Community Ecology 17 (1): 24-27, 2016
1585-8553/$20.00 © Akadémiai Kiadó, Budapest
DOI: 10.1556/168.2016.17.1.4

Minimal box size for fractal dimension estimation

E. Rosenberg

AT&T Labs, Middletown, NJ 07748, USA. Email: ericr@att.com

Keywords: Box counting, Fractal dimension, Resolution, Self-similar, Scale.

Abstract: We extend Kenkel's model for determining the minimal allowable box size $s*$ to be used in computing the box counting dimension of a self-similar geometric fractal. This minimal size $s*$ is defined in terms of a specified parameter ε which is the deviation of a computed slope from the box counting dimension. We derive an exact implicit equation for $s*$ for any ε. We solve the equation using binary search, compare our results to Kenkel's, and illustrate how $s*$ varies with ε. A listing of the Python code for the binary search is provided. We also derive a closed form estimate for $s*$ having the same functional form as Kenkel's empirically obtained expression.

Introduction

One of the most widely used fractal dimensions is the box counting dimension d_B. Suppose we wish to estimate d_B from N points, e.g., pixels in a 2-dimensional image. We cover the N points with boxes of side length s. If the object is 2-dimensional, the boxes are squares; if the object is 3-dimensional, the boxes are cubes; in general for an E-dimensional object, the boxes are E-dimensional hypercubes. Let $B(s)$ be the number of boxes containing at least one of the N points. When s is sufficiently large, all points lie in one box, so $B(s) = 1$. As s decreases, $B(s)$ increases or remains constant. For all sufficiently small box sizes, each non-empty box contains a single point, so $B(s) = N$.

The box counting dimension d_B of a geometric object is defined as (Mandelbrot 1983)

$$d_B = -\lim_{s\to 0} \frac{\log B(s)}{\log s} .$$

There are examples for which d_B does not exist (Falconer 2003) but in this paper we assume d_B exists. In practice, d_B is computed by evaluating $B(s)$ for a set of J values of s, which we denote by s_j, $j = 1, 2,..., J$. Then, typically, a line is fitted (often using linear regression) to the J pairs $(-\log s_j, \log B(s_j))$, and the slope of this line is the estimate of d_B. Assume the s_j values are ordered so that $s_1 < s_2 < ... < s_J$. We want $s_J - s_1$ as large as possible, so as to minimize the error in estimating d_B from the J pairs $(\log s_j, \log B(s_j))$. However, we cannot make s_1 arbitrarily small, since then, as noted above, $B(s)$ approaches N, and the slope of the $(\log s, \log B(s))$ curve approaches 0. The study of the minimal and maximal box sizes has received a great deal of attention; see, e.g., the large number of references in Kenkel (2013).

A recent paper by Kenkel (2013) introduces a simple but very useful probabilistic model to obtain a relationship between N and s_1. The model estimates the expected value of $B(s)$ as a function of s and d_B. Then s_1 is chosen to be the value at which the "local slope" of the $(-\log s, \log B(s))$ curve deviates from d_B by no more than 0.001, where this accuracy was chosen since estimates of d_B are often expressed to three or four decimal places. Using 28 simulations in which s was decreased until the local slope is within 0.001 of d_B, Kenkel obtained the following approximation for the minimal usable box size:

$$s_1 \approx \left(\frac{N}{10}\right)^{-1/d_B} . \qquad (1)$$

Since (1) was obtained numerically, and only for the accuracy 0.001, this result provides no guidance on how s_1 varies as a function of the accuracy. Indeed, in many applications, e.g., in the analysis of neurons (Karperien 2013), d_B is only estimated to two decimal places. Letting ε be the desired accuracy, in this paper we generalize Kenkel's result to compute s_1 for arbitrary ε. We show that s_1 can be computed, for any ε, by standard one-dimensional search techniques such as binary search.

The original model

Following Kenkel (2013), consider first the 1-dimensional case. If we randomly select N points on [0, 1] then the probability that a given box (i.e., interval) of size s does not contain any of these points is $(1-s)^N$. The probability that a given box of size s contains at least one of N randomly selected points is $1 - (1-s)^N$. Hence the expected number $B(s)$ of nonempty boxes of size s is given by $B(s) = [1 - (1 - s)^N]/s$.

Considering next the 2-dimensional case, if we randomly select N points on the unit square [0, 1]×[0, 1] then the probability that a given box of size s does not contain any of these points is $(1 - s^2)^N$. The probability that a given box of size s

contains at least one of N randomly selected points is $1 - (1- s^2)^N$. Hence the expected number $B(s)$ of nonempty boxes of size s is given by $B(s) = [1 - (1 - s^2)^N]/s^2$.

