Estimating topological entropy via a symbolic data compression technique

Yoshito Hirata* and Alistair I. Mees
Centre for Applied Dynamics and Optimization, Department of Mathematics and Statistics, The University of Western Australia, 35 Stirling Highway, Crawley, WA 6009, Australia

(Received 5 June 2002; revised manuscript received 8 November 2002; published 12 February 2003)

We estimate topological entropy via symbolic dynamics using a data compression technique called the context-tree weighting method. Unlike other symbolic dynamical approaches, which often have to choose ad hoc parameters such as the depth of a tree, the context-tree weighting method is almost parameter-free and infers the transition structure of the system as well as transition probabilities. Our examples, including a Markov model, the logistic map, and the Hénon map, demonstrate that the convergence is fast: one obtains the theoretically correct topological entropy with a relatively short symbolic sequence.

DOI: 10.1103/PhysRevE.67.026205 PACS number(s): 05.45.Tp, 05.10.−a

I. INTRODUCTION

Topological entropy is an important index for dynamical systems. It was proposed primarily as an indicator of chaos: how complex dynamical systems are. Now topological entropy plays a role in characterizing dynamical systems theoretically. For example, for a continuous map on an interval, positive topological entropy implies the existence of periodic points with a period not equal to a power of 2 ([1], p. 504). It also works in practical problems. It gives an upper bound for metric entropy ([2], p. 194), showing the minimum number of symbols necessary to encode a point using an infinite symbolic sequence.

As the definition of topological entropy is not suitable for practical calculation, several methods have been proposed for estimating it [3–13].

Estimating topological entropy is more difficult when one wants to approximate it from observed time series because we often have to choose ad hoc parameters. When one does not have a fitted continuous model, one may find the topological entropy by counting the number of distinct admissible substrings [3] or counting the number of periodic points [4,5]. For the first method, one wants to choose the length for counting distinct symbolic sequences as long as possible to find a good estimate, although it is not clear what length is reasonable. The second method is likely to fail to detect periodic points, especially for higher periods.

In this paper, we propose a method for estimating the topological entropy from observed time series using a data compression technique. Our assumption is that one only has a finite amount of data and one knows how to encode it into a symbolic sequence using a finite number of symbols. The strength of the proposed method is that we select a Markov model with states which are variable length substrings of the past symbols, closer to be the most relevant for the observed time series in terms of code length. As a set of substrings conditioning the past is chosen automatically from a given symbolic sequence, we do not have any *ad hoc* parameters to specify.

In Sec. II, we define topological conjugacy, which facilitates finding the topological entropy by representing the dynamical system in a different way. In Sec. III, we approximate a dynamical system on symbols with a Markov model using one of data compression techniques, the context-tree weighting method [14,15], and estimate the topological entropy of the original system by finding that for the Markov model. In Sec. IV, we present three examples, namely, a Markov model, the logistic map, and the Hénon map, showing how well we can estimate their topological entropies using the proposed method. Section V concludes this paper.

II. TOPOLOGICAL CONJUGACY

In this section, we define topological conjugacy. The aim of this section is to explain that if one has a dynamical system topologically conjugate with the original system, we can find the topological entropy of the original system by finding that of the conjugate system.

There are several possible definitions of topological entropy which are all equivalent ([1], pp. 105–118). We use the definition proposed in Ref. [16].

Let us assume that there are two dynamical systems which behave in the same manner, but in different *coordinates*. These two systems are transformed into each other by a topological conjugacy. Formally, a topological conjugacy is defined in the following way.

Let M be a metric space and $f:M\to M$ a map on it. We call the pair (M,f) a dynamical system.

Suppose that there are two dynamical systems (M,f) and (N,g). Let $h:M\to N$ be a continuous map. Suppose that h is one-to-one and onto, and its inverse is continuous. Then we call h a topological conjugacy and the relation $h(f(x))=g(h(x))$ holds for any $x\in M$. We also say that (M,f) and (N,g) are topologically conjugate.

Topological entropy is invariant under topological conjugacy ([1], p. 109).

When two dynamical systems are topologically conjugate, one of them may be a symbolic dynamics. We call (N,g) a symbolic dynamics when N is a shift space consisting of infinite symbolic sequences with a finite number of symbols and g is a shift on it.

*Electronic address: yoshito@maths.uwa.edu.au

III. ESTIMATING TOPOLOGICAL ENTROPY FROM TIME SERIES

In this section, we describe a method for estimating topological entropy using a symbolic sequence. We would like to build a Markov model from a symbolic sequence because one can easily find topological entropy for a Markov model by finding the largest eigenvalue of its adjacency matrix [17]. There is some previous work on building Markov models or graphs for estimating the topological entropy when a map is known [8,10,13] or a map is estimated [11].

First, we will build a context tree, a tree showing a conditional structure in the symbolic sequence, by following Willems [15]. At this stage, the context tree does not represent well which substring appears and which does not. Next the built context tree is pruned to obtain the best context tree in terms of code length. This may not be immediately suitable for finding topological entropy. Therefore in the third step, the pruned context tree is changed into a Markov model by extending the pruned context tree. Then finding the largest eigenvalue for the adjacency matrix of the Markov model gives the topological entropy, which we are looking for.

In what follows, suppose that we have a topological conjugacy of an original system into the corresponding symbolic dynamics, and that we have a finite symbolic sequence generated by an observed time series. We also assume that the original dynamics is topologically transitive ([1], p. 27) when we restrict the space to the attractor. Then the topological conjugacy forces the corresponding symbolic dynamics to be also topologically transitive ([17], p. 205). Therefore, if we have an aperiodic observed time series of infinite length, we can expect to see every admissible substring of finite length.

A. What is a context tree

Our *symbolic dynamical model* is a context tree. A context tree shows probabilistic structure conditioned on past substrings of a symbolic sequence. It was first used in dynamical systems in Refs. [18,19]. It has been applied for testing stationarity [19,20] and estimating the entropy rate [21]. In this paper, we only deal with cases where one needs two symbols, although general cases can be also handled similarly [18–21].

Let X be an alphabet, that is, a finite set of distinct symbols, and C_i $(i=0,1,\ldots,|C|-1)$ be substrings of a symbolic sequence on X. Assume that a set of C_i's satisfies the suffix condition, meaning that the substring corresponding to each C_i is not a suffix of C_j for any $j \neq i$ ([22], p. 120). Then these substrings can be represented as leaves of a tree, rooted on their *last* symbols and sharing paths corresponding to common suffices. (So the most recent symbols are closest to the root in such a tree.) We call this tree a context tree and each of the substrings corresponding to its leaves a *context*. We call the tree Markov if the set of C_i's forms a Markov chain.

We show an example of a context tree in Fig. 1. Its contexts are 0, 001, 101, and 11. Descending the tree corresponds to going back into the past (i.e., to earlier symbols). For example, to reach the node 001 from the root of the tree, we descend the arc 1 starting from the root, showing a symbol 1 time unit in the past, followed by descending the arc 0 (2 time units past), and the arc 0 (3 time units past).

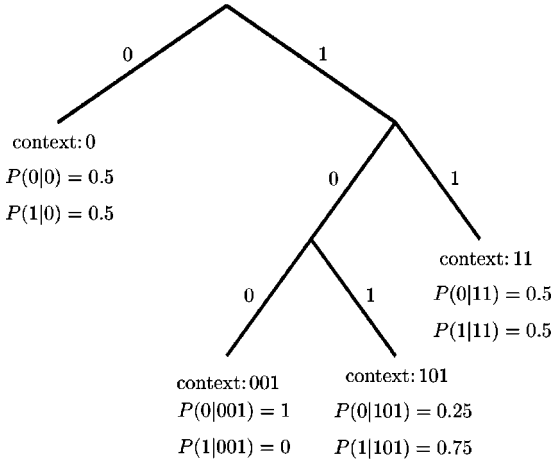

FIG. 1. An example of a context tree.

A set of contexts C_i gives a description of conditional structure. We denote the conditional probability of the next outcome X given the current context C_i by $P(X|C_i)$. For example, if the conditional probability at context C_i is $P(0|C_i)=1$, the behavior is deterministic for this context: one always has 0 after C_i.

