Práctica 1. Extra: Fractales de Sierpinski y dimensión de Hausdorff

Guillermo Martín Sánchez

27/02/20

1 Introducción

En esta práctica vamos a generar dos conjuntos fractales: el Triángulo de Sierpinski y la Alfombra de Sierpinski. Vamos a estimar también un rango donde se encuentra su dimensión de Hausdorff.

2 Material usado

2.1 Ejercicio 1

Programa un código en python para obtener una muestra (de 'resolución numérica' arbitraria) de la Alfombra de Sierpinski.

Hemos usado una técnica para obtener un número arbitrario n de puntos dentro de un fractal llamada en inglés *chaos game*. La técnica consiste en, dado un punto del fractal x_0, repetir un proceso iterativo para obtener tantos puntos como queramos dentro de este.
Para el Triángulo de Sierpinski, hay que elegir tres puntos del plano que formen un triángulo equilátero y un punto x_0 perteneciente al conjunto. En nuestro caso hemos elegido los vértices $p_0 = (0,0)$, $p_1 = (2,0)$ y $p_2 = (1,\sqrt{3})$; y $x_0 = (1,0)$. A partir de ahí, iteramos n veces el siguiente proceso:

- *Paso 1:* Elegir aleatoriamente un vértice p_0, p_1 o p_2, al que llamamos p
- *Paso 2:* $x_{i+1} = \frac{1}{2}(x_i + p)$

Se puede demostrar que este método devuelve n puntos pertenecientes al Triángulo de Sierpinski.

De manera análoga, para la Alfombra de Sierpinski, hay que elegir ocho puntos del plano que formen un cuadrado (4 vértices y los puntos medios de cada lado) y un punto x_0 perteneciente al conjunto. En nuestro caso hemos elegido los puntos $p_0 = (0,0)$, $p_1 = (1,0)$, $p_2 = (2,0)$, $p_3 = (2,1)$, $p_4 = (2,2)$,

$p_5 = (1,2)$, $p_6 = (0,2)$ y $p_7 = (0,1)$; y $x_0 = (\frac{1}{3}, 0)$. A partir de ahí, iteramos n veces el siguiente proceso:

- *Paso 1:* Elegir aleatoriamente un punto p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7 o p_8 al que llamamos p
- *Paso 2:* $x_{i+1} = \frac{1}{3}(x_i + p)$

Se puede demostrar que este método devuelve n puntos pertenecientes a la Alfombra de Sierpinski. Sin embargo, estos suelen estar focalizados en unos de las ocho copias que forman el conjunto. Como el fractal es autosimilar, basta descartar los puntos que queden fuera.

2.2 Ejercicio 2

A continuación, de acuerdo con la medida de Hausdourff (con refinamiento de recubrimientos), programa un código en python para obtener la dimensión de Hausdorff de la muestra anterior para la Alfombra de Sierpinski, y comprueba que tiende a log 8 /log 3.

La definición de Medida de Hausdorff para cierta dimensión d es, a partir de todos los posibles recubrimientos del conjunto por $\delta - balones$ ($\{U_i\}_{i\in I}$):

$$H^d(E) := \lim_{\delta\to 0}\left(\inf_k\left\{\sum_i ||U_i||^d\right\}_k\right) \tag{1}$$

Este límite puede ser finito o infinito. Denominamos $d = \sup\{d_0 \geq 0 : H^{d_0}(E) = \infty\} = \inf\{d_0 \geq 0 : H^{d_0}(E) = 0\}$. De hecho, se cumple $H^{d'}(E) = 0 \quad \forall d' > d$ y $H^{d'}(E) = \infty \quad \forall d' < d$.

