Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 93 (2005) 91–97
Information Processing Letters
www.elsevier.com/locate/ipl

Fast estimation of fractal dimension and correlation integral on stream data ☆

Angeline Wong [a], Leejay Wu [a,*], Phillip B. Gibbons [b], Christos Faloutsos [a]

[a] *Department of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891, USA*
[b] *Intel Research, 417 South Craig Str., Suite 300, Pittsburgh, PA 15213, USA*

Received 24 November 2003

Communicated by S. Albers

Abstract

In this paper we give a very space-efficient, yet fast method for estimating the fractal dimensionality of the points in a data stream. Algorithms to estimate the fractal dimension exist, such as the straightforward quadratic algorithm and the faster $O(N \log N)$ or even $O(N)$ box-counting algorithms. However, the sub-quadratic algorithms require $\Omega(N)$ space. In this paper, we propose an algorithm that computes the fractal dimension in a single pass, using a constant amount of memory relative to data cardinality. Experimental results on synthetic and real world data sets demonstrate the effectiveness of our algorithm.

Keywords: Approximation algorithms; Databases; Randomized algorithms; Fractal dimension; Box counting

1. Introduction

The intrinsic dimensionality of a set has numerous related definitions, one of the more intuitive being the number of parameters required to fully describe or represent each member of a data set. In general, none of these definitions readily lend themselves to measurement; however, for self-similar sets, the fractal dimension provides a reasonable estimation method. The fractal dimension has been shown to facilitate selectivity estimation, range queries [4,8], nearest-neighbor queries [2,3,6], similarity searches [2], dimensionality reduction [10], outlier detection [7], and other applications.

☆ This material is based upon work supported by the National Science foundation under Grants No. IIS-9988876, IIS-0083148, IIS-0113089, IIS-0209107, IIS-0205224; by the Pennsylvania Infrastructure Technology Alliance (PITA) Grant No. 22-901-0001; and by the Defense Advanced Research Projects Agency under Contract No. N66001-00-1-8936. Additional funding was provided by donations from Intel. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA, or other funding parties.

* Corresponding author.
E-mail addresses: angeline.wong@alumni.cmu.edu (A. Wong), lw2j@cs.cmu.edu (L. Wu), phillip.b.gibbons@intel.com (P.B. Gibbons), christos@cs.cmu.edu (C. Faloutsos).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2004.09.018

The development of fast approximation algorithms has reduced the computational complexity of estimating fractal dimensions from $O(N^2)$, where N is the number of points in the dataset, to linear [10]. However, the memory costs have remained at $\Theta(N)$. Herein we propose and evaluate the Tug-of-War algorithm, which combines $O(N)$ computational and $O(1)$ storage complexities.

2. Background and definition of fractals and fractal dimensions

Suppose we have a self-similar dataset of N points a_i in $\mathcal{R}^E$, such as the Sierpinski triangle (Fig. 1). What is its intrinsic dimensionality? While with certain types of point sets such as uniform lines, planes, and solids, the intrinsic dimensionality would be the corresponding obvious integer, for real point sets a fractional dimension—or fractal dimension—may more precisely reflect the lack of complete independence between dimensions.

A series of "fractal dimensions" has been defined, of which we shall consider one: the correlation fractal dimension, D_2. For a point-set that shows self-similarity in the range of scales (r_1, r_2), define D_2 as follows:

$$D_2 \equiv \frac{\partial \log \sum_i p_i^2}{\partial \log(r)}, \quad r \in [r_1, r_2],$$

where p_i is the occupancy of the ith cell when the E-dimensional address space is divided into (hyper)-cubic grid cells with sides of length r [2]. Alternatively, $\sum_i p_i^2$ is given by the second frequency moment, $F_2 = \sum_i |\{j: a_j = i\}|^2$, also called the *repeat rate* [1]. The method we propose approximates this correlation fractal dimension D_2.

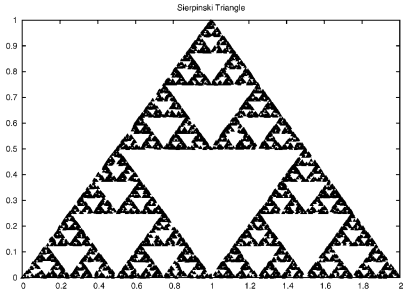

Fig. 1. The Sierpinski triangle.

