Estimating topological entropy using the context-tree weighting method

Yoshito Hirata and Alistair Mees

Centre for Applied Dynamics and Optimization
Department of Mathematics and Statistics
The University of Western Australia
35 Stirling Highway, Crawley, WA 6009, AUSTRALIA
Phone:+61-8-9380-7173, Fax:+61-8-9380-1028
Email: yoshito@maths.uwa.edu.au

Abstract

We propose a method for estimating topological entropy from observed time series. Estimating topological entropy is difficult, especially from observed time series. Established methods often require adjusting ad hoc parameters. The proposed method does not need such ad hoc parameters because it employs a data compression technique called the context-tree weighting method. We present some examples including some one-dimensional maps and Hénon map, demonstrating that the estimated topological entropy converges to the theoretically correct topological entropy, and that its convergence is fast.

Keywords topological entropy, symbolic dynamics, context-tree weighting method.

1. Introduction

The technique described can be used to estimate topological entropy of the processes that give time series of real values that are then quantized, or the processes that give symbolic sequences directly.

Estimating topological entropy is difficult, especially from observed time series. There are two known methods for estimating the topological entropy from an observed time series: counting the number of distinct admissible substrings [1], and counting the number of periodic points of some periods [2]. Both methods need to use ad hoc parameters. The first method requires specifying the length of the distinct substrings to count. while the second method needs detect all periodic points of some large period, and possibly also those of all smaller periods.

We propose a method for estimating topological entropy from observed time series via symbolic dynamics using a data compression technique called the context-tree weighting method [3, 4]. The context-tree weighting method encodes a symbolic sequence without any prior knowledge except for the number of symbols. It has been used to test stationarity in dynamical systems [5]. Our key idea is to use the context-tree weighting method to build the best Markov model describing the dynamics of a symbolic sequence.

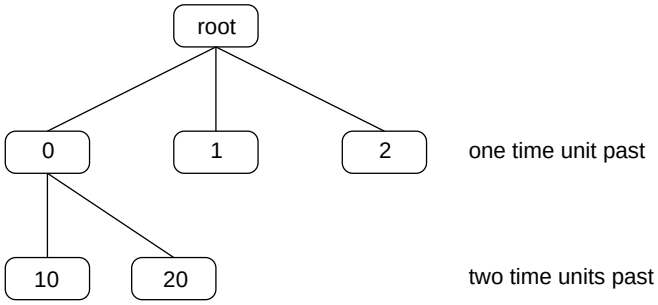

Figure 1: An example of context trees.

2. Approximating Dynamics Using a Markov Model

We assume that one already has a well defined partition for converting a time series into a symbolic sequence, and that this symbolic sequence is neither periodic nor eventually periodic.

Topological entropy for Markov models is easily found. Let P be the transition matrix of a Markov model, where P_{ij} is the conditional probability that the current state i is followed by a state j. Define the adjacency matrix T for the Markov model by $T_{ij} = 1$ if $P_{ij} > 0$, and $T_{ij} = 0$ otherwise. Let κ be the largest absolute eigenvalue for T. Then the topological entropy of the Markov model is equal to $\log \kappa$ [6, p. 120]. This motivates us to approximate symbolic dynamics by a Markov model [1]. Throughout, we use 2 for the base of the logarithm.

For constructing a Markov model, we first build a context tree, or *tree machine.* An example of a context tree is shown in Fig. 1. Each context of the tree is represented by a sub-

[1]There are some previous works on building Markov models or graphs for calculating topological entropy [7, 8].

string of the past symbols ordered chronologically. In the context tree, the most recent symbol is the closest to the root. Therefore, if one wants to reach context 20 in the example of Fig. 1, one starts from the root, and descends arc 0 (one time unit past), followed by arc 2 (two time units past).

To build this sort of context tree from a symbolic sequence, we employ a data compression technique, the context-tree weighting method [3, 4].

Following [4], we construct a context tree for a symbolic sequence $x_1x_2\cdots x_n$ over alphabet $X = \{0, 1, \cdots, |X|-1\}$. Firstly attach, in front of $x_1x_2\cdots x_n$, a symbol ϵ, representing the beginning of the symbolic sequence. Secondly prepare a tree consisting of just the root, λ. For $i = 1$ to $(n-1)$, add, into the tree, a context $\epsilon x_1x_2\cdots x_{i-1}$ for x_i. While descending the tree from the root to reach $\epsilon x_1x_2\cdots x_{i-1}$, we pass all the nodes for its suffixes. At each of them, whether it is an internal node or not, count the next symbol x_i.