B. Using a context tree to evaluate topological entropy

In the preceding subsection, we explained a context tree as a tool for representing a conditional structure in a symbolic sequence. This section shows how to obtain this kind of context tree from a symbolic sequence. We utilize insights from data compression, specifically universal coding, whose techniques compress data generated from any stationary, probabilistic source asymptotically in the shortest length without using prior knowledge.

The steps for obtaining a context tree for evaluating topological entropy are as follows.

(1) Build a context tree in essentially the same way as Willems did in his context-tree weighting method [15]. (We will outline the method shortly.)

(2) Prune the context tree to obtain the best model in terms of the code length.

(3) Convert the pruned context tree into a Markov chain.

In more detail, the steps are as follows.

1. Building a context tree from a symbolic sequence

Suppose one has a symbolic sequence $x_1x_2\cdots x_N$ of length N over X. Let ϵ be a symbol marking the beginning of the symbolic sequence: we change the original sequence $x_1x_2\cdots x_N$, into $\epsilon x_1x_2\cdots x_N$, which is a sequence over the new alphabet $X\cup\{\epsilon\}$.

For symbol x_i, the past subsequence is $\epsilon x_1x_2\cdots x_{i-1}$ if $i>1$, and ϵ if $i=1$. Using all the past subsequences, we build a context tree as follows.

First we initialize the tree to just contain the root with a single child ϵ. We record at the child node whether the next

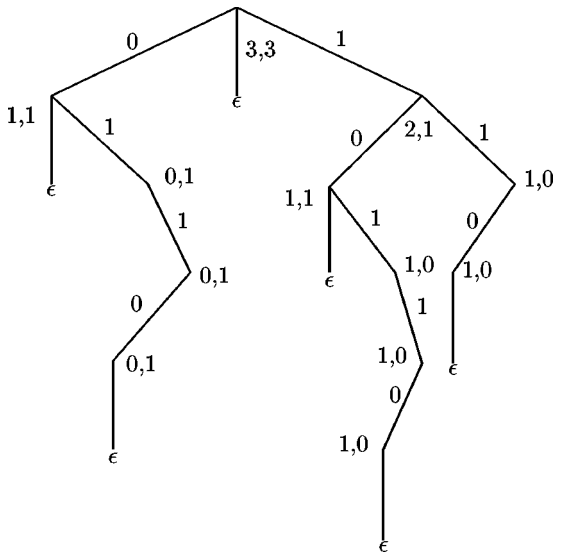

FIG. 2. Building an unbounded-depth context tree from a symbolic sequence 011010. We start building it by preparing a tree containing just a root with a child ϵ. Then we insert all the past subsequences into it. At each leaf, where one finds ϵ, we mark the next symbol. Using this marking, we find the counts of the next symbols 0 and 1 for every internal node by going up the tree from the leaves. Two numbers shown next to each context show, from the left, the counts for the next symbols 0 and 1 observed after the context in the symbolic sequence ϵ011010.

symbol x_1 is 0 or 1. This process corresponds to the case where $i=1$.

Suppose that we have built the tree using the subsequences up to $\epsilon x_1 x_2 \cdots x_{i-1}$ if $i>1$ or ϵ if $i=1$. Then we insert the subsequence $\epsilon x_1 x_2 \cdots x_i$ into the existing tree by following the symbols $x_i, x_{i-1}, \ldots, \epsilon$ backwards in time, while descending the tree from the root; we extend the tree as required, until the symbol ϵ has been added. At the leaf corresponding to the context $\epsilon x_1 x_2 \cdots x_i$, we record whether the next symbol x_{i+1} is 0 or 1. After the construction, every past sequence is represented by a unique leaf.

The construction we have given here is clearly inefficient, but it is not hard to encode the same process so that the tree can be constructed in space of order N.

The context tree does not at present have the conditional probabilities seen in Fig. 1. We will allocate these using next-symbol counts.

For any given node, let n_0 and n_1 be the counts for the next symbols 0 and 1, respectively. The counts at any leaf have already been allocated: they are either $(n_0=1,\ n_1=0)$ or $(n_0=0,\ n_1=1)$. The count at any internal node is the cumulative sum of the counts of its descendants, which can be generated by tracing the tree upwards from the leaves. We could immediately define a conditional probability at each node, but for this paper we will be pruning the tree first, which we describe shortly.

An example of the construction we have just described is shown in Fig. 2. The given sequence is 011010. First we add ϵ at the beginning of the sequence, obtaining ϵ011010. The past subsequences are ϵ, $\epsilon 0$, $\epsilon 01$, $\epsilon 011$, $\epsilon 0110$, and $\epsilon 01101$ for the symbols $x_1=0$, $x_2=1$, $x_3=1$, $x_4=0$, $x_5=1$, and $x_6=0$, respectively. We add each past subsequence into the context tree we are building. At each leaf, we mark the next symbol, which assigns its counts. Using this marking, we obtain the counts at each internal node.

2. Batch pruning

The context tree built above is not suitable for estimating topological entropy, since it does not show which substrings can appear and which are forbidden. Instead of using the context tree as it is, we prune it in a way that is optimal in terms of ''code length'' by importing a method from data compression. This step was also discussed in Ref. [18].

(*a*) *Two code lengths for each context.* We call a coding technique *universal* if it achieves the shortest length for any stationary information source asymptotically. There are several universal codings [22]. When these methods choose a model from a class of models, the selected model can be regarded the best in the class for the given data. We use the minimum description length (MDL) principle, which has been applied for modeling a continuous dynamical system from observations [23]. In the present work, we use the MDL principle for finding the best symbolic dynamical model, that is, the best context tree.

We find the best context tree as a subtree of the context tree built above. At each node we decide whether we keep its children or prune them. For this, we define two code lengths at each node: the code length when using only the counts at that node, and the code length when also using knowledge about its children. If the first code length is shorter than or equal to the second code length, pruning the children gives a shorter code length.

Observe that each node in a context tree corresponds to a unique substring represented by the recent symbols. In what follows, we use a substring for showing the corresponding node.

Given a node γ, the code length for the next symbol s is given by $-\log P(s|\gamma)$, where $P(s|\gamma)$ is the conditional probability of the next symbol s given node γ [22]. Therefore the current problem, how to define the two types of code lengths, is equivalent to how to decide the conditional probabilities for the two different situations. We prepare the two code lengths by modifying slightly the method proposed by Willems [15]. Throughout this paper, we use 2 for the base of the logarithm.

(*b*) *Code length when using just counts.* First we define a code length at node γ using just the counts for the next symbols. Suppose that the counts for the symbols 0 and 1 appearing after a substring γ are a and b. When we do not have any prior knowledge about the next-symbol distribution, the Krichevsky-Trofimov estimator [14,15] gives a good estimate for the conditional probability. Under the Krichevsky-Trofimov estimator, we always add 1/2 to the count for each symbol at each node: that is, the count is always 1/2 greater than the number actually observed. Then the conditional probabilities for the next symbols 0 and 1 are estimated as

$$\frac{a+1/2}{a+b+1}, \quad \frac{b+1/2}{a+b+1}, \tag{1}$$

respectively. The estimator gives an estimate for the conditional probability even if the counts are both 0. Given the counts a and b, the Krichevsky-Trofimov estimator gives the code lengths for the next symbols 0 and 1 as

$$-\log\frac{a+1/2}{a+b+1}, \quad -\log\frac{b+1/2}{a+b+1}, \tag{2}$$

respectively. In this way, one can evaluate the code length for the next symbol given the recent substring γ and its next-symbol counts a and b.

We are interested in the ''total cost'' of coding all the symbols appearing after the substring γ in a symbolic sequence. Denote by a_γ and b_γ the number of symbols 0 and 1 appearing after a substring γ in the whole symbolic sequence. Let

$$P_e(a_\gamma,b_\gamma)=\frac{\frac{1}{2}\frac{3}{2}\cdot\cdots\cdot\left(a_\gamma-\frac{1}{2}\right)\frac{1}{2}\frac{3}{2}\cdot\cdots\cdot\left(b_\gamma-\frac{1}{2}\right)}{1\,2\cdot\cdots\cdot(a_\gamma+b_\gamma)}. \tag{3}$$

We define $P_e(0,0)=1$ for convenience. Then the code length for describing all the next symbols appearing after context γ in the symbolic sequence is given by $-\log P_e(a_\gamma,b_\gamma)$. This gives the code length for node γ when one does not use the knowledge about its children.