Finally, consider the case where the N points are sampled from a self-similar fractal set with box counting dimension d_B. For example, N might be the total number of pixels in an image of a real-world fractal, such as a fern leaf. The expected number $B(s)$ of nonempty boxes of size s is given by (Kenkel 2013)

$$B(s) = \frac{1 - \left(1 - s^{d_B}\right)^N}{s^{d_B}} . \tag{2}$$

This equality is the basis for the analysis of the remainder of this paper.

From (2), Kenkel numerically computes the "local slope" of the ($\log s$, $\log B(s)$) curve, where the "local slope" $m(s)$ is defined by $m(s) \equiv (\log B(s + \delta) - \log B(s))/(\log (s + \delta) - \log s)$ for some small increment δ. Then, s_1 is the value for which $m(s) + d_B = 0.001$ and empirically s_1 is found to be well approximated using (1). From (1) we obtain $s_1^{d_B} = 10/N$. Substituting this in (2), we have

$$B(s_1) = \frac{1 - \left(1 - (10/N)\right)^N}{10/N} .$$

which yields, for large N, the estimate $B(s_1) = N/10$. This means no box size s should be used, in the estimation of d_B, for which $B(s) > N/10$ (Kenkel 2013).

The generalized model

In this section, we generalize the original model by determining the minimal box size when the constant 0.001 described in Section 2 is replaced by a positive parameter ε. Rather than work with a local slope of the ($\log s$, $\log B(s)$) curve, we work with an actual derivative, which can be obtained in closed form. Using the chain rule for derivatives, we have

$$\frac{d \log B(s)}{d \log s} = \left(\frac{d \log B(s)}{d s}\right)\left(\frac{d s}{d \log s}\right) = \left(\frac{d \log B(s)}{d s}\right)\left(\frac{d \log s}{d s}\right)^{-1} = \left(\frac{d \log B(s)}{d s}\right) s . \tag{3}$$

From (2), taking the derivative of $\log B(s)$ we have

$$\frac{d \log B(s)}{d s} = \frac{d_B N s^{-1}\left(1 - s^{d_B}\right)^{N-1} - d_B s^{-d_B - 1}\left(1 - \left(1 - s^{d_B}\right)^N\right)}{\left(1 - \left(1 - s^{d_B}\right)^N\right) s^{-d_B}} . \tag{4}$$

Combining (3) and (4) yields

$$\frac{d \log B(s)}{d \log s} = \frac{d_B N\left(1 - s^{d_B}\right)^{N-1} - d_B s^{-d_B}\left(1 - \left(1 - s^{d_B}\right)^N\right)}{\left(1 - \left(1 - s^{d_B}\right)^N\right) s^{-d_B}} . \tag{5}$$

Denoting the right hand side of (5) by $F(s)$, our task is to find the value s^* such that $F(s^*) = -d_B + \varepsilon$. For $s < s^*$ we have $F(s) > - d_B + \varepsilon$, so such box sizes should not be used in the estimation of d_B. Thus s^* is the minimal usable box size. As a first step towards calculating s^*, define

$$\alpha \equiv (1 - s^{d_B})^{N-1} . \tag{6}$$

While α is actually a function of s, for notational simplicity we write α rather than $\alpha(s)$. The equation $F(s) = -d_B + \varepsilon$ can now be rewritten as

$$\frac{d_B N\alpha - d_B s^{-d_B}\left(1 - \alpha(1 - s^{d_B})\right)}{\left(1 - \alpha(1 - s^{d_B})\right) s^{-d_B}} + d_B = \epsilon . \tag{7}$$

Dividing both sides by d_B yields

$$\frac{N\alpha - s^{-d_B}\left(1 - \alpha(1 - s^{d_B})\right)}{\left(1 - \alpha(1 - s^{d_B})\right) s^{-d_B}} = \frac{\epsilon}{d_B} - 1 . \tag{8}$$

Define $\nu \equiv (\varepsilon/d_B) - 1$. Multiplying the numerator and denominator of the left hand side of (8) by s^{d_B} yields $N\alpha s^{d_B} - 1 + \alpha - \alpha s^{d_B} = \nu(1 - \alpha + \alpha s^{d_B})$, which simplifies to $\alpha s^{d_B}(N - 1 - \nu) = (\nu + 1)(1 - \alpha)$. Recalling the definition of ν, more algebra yields

$$\frac{N d_B}{\epsilon} - 1 = \left(\frac{1 - \alpha}{\alpha}\right) s^{-d_B} . \tag{9}$$

Finally, from (9) and the definition (6) of α, we obtain an implicit equation for s^*:

$$\left(\left(1 - s^{d_B}\right)^{1-N} - 1\right) s^{-d_B} = \epsilon^{-1} N d_B - 1 . \tag{10}$$

Equation (10) is exact; no approximations were made. The parameter ε appears only in the right hand side, which is independent of s.