Using the counts obtained at each node, we compute a coding distribution, which is used for evaluating the context tree later. Suppose that at context c, we have occurrences c_i for symbol i. Then, we estimate the probability of the next symbol j, using Krichevsky-Trofimov estimator, by $(c_j + 1/|X|)/\sum_j(c_j+1/|X|)$. The code length for coding the next symbol j given c is $-\log\{(c_i + 1/|X|)/\sum_j(c_j + 1/|X|)\}$. It is worth mentioning that the code length takes into account the modeling cost of using the estimator.

Using the Krichevsky-Trofimov estimator, we define P_e^c for context c:

$$P_e^c = \frac{\prod_{i=0}^{|X|-1}\prod_{j=0}^{c_i-1}(j+1/|X|)}{\prod_{k=0}^{\sum_j c_j-1}(k+1/|X|)}. \tag{1}$$

The value $-\log P_e^c$ can be interpreted as the total coding length necessary for coding, without prior knowledge, symbols appearing after the context c in $x_1x_2\cdots x_n$.

Although we can use P_e^c for coding sequences, we can do better by *mixing*: define P_w^c for context c by

$$P_w^c = \begin{cases} \frac{1}{2}\{P_e^c + P_w^{\epsilon c}\prod_{i=0}^{|X|-1}P_w^{ic}\}, & \text{if } c \text{ is internal,} \\ P_e^c, & \text{otherwise.} \end{cases} \tag{2}$$

The quantity $-\log P_w^c$ can be regarded as the total code length for symbols appearing after the context c in the original symbolic sequence when one uses the mixing strategy. Therefore $-\log P_w^\lambda$ can be interpreted as the total code length for $x_1x_2\cdots x_n$. Using arithmetic coding, the code length for the entire sequence is at most 2 bits more than $-\log P_w^\lambda$ [3].

We are interested in finding the best model rather than coding the sequence. To do so we find a subtree in a context tree which can encode the next symbol appearing after the given symbolic sequence with the minimum code length. When a context c is internal, it holds that

$$-\log P_w^c \le 1 + \min\{-\log P_e^c, \sum_{i\in\{\epsilon,0,1,\cdots,|X|-1\}} -\log P_w^{ic}\}. \tag{3}$$

Therefore, we search the context tree depth-first, and at each node c, we prune its children if $-\log P_e^c \le 1 - \sum_i \log P_w^{ic}$.

Pruning makes the context tree simpler. But the context tree is sometimes not Markov. One can *Markovise* the pruned context tree using the following algorithm:

1. Build a tree such that it contains every prefix of each context in the pruned context tree.
2. End the algorithm if every path with a positive probability has its destination among the leaves of the tree.
3. Find paths with positive probabilities which do not have their destinations among the leaves, and extend the tree so that it contains these destinations. We also add their prefixes if needed. Go back to Step 2.

For a context with counts $\{c_i\}$, we give the conditional probability of the next symbol i by $c_i/\sum_j c_j$, obtaining a Markov model for a symbolic sequence $x_1x_2\cdots x_n$.

3. Examples

We demonstrate the performance of the proposed method with the following maps:

$$x_{t+1} = \begin{cases} \frac{3}{2}x_t + \frac{2}{5}, & x_t \in \left[0, \frac{2}{5}\right], \\ -\frac{5}{2}(x_t - \frac{4}{5}), & x_t \in \left(\frac{2}{5}, \frac{4}{5}\right], \\ \frac{4}{3}(x_t - \frac{4}{5}), & x_t \in \left(\frac{4}{5}, 1\right], \end{cases} \tag{4}$$

$$x_t = 3.98{x_t}^3 - (3.98-1)x_t, \tag{5}$$

$$\begin{cases} x_{t+1} = 1 - 1.4{x_t}^2 + 0.3y_t, \\ y_{t+1} = x_t. \end{cases} \tag{6}$$

The map in Eq. (4) is Markov when one divides $[0, 1]$ into $\left[0, \frac{4}{15}\right]$, $\left[\frac{4}{15}, \frac{2}{5}\right]$, $\left[\frac{2}{5}, \frac{4}{5}\right]$, and $\left[\frac{4}{5}, 1\right]$. The maps in Eqs. (5) and (6) are called the antisymmetric cubic map [9, p. 161] and the Hénon map respectively. For the one-dimensional maps, we obtained symbolic sequences by dividing the phase space at their critical points. For the Hénon map, we used the partition proposed in [10].