(*c*) *Code length when using knowledge about children.* When one uses the knowledge about its children, the code length for each node is defined using more complicated formulae. The point is that now we may be able to encode more cheaply because there is additional information available.

First, we review how the counts at a node and knowledge about its children are ''merged'' in the context-tree weighting method [15].

At each node γ, a weighted probability P_w^γ is defined as

$$P_w^\gamma=\begin{cases}\frac{1}{2}P_e(a_\gamma,b_\gamma)+\frac{1}{2}P_w^{0\gamma}P_w^{\epsilon\gamma}P_w^{1\gamma} & \text{if } \gamma \text{ is internal,}\\ P_e(a_\gamma,b_\gamma) & \text{otherwise.}\end{cases} \tag{4}$$

Although it looks slightly different, this definition is equivalent to one in Ref. [15]. Let λ be an empty substring corresponding to the root of a tree. Using this weighted probability, we define the coding distribution $P_c(x_1^t)$ for a symbolic sequence x_1^t as

$$P_c(x_1^t)=P_w^\lambda(x_1^t|\epsilon), \tag{5}$$

for all $x_1^t\in\{0,1\}^t, t=0,1,\ldots$. The code length $L(x_1^t)$ for sequence x_1^t is upper bounded by this $P_c(x_1^t)$ [14]:

$$L(x_1^t)\leqslant -\log P_c(x_1^t)+2. \tag{6}$$

In this way, Willems [15] assigned the probability for x_1^t and used it for compressing a symbolic sequence with arithmetic coding.

Our aim is to find the *most likely symbolic mechanism* given a symbolic sequence. Therefore, instead of using arithmetic coding, we rather focus on the code length for the sequence and select the best model in terms of code length by finding the context tree which is that subtree of the given context tree which minimizes the code length.

Modifying Eq. (4) slightly gives a code length when coding using a subtree. Willems [15] proved that

$$-\log P_w^\gamma\leqslant 1+\min\left\{-\log P_e(a_\gamma,b_\gamma),\sum_{i\in\{0,\epsilon,1\}}-\log P_w^{i\gamma}\right\}. \tag{7}$$

This suggests that we define $\tilde{P}_w^\gamma$ so that

$$-\log\tilde{P}_w^\gamma=\begin{cases}1+\min\left\{-\log P_e^\gamma,\sum_{i\in\{0,\epsilon,1\}}-\log\tilde{P}_w^{i\gamma}\right\} & \text{if } \gamma \text{ is internal,}\\ -\log P_e^\gamma & \text{otherwise.}\end{cases} \tag{8}$$

The first term 1 of $1+\min\{-\log P_e^\gamma,\Sigma_{i\in\{0,\epsilon,1\}}-\log\tilde{P}_w^{i\gamma}\}$ can be regarded as the cost of describing the topology of the tree, or the cost for selecting a certain context tree. (That is, it is the cost of the binary decision ''prune'' or ''do not prune.'') Thus the quantity $-\log\tilde{P}_w^\gamma$ can be interpreted as the code length for node γ when one uses the knowledge about its children. By combining the counts with the children's information in this way, the code length $-\log\tilde{P}_w^\gamma$ for node γ using the knowledge about its children is defined.

(*d*) *Pruning by comparing two code lengths.* Now we are ready to describe the pruning. In principle, at each node we prune its children if coding using just its counts gives the shorter code length than coding using the knowledge of its children. But when we prune the context tree, we have to decide the priority of the nodes: which node should be tested for pruning first.

We choose to prune from the bottom. Specifically, we search the tree depth first, and at each node γ we prune the children of γ if $-\log P_e^\gamma\leqslant-\log\tilde{P}_w^\gamma$, where coding using just the counts works better than coding using the knowledge about the children.

3. Markovizing

Pruning makes a context tree simpler. However, it may be still hard to use it for purposes such as estimating topological

entropy because sometimes it is not Markov. It has been stated elsewhere that a tree is Markov if it contains all its subtrees in itself [18]. This is a necessary condition because every past context contains the recent symbols necessary for reconstructing the current context. It is also sufficient when one uses a complete context tree, for which each internal node has every child.

However, we would like to use an incomplete context tree, as illustrated earlier: that is, we do not create children if they have zero counts. An incomplete tree is smaller than the corresponding complete tree, but in this case the subtree condition is not sufficient. It is possible that some contexts, which are required by other contexts to make the tree Markov, may correspond to leaves of the complete tree which have been omitted in the incomplete tree because they did not occur in the original symbolic sequence. We can solve this situation by inserting these missing future contexts into the context tree.

Applying the following algorithm changes the pruned context tree into a Markov tree.

(1) Construct a context tree containing every prefix of each context in the pruned context tree.

(2) For each leaf C of the tree, assign the conditional probability of the next symbols 0 and 1 by $a_C/(a_C+b_C)$ and $b_C/(a_C+b_C)$, respectively.

(3) End the algorithm if for each context all the admissible next symbols give the corresponding next contexts to transit among the leaves of the tree.

(4) Extend the context tree so that it contains the missing contexts. Go back to step (2).

Step (1) ensures that every context has all the possible immediately proceeding contexts among the leaves of the context tree. Steps (2), (3), and (4) ensure that each context has all the appropriate contexts immediately following among the leaves of the tree. By checking the Markov property for the both time directions, we change the pruned context tree into Markov.

This algorithm will finish within a finite time because we add 2^D nodes at most, where D is the maximum depth of the pruned tree. Typically, most trees we worked were Markovized after step (1), and around 20 loops of steps (2)–(4) were enough for every tree we tested. We show an example of the algorithm in Fig. 3. Let us suppose that we obtained a context tree given in Fig. 3(a). Assume that at each node there can appear both 0 and 1 as the next symbols. First we add nodes 0101 and 1101 so that we have all the prefixes for nodes 01011 and 11011 in the context tree. Then, we obtain the tree shown in Fig. 3(b). But node 00 does not have a destination when the next symbol is 1. Therefore we add a node 001 into the tree [Fig. 3(c)]. However, the node 001 does not have a destination when the next symbol is 1. Hence we insert a node 0011 [Fig. 3(d)], completing the Markovization.

For each context C, we define the conditional probability of the next symbols 0 and 1 simply by $a_C/(a_C+b_C)$ and $b_C/(a_C+b_C)$, respectively.

Now we have a Markov chain on the contexts, helping find the topological entropy.

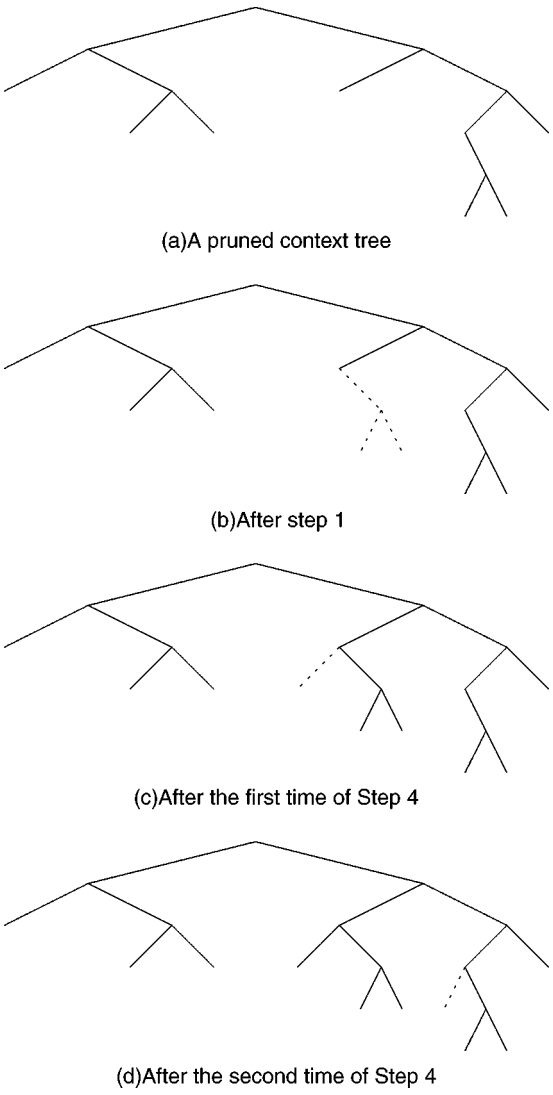

FIG. 3. An example of Markovizing a pruned context tree. Edges going down with moving to the left (the right) correspond to symbol 0 (1).

