it seems to be vice versa in the case of Markov- or finite-state sources.

References

[1] Shannon C E 1948 *Bell Syst. Tech. J.* **27** 379
[2] McMillan B 1953 *Ann. Math. Stat.* **24** 196
[3] Khinchin A I 1957 *Mathematical Foundations of Information Theory* (New York: Dover)
[4] Grassberger P 1988 *Phys. Lett.* A **128** 369
[5] Ebeling W and Nicolis G 1992 *Chaos Solitons Fractals* **2** 1
[6] Schürmann T and Grassberger P 1995 *Chaos* **6** 414
[7] Hilberg W 1990 *Frequenz* **44** 243
[8] Queffélec M 1987 *Substitution Dynamical Systems—Spectral Analysis* (*Lecture Notes Mathematics vol 1294*) (Berlin: Springer)
[9] Grassberger P 1986 *J. Theor. Phys.* **25** 907
[10] Ebeling W 1992 *Statistical Physics and Thermodynamics of Nonequilibrium Systems* ed W Ebeling and W Muschik (Singapore: World Scientific)
[11] Procaccia I, Thomae S and Tressor C 1997 *Phys. Rev.* A **35** 1884
[12] Schroeder M R 1991 *Fractals Chaos Power Laws* (New York: Freeman)
[13] Gramss T 1994 *Phys. Rev.* E **50** 2616
[14] Lempel A and Ziv J 1976 *IEEE Trans. Inf. Theory* **22** 75
[15] Ziv J and Lempel A 1977 *IEEE Trans. Inf. Theory* **23** 337
[16] Kolmogorov A N 1965 *Probl. Inf. Transmiss.* **1** 1
[17] Solomonoff R J 1964 *Inf. Control* **7** 224
[18] Chaitin G J 1966 *J. Assoc. Comput. Mach.* **13** 547
[19] Ziv J and Lempel A 1978 *IEEE Trans. Inf. Theory* **24** 530
[20] Grassberger P 1989 *IEEE Trans. Inf. Theory* **35** 669
[21] Kontoyiannis I, Algoet P H, Suhov Yu M and Wyner J 1998 *IEEE Trans. Inf. Theory* **44** 1319
[22] Kontoyiannis I 1997 *NSF Technical Report* No 97 Dept. Stat., Stanford University
[23] Louchard G and Szpankowski W 1996 *Preprint*
[24] Shields P 1993 *IEEE Trans. Inf. Theory* **39** 520
[25] DeLuca A 1981 *Inform. Process. Lett.* **12** 193
DeLuca A 1995 *Inform Process. Lett.* **54** 307
[26] Droubay X 1995 *Inform. Process. Lett.* **55** 217
[27] Sulston K W, Burrows B L and Chishti A N 1995 *Physica* A **217** 146
[28] Knuth D E 1973 The art of computer programming *Fundamental Algorithms* vol 1 (Reading, MA: Addison-Wesley)
[29] Ebeling W 1997 *Physica* D **109** 42 and references therein