Hindawi
Complexity
Volume 2020, Article ID 7163254, 9 pages
https://doi.org/10.1155/2020/7163254

Research Article

Calculating Permutation Entropy without Permutations

Alexander K. Vidybida

Bogolyubov Institute for Theoretical Physics, 14-b Metrolohichna Str, Kyiv 03143, Ukraine

Correspondence should be addressed to Alexander K. Vidybida; vidybida@bitp.kiev.ua

Received 5 May 2020; Revised 14 July 2020; Accepted 5 August 2020; Published 23 October 2020

Academic Editor: Eric Campos

A method for analyzing sequential data sets, similar to the permutation entropy one, is discussed. The characteristic features of this method are as follows: it preserves information about equal values, if any, in the embedding vectors; it is exempt from combinatorics; and it delivers the same entropy value as does the permutation method, provided the embedding vectors do not have equal components. In the latter case, this method can be used instead of the permutation one. If embedding vectors have equal components, this method could be more precise in discriminating between similar data sets.

1. Introduction

Because of technical progress in the areas of sensors and storage devices, a huge amount of raw data about time course of different processes, such as ECG, EEG, climate data recordings, and stock market data, have become available. These data are redundant. The data processing and classification, aimed at extracting meaningful for nonspecialist characteristics, is based on reducing the excess of redundancy. As a result, new data are obtained, small in size and digestible by a human being. Examples of those reduced data for time series can be mean value, variance, Lyapunov exponents, correlation dimension, and attractor dimension.

A remarkable method suitable for reducing the excess of redundancy in time series has been proposed by Bandt and Pompe in [1], known as permutation entropy. This method is simple and transparent, is robust with respect to monotonic distortions of the raw data, and is suitable for estimating the dynamical complexity of the underlying dynamical process. Many interesting results, e.g. [2–5], have been obtained with straightforward application of the permutation entropy methodology in its initial form, as it is described in [1]. Nevertheless, this method is subjected to a critique for not taking into account absolute values of the raw data and for not treating properly a possibility of having equal values in the embedding vector (ties), [6, 7]. In this connection, it should be taken into account that any redundancy reduction method leaves out some types of information, which may be useless for one process/task and may carry useful information for another one. In the latter case, the bare idea in [1] about how to treat equal values can/should be modified in order to meet a purpose of concrete situation. Examples of such a modification can be found in [8, 9] for taking into account absolute values or in [10, 11] for treating equal values. Interesting modification of the permutation entropy method has been proposed in [12] for 3-tuple EEG data.

In the standard permutation entropy methodology, it is preferable that embedding vectors have all their components different. Otherwise, they cannot be plainly symbolized by a permutation without using additional rules, which actually treat equal values as not being such. Situation with equal values in the embedding vector may arise for high embedding dimension, for crude quantization of measured data, for very long data sequences, and when observed dynamical system has intrinsically only a small number of possible outputs.

This note is aimed at discussing a slightly different symbolization technique of embedding vectors, which does not refer to combinatorics and which is capable of preserving information about equal values in embedding vectors. Instead of permutation, an embedding vector is

emblematized with a single integer number of base D, where D is the embedding dimension. In the case of no ties (no equal components in the embedding vectors), the technique is equivalent to the standard permutation entropy methodology. In the opposite case, it may discriminate between similar data sets better than the permutation entropy method does.

2. Permutation Entropy

Consider a finite sequence

$$X = (x_0, x_1, \ldots, x_{N-1}), \quad x_i \in \mathbb{R}^1,\ i = 0, 1, 2, \ldots, N-1, \tag{1}$$

of measurements. By choosing the embedding dimension $D < N$, the data (1) can be embedded into a D-dimensional space by picking out consecutive D-tuples from X. As a result, a set of D-dimensional embedding vectors are obtained:

$$\mathbf{V} = \{V_0, V_1, \ldots, V_{N-D}\}, \quad V_i \in \mathbb{R}^D,\ i = 0, 1, 2, \ldots, N-D, \tag{2}$$

where each vector has the following form:

$$\begin{aligned} V_0 &= (x_0, x_1, x_2 \ldots, x_{D-1}), \ldots, \\ V_i &= (x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+D-1}), \ldots, \\ V_{N-D} &= (x_{N-D}, x_{N+1-D}, \ldots, x_{N-1}). \end{aligned} \tag{3}$$

An additional parameter of the embedding procedure is delay $\tau = 1, 2, \ldots$. In the above definition, we put $\tau = 1$ for simplicity. With $\tau \neq 1$ one would have $V_i = (x_i, x_{i+\tau}, x_{i+2\tau}, \ldots, x_{i+(D-1)\tau})$ instead of (3).

The data represented in (2) and/or (3) are even more redundant than those represented in (1) since for $D \ll N$, most data values from (1) are represented in (3) D times. In the permutation entropy technique [1], each embedding vector from (2) and/or (3) is replaced with a permutation π of D integers {0, 1, 2, …, $D-1$}, which is defined by the order pattern of values composing the vector. For any embedding vector $V = (x_0, x_1, \ldots, x_{D-1})$, the permutation π, which symbolizes it, is calculated as follows. Arrange all components of V either in the descending [13]:

$$\begin{aligned} V &= (x_0, x_1, \ldots, x_{D-1}) \longrightarrow V_\pi \\ &= \left(x_{r_0}, x_{r_1}, \ldots, x_{r_{D-1}}\right), x_{r_0} > x_{r_1} > \cdots > x_{r_{D-1}}, \end{aligned} \tag{4}$$