C. Finding topological entropy for a Markov model

After obtaining a Markov model, one can easily find its topological entropy. For states i, j of a Markov model, let P_{ij} be a conditional probability that given the current state i, the next state is j. Then we define an adjacency matrix T for the Markov model in the following way:

$$T_{ij}=\begin{cases}1 & \text{if } P_{ij}>0,\\ 0 & \text{otherwise.}\end{cases} \tag{9}$$

Let κ be an eigenvalue of T with the largest absolute value. It can be shown that κ is always positive, and that the topological entropy of this Markov model is given by $\log\kappa$ ([17], p. 120).

D. Summary

In short, we construct a context tree for estimating topological entropy by building a big context tree by following

Willems [15], pruning it to have the best context tree in terms of code length, and Markovizing it by extending the pruned context tree.

The leaves of the Markovized context tree form a Markov chain. Finding the largest eigenvalue of the adjacency matrix of the Markov chain and taking its logarithm gives an estimate for the topological entropy.

IV. EXAMPLES

In Sec. III, we explained how to estimate the topological entropy using the context-tree weighting method. To show the efficacy of the proposed method, we present in this section some examples: a Markov model, the logistic map, and the Hénon map.

A. Another method for estimating topological entropy from time series

In the following examples, we compare our method with previous work: counting the number of the distinct admissible substrings of finite lengths [3].

We briefly summarize the method described in Ref. [3]. 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

$$h(n)=\log N(n)-\log N(n-1). \tag{10}$$

However, it was reported [3] that this estimate would oscillate. To overcome this, we averaged the estimates for several lengths and obtained another estimate $\overline{h}(n)$:

$$\overline{h}(n)=\frac{1}{n-m}\sum_{i=m+1}^{n} h(i)=\frac{1}{n-m}\log\frac{N(n)}{N(m)}, \tag{11}$$

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

One wants to choose the substring length n long to obtain a good estimate. But n has to be short enough so that one can observe all the possible admissible substrings of length n. But there is no prescription for choosing n. Therefore, given the length l of a symbolic sequence, we chose n to be $\lfloor \log l/\log|X| \rfloor$ arbitrarily and estimated $N(n)$ and $N(m)$ by counting substrings appearing in a given symbolic sequence.

B. Markov model

For the first example, we try to find the topological entropy of the following map of $[0,1]$ into itself:

$$x_{t+1}=\begin{cases}\dfrac{(1-a)(x_t-a)}{a}+1 & x_t\in[0,a)\\[2ex] \dfrac{x_t-a}{a-1}+1 & x_t\in[a,1].\end{cases} \tag{12}$$

Here we take $a=0.7$.

It is known that when we divide $[0,1]$ at the critical point a into two intervals $[0,a)$ and $[a,1]$, we have one-to-one correspondence between a point in $[0,1]$ and an infinite symbolic sequence to make the symbolization a topological conjugacy.

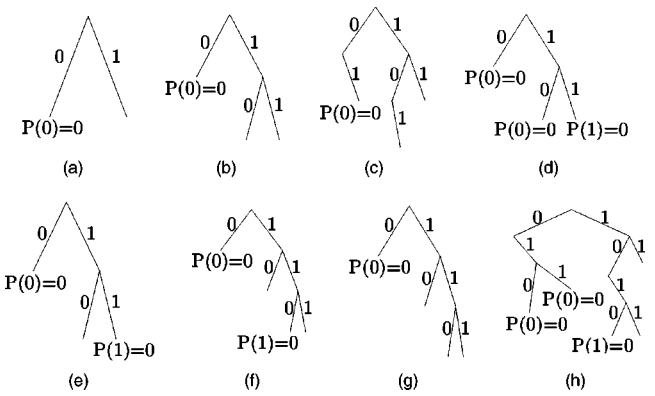

FIG. 4. Classification of Markovized pruned context trees obtained from different symbolic sequences in the numerical experiments for the Markov model. At each context of the context trees, we show symbols which do not appear using probabilities if some of them exist. Trees (a), (b), and (c), which appeared dominantly, achieved the correct topological entropy, 0.6942. Tree (d), with the topological entropy 0, appeared when the symbolic sequences were periodic. Trees (e), (f), (g), and (h) have 0.4057, 0.5972, 0.5514, 0.5514 as their topological entropy, respectively. See Table I for the numbers of occurrences for each tree.

This map is equivalent to a Markov chain with two states 0 and 1 such that $1-P(0|0)=P(1|0)=1$ and $1-P(0|1)=P(1|1)=1-a$. Therefore, we obtain the topological entropy by finding the maximum eigenvalue of the following adjacency matrix:

$$\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}. \tag{13}$$

Finding its maximum eigenvalue and taking the logarithm, we have 0.6942 as the theoretical topological entropy of this system. It should be remarked that this Markov model does not produce a substring 00, which is forbidden.

TABLE I. The numbers of Markovized pruned context trees observed in the numerical simulation of a Markov model. Tree types from (a) to (h) correspond to ones shown in Fig. 4. Trees (a), (b), and (c) achieved the correct topological entropy. Tree (d) corresponds to the periodic cases. We see that from any aperiodic symbolic sequences of length more than 150, we obtained the correct topological entropy.

	Tree type							
Length	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
50	86	6	2	2	0	1	1	1
100	93	3	0	2	1	0	0	0
150	97	0	0	2	0	0	0	0
200	94	3	0	2	0	0	0	0
250	97	0	0	2	0	0	0	0
300	96	1	0	2	0	0	0	0

TABLE II. Maximum, minimum, mean, and standard deviation of the topological entropy estimated from 97 aperiodic symbolic sequences using the proposed method, the Markov model.

Length	Maximum	Minimum	Mean	Standard deviation
50	0.6942	0.5514	0.6913	0.0203
100	0.6942	0.4057	0.6912	0.0291
150	0.6942	0.6942	0.6942	0.0000
200	0.6942	0.6942	0.6942	0.0000
250	0.6942	0.6942	0.6942	0.0000
300	0.6942	0.6942	0.6942	0.0000

In the following way, we generated a set of symbolic sequences of length n. We first prepared 99 initial points 0.01,0.02, . . . , and 0.99. For each initial point, we applied the map, generating $(n+1000)$ points. Abandoning the first 1000 points, we converted the rest into a symbolic sequence of 0 and 1 using the partition mentioned above. We observed that sequences obtained from 0.7 and 0.79 were of period 3.

We tested the proposed method with symbolic sequences of lengths 50, 100, 150, 200, 250, and 300. We focused on their topological structures and classified them depending on the varieties of included contexts, and their possible transitions. The classification was shown in Fig. 4. Table I shows the numbers of context trees which appeared using the tested symbolic sequences. The estimates for each length are summarized in Table II.

When the sequences were periodic, we obtained a context tree shown in Fig. 4(d) with the topological entropy 0 for every length investigated here.

For the lengths $n=150$, 200, 250, and 300, all the tested aperiodic sequences achieved the correct topological entropy 0.6942. The numbers of tested aperiodic sequences not achieving the correct topological entropy were three for length 50, and one for length 100. Clearly longer symbolic sequences improve the chance of obtaining the correct topological entropy.

We found, in this numerical experiment, three types of trees achieving the correct topological entropy as shown in Fig. 4. Although these three models look different, they are equivalent to the original model in the sense that they forbid a substring 00. Hence we say that we selected a *topologically* correct model from the aperiodic sequences even with length 150.