We compute a solution s^* of (10) using binary search over [0,1]; a code listing is provided in the Appendix. The binary

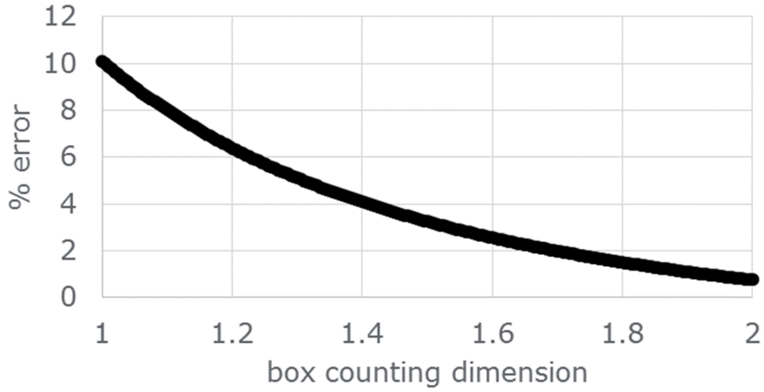

Figure 1. Percent error in the minimal usable box size for N =1000 and ε =0.001, for $1 \le d_B \le 2$.

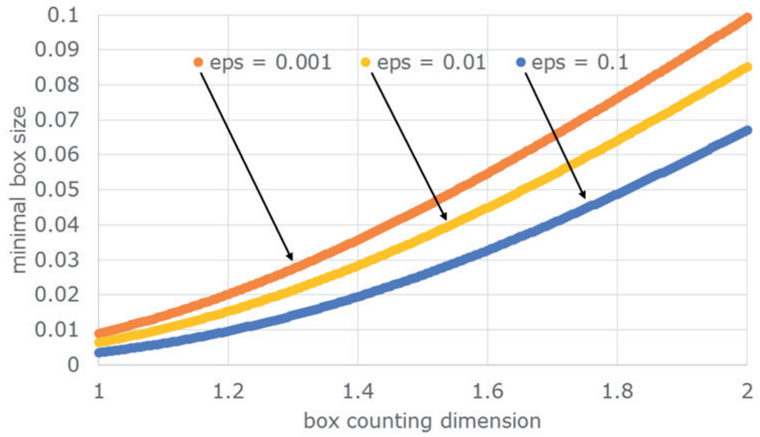

Figure 2. Minimal usable box size for N =1000 and for three values of ε, for $1 \le d_B \le 2$.

search is halted when the width of the interval containing s^* is less than 1.0×10^{-7}, which takes 24 iterations. For $N = 1000$ and ε =0.001, the percent error $100(s_1 - s^*)/s^*$, where s_1 is the estimate defined by (1), is plotted in Figure 1 as a function of d_B for $1 \le d_B \le 2$. The error in the approximation (1) is highest (about 10%) when $d_B = 1$.

Since, as noted in Section 1, in many applications d_B is estimated to only two decimal places, Figure 2 compares, for N =1000, the value s^* for ε=0.1, ε=0.01, and ε=0.001, all for the same range $1 \le d_B \le 2$.

Approximating the minimal usable box size

We now derive a closed form approximation to s^*. Defining $x = s^{d_B}$, we rewrite (10) as

$$x^{-1}\left((1-x)^{1-N} - 1\right) = \epsilon^{-1} N d_B - 1\,. \tag{11}$$

We have $x < 1$ since $s < 1$ and $d_B > 0$. Ignoring terms of degree four and higher, the Taylor series expansion of $(1-x)^{1-N}$ yields $(1-x)^{1-N} \approx 1-(1-N)x + (1-N)(-N)x^2/2 - (1-N)(-N)(-N-1)x^3/6$. For large N we have $(1-x)^{1-N} \approx 1 + Nx + (N^2/2)x^2 + (N^3/6)x^3$. Substituting this in (11) yields

$$\begin{aligned} x^{-1}\left((1-x)^{1-N} - 1\right) &\approx x^{-1}\left(1 + Nx + (N^2/2)x^2 + (N^3/6)x^3 - 1\right) \\ &= N + (N^2/2)x + (N^3/6)x^2 \\ &= \epsilon^{-1} N d_B - 1 \approx \epsilon^{-1} N d_B\,. \end{aligned}$$