or in the ascending [11,14]:

$$\begin{aligned} V &= (x_0, x_1, \ldots, x_{D-1}) \longrightarrow V_\pi \\ &= \left(x_{r_0}, x_{r_1}, \ldots, x_{r_{D-1}}\right), \quad x_{r_0} < x_{r_1} < \cdots < x_{r_{D-1}}, \end{aligned} \tag{5}$$

order keeping their subscripts unchanged (actually, in (4) and (5), equal values (ties) are as well admitted. Here, we exclude such a possibility for the sake of clarity. The equal values are discussed in the next section.) The permutation π which corresponds to V is obtained as the row of the subscripts in the rearranged vector V_π from either (4) or (5):

$$\pi \equiv \pi(V) = (r_0, r_1, \ldots, r_{D-1}). \tag{6}$$

From the set of embedding vectors $\mathbf{V}$, calculate a new set Π of order patterns by replacing each vector in (2) by the corresponding permutation:

$$\Pi = \{\pi_0, \pi_1, \ldots, \pi_{N-D}\}. \tag{7}$$

Now, empirical probability of each permutation, $p(\pi_i)$, can be obtained by dividing the number of occurrences of π_i in Π by the total number of elements in Π. The permutation entropy of $\mathbf{V}$ is the Shannon entropy of the probability distribution $p(\pi_i)$:

$$H(\mathbf{V}) \equiv H(\Pi) = -\sum_{i=0}^{K-1} p(\pi_i) \log\left(p(\pi_i)\right), \tag{8}$$

where K is the number of different permutations in Π.

2.1. Treatment of Equal Values. Equal values in an embedding vector are, to an extent, inconvenient. Indeed, if $x_r = x_s$ for some $0 \le r, s < D$ in a vector $V = (x_0, x_1, \ldots, x_{D-1})$, then r and s should be placed side by side in the permutation (6), but which one should go first? Due to the sameness of values, it is impossible to uniquely determine a corresponding permutation without introducing additional rules. In some cases, the possibility of equal values can be ignored due to their low probability. This is reasonable when the embedding dimension is low and/or a chaotic process data are recorded with high precision [1, 15, 16]. If equal values are inevitable, the following rule is applied (in some cases, e.g. [10, 11], the opposite inequality sign is used here):

$$\textbf{if} \quad x_s = x_r \ \textbf{and}\ s > r \ \textbf{then}\ s \text{ goes first.} \tag{9}$$

The rule (9) has different meaning depending on whether (4) or (5) convention is used. Namely, in the case of (4), an embedding vector with all components equal will be equivalent to a vector with monotonically ascending components. If (5) is adopted, then that same vector will be equivalent to a vector with monotonically descending components (Figure 1).

Without knowing a real system, it is not clear which case is better and whether it is good or bad to label a sequence of same values as being decreasing or increasing. Actually, the permutation symbolization technique aims at reducing redundancy. Discrimination between constant and either increasing or decreasing sequences of data may appear to be excessive in some cases. On the contrary, when a system, which generates data, has a few possible outputs, the data were subjected to a crude quantization, or embedding dimension is large, it may happen to be useful if the presence of equal values in the embedding vector results in the order pattern preserving this fact. One possible approach to do this is discussed in the next section.

3. Arithmetic Entropy

3.1. Symbolization. The following symbolization is aimed to keep information about equal values in embedding vectors.

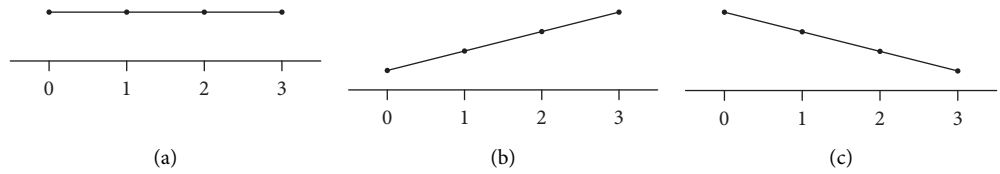

Figure 1: In the standard permutation entropy symbolization, a sequence of same values (a) is equivalent either to ascending (b) or descending (c) sequence, if either (4) or (5) convention is used.

Having a vector $V = (x_0, x_1, \ldots, x_{D-1})$, construct a sequence of integers α:

$$V = (x_0, x_1, \ldots, x_{D-1}) \longrightarrow \alpha = \alpha(V) = (a_0, a_1, \ldots, a_{D-1}), \tag{10}$$

by using the following rule: find the smallest component, c_0, in V. If c_0 is found at places $r_1, r_2, \ldots$, put number 0 at those places in $\alpha(V)$. Find the next smallest component c_1, $c_1 > c_0$ in V. If c_1 is found at places $s_1, s_2, \ldots$, put number 1 at those places in α. Proceed this way until components of V are exhausted. At this stage, all D components of α will be determined. α obtained this way is used as a symbol of embedding vector V.

For example, consider $V = (7, 15, 7, 25, 15)$. The corresponding symbol, or the order pattern, is $\alpha = (0, 1, 0, 2, 1)$. Here, information about equal values and their positions is preserved.

If V has no equal components, it can be proven (Appendix A) that $\alpha = \pi^{-1}$. This means that α is the inverse permutation of the one obtained for V if convention (5) is used. Since correspondence between permutations and their inverse is one-to-one, it does not matter which one, π or α, is used for calculating entropy. This further means that for a data set and embedding method, which does not deliver equal values in the embedding vectors, symbolization used here is equivalent to the permutation one (it seems that in paper [5], the symbolization method described here is used. But, as it may be concluded from [5] (equation (6)), the issue of equal values is not addressed. Similar approach is used in [12, 17], again without considering equal values.) while calculating entropy.