2.1. Survey of methods for estimating fractal dimensions

The following subsections briefly explain two salient methods for computing D_2 fractal dimensions. The first, pair-counting, guarantees perfect accuracy but involves the enumeration of all $O(N^2)$ pairs. The second, box-counting, sacrifices some accuracy in exchange for incurring an only $O(N)$ computational cost and potentially $O(N)$ storage cost.

2.1.1. Pair-counting

One immediate method involves calculating the number of pairs of points within a given distance of each other, hence the name "pair-counting". The average number of neighbors within a given radius r, C_r, is exactly twice the total number of pairs within distance r of each other, divided by the number of points N in the set.

Self-similar sets, by definition, obey a power-law relationship [2]:

$$C_r \propto r^{D_2}$$

within limits—below or above certain radii, a finite set must stray from strict adherence to that law. Therefore, the fractal dimension can be computed as the slope of the linear relationship between $\log(C_r)$ and $\log(r)$. We can also compute the correlation between these two quantities within the region of radii where the set is in fact linear or nearly so; this correlation then serves as a measure of reliability and self-similarity.

Computing the quantity D_2 exactly requires determining the distance between every two points. While the distances need not be stored, they must be computed at least once. This $O(N^2)$ computational cost proves prohibitive for realistically large data sets. In practice, one trades accuracy for performance.

2.1.2. Box-counting

The practical box-counting algorithm provides one such compromise [2,9,10]. This algorithm derives its name from the imposition of nested hypercube grids on the data, followed by counting the occupancy of each grid cell [2], thus focusing on individual points instead of on pairs. The sum of squared occupancies $S_2(r)$ for a particular grid side length r is defined as $S_2(r) \equiv \sum_i C_i^2$, where C_i is the count of points in the ith cell. Substituting these counts for the pair counts in

the same power-law relationship estimates the fractal dimension. Given sufficient space to store all the counters simultaneously, all counts can be made in a single pass over the data. Even when performing one pass per radius to minimize the storage cost, the computational cost is still only linear with respect to N [10]. This algorithm has a storage complexity of $\Theta(N)$, as every datum might end up in its own cell, given a sufficiently small radius.

3. Proposed method: Tug-of-War

Accepting an $O(N)$ computational cost seems reasonable, as one must consider every point at least once, unless sampling. However, we can drastically reduce the storage cost to the point where it proves constant with respect to N. Our proposed method provides such space-efficient performance without excessively sacrificing speed or accuracy.

3.1. Preliminaries

Consider the box-counting method. It divides the data space according to a variety of granularities and for each granularity, computes the sum of squared occupancies. If one considers the individual cells of the hypergrid as corresponding to events or outcomes, then it becomes clear that this counting problem is nothing more than discretization followed by the determination of the second moment. Previous work by Alon, Matias, and Szegedy provides a theoretical framework for efficiently estimating second moments of item frequencies [1]. Theorem 2.2 and its proof from that paper show that one can use a set of four-wise independent hash functions h_{uc} to compute a value Z whose square approximates the second moment F_2 (which is $S_2(r)$ for a given grid side r) within probably approximately correct (PAC) bounds.[1] In addition, the suggested method relies on a compact set of counters whose cardinality need not depend on the cardinality of the data set or of a full hypergrid. For compactness, we must direct those seeking mathematical proof of the efficacy of this estimation method to that earlier work.

[1] The subscripts u and c refer to their use as indices derived from the accuracy and confidence parameters.

The algorithm in the next subsection expands upon this by showing how to efficiently translate multidimensional coordinates into discrete events. We also present suitable four-wise independent hash functions and how to generate them, since Alon et al. [1] describe the four-wise independence property the functions should have and one possible, general method to generate such, but do not specify explicitly the format of the functions or how they ought to be generated.

3.2. Proposed algorithm

Our Tug-of-War algorithm efficiently approximates the correlation fractal dimension by extending this algorithm to compute the second moment F_2 for multidimensional points and for a *range* of grid sides r. Recall that Alon et al. [1] define probably approximately correct bounds. The user-specified parameter s_1 affects the level of approximation, while s_2 determines the corresponding probability. For any positive integral values of s_1 and s_2, let $\lambda = \sqrt{16/s_1}$ and $\epsilon = 1/e^{s_2/2}$. Then, the probability that the estimate of the second moment has a greater relative error than λ is at most ϵ [1]. Note that these terms are both independent of N.