For comparison, we estimated topological entropy of each map by counting the distinct admissible substrings with a finite length [1]. For a map with a partition, let $N(n)$ be the number of admissible substrings with length n. Then an estimate $h(n)$ of the topological entropy for length n is given by

$$h(n) = \log N(n) - \log N(n-1). \tag{7}$$

However it was reported that this estimate would oscillate [1]. Instead, we averaged the estimates for some lengths and obtained another estimate $\bar{h}(n)$:

$$\bar{h}(n) = \frac{1}{n-m} \sum_{i=m}^{n} h(i), \tag{8}$$

where $m = \lfloor n/2 \rfloor$. It can be easily shown that $\bar{h}(n)$ converges to the topological entropy.

Given the length l of a symbolic sequence, we chose n to be $\lfloor \log l / \log |X| \rfloor$ and estimated $N(n)$ and $N(m)$ by counting substrings appearing in a given symbolic sequence.

All the tested examples are listed in Table 1. For Eqs. (5) and (6), we plotted the estimates obtained using the conventional and proposed methods in Fig. 2. Estimates obtained using the proposed method showed good agreement with their theoretical values or values reported in previous works. In most cases, the estimates were better than the conventional technique.

4. Conclusion

We proposed a method estimating topological entropy. The context-tree weighting method enables us to construct a Markov model from a symbolic sequence without any ad hoc parameters. One can obtain the topological entropy by building the adjacency matrix associated with the Markov model, finding its eigenvalue with the maximum absolute value, and taking its logarithm. We demonstrated the efficacy of the method with some bimodal maps and the Hénon map. Estimates obtained by the proposed method converged to their theoretical values, or to literature values, and convergence was rapid compared to the standard method [1].

Acknowledgement

We appreciate Dr Kevin Judd and Mr Devin Kilminster for their helpful comments.

References

[1] J. Crutchfield and N. Packard, Symbolic dynamics of one-dimensional maps: Entropies, finite precision, and noise, Int. J. Theor. Phys., vol. 21, pp. 433–466, (1982).

[2] D. Auerbach, P. Cvitanović, J.-P. Eckmann, G. Gunaratne, and I. Procaccia, Exploring chaotic motion through periodic orbits, Phys. Rev. Lett., vol. 58, pp. 2387–2389, (1987).

[3] F. M. J. Willems and Y. M. Shtarkov and T. J. Tjalkens, The context-tree weighting method: Basic properties, IEEE Trans. Inf. Theory, vol. 41, pp. 653–664, (1995)

[4] F. M. J. Willems, The context-tree weighting method: Extensions, IEEE Trans. Inf. Theory, vol. 44, pp. 792–798, (1998).

[5] M. B. Kennel and A. I. Mees, Testing for general dynamical stationarity with a symbolic data compression technique, Phys. Rev. E, vol. 61, pp. 2563–2568, (2000); M. B. Kennel and A. I. Mees, Data compression, dynamics, and stationarity, in *Nonlinear Dynamics and Statistics*, edited by A. I. Mees, pp. 387–412, Birkhäuser, Boston USA, 2001.

[6] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding*, Cambridge University Press, Cambridge, UK, 1995.

[7] G. D'Alessandro and P. Grassberger and S Isola and A Politi, On the topology of the Hénon map, J. Phys. A, vol. 23, pp. 5285–5294, (1990); P. Góra and A. I. Boyarsky, Computing the topological entropy of general one-dimensional maps, Trans. Amer. Math. Soc., vol. 323, pp. 39-49, (1991); . Balmforth and E. A. Spiegel, Topological entropy of one-dimensional maps: Approximations and bounds, Phys. Rev. Lett., vol. 72, pp. 80–83, (1994).

[8] G. Froyland, O. Junge, and G. Ochs, Rigorous computation of topological entropy with respect to a finite partition, Physica D, vol. 154, pp. 68–84, (2001).

[9] B.-L. Hao, *Elementary Symbolic Dynamics and Chaos in Dissipative Systems*, World Scientific, Singapore, 1989.

[10] P. Grassberger and H. Kantz, Generating partitions for the dissipative Hénon map, Physics Letters, vol. 113A, pp. 235–238, (1985); Points for partitioning in this case are listed in [8].

[11] J. Jacobs, E. Ott, and B. R. Hunt, Calculating topological entropy for transient chaos with an application to communicating with chaos, Phys. Rev. E, vol. 57, pp. 6577–6588, (1998).

[12] L. Block and J. Keesling, Computing the topological entropy of maps of the interval with three monotone pieces, J. Stat. Phys., vol. 66, pp. 755–774, (1992).