3.2. Arithmetization. Expect that embedding vector V in (10) has exactly d unique components, where $d \le D$. In this case, the corresponding symbol $\alpha(V)$ will be a sequence of D numbers chosen from the set $\mathbf{d} = \{0, 1, \ldots, d-1\}$ in such a way that not any element from $\mathbf{d}$ is missed. The latter can be formulated as the following condition:

$$\bigwedge_{b \in \mathbf{d}} b \in \alpha(V). \tag{11}$$

The sequence $\alpha(V)$ can be considered as a single integer $A(V)$, in a base-D positional numeral system with digits $a_{D-1}a_{D-2}\ldots a_0$:

$$A(V) \equiv A = a_0 + a_1 D + a_2 D^2 + \cdots + a_{D-1} D^{D-1}. \tag{12}$$

(For a single embedding vector, d might be chosen as radix instead of D. But d may be different for different vectors. And a same integer may have different representation for different bases with (11) satisfied, e.g., $0112_3 = 1110_2$.). It is clear that there is one-to-one correspondence between order patterns α and integers obtained as shown in (12). Therefore, a set of order patterns, constructed as described in Section 3.1, can be replaced with a set $\mathscr{A}$ of integers obtained as shown in (12):

$$\mathscr{A} = \{A_0, A_1, \ldots, A_{N-D}\}, \quad \text{where } A_i \equiv A(V_i). \tag{13}$$

The empirical probabilities $p(A_i)$ to find an integer A_i among those in $\mathscr{A}$ can be calculated as usual, and we have for the arithmetic entropy:

$$H_a(\mathbf{V}) \equiv H_a(\mathscr{A}) = -\sum_{i=0}^{L-1} p(A_i) \log\left(p(A_i)\right), \tag{14}$$

where L is the number of different integers in $\mathscr{A}$.

For a data sequence and embedding method which does not deliver equal values in the embedding vectors, all $d_i = D$ and the integers A_i will represent corresponding permutation order patterns unambiguously. In this case, $A_{\min} \le A_i \le A_{\max}$, where $A_{\min}$ corresponds to pattern $\alpha_{\min} = (D-1, D-2, \ldots, 1, 0)$:

$$A_{\min} = D - 1 + (D-2)D + (D-3)D^2 + \cdots + D^{D-2}. \tag{15}$$

And $A_{\max}$ corresponds to pattern $\alpha_{\max} = (0, 1, \ldots, D-1)$:

$$A_{\max} = D + 2D^2 + 3D^3 + \cdots + (D-1)D^{D-1}. \tag{16}$$

In this case, only $D!$ integers will be used from $[A_{\min}; A_{\max}]$ due to condition (11).

3.3. How Many New Possible Order Patterns Are Got? If it is decided to treat the order patterns generated from embedding D-vectors with some components equal as not equivalent to those from vectors with all components different, then the number of all possible patterns will be greater than $D!$. Here we attempt to estimate how many new patterns can be obtained.

Any new pattern appears from embedding D-vector with d different components, where $d \in \{1, 2, \ldots, D-1\}$. So, having d fixed, the number of corresponding new patterns is equal to the number $N(D, d)$ of base-D D-digit integers constructed from digits $\{0, 1, \ldots, d-1\}$ in such a way that each of the d digits is used at least once. This number can be calculated as

$$N(D,d) = d!\left\{ \begin{matrix} D \\ d \end{matrix} \right\}, \tag{17}$$

where $\left\{ \begin{matrix} D \\ d \end{matrix} \right\}$ is the Stirling numbers of the second kind ([18], Part 5, Section 2). Considering all possible values for d, we have for the total number of possible new patterns:

$$N(D) = \sum_{0<d<D} d!\left\{ \begin{matrix} D \\ d \end{matrix} \right\} = b(D) - D!, \tag{18}$$

where $b(D)$ are known as the ordered Bell numbers, see ([19], p. 337) for naming discussion. Calculating (the Stirling numbers were calculated with stirling2(D, d) function in the "maxima" computer algebra system (http://maxima.sourceforge.net/)) $N(D)$ for $D \in \{2,3,4,5,6,7\}$, we see that the number of new patterns is normally greater than $D!$, see Figure 2 and also Table 1. Of course, the possible new patterns may only be significant when they can be observed (see discussion about this in [7]). This depends on the process under study and embedding method.

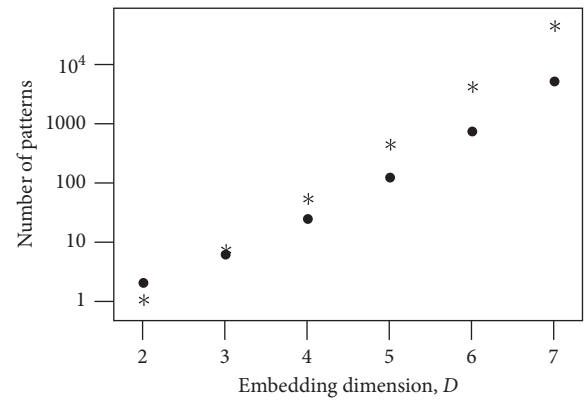

Figure 2: Comparison of the possible number of patterns. (•): equal values are treated as described in Section 2.1, $D!$; (∗): additional possible patterns due to equal values, $N(D)$ graph, equation (18).