Suppose we have a self-similar dataset M of N points a_i in $\mathcal{R}^E$, from which we can form a sequence $A = (a_1, a_2, \ldots, a_N)$ in which each a_i can be further expanded into an E-dimensional tuple $(a_{i1}, \ldots, a_{iE})$.

For each radius, we generate $s_1 s_2$ 4-tuples of random prime numbers $\{\alpha, \beta, \gamma, \delta\}$, which each define a four-wise independent hash function h_{uc} of the form

$$\mathit{hashParity}(\mathit{result}),$$

where *hashParity* maps *result* to -1 or 1 according to the value of the exclusive-or of its bits and

$$\mathit{result} = \sum_{j=1}^{E} \alpha^j x_{ij}^3 + \beta^j x_{ij}^2 + \gamma^j x_{ij} + \delta^j \bmod q,$$

where

$$x_{ij} = \left\lfloor \frac{a_{ij}}{r} \right\rfloor$$

and q is a (large) prime number. These polynomials are guaranteed to be irreducible and the hash functions four-wise independent, therefore permitting the use of the original PAC bounds.

Estimating one second moment requires one counter per hash function [1]; estimating the fractal dimension requires one second moment estimation for each of the $|R| = |\{r\}|$, and thus needs $|R|s_1s_2$ counters and hash functions. Every hash function is applied to every E-tuple to increment or decrement the corresponding counters. For each radius r, estimation of the second moment of the corresponding hypergrid proceeds by first partitioning the s_1s_2 counters into s_2 sets of s_1 counters, squaring the counters' values, computing the mean of each subset, and then determining the median of these means. As per [1], this yields Z^2, an estimate for $F_2 = S_2(r)$. Computation of D_2 at that point proceeds in the usual direction.

3.3. Computational and memory complexities

The computational complexity of the Tug-of-War method is linear in the size N of the database or current data stream. The runtime is dependent on N, the embedding dimensionality E, the number of radii $|R| = |\{r\}|$, the accuracy parameter s_1, and the confidence parameter s_2. E, $|R|$, s_1, and s_2 are independent of N, so the runtime complexity of $O(EN|R|s_1s_2)$ is actually just linear—$O(N)$—with respect to N.

The implemented four-wise independent hash functions each use four hash keys and one hash function is needed for each of the $|R|s_1s_2$ counters. Thus the amount of memory required for the hash keys is $4|R|s_1s_2 \times$ `sizeof(int)`, which is $O(|R|s_1s_2)$, or $O(1)$ with respect to N. The storage required for the counters is $|R|s_1s_2 \times$ `sizeof(long)`, which is also $O(|R|s_1s_2)$, so again $O(1)$ with respect to N. Additional, constant overhead is needed to compute the hash values. The total memory complexity for the Tug-of-War method is therefore $O(1)$ with respect to N, but linear with respect to $|R|$, s_1, or s_2.

Since all the counters and functions may fit in memory simultaneously, the Tug-of-War method can process all the data in a single pass. Streams should present no problems, since at any point one can economically compute the median of means of the counters for each of the radii. Then, to estimate the appropriate partial derivative, we perform a linear regression on the logarithms of the per-radius estimates. This latter regression can also be performed incrementally using recursive least-squares if necessary.

4. Experiments

In this section, we present empirical verification of the speed and accuracy of the Tug-of-War algorithm. Our C++-based implementation was tested on both real-world and synthetic data on the Linux platform, with 31 different radii, an accuracy parameter s_1 of 30, and confidence parameter s_2 of 5.

The same set of randomly-generated hash functions was used for all tests. As per [1], any functions that meet the four-wise independence criterion will preserve the PAC bounds.

4.1. Quality of the estimation

Estimating the fractal dimension of a diagonal line provides a simple first test. The result would clearly be 1 were the set infinite; a finite set should be close. The test diagonals consist of 1000 points with E-dimensional embedding $\{(i, \ldots, i) \mid i = 1..1000\}$. The Tug-of-War method estimated D_2 fractal dimensions 1.045, 1.050, and 1.038 with correlations 0.997, 0.999, and 0.993 for $E = 2$, 3, and 4, respectively. Further evidence that the Tug-of-War method performs correctly comes from a comparison of the log-log plots produced by this and the box-counting methods (Fig. 2). The estimated fractal dimension closely matches the theoretical value of 1, the reported correlations approach 1, and the linear regions within the log-log plots prove nearly identical.