Dividing by N, we obtain $(N^2/6)x^2 + (N/2)x + (1 - \varepsilon^{-1} d_B) \approx 0$. Using $1 - \varepsilon^{-1} d_B \approx -\varepsilon^{-1} d_B$ we get

$$x = \frac{-\frac{N}{2} \pm \sqrt{\frac{N^2}{4} - 4\frac{N^2}{6}(-\epsilon^{-1} d_B)}}{(2/6)N^2} \approx \frac{N\sqrt{\frac{2}{3}\epsilon^{-1} d_B}}{(1/3)N^2}$$

$$= \frac{1}{N}\sqrt{\frac{6 d_B}{\epsilon}}\,.$$

By definition, $x = s^{d_B}$, so $s^{d_B} = (1/N)\sqrt{(6d_B/\varepsilon)}$. Solving for s yields an approximation $\widetilde{s}$ for the minimal usable box size:

$$\widetilde{s} = \left(N\sqrt{\frac{\epsilon}{6\,d_B}}\right)^{-1/d_B}\,. \tag{12}$$

The estimate $\widetilde{s}$, which is a decreasing function of ε, looks identical to (1), except that the empirically derived constant 0.1 in (1) is replaced by $\sqrt{(\varepsilon/(6d_B))}$ in (12). However, $\widetilde{s}$ is not a good approximation to s^*. For example, with $N = 1000$, $d_B = 2$, and $\varepsilon = 0.001$, we have $s^* \approx 0.099$ while

$\widetilde{s} \approx 0.331$. The error arises from the third order Taylor series approximation, since for $x = (s^*)^2$ we have $(1-x)^{1-N} \approx 19700$ while $1 + Nx + (N^2/2)x^2 + (N^3/6)x^3 \approx 219$. Thus, while (12) yields the functional form obtained by Kenkel, in practice s^* should be computed using (10) and binary search.

Concluding remarks

We generalized Kenkel's model for finding the minimal usable box size s^* for computing the box counting dimension. For box sizes smaller than s^*, the slope of the $(-\log s, \log B(s))$ curve flattens out, and deviates by more than a specified accuracy ε from d_B. Whereas Kenkel considered only the choice ε =0.001, we derived an exact implicit equation for s^* for any ε. Binary search was used to actually compute s^*, and a Python implementation is provided in the Appendix. We also used the implicit equation to derive a closed form approximation for s^* having the same functional form as Kenkel's empirically obtained expression.

Acknowledgments. The author thanks N. Kenkel and an anonymous reviewer for their valuable comments and suggestions.

References

Falconer, K. 2003. *Fractal Geometry: Mathematical Foundations and Applications*, 2nd edn. Wiley, Chichester.

Karperien A., H. Ahammer and H.F. Jelinek. 2013. Quantitating the subtleties of microglial morphology with fractal analysis. *Frontiers in Cellular Neuroscience* 7: 3, doi: 10.3389/fncel.2013.00003.

Kenkel, N.C. 2013. Sample size requirements for fractal dimension estimation. *Community Ecol.* 14: 144–152.

Mandelbrot, B.B. 1983. *The Fractal Geometry of Nature*. W.H. Freeman, New York.

Received November 3, 2015
Revised December 20, 2015
Accepted January 26, 2016

Appendix: Binary search code

Python code used to implement the binary search method used in Section 3 to compute the s^* solving (10). The code performs binary search over the interval [0, 1] to compute s* to an accuracy of 10^{-7} for ε = 0.001 and for d_B from 1 to 2, in increments of 1/200. The variable Estimate is the estimate given by (1).

Binary Search

```
import math
N = 1000.
epsilon = .001
for i in range(201):
    d = 1.0 + float(i)/200.
    RHS = -1.0 + N*d/epsilon
    Estimate = (0.1*N)**(-1.0/d)
    low = 10**(-8)
    high = 1.0
    while (low + 10**(-7) <= high):
        s = (low + high)/2.0
        x = s**d
        LHS = ((1.0-x)**(1.0-N) -1.0)/x
        if LHS < RHS:
            low = s
        else:
            high = s
    error = 100*(Estimate-s)/s
    print " d ", d, " s ", s, " error ", error
```