Table 1: Total number of possible patterns in the AE symbolization.

D	2	3	4	5	6	7
$N(D) + D!$	3	13	75	541	4683	47293

3.4. Coding. Certainly, there are several possible implementations of the algorithm discussed in Sections 3.1 and 3.2. Here, the one used for the examples in Section 4, and Appendix C, is shown. It is a C++ program. It is expected that the sequence (1) is organized into a one-dimensional array `X[N]`. For calculating the arithmetic order pattern of vector V_i shown in (3), it is necessary to pass a pointer to `X[i]` to the function `get_numerical_pattern`, below, as its third argument: `data_point = X + i`.

In the below example, `X[i]` is declared as `double`, but it can be of any type with appropriate sorting defined. The returning value is declared as `mpz_class`, which is a GNU multiple precision integer (https://gmplib.org/). This is used because for embedding dimensions $D > 15$, the returned number representing an order pattern may exceed 64 bits in size (it makes sense to use large embedding dimensions only for very long sequences of data. Otherwise, any observed pattern appears only once, which is unfavorable for estimating probabilities). For smaller D, `mpz_class` can be replaced with `int` or `long` everywhere in the code.

```
(1) #include <gmp.h>
(2) #include <gmpxx.h>
(3) #include <forward_list>
(4)
(5) /**
(6)   Function calculates numerical representation of order pattern
(7)   of an embedding vector V_i = {(x_i, x_{i + tau},...).
(8)   Here D is the embedding dimension and tau is the delay.
(9)   The data_point points to the first component of V_i in the
(10)  array of raw data.
(11) */
(12) mpz_class get_numerical_pattern (int D, int tau, double * data_point)
(13) {
(14)   int k;
(15)   std:forward_list<double> FL;
(16)   auto it = FL.before_begin();
(17)   for (k = 0; k < D; k++) it = FL.emplace_after (it, data_point [k*tau]);
(18)   FL.sort ();
(19)   FL.unique ();
(20)
(21)   int * pDpnm = new int [D]; //order pattern will be here
(22)   int tag = 0;
(23)   for (auto it = FL.begin (); it != FL.end (); ++it)
(24)     {
(25)       for (k = 0; k < D; k++)
(26)         if  (*it == data_point  [k*tau]) pDpnm [k] = tag;
(27)       tag++;
(28)     }
(29)
(30)   mpz_class pnum = 0; // arithmetic order pattern (initial value)
(31)   mpz_class digval = 1; // initial value of a single digit
```

```
(32)  for (k = 0; k < D; k++)
(33)    {
(34)      pnum += pDpnm [k]*digval;
(35)      digval *= D;
(36)    }
(37)  return pnum;
(38) }
```

This code is transparent and does not refer to combinatorics. At the same time, provided an embedding vector does not have equal components, when loop at lines 23–28 above is complete, we obtain in the array pDpnm [D] a permutation π^{-1}, where π is the permutation for that vector obtained in accordance with the standard rules of [1] reproduced in Section 2 with (5) adopted.

4. Example

The discussed methodology has been tested at two surrogate sequences. The purpose was to demonstrate that for a pair of sequences, the standard permutation entropy method gives roughly the same entropy, whereas the arithmetic entropy may be considerably different.

For calculating standard permutation entropy in situation when equal components in embedding vectors are possible, we replace the following fragment:

```
(i)  for (auto it = FL.begin (); it != FL.end ();
     ++it)
(ii)   {
(iii)    for (k = 0; k < D; k++)
(iv)       if (*it == data_point [k*tau]) pDpnm
     [k] = tag;
(v)      tag++;
(vi)   },
```

in the code of Section 3.4, with the following one:

```
(i)  for (auto it = FL.begin (); it != FL.end ();
     ++it)
(ii)   {
(iii)    for (k = D−1; k >= 0; k--)
(iv)       if (*it == data_point [k*tau]) pDpnm
     [k] = tag++;
(v)    }
```

With such a replacement, we get in the array pDpnm [D] above the permutation, which is inverse to the one obtained for V_i in the standard permutation entropy symbolization with rules (5) and (9) adopted. As it was mentioned above, usage of inverse permutations instead of the initial ones delivers the same value for the standard permutation entropy.

The two sequences, S1 and S2, are obtained as follows: by means of function gsl_rng_uniform_int from the GNU Scientific Library [20], we generate random numbers from the set $\{0, 1, \ldots, 4\}$, which are equally probable. Each obtained random number "val" is written into the S1. The same number is written into S2 provided it is not equal to the number written to S2 at the previous step. If it does, then the number (val + 1) (**mod** 5) is written instead. This introduces a nonzero correlation between consecutive values in S2. For example, in S2, any two consecutive values are always different. Examples of S1 and S2 are as follows:

$$\begin{aligned} \mathrm{S1} &= (2, 2, 0, 3, 4, 0, 1, 3, 4, 4, 0, 3, 3, 2, 2, 4, 4, 2, 0, 1, \ldots), \\ \mathrm{S2} &= (2, 3, 0, 3, 4, 0, 1, 3, 4, 0, 1, 3, 4, 2, 3, 4, 0, 2, 0, 1, \ldots). \end{aligned} \tag{19}$$

1 000 000 long S1 and S2 were produced and both permutation and arithmetic entropy have been calculated. The results are shown in Tables 2 and 3.

