Coding Example: Minkowski-Bouligand Dimension

This problem is an extension of the previous case study. In this lesson, we'll learn how to find the fractal dimension of the Mandelbrot set.

> **Note:** You should look at the ufunc.reduceat method that performs a (local) reduce with specified slices over a single axis.

Problem Description

We now want to measure the fractal dimension of the Mandelbrot set using the Minkowski–Bouligand dimension. To do that, we need to do box-counting with a decreasing box size (see figure below). As you can imagine, we cannot use pure Python because it would be way too slow. The goal of the exercise is to write a function using NumPy that takes a two-dimensional float array and returns the fractal dimension. We'll consider values in the array to be normalized (i.e. all values are between 0 and 1).

Given below is the Minkowski-Bouligand dimension:

The Minkowski–Bouligand dimension of the Great Britain coastlines is approximately 1.24.

Complete Solution

Here's the detailed solution to find the fractal dimension:

```
# -----------------------------------------------------------------------------
# From Numpy to Python
# Copyright (2017) Nicolas P. Rougier - BSD license
# More information at https://github.com/rougier/numpy-book
# -----------------------------------------------------------------------------
import numpy as np

#calculate the fractal dimensions
def fractal_dimension(Z, threshold=0.9):
    def boxcount(Z, k):
        S = np.add.reduceat(
            np.add.reduceat(Z, np.arange(0, Z.shape[0], k), axis=0),
                               np.arange(0, Z.shape[1], k), axis=1)
        return len(np.where((S > 0) & (S < k*k))[0])
    Z = (Z < threshold)
    p = min(Z.shape)
    n = 2**np.floor(np.log(p)/np.log(2))
    n = int(np.log(n)/np.log(2))
    sizes = 2**np.arange(n, 1, -1)
    counts = []
    for size in sizes:
        counts.append(boxcount(Z, size))
    coeffs = np.polyfit(np.log(sizes), np.log(counts), 1)
    return -coeffs[0]


if __name__ == '__main__':
    from scipy import misc
    import matplotlib.pyplot as plt
    import matplotlib.patches as patches
```

```
Z = 1.0 - misc.imread("GreatBritain.png")/255


print(fractal_dimension(Z, threshold=0.25))

sizes = 128, 64, 32
xmin, xmax = 0, Z.shape[1]
ymin, ymax = 0, Z.shape[0]
fig = plt.figure(figsize=(10, 5))

for i, size in enumerate(sizes):
    ax = plt.subplot(1, len(sizes), i+1, frameon=False)
    ax.imshow(1-Z, plt.cm.gray, interpolation="bicubic", vmin=0, vmax=1,
              extent=[xmin, xmax, ymin, ymax], origin="upper")
    ax.set_xticks([])
    ax.set_yticks([])
    for y in range(Z.shape[0]//size+1):
        for x in range(Z.shape[1]//size+1):
            s = (Z[y*size:(y+1)*size, x*size:(x+1)*size] > 0.25).sum()
            if s > 0 and s < size*size:
                rect = patches.Rectangle(
                    (x*size, Z.shape[0]-1-(y+1)*size),
                    width=size, height=size,
                    linewidth=.5, edgecolor='.25',
                    facecolor='.75', alpha=.5)
                ax.add_patch(rect)

plt.tight_layout()
plt.savefig("output/fractal-dimension.png")
plt.show()
```

Further Readings

- How To Quickly Compute the Mandelbrot Set in Python, Jean Francois Puget, 2015.
- My Christmas Gift: Mandelbrot Set Computation In Python, Jean Francois Puget, 2015.
- Fast fractals with Python and NumPy, Dan Goodman, 2009.
- Renormalizing the Mandelbrot Escape, Linas Vepstas, 1997.

Now that we have learned Temporal Vectorization, let’s look at “Spatial Vectorization” in the next lesson.