Vamos a usar este último hecho para estimar el valor de d. Para ello, hacemos un *grid* del plano, dividiéndolo en n^2 cuadrados de cada vez menor área (para intentar estimar el límite). Estos cuadrados van a ser nuestra aproximación a los $\delta - balones$ de la definición. Calculamos cuantos k de ellos contienen puntos y por tanto pertenecen al recubrimiento (los denominamos $\{C_i\}_{i=1}^k$). Entonces, si la longitud de un lado de estos cuadrados es $\epsilon = \frac{2}{n}$, tenemos que $||C_i|| = \sqrt{2}\epsilon \quad \forall i$. De esta manera, para una d dada aproximamos:

$$H^d_\delta(E) := \inf_k\left\{\sum_i ||U_i||^d\right\}_k \qquad 0 < ||U_i|| < \delta \tag{2}$$

por

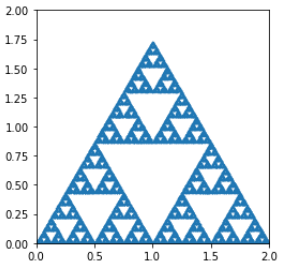
$$H^d_\epsilon(E) := \sum_{i=1}^k ||C_i||^d \quad ||C_i|| = \sqrt{2}\epsilon \tag{3}$$

que a su vez, como $||C_i|| = \sqrt{2}\epsilon \; \forall i$ podemos escribir como $H^d_\epsilon(E) = k(\sqrt{2}\epsilon)^d$. De esta manera, estudiamos $\lim_{\delta\to 0} H^d_\delta(E)$ mediante la aproximación $\lim_{\epsilon\to 0} H^d_\epsilon(E)$. Buscamos pues un d tal que $H^{d'}_\epsilon(E) = 0 \quad \forall d' > d$ y $H^{d'}_\epsilon(E) = \infty \quad \forall d' < d$

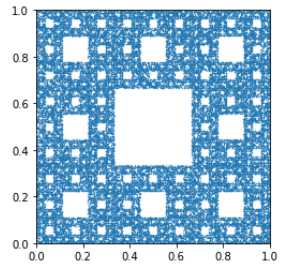
3 Resultados y discusión

3.1 Ejercicio 1

Representamos los conjuntos en las siguientes figuras:



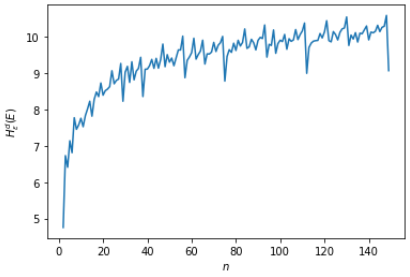
Triángulo con $n = 50000$ puntos



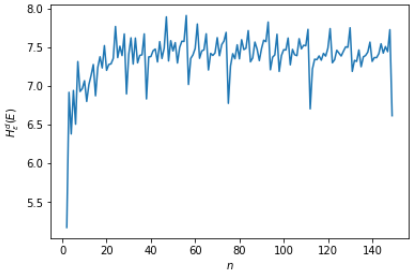
Alfombra con $n = 50000$ puntos

3.2 Ejercicio 2

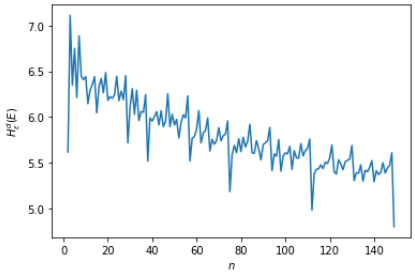
Sabemos que la dimensión exacta del Triángulo de Sierpinski es $d = \frac{log(3)}{log(2)} \approx 1.58$ y la de la dimensión exacta de la Alfombra de Sierpinski es $d = \frac{log(8)}{log(3)} \approx 1.89$. Como podemos ver nuestro método correctamente estima estos valores ya que para valores más bajos de d vemos que el límite tiende a ∞ y para valores más altos a 0. Hay que decir sin embargo que esto se ve más claro en el caso de la Alfombra que en la del Triángulo.