Notice that arithmetic entropy is considerably greater than the permutation one. This is due to the high frequency of embedding vectors with equal components. Also, from Table 2 with $\tau = 1$, it can be seen that arithmetic entropy discriminates better between S1 and S2. However, the case with delay $\tau = 2$ shown in Table 3 is not similarly conclusive. This might be due to the construction method of the S2 sequence. Namely, by pulling from S2 embedding vectors with delay 2, we may get vectors with equal adjacent components, similarly to the S1 case. This alleviates difference between S1 and S2. For $\tau = 1$, embedding vectors for S2 do not have equal adjacent components. One more example is in Appendix C.

5. Conclusions and Discussion

In this note, we have discussed a method for calculating entropy in a sequence of data, which is similar to the permutation entropy method. The characteristic features of this method are as follows:

(i) It treats equal components in the embedding vectors as being equal instead of ordering them artificially

(ii) It is entirely exempt from combinatorics, labeling order patterns by integers instead of permutations

(iii) If embedding vectors do not have equal components, this method delivers exactly the same value for the entropy as does the standard permutation entropy one

In the symbolization procedure discussed in Section 3.1, new order patterns may appear as compared to the standard permutation method (Section 3.3). Those new patterns arise from embedding vectors with some components being equal to each other. In the standard permutation entropy method, the embedding vectors characterized by those new patterns, if any, are labeled by permutations as if there were no equal components. This is made possible through ordering equal values in accordance with the rule (9).

Mathematically, replacing embedding vectors with their order patterns means constructing a quotient set from the set of all embedding vectors with respect to some equivalence relation [10, 21, 22]. In the case of permutation entropy, the corresponding equivalence relation is defined by using (9) and either (4) or (5). Denote it by $\sim_P$. For arithmetic

Table 2: Comparison of permutation entropy (PE) and arithmetic entropy (AE) for embedding delay $\tau = 1$.

$D = 3$	PE	AE	$D = 4$	PE	AE
S1	2.497	3.684	S1	4.390	6.165
S2	2.368	2.919	S2	4.187	5.238

Entropy is given in bits.

Table 3: Same as Table 2 for embedding delay $\tau = 2$.

$D = 3$	PE	AE	$D = 4$	PE	AE
S1	2.498	3.684	S1	4.393	6.166
S2	2.407	3.676	S2	4.224	6.098

entropy, the corresponding equivalence relation is defined by using the algorithm described in the first paragraph of Section 3.1. Denote it by $\sim_A$. It is clear that for two embedding vectors U and V, if $U \sim_A V$, then $U \sim_P V$. Namely, if U and V have the same arithmetic order pattern, then they do have the same permutation order pattern. That means that $\sim_P$ is coarser relation than $\sim_A$. Other equivalence relations could be offered, which are courser than $\sim_P$, finer than $\sim_A$, lying in between, or incomparable with the both, see [12]. A symbolization which still uses permutations, but is equivalent to the one discussed here, as regards the treatment of equal values in embedding vectors, has been proposed in [11], see discussion in Appendix B. Which one is better depends on the data sequence and which kind of redundancy is intended to strip.

Appendix

A. Equivalence with Permutations

The following theorem proves the statement made in Section 3.1.

Theorem A.1. *Suppose that an embedding vector $V = (x_0, x_1, \ldots, x_{D-1})$ does not have equal components. Then its symbolic pattern $\alpha(V)$, obtained as described in Section 3.1 after equation (10), represents permutation which is inverse to the $\pi(V)$—the permutation obtained in the standard permutation entropy approach with convention (5) adopted:*

$$\alpha = \pi^{-1}. \tag{A.1}$$

Proof. Since V has no equal components, then $\alpha(V) = (a_0, a_1, \ldots, a_{D-1})$ represents some permutations of sequence $(0, 1, \ldots, D-1)$. Furthermore, the procedure of obtaining $\alpha(V)$ from V does not change the rank order: for any $0 \le i, j < D$, if $x_i < x_j$, then $a_i < a_j$, and vice verse. If so, then $\alpha(V)$ can be used for calculating standard permutation $\pi(V)$:

$$\pi(V) = \pi(\alpha(V)). \tag{A.2}$$

In this course, after arranging elements of α as required in (5), one obtains

$$(a_0, a_1, \ldots, a_{D-1}) \longrightarrow \left(a_{r_0}, a_{r_1}, \ldots, a_{r_{D-1}}\right) = (0, 1, \ldots, D-1). \tag{A.3}$$

Obtained permutation $\pi = (r_0, r_1, \ldots, r_{D-1})$ acts as follows:

$$i \longrightarrow r_i, \quad i = 0, 1, \ldots, D-1. \tag{A.4}$$

Now, take into account that $\alpha(V)$ has number i at position r_i (A.3). That means that order pattern $\alpha(V)$, if treated as a permutation, acts as follows:

$$i \longleftarrow r_i, \quad i = 0, 1, \ldots, D-1. \tag{A.5}$$

The latter just means that $\alpha(V) = (\pi(V))^{-1}$.

Due to this theorem, the method discussed in this note is equivalent to the standard permutation entropy method if in any embedding vector, any two components are different. □

B. Comparison with Modified Permutation Entropy

Several versions of the modified permutation entropy symbolization have been proposed. We analyze here those proposed in [10, 11]. Considering firstly [10], the symbolization proposed is obtained as follows: having an embedding vector $V = (x_0, x_1, \ldots, x_{D-1})$ arrange its components as shown in (5), with their subscripts retained, if there are equal components, arrange their subscripts similar to (9) rule, or any other way. Before fetching the row of subscripts in the resulting vector $V_\pi = (x_{r_0}, x_{r_1}, \ldots, x_{r_{D-1}})$ as the modified symbol, do the following preparation. If there is a group of equal components $x_{s_0} = x_{s_1} = \cdots = x_{s_l}$ in V_π, then replace all subscripts in this group by the smallest among $\{s_0, s_1, \ldots, s_l\}$. Do this with all groups of equal components in V_π. Use the row of subscripts in the such way modified V_π as the modified symbol of V. This way modified symbolization retains some information about equal components in V. Let us denote this type of symbolization as MPE and corresponding symbol as $\mu(V)$.