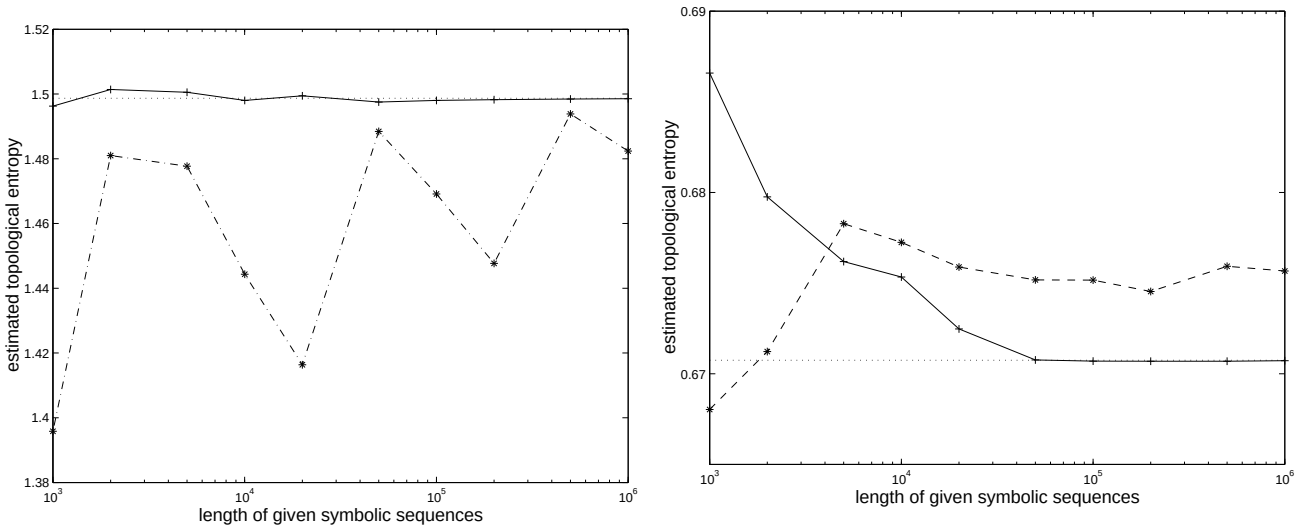

Figure 2: Convergence of estimated topological entropy obtained for Eq. (5) (left) and Eq. (6) (right). In each graph, we plotted the mean of estimates obtained using the proposed method ($+$) and that of the conventional method ($*$). In the left graph, the dotted line is the theoretical value for the topological entropy obtained with a technique described in [12]. In the right graph, the dotted line is the topological entropy calculated in a previous work [11].

Table 1: Maximum, minimum, mean, and standard deviation of the topological entropy estimated for each map. At each length, we used 99, 100 and 119 symbolic sequences generated from different initial points for Eqs. (4), (5), and (6), respectively. Theoretical values and a previous work are listed together. For Eq. (4), we obtained the theoretical value from its adjacency matrix. For Eq. (5), we employed a technique in [12] for finding the theoretical value. For Eq. (6), we listed the most precise value known so far [11]. Also shown are the means of topological entropies estimated using a conventional method used in [1].

map	length	proposed method				conv.	theo.
		max.	min.	mean	sta. dev.	mean	
Eq. (4)	50	1.00000	0.38829	0.93293	0.11289	0.85749	0.94678
	100	1.00000	0.66786	0.94579	0.04957	0.87113	
	150	1.00000	0.74633	0.94465	0.03049	0.89545	
	200	0.94678	0.82710	0.94482	0.01320	0.90261	
	250	0.94678	0.91319	0.94644	0.00338	0.91235	
	300	0.94678	0.94678	0.94678	0.00000	0.91830	
	350	0.94678	0.89034	0.94621	0.00567	0.92120	
	400	0.94678	0.94678	0.94678	0.00000	0.92252	
	450	0.94678	0.94678	0.94678	0.00000	0.92312	
	500	0.94678	0.94678	0.94678	0.00000	0.92348	
Eq. (5)	1,000	1.50140	1.45203	1.49627	1.2700×10^{-2}	1.39582	1.49869
	10,000	1.50140	1.48724	1.49802	2.6874×10^{-3}	1.44433	
	100,000	1.49878	1.49656	1.49802	4.3724×10^{-4}	1.46913	
	1,000,000	1.49863	1.49843	1.49855	4.6567×10^{-5}	1.48238	
Eq. (6)	1,000	0.72084	0.63728	0.68659	1.4140×10^{-2}	0.66803	0.67075
	10,000	0.67936	0.66888	0.67534	2.145×10^{-3}	0.67726	± 0.00004
	100,000	0.67107	0.67001	0.67070	1.870×10^{-4}	0.67517	
	1,000,000	0.67079	0.67056	0.67072	3.562×10^{-5}	0.67567	