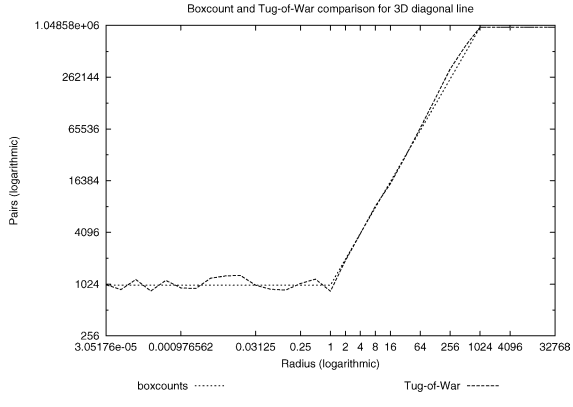

Fig. 2. Comparison of box-count and Tug-of-War log-log plots for diagonal line with $E = 3$.

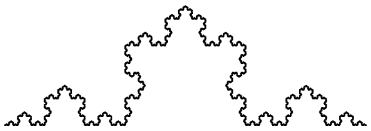

Fig. 3. Koch snowflake.

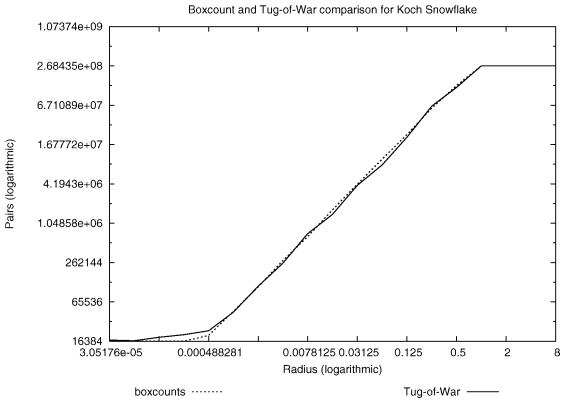

Fig. 4. Tug-of-War vs. box-counting, Koch snowflake.

4.1.1. Synthetic datasets

Additional testing was performed with more complex fractals to again compare experimental estimates of the fractal dimension with theoretical values. The results of two well-known fractals, the Sierpinski triangle and the Koch snowflake (Figs. 1 and 3), are discussed here.

The Sierpinski5K set consists of the first 5000 points generated through a breadth-first recursive generation of the Sierpinski triangle. On this set, the Tug-of-War method estimates D_2 as 1.520 with a near-perfect correlation of 0.999, reasonably close to the theoretical value of $\log_2 3 \approx 1.585$ [5]. Previous measurements using box-counting determined the D_2 to be 1.587.

The theoretical fractal dimension of the Koch snowflake is $\log_3 4 \approx 1.262$. For a Koch dataset containing 16,385 points, Tug-of-War produces an estimated D_2 of 1.255 and a correlation of 0.999. A comparison of the box-counting and Tug-of-War log-log trends (Fig. 4) further supports the high quality of the results.

4.1.2. Real datasets

Real point-sets behave like fractals more than one might expect [2]. Three such datasets—Montgomery County, MD road intersections, Long Beach County, CA road intersections (Fig. 5), and an fMRI (Fig. 6)—are discussed here.

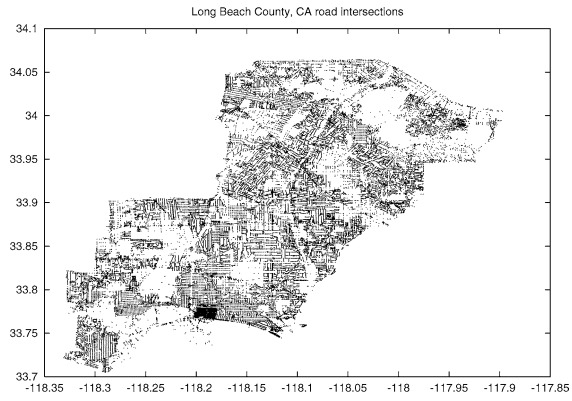

Fig. 5. Long Beach County, CA.

