ELSEVIER

2 June 1997

Physics Letters A 229 (1997) 375–378

PHYSICS LETTERS A

A vectorized algorithm for correlation dimension estimation

Eran Toledo [a], Sivan Toledo [b], Yael Almog [a], Solange Akselrod [a,*]

[a] *The Abramson Center for Medical Physics, Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel*
[b] *Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA*

Received 4 December 1996; accepted for publication 24 February 1997
Communicated by A.R. Bishop

Abstract

We present an algorithm for the efficient and reliable computation of the Grassberger–Procaccia correlation dimension in cases where the correlation integral is constructed using all pairs of points. This algorithm is explicitly designed for modern super-scalar processors. © Published by Elsevier Science B.V.

PACS: 89.80.+h; 05.45.+b
Keywords: Correlation dimension; Nonlinear; Algorithm; GPA

Nonlinear dynamics techniques have recently become valuable tools for analyzing and evaluating dynamical systems in many fields. The Grassberger–Procaccia correlation dimension [1], in particular, is used by many investigators since it is applicable for experimental data and provides a quantitative measure of the system's complexity. In this Letter we introduce a vectorized algorithm for the estimation of the correlation dimension from time series. This algorithm computes the correlation dimension accurately, unlike other proposed efficient algorithms in the literature. In addition, it is well-suited for high-performance modern processors.

Estimation of the correlation dimension (commonly referred to as D_2) involves embedding the time series with delay coordinates and estimating the probability density function of the distance between pairs of points in an m-dimensional space [2]. The formal definition of D_2 is

$$D_2 = \lim_{r \to 0} \frac{\mathrm{d} \ln C(r)}{\mathrm{d} \ln r}. \tag{1}$$

$C(r)$ is the correlation integral,

$$\begin{aligned} C(r) &\equiv \int_0^r \mathrm{d}^m r' \; c(r') \\ &\equiv \lim_{N \to \infty} \frac{1}{N(N-1)} \sum_{i \neq j} H\left(r - \| x_i - x_j \|\right) \\ &= \lim_{N \to \infty} \frac{1}{N} \sum_j C_i(r), \end{aligned} \tag{2}$$

* Corresponding author. School of Physics, Tel Aviv University, Tel Aviv, Israel. E-mail: sol@novelty.tau.ac.il.

PII S0375-9601(97)00186-2

where $c(r)$ is the standard correlation function, $H(x)$ is the Heaviside step function. $C_i(r)$, the pointwise correlation integral, is a histogram of the distance between point x_i and all the other $N-1$ points in an m-dimensional space,

$$C_i(r) = \lim_{N\to\infty} \frac{1}{N-1} \sum_{j\neq 1} H\big(r - \| x_i - x_j \|\big). \quad (3)$$

The correlation integral can also be interpreted as the mean over all the pointwise correlation integrals. Real time series can only provide an estimation of the correlation integral, since the length of the time series is limited.

The most straightforward implementation of correlation dimension estimation of a chaotic time series of length N has a computational complexity of $O(N^2)$. Furthermore, in practice, a reliable estimation of the correlation dimension must rely on the estimation for several embedding dimensions, thus increasing the computational complexity to $O(N^2 \cdot N_D)$, where N_D is the number of embedding dimensions. This high complexity places practical bounds on the signal's length that can be analyzed and has urged investigators to look for more efficient algorithms [3–7].

One possible way to reduce the complexity is to estimate the correlation integral from a sample of size M from the pointwise correlation integrals, reducing the complexity to $O(N \cdot M \cdot N_D)$ [6]. Although the complexity is reduced, this reduction is at the expense of accuracy. Chaotic attractors have a correlation integral of the form $C(r) \propto r^{D_2}$ for small r, i.e. as the distance approaches zero the probability to find pairs of points closer than that distance decreases exponentially to zero. Therefore, a large sample of pointwise correlation integrals is required to accurately determine the probability density function of the distance between points, rendering the correlation integral a poor estimator, when obtained from a small sample of the pointwise correlation integrals.

Another way to overcome the high complexity is to use the fact that D_2 is defined for small r. Therefore, in order to estimate D_2, knowledge of $C(r)$ is needed only for small r [3,7]. The core of the efficient algorithms described in Refs. [3–5,7] is placing the time series in a smart data structure (coarse grid or m-dimensional trees) that reduces the time needed for finding all the neighbors of point x_i within distance ε to $O(N \cdot C(\varepsilon))$ and the total complexity to $O(N^2 \cdot C(\varepsilon) \cdot D)$. Those algorithms achieve the maximal accuracy available and a considerable reduction in complexity, since $C(\varepsilon) \ll 1$ for small ε. The major limitation of those algorithms is that the estimation of the correlation integral is calculated for a bounded range of r. This limitation becomes significant when real signals are discussed, since noise plays a crucial role in real data. It is a well known fact that additive noise obscures the correlation integral for $r < \eta$, where η represents the amount of noise added to the chaotic signal [8]. However, in most cases the amount of noise is unknown a priori, therefore the estimation of $C(r)$ is required over the entire range of r [8]. Furthermore, the existence of noise compels us to estimate $C(r)$ for larger distances, thus causing the algorithms optimized for small r to be inefficient. In those cases they become even more wasteful than the straightforward algorithm [3].

As mentioned above, there are cases in which an analysis over all the $N(N-1)$ pairs is required and an algorithm with computational complexity of $O(N^2 \cdot D)$ is unavoidable. The only improvement relevant for those cases is lowering the coefficient before the N^2D. We present in this paper an algorithm, optimized for vector computation and, therefore, reduce CPU time by a factor of 2–3 relative to a naive algorithm, when using advanced CPUs.