By comparing the values presented in Table 1 with the data of TABLE 1 in [10], we see that the total number of possible patterns is bigger in Table 1. Therefore, it could be expected that MPE symbolization used in [10] is coarser than that discussed in this note. An additional hint in the same direction is that for some embedding vectors, symbolization in [10] gives the same result, while the method discussed in this note gives two different results. Here is one example: $V_1 = (3, 5, 3, 5)$, $V_2 = (3, 5, 5, 3)$. MPE symbolization in [10] gives both for V_1 and V_2 the same order pattern $(0, 0, 1, 1)$, whereas AE symbolization gives $(0, 1, 0, 1)$ for V_1 and $(0, 1, 1, 0)$ for V_2 resulting in two different numerical patterns, 68 and 20 calculated for $D = 4$ as shown in (12). Notice now that for any two embedding vectors V_1 and V_2,

$$\textbf{if } \alpha(V_1) = \alpha(V_2), \textbf{ then } \mu(V_1) = \mu(V_2). \tag{B.1}$$

Indeed, the MPE symbol of any vector V is obtained through rearranging components of V in accordance with their rank order. Symbol $\alpha(V)$, if considered as a vector, has the same rank of its components as does V. Therefore, $\alpha(V)$ can be used for calculating $\mu(V)$ instead of V itself. If so, then (B.1) becomes evident. The above reasoning proves that MPE and AE methods of symbolization are comparable, and AE is finer than MPE.

Consider now the symbolization used for modified permutation entropy proposed in [11]. In this symbolization, each embedding vector V is symbolized with $\sigma(V)$. $\sigma(V)$ has the following structure:

$$\sigma(V) = (\pi(V), \mathbf{e}(V)), \tag{B.2}$$

where $\pi(V)$ is a permutation. The second half in (B.2), $\mathbf{e}(V) = (e_1, e_2, \ldots, e_{D-1})$, keeps information about equal components in V. Call this symbolization MPE2. The symbol $\sigma(V)$ is obtained as follows: arrange components in $V = (x_0, x_1, \ldots, x_{D-1})$ in the ascending order keeping their subscripts. As a result, we obtain a sequence of groups consisting of equal components. Each group may have from one to D elements. Of course, in the latter case, there will be only one group. The value composing each group in the sequence increases from left to right. Arrange subscripts in each group in the ascending order. Denote this way prepared sequence of components with their initial subscripts as $\widetilde{V} = (v_{r_0}, v_{r_1}, \ldots, v_{r_{D-1}})$. The row of subscripts in $\widetilde{V}$ is $\pi(V)$ from (B.2). This is the standard PE symbol with (5) adopted with the only difference that in the rule (9), the opposite inequality sign is used. The sequence $\mathbf{e}(V)$ is composed of $D-1$ zeros and ones by the following rule: **if** $v_{r_{i-1}} = v_{r_i}$, **then** $e_i = 1$; **otherwise** $e_i = 0$.

Theorem B.2. *Symbolization MPE2 produces the same partition of a set of embedding vectors as does the AE one described in Section 3.1.*

Proof. In order to prove this statement, we need to show that for any V_1 and V_2, the following equivalence holds:

$$\alpha(V_1) = \alpha(V_2) \Leftrightarrow \sigma(V_1) = \sigma(V_2). \tag{B.3}$$

It is easily seen that $\sigma(V)$ can be unambiguously recovered from $\alpha(V)$. Indeed, V and $\alpha(V)$ considered as a vector have the same rank order of components. And calculation of $\sigma(V)$ is based exclusively on the rank order. Therefore,

$$\sigma(V) = \sigma(\alpha(V)). \tag{B.4}$$

Thus, vectors with same α will have same σ. This proves the one half of (B.3). In order to prove the second half, we need to show how $\alpha(V) = (a_0, a_1, \ldots, a_{D-1})$ can be unambiguously recovered from $\sigma(V)$. For this purpose, we use the equality (B.4). So, if we arrange $\alpha(V)$ in the ascending order retaining subscripts, we obtain, instead of $\widetilde{V}$, above, a vector $\widetilde{\alpha} = (\widetilde{a}_{r_0}, \widetilde{a}_{r_1}, \ldots, \widetilde{a}_{r_{D-1}})$. This vector consists of groups of equal values: the first group has only zeros, the second one has only "1," and the last one has only "$d-1$," where d is the number of unique components in V or $\alpha(V)$. The sequence of subscripts in $\widetilde{\alpha}$ is the permutation, which constitutes the first part in $\sigma(V)$. If one would have $\widetilde{\alpha}$ without subscripts inherited from $\alpha(V)$, in the form of $\beta = (b_0, b_1, \ldots, b_{D-1})$, the required α might be obtained by applying permutation $\pi(V)$ to β. Namely, for $i = 0, 1, \ldots, D-1$, $a_{r_i} = b_i$, where r_i is taken from the permutation $\pi(V)$: $r_i = \pi_i$. The required sequence β can be recovered from the second part of σ. For this purpose, do the following reprocessing of $\mathbf{e}(V)$. Replace $\mathbf{e}(V)$ with the following sequence: $(0, e_1, e_2, \ldots, e_{D-1})$. With the obtained new sequence, proceed as follows: at the step number one, if $e_1 = 1$, replace it with 0, otherwise replace it with 1. Similarly, at the step number i, if $e_i = 1$, replace it with the number put at the previous step in place of e_{i-1}, otherwise, replace e_i with that same number incremented by 1. After replacing e_{D-1}, we obtain the required sequence β. This completes the proof. □