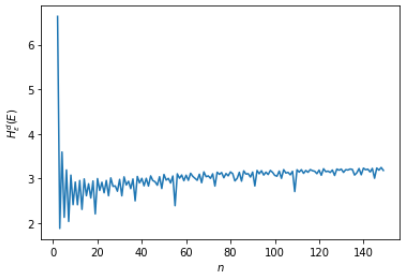
Triángulo con $d = 1.50$



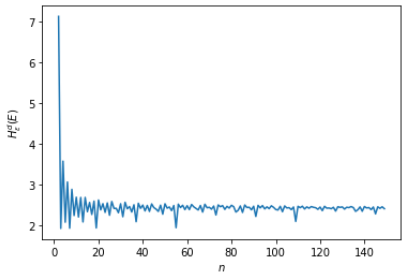
Triángulo con $d = 1.58$



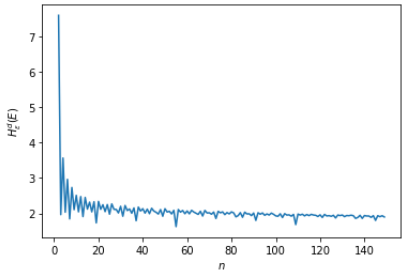
Triángulo con $d = 1.66$



Alfombra con $d = 1.82$



Alfombra con $d = 1.89$



Alfombra con $d = 1.95$

4 Conclusión

Aproximaciones numéricas nos permiten estudiar propiedades de naturaleza no finita como puede ser la dimensión de Hausdorff de conjuntos fractales. La dimensión de Hausdorff es una generalización útil del concepto de dimensión para estudiar y clasificar conjuntos extraños como aquellos que tienen perímetro infinito y área nula como muchos fractales del plano, para los cuales las definiciones de dimensiones enteras no son informativas.

5 Código

```
# -*- coding: utf-8 -*-
"""
Extra: Sierpinski Triangle
"""

import numpy as np
import matplotlib.pyplot as plt

# Generate n points in Sierpinski Triangle
def generate_triangle(n):
    p0 = [0,0]
    p1 = [2,0]
```

```
    p2 = [1,np.sqrt(3)]
    p = np.array([p0, p1, p2])
    vi = 1/2 * (p[0] + p[1])

    varr = np.array([vi])
    for i in range(n):
        r = np.random.randint(3)
        vi = 1/2 * (vi + p[r])
        varr = np.append(varr,[vi], axis=0)

    return varr

# Generate n points in Sierpinski Carpet
def generate_carpet(n):
    p0 = [0,0]
    p1 = [1,0]
    p2 = [2,0]
    p3 = [2,1]
    p4 = [2,2]
    p5 = [1,2]
    p6 = [0,2]
    p7 = [0,1]
    p = np.array([p0, p1, p2, p3, p4, p5, p6, p7])
    vi = 1/3 * (p[0] + p[1])

    varr = np.array([vi])
    for i in range(n):
        r = np.random.randint(8)
        vi = 1/3 * (vi + p[r])
        varr = np.append(varr,[vi], axis=0)

    return varr

# Compute for some d and n the estimation of the Hausdorff d-Volume for
# a cover of epsilon = 2/(n-1) squares
def volume_d(set, n, d):
    x = np.linspace(0,2,n)
    eps = 2/(n-1)
    count = 0
    for i in range(n-1):
        inx = (x[i] < set[:,0]) & (x[i+1] > set[:,0])
        for j in range(n-1):
            iny = (x[j] < set[:,1]) & (x[j+1] > set[:,1])
            filled = np.any(inx & iny)
            count = count + filled
    return count*((np.sqrt(2)*eps)**d)
```

```
#%%
"""
Generate Sierpinski Triangle
"""
set = generate_triangle(50000)
plt.scatter(set[:,0],set[:,1], s=0.1)
plt.axis([0, 2, 0, 2]);
plt.gca().set_aspect('equal', adjustable='box')

#%%
"""
Generate Sierpinski Carpet
"""

set = generate_carpet(50000)
plt.figure()
plt.scatter(set[:,0],set[:,1], s=0.1)
plt.axis([0, 1, 0, 1]);
plt.gca().set_aspect('equal', adjustable='box')

#%%
"""
Compute Sierpinski's Carpet Hausdorff d-Volume
"""
set = generate_carpet(50000)
narr = range(2,150)

d = 1.95
Hd = []
for n in narr:
    Hd.append(volume_d(set,n,d))

plt.figure()
plt.plot(narr,Hd)
plt.xlabel('$n$')
plt.ylabel('$H^{d}_{\epsilon}(E)$')

#%%
"""
Compute Sierpinski's Triangle Hausdorff d-Volume
"""
set = generate_triangle(50000)
narr = range(2,150)

d = 1.66
```

```
Hd = []
for n in narr:
    Hd.append(volume_d(set,n,d))

plt.figure()
plt.plot(narr,Hd)
plt.xlabel('$n$')
plt.ylabel('$H^{d}_{\epsilon}(E)$')
```