TABLE III. Maximum, minimum, mean, and standard deviation of the topological entropy estimated from 97 aperiodic symbolic sequences using Ref. [3], the Markov model.

Length	Maximum	Minimum	Mean	Standard deviation
50	0.5283	0.5283	0.5283	0
100	0.6160	0.6160	0.6160	0
150	0.5946	0.5946	0.5946	0
200	0.5946	0.5946	0.5946	0
250	0.5946	0.5946	0.5946	0
300	0.5111	0.5111	0.5111	0

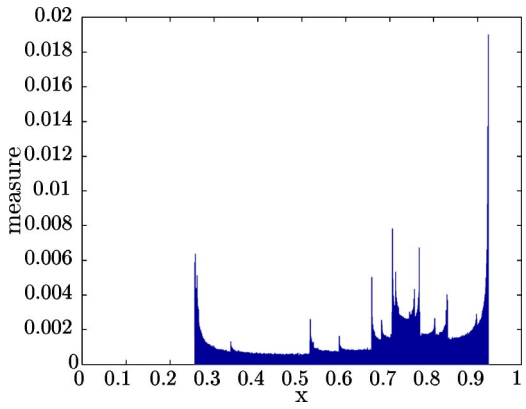

FIG. 5. A measure observed from the logistic map. We generated 1 001 000 points from an initial point 0.1. After throwing away the first 1000 points, we split $[0,1]$ into 1000 bins of the equal size and built the histogram using the remaining 1 000 000 points.

Lastly, we compare the performance of the proposed method with that of Ref. [3]. The estimates obtained using Ref. [3] are listed in Table III. The values 0 for the standard deviations showed that our selection of n and m was appropriate. The estimates occurred widely. The comparison of the estimates using the proposed method with those obtained using Ref. [3] shows that the proposed method works effectively.

If the system is Markov, the proposed method seems to provide the theoretically correct topological entropy, given only a short aperiodic symbolic sequence.

C. Logistic map

The previous example is a Markov model, which is expressed exactly using a context tree of finite depth. However, in general, a context tree of a finite depth may not describe the original dynamics completely as one may need an infinite depth. The next map, the logistic map, illustrates this.

The logistic map is defined as

$$x_{t+1}=rx_t(1-x_t). \tag{14}$$

Here we only focus on the parameter $r=3.7$, which is the same as one used in Ref. [3].

We calculated the theoretical value of the topological entropy under this parameter using Ref. [6]. Utilizing the first 20 symbols of the kneading sequence, we found that the topological entropy is 0.550 701, which is within 10^{-6} of the correct value.

Shown in Fig. 5 is a measure exhibited under the parameter $r=3.7$. As the orbit looks dense in the invariant set, or the interval approximately equal to $[0.2567,0.9250]$, the dynamics seems topologically transitive on the invariant set. We took a partition consisting of $[0,0.5)$ and $[0.5,1]$ to divide the invariant set, assigning symbols 0 and 1, respectively.

We tested the efficacy of the proposed method using the logistic map with $r=3.7$. Suppose that one wants to have symbolic sequences of length n. We first had initial points 0.01,0.02, . . . , and 0.99, mapping them $(1000+n)$ times,

TABLE IV. Maximum, minimum, mean, and standard deviation of the topological entropy estimated from 99 symbolic sequences using the proposed method, the logistic map.

Length	Maximum	Minimum	Mean	Standard deviation
100	0.6942	0.4205	0.5772	6.997×10^{-2}
200	0.5973	0.4856	0.5494	1.081×10^{-2}
1000	0.5515	0.5365	0.5512	2.117×10^{-3}
10 000	0.5515	0.5501	0.5511	3.406×10^{-4}
100 000	0.5507	0.5502	0.5507	7.146×10^{-5}
1 000 000	0.5507	0.5507	0.5507	3.386×10^{-6}

TABLE V. Maximum, minimum, mean, and standard deviation of the numbers of contexts of context trees obtained from symbolic sequences, the logistic map.

Length	Maximum	Minimum	Mean	Standard deviation
100	15	2	5.2323	2.2894
200	21	4	7.1313	1.9041
1000	22	6	8.3636	2.5252
10 000	135	8	51.8586	22.7699
100 000	219	118	153.6667	24.1433
1 000 000	505	310	411.4141	38.9885

respectively. For each initial point, we threw away the first 1000 points, changing the remaining n points into a symbolic sequence of 0 and 1 using a partition containing $[0,0.5)$ and $[0.5,1]$ as mentioned above. We observed that none of the symbolic sequences were periodic.

We list in Table IV the maximum, the minimum, the mean, and the standard deviation of our estimated topological entropy for each length. We also show in Fig. 6 the histograms of the estimated topological entropy for each length.

We have several interesting observations. The first observation is that the estimated topological entropy approached the theoretical value of the topological entropy when the length of symbolic sequence got longer. But the convergence was not monotonic. Figure 7 shows how the topological entropy estimated from a sequence changed according to the increase of the length. Although it approached the theoretical value, the error sometimes got bigger than that of the previous values.

The second observation is that the estimated topological entropy was sometimes overestimated, sometimes underestimated. For each length we tested, the maximum was always over the theoretical value.

The third observation is that the mean value tended to be more than the theoretical value. It means that we tend to overestimate the topological entropy. If one has some different time series, it may be possible to obtain an upper bound for the topological entropy.

The fourth observation is that the numbers of contexts were relatively small. The numbers of contexts were listed in Table V. Even with symbolic sequences of length 1 000 000, the maximum number of the contexts was 505. We postpone discussing the number of contexts until the next example, as we do not have the previous works to compare.

The fifth observation is that the convergence was fast. See histograms shown in Fig. 6. For length 1000, we found a strong peak at a point slightly bigger than the theoretically correct value. But we also had a small peak at 0.5365, which made the standard deviation big. When we changed the length to 10 000, 100 000, and 1 000 000, we saw that the peak of the estimated topological entropy became sharp at 0.5507, the theoretical value.

We compare the convergence of the estimates obtained using the proposed method with that of Ref. [3], which were listed in Table VI. As the standard deviations were small, we assert that the lengths of substrings were appropriate. Now we can check the validity of the lengths of substrings because there are several symbolic sequences for each length. But we remark that when there is only a symbolic sequence, we do not have any way to check the validity of the lengths of substrings when using Ref. [3]. The convergence of both the methods are compared in Fig. 8. The estimates obtained

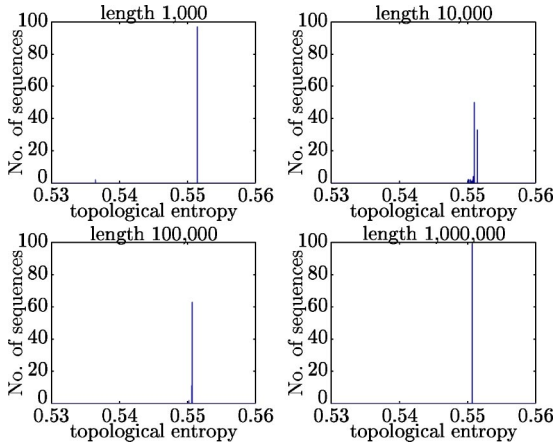

FIG. 6. Histograms of the estimated topological entropy using symbolic sequences of lengths 1000 (top left), 10 000 (top right), 100 000 (bottom left), and 1 000 000 (bottom right), the logistic map.

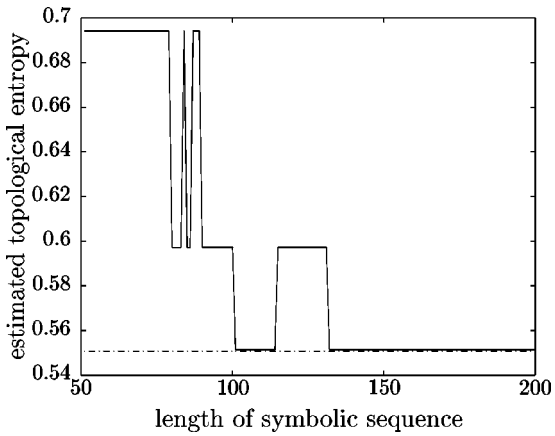

FIG. 7. Change of the estimated topological entropy along the length of a symbolic sequence generated from an initial point 0.1, the logistic map. Observe that the estimated topological entropy did not converge to the theoretical value monotonically.