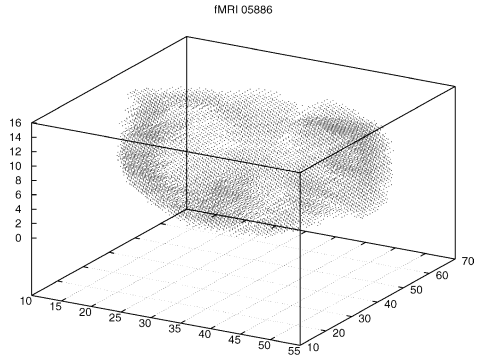

Fig. 6. fMRI 05886.

The Montgomery County and Long Beach County planar datasets contain 27,282 and 36,548 points, respectively. The 05886 fMRI dataset has 144,768 points in three dimensions. The Tug-of-War method estimates D_2 of 1.557, 1.758, and 2.521 with correlations 0.996, 0.999, and 0.998 for these datasets, respectively.

As one might indeed expect, due to their non-uniformity, these sets have intrinsic dimensionalities significantly less than their embedding dimensionalities [2]. Even discounting the high correlations and close matches of the box-count and Tug-of-War plots, the Tug-of-War results agree with previous measurements (see Figs. 7 and 8). While we know of no previous measurements for the fMRI dataset for comparison, box-counting has been used to estimate the

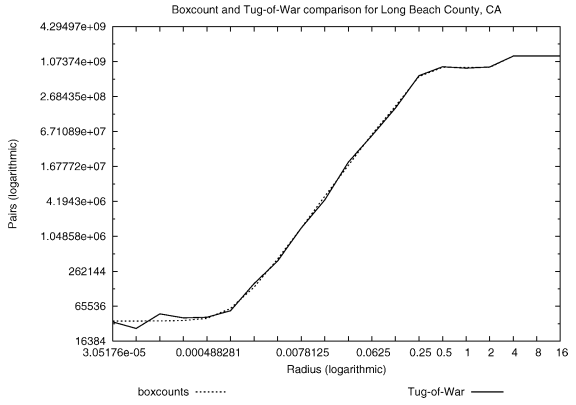

Fig. 7. Tug-of-War vs. box-counting, Long Beach County.

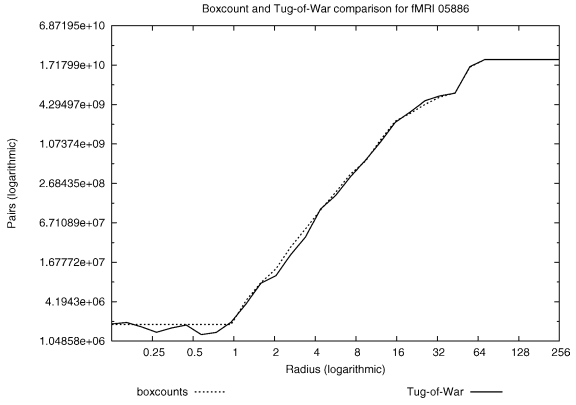

Fig. 8. Tug-of-War vs. box-counting, fMRI.

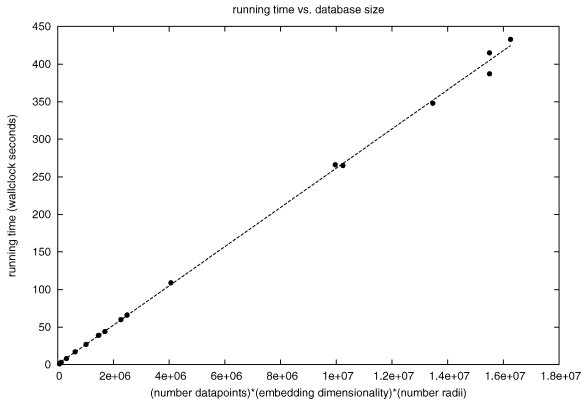

Fig. 9. Running time against database size.