C. Example

Here, we consider the sequence of digits in decimal expansion of $\sqrt{2}$ (see ([6], Section 4.1)). The first 1 million digits in the decimal expansion of $\sqrt{2}$ have been downloaded from https://catonmat.net/tools/generate-sqrt2-digits and https://apod.nasa.gov/htmltest/gifcity/sqrt2.1mil.

Denote this sequence S1. The first 10 millions digits in the decimal expansion of $\sqrt{2}$ has been downloaded from https://apod.nasa.gov/htmltest/gifcity/sqrt2.10mil.

Denote this sequence S10. Both PE and AE were calculated for both S1 and S10 for different embedding dimensions D with delay $\tau = 1$. The D values were chosen based on the number of occurrences of different order patterns in S1 (Table 4). Based on the data of Table 4, we skip $D = 6$ and $D = 7$ cases because the number of occurrences of some arithmetic entropy patterns is too small for calculating probabilities. The values obtained for entropy are presented in Tables 5 and 6.

The data, which are obtained numerically, can be checked analytically. Indeed, the number $\sqrt{2}$ is believed to be base 10 normal [23]. This means that any combination of n digits can be found in the expansion with probability 10^{-n}. For example, if $n = 2$, there are 10 combinations $\{(0,0), (1,1), \ldots, (9,9)\}$ with AE pattern $\alpha_0 = (0,0)$, 45 combinations $\{(0,1), (0,2), \ldots, (0,9), (1,2), \ldots, (1,9), (2,3), \ldots, (8,9)\}$ with AE pattern $\alpha_1 = (0,1)$, and the same amount with AE pattern $\alpha_2 = (1,0)$. This gives for the probabilities $p(\alpha_0) = 0.1$, $p(\alpha_1) = 0.45$, and $p(\alpha_2) = 0.45$. And $\mathrm{AE}_2 = -0.1\log_2(0.1) - 2 \cdot 0.45\log_2(0.45) = 1.369$. In the PE symbolization, both α_0 and α_1 correspond to $\pi_1^{-1} = (0,1)$ and α_2 corresponds to $\pi_2^{-1} = (1,0)$ (we use here the rule (9) with inverse inequality sign). This gives $\mathrm{PE}_2 = -0.55\log_2(0.55) - 0.45\log_2(0.45) = 0.993$.

From Tables 5 and 6, we see that AE is usually bigger than PE. This could be explained by the bigger total number of patterns available in the AE symbolization. Perhaps, for the same reason, normalized AE is smaller than NPE for small D. What seems unexpected, it is the opposite behavior of NPE and NAE with growing D. Namely, NAE is increasing and NPE is decreasing function of D for the parameter set considered ($D = 7$ and 8 were considered for S10.

Table 4: Occurrences of different AE patterns in S1. Compare with Table 1.

D	2	3	4	5	6	7
$n_{\min}$	100169	10055	1024	105	14	1
$n_{\max}$	450143	120545	21274	2661	300	47
N_{tot}	3	13	75	541	4683	47293

$n_{\min}$ denotes the smallest number of repetitions in S1 for a pattern, $n_{\max}$ is the biggest number of repetitions, and N_{tot} is the total number of patterns found.

Table 5: Arithmetic entropy, permutation entropy, and normalized entropies for the first 1 000 000 digits of $\sqrt{2}$.

D	2	3	4	5
AE	1.369	3.477	6.067	8.992
PE	0.993	2.563	4.536	6.816
NAE	0.864	0.940	0.974	0.990
NPE	0.993	0.992	0.989	0.987

Entropy is given in bits. NAE is calculated as $\text{AE}/\log_2(N_{\text{tot}})$, where N_{tot} is taken from the bottom row of Table 4.

Table 6: Same as in Table 5 for the first 10 000 000 digits.

D	2	3	4	5
AE	1.369	3.476	6.066	8.991
PE	0.993	2.563	4.537	6.817
NAE	0.864	0.939	0.974	0.990
NPE	0.993	0.992	0.990	0.987

The results, as regards decreasing and increasing, support those observed for smaller D). As it is illustrated in the previous paragraph, the D-tuples of digits from the expansion sequence are distributed unevenly between different order patterns both for PE and AE (this might explain dispersion of the patterns' frequencies observed in ([6], Figure 8)).The abovementioned behavior with increasing D suggests that the unevenness decreases for AE and increases for PE, at least in some "normalized" sense. This is for the $\sqrt{2}$ expansion. Whether a similar behavior takes place for other sequences and a possible practical utilization of this fact require additional study.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

In this paper, the following free software has been used: (i) linux operating system (https://getfedora.org/); (ii) GNU Scientific Library [20] (https://www.gnu.org/software/gsl/); (iii) GNU Multiple Precision Arithmetic Library (https://gmplib.org/); (iv) Maxima, a free Computer Algebra System (http://maxima.sourceforge.net/); and (v) RefDB, a free Reference Manager created by Markus Hoenicka (http://refdb.sourceforge.net/).