TABLE VI. Maximum, minimum, mean, and standard deviation of the topological entropy estimated from 99 symbolic sequences using Ref. [3], the logistic map.

Length	Maximum	Minimum	Mean	Standard deviation
100	0.5283	0.4595	0.5221	1.988×10^{-2}
200	0.5504	0.5344	0.5480	5.778×10^{-2}
1000	0.5615	0.5615	0.5615	0
10 000	0.5581	0.5581	0.5581	0
100 000	0.5541	0.5541	0.5541	0
1 000 000	0.5498	0.5498	0.5498	0

using Ref. [3] varied greatly, although those of the proposed method got closer to the theoretical value. Judging from the convergence, the proposed method is better than the method in Ref. [3].

We also discuss the convergence quantitatively. The change of standard deviation with length is shown in Fig. 9. Linear fitting to a log-log plot gave the slope -0.9597, which is close to 1.

To examine how fast the convergence was from a different viewpoint, we compared the standard deviation of the estimated topological entropy with that of the metric entropy. Let P_i be the stationary distribution for context i. Then the metric entropy of a Markov model is given by

$$\sum_{i,j} -P_i P_{ij} \log P_{ij} . \tag{15}$$

We estimated P_i using the sum of the counts on context i divided by the sum of all the counts on all the contexts.

Table VII shows the maximum, the minimum, the mean, and the standard deviation of the estimated metric entropy. The mean of the estimates for the length 1 000 000 agreeing with the Lyapunov exponent suggests that Pesin's identity ([2], p. 198) holds.

The standard deviations of the topological entropy obtained from various lengths of symbolic sequences are compared with those of the metric entropy in Fig. 9. Linear fitting in the log-log plot gave the slope for the metric entropy

TABLE VII. Maximum, minimum, mean, and standard deviation of the metric entropy estimated from symbolic sequences of lengths 100, 200, 1000, 10 000, 100 000, and 1 000 000, the logistic map. The Lyapunov exponent is 0.5121 ± 0.0001 (obtained by iterating 10 000 000 times for each initial point and averaging over 999 initial points), which agrees with the mean estimate obtained with symbolic sequences of length 1 000 000.

Length	Maximum	Minimum	Mean	Standard deviation
100	0.6666	0.3258	0.524	8.161×10^{-2}
200	0.5748	0.3663	0.5077	3.485×10^{-2}
1000	0.5493	0.4847	0.5221	1.108×10^{-2}
10 000	0.5333	0.5044	0.5185	6.117×10^{-3}
100 000	0.5157	0.5093	0.5127	1.243×10^{-3}
1 000 000	0.5131	0.5109	0.5121	4.009×10^{-4}

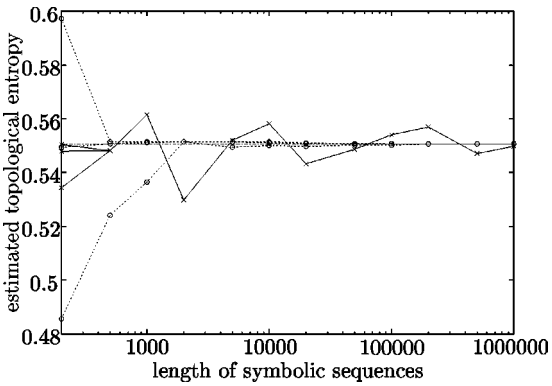

FIG. 8. The comparison in convergence of the estimates obtained from symbolic sequences generated from the logistic map between the proposed method and a method in Ref. [3]. The dotted lines show, from the top, the maximum, the mean, and the minimum of the estimates obtained using the proposed method. The solid lines show, from the top, the maximum, the mean, and the minimum of the estimates obtained using Ref. [3]. The broken line shows the theoretical value obtained using Ref. [6]. This graph indicates that the proposed method converges to the theoretical value faster than that in Ref. [3].

-0.5334. It showed that the topological entropy converged faster than the metric entropy. This is quite natural because, although the topological entropy can be obtained by finding only a structure that forbids appropriate substrings, for the metric entropy one must estimate not only the topological structure but also the conditional probability of the next outcome. Figure 9 provides support for the fact that topological entropy is often preferred to metric entropy for analyzing observed time series.

This example of the logistic map shows that we can use the proposed method for systems whose corresponding symbolic sequences are not Markov chains with a finite number of states.

D. Hénon map

The previous two examples are one-dimensional maps. But our method is not restricted to one-dimensional maps if we have a well defined partition which changes a time series into a symbolic sequence. Here we present an analysis of the Hénon map.

The Hénon map is defined as

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \end{pmatrix} = \begin{pmatrix} 1 - a x_t^2 + b y_t \\ x_t \end{pmatrix}.$$

We used a set of standard parameters $a=1.4$ and $b=0.3$.

The topological entropy for the Hénon map under these parameters has been estimated using different techniques. Biham and Wenzel [5] estimated 0.6708 ± 0.0003 (in base 2) using the numbers of periodic points. D'Alessandro *et al.* [8], using a graph, gave an estimate, which is, according to Ref. [13], close to 0.4651 in the natural base (0.6710 in base 2). Jacobs *et al.* [12] estimated $0.464\,93\pm0.000\,03$ in the natural base, $0.670\,75\pm0.000\,04$ in base 2, using the *stretch-*

TABLE VIII. Comparison of the numbers of periodic points for the Hénon map between the proposed method and the theoretical one found in Auerbach *et al.* [4] and Biham and Wenzel [5]. For each length, we used a symbolic sequence generated using an initial point (0.1,0.1).

	Length				
Period	1000	10 000	100 000	1 000 000	Theoretical
1	1	1	1	1	1
2	3	3	3	3	3
3	1	1	1	1	1
4	7	7	7	7	4
5	1	1	1	1	1
6	21	15	15	15	15
7	29	29	29	29	29
8	63	63	63	63	63
9	73	64	55	55	55
10	123	103	103	103	103
11	155	155	155	155	155
12	289	247	247	247	247
13	443	443	417	417	417
14	787	675	647	647	647
15	1231	1141	1081	1081	1081
16	2079	1743	1711	1695	1695
17	3180	2891	2806	2823	2823
18	5151	4416	4245	4263	4263
19	8057	7183	6936	6917	6917
20	13167	11287	10827	10807	10807
21	20987	18509	17564	17564	17543
22	34235	28955	27151	27107	27107
23	54764	46668	44322	44368	44391
24	88521	73863	69903	69927	69951
25	141351	118701	112426	112476	112451
26	227555	188321	177349	177349	177375
27	364303	301438	284203	284041	284041
28	586775	480375	449967	449463	449519

ing factor. The lower bound is given in Ref. [9] as 0.670 (in base 2). The upper bound is given in Ref. [13] as 0.4687 (in the natural base), which is equivalent to 0.6762 in base 2.

There is a proposed method for constructing a generating partition of the Hénon map by connecting the *primary tangencies* [24]. One can find points to connect in Ref. [13]. Assuming that this partition is also a topological conjugacy, we employed this partition to obtain, from a time series, a symbolic sequence.

We tested our method for symbolic sequences of lengths 1000, 10 000, 100 000, and 1 000 000. For each length, we tried to prepare 121 time series of initial points $(x_0,y_0)=(0.1u,0.1v)$ for $u,v=0,1,\ldots,10$. However, it turned out that initial points (0,1) and (1,1) were not appropriate because they escaped from the attractor. Therefore, we only used time series of the remaining initial points.

We generated each symbolic sequence in the following way: Suppose that one wants to generate a symbolic sequence of length n. From each initial point, we applied the map $(1000+n)$ times. After abandoning the first 1000 points, we converted the remaining n points into symbols of 0 and 1 using the partition found in Ref. [13].