In order to estimate the correlation integral from which we estimate D_2 we need to calculate the histogram of the distances between all pairs of points. The most naive way is to iterate through all the distances of the histogram and for each distance count how many pairs of points are distant from each other by that specific distance. This is indeed a very time-consuming algorithm and it is undoubtedly preferable to update the histogram while iterating over all the pairs rather than iterating over the bins of the histogram.

The more sophisticated algorithm consists of three phases: for every pair of m-dimensional points we calculate the distance between the two points, then we compute the logarithm of the distance, and finally we add 1 to the bin of the histogram containing the logarithm of that distance. Although computing the

logarithm of the distance is not obligatory, we prefer to do so since the definition of the correlation dimension involves the logarithm of the distance. Therefore, building the histogram out of linear distances would cause biasing of the slope of $C(r)$ towards the larger r [9].

The drawback of this implementation is its unfitness for modern processors, since the modern superscalar processors achieve high performance computation speed when the process can be fed into a pipeline [10]. When a process is composed of simple tasks like addition, subtraction and so on, the processor can start one task before it has finished the previous task. This increases the net computation speed considerably although each single task takes the same time. When computing the distance, computing the logarithm and then updating the histogram the processor is unable to use its pipeline capabilities, thus slowing the computation speed considerably.

The innovation of our algorithm consists in separating the three phases that manipulate the pairs, namely the calculation of the distance between a group of points, the computation of the logarithm of the distances and the update of the histogram. The separation of the three parts and the processing of a group of pairs enables the processor not only to use its pipeline capabilities but also to use CPU vector instructions, which accelerate the computation time by a factor of 2–3.

The separation of the computation into simple elements is enabled using the following recursion rules: Let

$$R_{i,j}^{m,\tau} = \sum_{k=0}^{m-1} \left(x_{i+k\tau} - x_{j+k\tau} \right)^2 \tag{4}$$

be the square of the Euclidean distance between the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with delay coordinate τ in embedding dimension m. It is easily seen that

$$R_{i,j}^{m,\tau} = R_{i-\tau,j-\tau}^{m+1,\tau} - \left(x_{i-\tau} - x_{j-\tau} \right)^2 \tag{5}$$

and that

$$R_{i,j}^{m,\tau} = R_{i-\tau,j-\tau}^{m,\tau} - \left(x_{i-\tau} - x_{j-\tau} \right)^2 + \left(x_{i+(m-1)\tau} - x_{j+(m-1)\tau} \right)^2. \tag{6}$$

Consider a table d whose elements are $d_{i,j} = (x_i - x_j)^2$ where $d_{i,j} = d_{j,i}$. The three phases are performed on the diagonals of d and R rather than on the rows. Eqs. (5) and (6) show that after the initialization, every distance between the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in embedding dimension m can be easily calculated using the distance between points $x_{i'}$, and $x_{j'}$, on the same diagonal ($i - j = i' - j'$) and the already calculated ($m + 1$)-dimensional distance between points on the same diagonal of R. Therefore, the calculation of $(x_i - x_{i+p})^2$ for all i and a fixed p is fed into the processor's pipeline. Then the calculation of the distances is fed to the pipeline using Eq. (5) for an embedding dimension lower than the maximal dimension desired and using Eq. (6) for the maximal dimension. After calculating all the distances in the matrix R, the logarithm is computed for all the elements of the matrix. Since the matrix is arranged in the computer memory as a vector, a vectorized logarithm instruction can be issued, saving considerable CPU time. We then update the histogram and repeat the procedure for the increased values of p.

We compared our algorithm to the naive one on two computers. On a Silicon Graphics PowerChallenge with 200 MHz R10000 processor, we found an improvement by a factor of 2 in CPU time. On a 143 MHz Sun UltraSparc the improvement was by a factor of 3.

We claim that in cases which require computation of the correlation integral over all the $N(N-1)/2$ pairs, reduction in CPU time can be achieved through adaptation of the naive algorithm to the computer's architecture. Our algorithm by virtue of its structural simplicity achieves a considerable time saving. Furthermore, the algorithm provides the most accurate correlation integral estimation possible for the time series under study.

References

[1] P. Grassberger and I. Procaccia, Physica D 9 (1983) 189.
[2] N. Gershenfeld, An experimentalist's introduction to the observation of dynamical systems, in: Directions in chaos, Vol 2, ed. H. Bai-Lin (World Scientific, Singapore, 1988).
[3] P. Grassberger, Phys. Lett. A 148 (1990) 63.
[4] S. Bingham and M Kot, Phys. Lett. A 140 (1989) 327.
[5] P. Grassberger, T. Schreiber and C. Scaffrath, Int. J. Bifurc. Chaos 1 (1991) 521.

[6] G. Mayer-Kress, Application of dimension algorithms to experimental chaos, in: Directions in chaos, Vol 1, ed. H. Bai-Lin (World Scientific, Singapore, 1988).
[7] J. Theiler, Phys. Rev. A 36 (1987) 4456.
[8] T. Schreiber and H. Kantz, Chaos 1 (1995) 133.
[9] N.A. Gershenfeld, Physica D 55 (1992) 135.
[10] D.A. Patterson and J.L. Hennessy, Computer architecture, a quantitative approach, 2nd Ed. (Morgan Kaufmann, Los Altos, CA, 1995).