Conflicts of Interest

The author declares that there are no conflicts of interest.

Acknowledgments

The work was partially supported by the Program of Fundamental Research of the Department of Physics and Astronomy of the National Academy of Sciences of Ukraine "Mathematical models of nonequilibrium processes in open systems" (N 0120U100857).

References

[1] C. Bandt and B. Pompe, "Permutation entropy: a natural complexity measure for time series," *Physical Review Letters*, vol. 88, no. 17, Article ID 174102, 2002.

[2] A. Porta, S. Guzzetti, N. Montano, R. Furlan, M. Pagani et al., "Entropy, entropy rate, and pattern classification as tools to typify complexity in short heart period variability series," *IEEE Transactions on Biomedical Engineering*, vol. 48, no. 11, pp. 1282–1291, 2001.

[3] M. Zanin, L. Zunino, O. A. Rosso, and D. Papo, "Permutation entropy and its main biomedical and econophysics applications: a review," *Entropy*, vol. 14, no. 8, pp. 1553–1577, 2012.

[4] A. F. Bariviera, M. B. Guercio, L. B. Martinez, and O. A. Rosso, "A permutation information theory tour through different interest rate maturities: the libor case," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 373, no. 2056, Article ID 20150119, 2015.

[5] L. Tylová, J. Kukal, V. Hubata-Vacek, and O. Vyšata, "Unbiased estimation of permutation entropy in EEG analysis for Alzheimer's disease classification," *Biomedical Signal Processing and Control*, vol. 39, pp. 424–430, 2018.

[6] L. Zunino, F. Olivares, F. Scholkmann, and O. A. Rosso, "Permutation entropy based time series analysis: equalities in the input signal can lead to false conclusions," *Physics Letters A*, vol. 381, no. 22, pp. 1883–1892, 2017.

[7] D. Cuesta–Frau, M. Varela–Entrecanales, A. Molina–Picó, and B. Vargas, "Patterns with equal values in permutation entropy: do they really matter for biosignal classification?" *Complexity*, vol. 2018, Article ID 1324696, 15 pages, 2018.

[8] H. Azami and J. Escudero, "Amplitude-aware permutation entropy: illustration in spike detection and signal segmentation," *Computer Methods and Programs in Biomedicine*, vol. 128, pp. 40–51, 2016.

[9] Z. Chen, Y. Li, H. Liang, and Y. Jing, "Improved permutation entropy for measuring complexity of time series under noisy condition," *Complexity*, vol. 2019, Article ID 1403829, 12 pages, 2019.

[10] C. Bian, Q. Chang, D. Qianli, Y. Ma, and Q. Shen, "Modified permutation-entropy analysis of heartbeat dynamics," *Physical Review E*, vol. 85, no. 2, Article ID 021906, 2012.

[11] T. Haruna and K. Nakajima, "Permutation approach to finite-alphabet stationary stochastic processes based on the duality between values and orderings," *The European Physical Journal Special Topics*, vol. 222, no. 2, pp. 383–399, 2013.

[12] S. Berger, G. Schneider, F. E. Kochs, and D. Jordan, "Permutation entropy: too complex a measure for EEG time series?" *Entropy*, vol. 19, no. 12, Article ID 692, 2017.

[13] K. Keller, A. Unakafov, and V. Unakafova, "Ordinal patterns, entropy, and EEG," *Entropy*, vol. 16, no. 12, pp. 6212–6239, 2014.

[14] T. Gutjahr and K. Keller, "Ordinal pattern based entropies and the Kolmogorov–Sinai entropy: an update," *Entropy*, vol. 22, no. 1, Article ID 63, 2020.
[15] C. Bandt, "Ordinal time series analysis," *Ecological Modelling*, vol. 182, no. 3-4, pp. 229–238, 2005.
[16] W. Aziz and M. Arif, "Multiscale permutation entropy of physiological time series," in *in Proceedings of the 2005 Pakistan Section Multitopic Conference*, pp. 1–6, Karachi, Pakistan, December 2005.
[17] C. W. Kulp and L. Zunino, "Discriminating chaotic and stochastic dynamics through the permutation spectrum test," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 24, Article ID 033116, 2014.
[18] J. Riordan, *An Introduction to Combinatorial Analysis*, John Wiley, Hoboken, NJ, USA, 1958.
[19] N. Pippenger, "The hypercube of resistors, asymptotic expansions, and preferential arrangements," *Mathematics Magazine*, vol. 83, no. 5, pp. 331–346, 2010.
[20] M. Galassi, J. Davies, J. Theiler et al., *GNU scientific library reference manual*, Network Theory Ltd, 2009, https://www.freetechbooks.com/network-theory-ltd-p1818.html.
[21] K. Keller, M. Sinn, and J. Emonds, "Time series from the ordinal viewpoint," *Stochastics and Dynamics*, vol. 7, no. 2, pp. 247–272, 2007.
[22] A. B. Piek, I. Stolz, and K. Keller, "Algorithmics, possibilities and limits of ordinal pattern based entropies," *Entropy*, vol. 21, no. 6, Article ID 547, 2019.
[23] M. Queffeec, "Old and new results on normality,""Old and new results on normality," in *Dynamics & Stochastics, Lecture Notes–Monograph Series*, D. Denteneer, F. den Hollander, and E. Verbitskiy, Eds., pp. 225–236, Institute of Mathematical Statistics, 2006, https://imstat.org/, 48 edition.