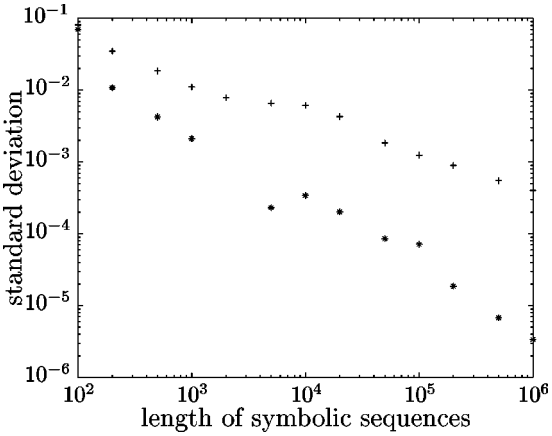

FIG. 9. The standard deviations of the estimated topological entropy (*) and that of the metric entropy (+), the logistic map. We obtained both the values for symbolic sequences of lengths 1000, 2000, 5000, 10 000, 20 000, 50 000, 100 000, 200 000, 500 000, and 1 000 000 except for that of the topological entropy for length 2000, where all its sequences happened to give the same estimates. Linear fitting gave the slopes in the log-log plot -0.9597 for the topological entropy, and -0.5334 for the metric entropy, respectively.

First, we compared the numbers of periodic points for showing how well the models are estimated using the context-tree weighting method. Here a periodic point with a period p means a point x satisfying $f^p(x)=x$ for map f. This x may be a periodic point with period of p's factor. For a Markov model with its adjacency matrix A, the number of periodic points with period p is given by the trace of A^p ([17], p. 38).

We used four symbolic sequences for lengths 1000, 10 000, 100 000, and 1 000 000 generated from an initial point (0.1,0.1) and found the numbers of periodic points using each of them. The numbers of periodic points estimated for each length are listed in Table VIII. For comparison, we also list, in the same table, the numbers of periodic points obtained by Auerbach *et al.* [4] for periods up to 10, and Biham and Wenzel [5] for periods from 11 to 28, which have been confirmed in Ref. [7]. Table VIII suggests that if we have a longer time series, the estimated number of periodic points becomes closer to the true number. In particular, when we used a symbolic sequence of length 1 000 000, we counted the number of periodic points up to period 20 correctly, up to period 27 within error of an orbit.

TABLE IX. Maximum, minimum, mean, and standard deviation of the topological entropy estimated from symbolic sequences, the Hénon map.

Length	Maximum	Minimum	Mean	Standard deviation
1000	0.7208	0.6373	0.6866	1.4140×10^{-2}
10 000	0.6794	0.6689	0.6753	2.145×10^{-3}
100 000	0.6711	0.6700	0.6707	1.870×10^{-4}
1 000 000	0.6708	0.6706	0.6707	3.562×10^{-5}

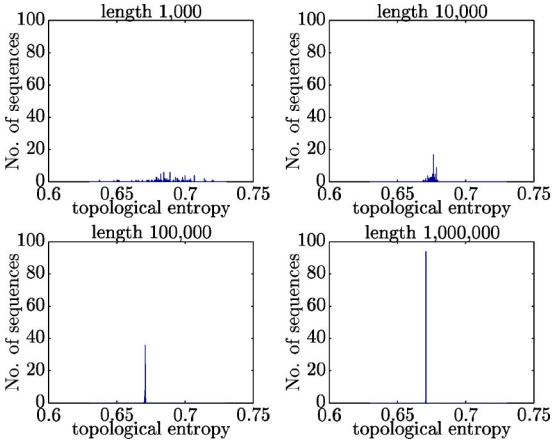

FIG. 10. Histograms of the estimated topological entropy using symbolic sequences of lengths 1000 (top left), 10 000 (top right), 100 000 (bottom left), and 1 000 000 (bottom right), the Hénon map.

It shows that modeling using the context-tree weighting method may be used to give an initial rough guess for the numbers of periodic points of unknown systems.

We list the maximum, the minimum, the mean, and the standard deviation of the topological entropy estimated from symbolic sequences of each length in Table IX.

The estimated topological entropy seems to converge to the values proposed by the previous works. In particular, all the values obtained using symbolic sequences of length 1 000 000 were within an estimate of the interval 0.6708 $\pm$0.0003 suggested in Ref. [5], and the mean of the estimates was within the interval 0.670 75$\pm$0.000 04 suggested in Ref. [12]. These show that, given a time series of sufficient length, the proposed method gives the topological entropy

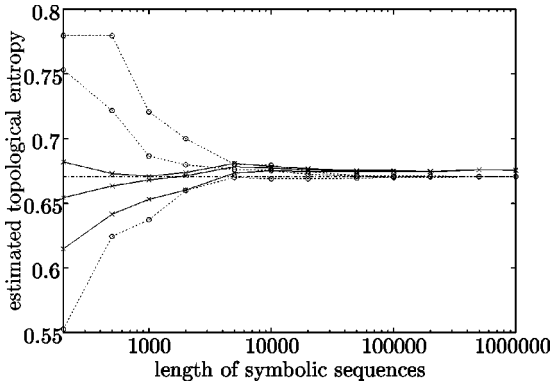

FIG. 11. The comparison in convergence of the estimates obtained from symbolic sequences generated from the Hénon map between the proposed method and a method in Ref. [3]. The dotted lines show, from the top, the maximum, the mean, and the minimum of the estimates obtained using the proposed method. The solid lines show, from the top, the maximum, the mean, and the minimum of the estimates obtained using Ref. [3]. The broken line shows the theoretical value obtained using Ref. [6]. This graph indicates that the proposed method converges to the theoretical value faster than that in Ref. [3].

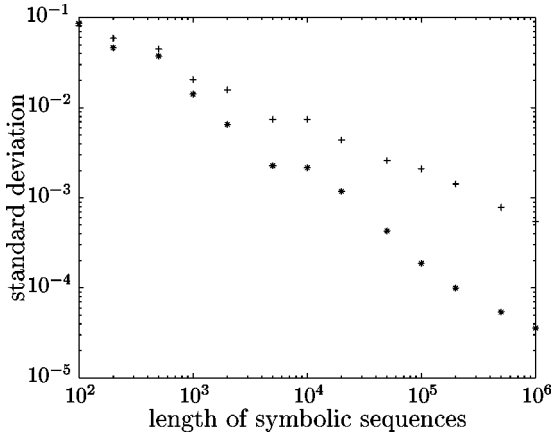

FIG. 12. The standard deviation of the estimated topological entropy (*) and that of the metric entropy (+), the Hénon map. We obtained both the values for symbolic sequences of lengths 1000, 2000, 5000, 10 000, 20 000, 50 000, 100 000, 200 000, 500 000, and 1 000 000. Linear fitting gave the slopes in the log-log plot -0.8893 for the topological entropy, and -0.5472 for the metric entropy, respectively.

agreeing with the theoretical value. Length 10 000 was enough to have at least a two-digit accuracy.

The mean of the estimates was always more than the theoretical value, as we found with the logistic map.

Now we would like to discuss the speed of the convergence. Histograms in Fig. 10, showed the speed of the convergence more clearly than the case of the logistic map. For length 1000, the estimated topological entropy was spread widely. Even the count of the highest peak was 6. When making the length longer, we observed that the peak was getting sharper. For length 1 000 000, we had 94 out of 119 estimates at a single bin with the width 10^{-4}.

Next we evaluated the speed of the convergence quantitatively. We plotted changes of the standard deviation according to the length of symbolic sequences in Fig. 12. Linear fitting gave the slope in log-log plot -0.8893. Using the values from length 10 000 to length 500 000, we found the slope -0.9802.

We do not have evidence enough to say anything on the value of the slope. It may be always -1 or it may depend on the number of symbols. The value of the slope here is an open problem.

To show the efficacy more clearly, we used also Ref. [3] for estimating topological entropy.

TABLE X. Maximum, minimum, mean, and standard deviation of the topological entropy estimated from symbolic sequences using Ref. [3], the Hénon map.