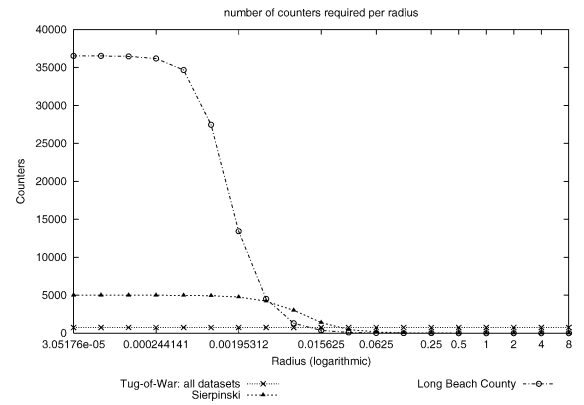

Fig. 10. Counter requirements for box-counting and Tug-of-War methods.

D_2 values for Montgomery County and Long Beach County datasets as 1.518 and 1.732, respectively [2].

Tug-of-War provides accuracy on both the synthetic and real-world data. The following subsection examines performance and scalability.

4.2. Running time and memory consumption

Multiple sets of differing size were generated using the Iterative Function System. Fig. 9 shows the linear relationship of running time (as the sole user of an Intel® Xeon™ CPU 2.80 GHz processor) to database size, calculated as the product of the number of points, their embedding dimensionality, and number of radii used.

The Tug-of-War method's greatest advantage over other computationally linear algorithms used to estimate fractal dimensions rests in its storage efficiency; both the storage cost and the PAC bounds depend on s_1 and s_2 but not on N. We calculated the number of counters necessary for the box-counting and Tug-of-War methods. Fig. 10 shows that the number of counters for the Tug-of-War method is constant across all radii and demonstrates that the total number of counters required to attain a 99.5% level of accuracy and precision (as measured by the correlation) when processing large databases is significantly less than that of box-counting.

5. Conclusions

The Tug-of-War method is a very fast and space-efficient approximation algorithm for accurately estimating the correlation fractal dimension. Its key fea-

tures are: it handles multi-dimensional data; it requires but a single pass through the data; it has a computational complexity of $O(N)$ and a space complexity of $O(1)$ with respect to N; and it closely matches the accuracy yielded by previous algorithms of similar $O(N)$ speed but $O(N)$ space complexity. Thus we advocate its use for large data sets and data streams.

References

[1] N. Alon, Y. Matias, M. Szegedy, The space complexity of approximating the frequency moments, in: Proc. 28th ACM Symp. on the Theory of Computing, May 1996, pp. 20–29.

[2] A. Belussi, C. Faloutsos, Estimating the selectivity of spatial queries using the 'correlation' fractal dimension, in: U. Dayal, P.M.D. Gray, S. Nishio (Eds.), Proc. 21th Internat. Conf. on Very Large Data Bases, Morgan Kaufmann, September 1995, pp. 299–310.

[3] S. Berchtold, C. Böhm, B. Braunmüller, D. Keim, H.-P. Kriegel, Fast parallel similarity search in multimedia databases, in: Proc. ACM SIGMOD Internat. Conf. on Management of Data, May 1997, pp. 1–12.

[4] C. Faloutsos, I. Kamel, Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension, in: Proc. ACM SIGACT-SIGMOD-SIGART PODS, Minneapolis, MN, May 24–26 1994, pp. 4–13; Also available as CS-TR-3198, UMIACS-TR-93-130.

[5] B. Mandelbrot, Fractal Geometry of Nature, W.H. Freeman, New York, 1977.

[6] B.-U. Pagel, F. Korn, C. Faloutsos, Deflating the dimensionality curse using multiple fractal dimensions, in: Proc. Internat. Conf. on Data Engineering, February 2000, pp. 589–598.

[7] S. Papadimitriou, H. Kitagawa, P.B. Gibbons, C. Faloutsos, LOCI: fast outlier detection using the local correlation integral, in: Proc. Internat. Conf. on Data Engineering, March 2003, pp. 315–326.

[8] A. Papadopoulos, Y. Manolopoulos, Performance of nearest neighbor queries in R-trees, in: Proc. 6th Internat. Conference on Database Theory, January 1997, pp. 394–408.

[9] M. Schroeder, Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise, W.H. Freeman and Company, New York, 1991.

[10] C. Traina Jr., A.J.M. Traina, L. Wu, C. Faloutsos, Fast feature selection using the fractal dimension, in: Proc. XV Brazilian Symp. on Databases, October 2000, pp. 158–171.