Length	Maximum	Minimum	Mean	Standard deviation
1000	0.6710	0.6531	0.6680	3.795×10^{-3}
10 000	0.6786	0.6753	0.6773	6.557×10^{-4}
100 000	0.6753	0.6749	0.6752	9.394×10^{-5}
1 000 000	0.6757	0.6756	0.6757	6.301×10^{-6}

TABLE XI. Maximum, minimum, mean, and standard deviation of the metric entropy estimated from symbolic sequences of lengths 1 000, 10 000, 100 000, and 1 000 000, the Hénon map. The mean value obtained for length 1 000 000 is close to 0.6048 ± 0.0002, the Lyapunov exponent obtained by Grassberger and Kantz [24], implying that Pesin's identity ([2], p. 198) holds under the standard parameter set.

Length	Maximum	Minimum	Mean	Standard deviation
1000	0.6817	0.5681	0.6339	2.049×10^{-2}
10 000	0.6388	0.6022	0.6218	7.433×10^{-3}
100 000	0.6134	0.6043	0.6084	2.093×10^{-3}
1 000 000	0.6067	0.6038	0.6054	5.458×10^{-4}

The result when using Ref. [3] is summarized in Table X. In this case, the lengths of substrings were slightly long. If the lengths of substrings were correctly chosen, we should have seen all the possible admissible substrings in a symbolic sequence and obtained the same numbers of substrings for each symbolic sequence, that should have made the standard deviations 0. The convergence of the estimates are compared in Fig. 11. The figure shows that the estimates obtained by the proposed method converged to the literature value faster than that in Ref. [3]. The comparison suggests that the proposed method gives the better estimates at most times.

The metric entropy was estimated as shown in Table XI. The mean value obtained for length 1 000 000 is close to 0.6048, the Lyapunov exponent obtained in Ref. [24]. Therefore, Pesin's identity seems to hold under the standard parameter set. We compare the standard deviation of the estimated topological entropy with that of the estimated metric entropy in Fig. 12. Linear fitting in the log-log plot gave the slope for the metric entropy -0.5472. Hence, we also saw in the Hénon map that the topological entropy converged faster than the metric entropy.

The numbers of contexts were listed in Table XII. Still these numbers were reasonably small enough to handle. For example, it took 10.5 sec using the same computer and MATLAB to find the maximum eigenvalue of the adjacency matrix of 1903×1903 for obtaining the topological entropy of 1 000 000 points data generated from (0.1,0.1), which was mentioned above.

The proposed algorithm did not take much time to construct a context tree. The codes were written in C++, calculated on a computer with CPU AMD-K6 350 MHz and 384 M bytes memory. We used 10 000 and 1 000 000 points data generated from (0.1,0.1) for the evaluation. For 10 000 points data, we had a context tree with 115 contexts and the maximum depth 14. To obtain the context tree for evaluating topological entropy took 0.94 sec. It took 0.09 sec for Markovizing the tree, where the algorithm did not enter the loop of steps (2)–(4). For 1 000 000 points data, we obtained a context tree with 1903 contexts and maximum depth 36. It took 2 min 18 sec for obtaining the context tree for evaluating topological entropy, and 15 sec for Markovizing the tree, doing the loop once.

TABLE XII. Maximum, minimum, mean, and standard deviation of the numbers of contexts of context trees obtained from symbolic sequences using the proposed method, the Hénon map.

Length	Maximum	Minimum	Mean	Standard deviation
1000	86	18	37.5	9.6
10 000	238	72	115.3	35.2
100 000	852	440	613.31	73.36
1 000 000	2348	1613	1888.8	127.8

However, the size of the tree may be slightly bigger than is found using Markov models, where the map is assumed to be known and one can generate desired orbits. For the same parameter set of the Hénon map, D'Alessandro *et al.* [8] have 676 nodes as the maximum. Froyland *et al.* [13] have 1162 nodes as the maximum. There is a trade-off between the accuracy and the size of the graph. Unfortunately, the previous works did not mention any accuracy, so we cannot further compare our method with the previous works. If the partition given in Ref. [13] is a topological conjugacy, we assert that $0.670\,72 \pm 0.000\,04$, which is obtained with 119 symbolic sequences of length 1 000 000 shown in Table IX, is one of the most precise estimates of the topological entropy of the Hénon map under the set of the standard parameters.

V. CONCLUSION

We have proposed a method for estimating the topological entropy given a time series of a finite length and its good partition. One can apply the method when the original dynamics is topologically transitive and the time series is not periodic. After converting the time series into a symbolic sequence, the method employs a technique called the context-tree weighting method [14,15] to model its dynamics and find the most reasonable Markov model in the sense of code length. By constructing the corresponding adjacency matrix and finding its maximum eigenvalue, one obtains an estimate of the topological entropy. The biggest advantage of our method is that one does not have to tune *ad hoc* parameters, which often appear in other methods.

We demonstrated the performance of the proposed method with a Markov model, the logistic map, and the Hénon map. In these models, whether they are Markov or not, we found that one can achieve the theoretical value if the time series is sufficiently long. When we have several symbolic sequences, the mean of the estimates tends to converge to the correct value from above. The convergence of the estimated topological entropy is fast: in our examples, its standard deviation almost varied inversely as the length of symbolic sequences and its decay rate was nearly twice that of the estimated metric entropy.

ACKNOWLEDGEMENT

We thank Devin Kilminster for checking the draft.

[1] A. Katok and B. Hasselbratt, *Introduction to the Modern Theory of Dynamical Systems* (Cambridge University Press, Cambridge, UK, 1999).
[2] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis* (Cambridge University Press, Cambridge, UK, 1997).
[3] J. Crutchfield and N. Packard, Int. J. Theor. Phys. **21**, 433 (1982).
[4] D. Auerbach, P. Cvitanović, J.-P. Eckmann, G. Gunaratne, and I. Procaccia, Phys. Rev. Lett. **58**, 2387 (1987).
[5] O. Biham and W. Wenzel, Phys. Rev. Lett. **63**, 819 (1989).
[6] L. Block, J. Keesling, S. Li, and K. Peterson, J. Stat. Phys. **55**, 929 (1989).
[7] P. Grassberger, H. Kantz, and U. Moenig, J. Phys. A **22**, 5217 (1989).
[8] G. D'Alessandro, P. Grassberger, S. Isola, and A. Politi, J. Phys. A **23**, 5285 (1990).
[9] Q. Chen, E. Ott, and L.P. Hurd, Phys. Lett. A **156**, 48 (1991).
[10] P. Góra and A.I. Boyarsky, Trans. Am. Math. Soc. **323**, 39 (1991).
[11] N. Balmforth, E.A. Spiegel, and C. Tresser, Phys. Rev. Lett. **72**, 80 (1994).
[12] J. Jacobs, E. Ott, and B.R. Hunt, Phys. Rev. E **57**, 6577 (1998).
[13] G. Froyland, O. Junge, and G. Ochs, Physica D **154**, 68 (2001).
[14] F.M.J. Willems, Y.M. Shtarkov, and T.J. Tjalkens, IEEE Trans. Inf. Theory **41**, 653 (1995).
[15] F.M.J. Willems, IEEE Trans. Inf. Theory **44**, 792 (1998).
[16] R. Bowen, Trans. Am. Math. Soc. **153**, 401 (1971).
[17] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding* (Cambridge University Press, Cambridge, UK, 1995).
[18] A.I. Mees, in *Nonlinear and Nonstationary Signal Processing*, edited by W.J. Fitzgerald, R.L. Smith, A.T. Walden, and P. Young (Cambridge University Press, Cambridge, UK, 2000), pp. 184–216.
[19] M.B. Kennel and A.I. Mees, Phys. Rev. E **61**, 2563 (2000).
[20] M.B. Kennel and A.I. Mees, in *Nonlinear Dynamics and Statistics*, edited by A. I. Mees (Birkhäuser, Boston, 2001), pp. 387–412.
[21] M.B. Kennel and A.I. Mees, Phys. Rev. E **66**, 056209 (2002).
[22] T.M. Cover and J.A. Thomas, *Elements of Information Theory* (Wiley, New York, 1990).
[23] K. Judd and A. Mees, Physica D **120**, 273 (1998).
[24] P. Grassberger and H. Kantz, Phys. Lett. **113A**, 235 (1985).