Fractal Compression for Dataset Complexity

Fractals and Complexity Measures

Akshay Gadre, Shijin Rajakrishnan

Memory Based Reasoning in AI, IIT Madras, Team 12

Abstract. Data compression is an extremely valuable tool used to reduce resource usage, such as storage and/or transmission capacities. Compression can be either lossy or lossless, and in this project we explore a lossy method of compression using Fractals. The basic idea behind this is to utilize the self-similarities often present in images, and exploit it to reduce redundancies.
We further extend the idea of compression to other domains and try to check if the self-similarity present within datasets can be adopted as a suitable measure of complexity, for example, in clustering tasks.
Clustering is a commonly used technique to classify large data sets. Multiple techniques of clustering are asymptotically exponential with increase in dimensionality of the data set. Fractal clustering is a technique whose complexity depends on the self-similarity of the data set instead of the dimensions of this data set. This study concretely tries to establish a measure of self-similarity of a data set and measure the relationship of the self similarity to the fractal dimensions.

1 Introduction

1.1 Background

Clustering of large data sets is a famous knowledge discovery technique. However many of the published algorithms fail to give asymptotically polynomial behaviour with respect to the dimensionality of dataset when it comes to clustering. In 2003, D. Barbara and P. Chen[1] conceptualized a Fractal Clustering algorithm which uses the self similarity within the data to cluster it. They use a concept called fractal dimension of the data set to measure the self similarity of the data.
There have been multiple image compression techniques developed to ensure better use of storage space and fast transfer of images. Some of the most famous ones like JPEG and GIF are members of a class of image compression techniques known as lossy image compression. Y Fisher(1992)[2] and M Barnsley(1994)[3] developed another lossy image compression technique called the Fractal Image Compression which uses the self-similarity within the image to store affine contractive maps in a storage file which can be decompressed to reconstruct the image to the accuracy at which it was stored. It uses the intrinsic self similarity property of fractals of convergence to a fixed point(attractor) irrespective of the starting image and hence the name.

1.2 Our approach

We first evaluate the viability and usefulness of the fractal compression technique by running it on several images, and observing the output image(which will be lossy).
Then we extend the idea to try to use the self similarity of the image(identified by the losses created during the compression/decompression), and correlate it with the complexity of datasets, and hence try to observe a relation between how complex the dataset is based on the self similarity property existing within that dataset.
We use binary matrices which have been stacked, and then treat it as a black-white image and compress it and then subsequently decompress it, and observe the errors incurred, which will be a measure of the self-similarity existing within that matrix.

2 Fractals and Iterated Function Systems

2.1 Fractals

Fractals are self replicating never ending iterative dynamic systems. Many naturally occurring shapes can be easily described using fractal geometry than possible with classical geometry. Fractals are constructed by repeatedly applying a transformation on a starting structure. Some of the most famous examples of fractals are the Koch snowflake, Sierpinski's triangle and Sierpinski's pyramid.
One of the most important features of fractal is that it is a contractive map. It means that irrespective of what structure you start with, if you iterate the contractive transformations defined to construct the fractal, you will end up with the same final fractal. This property will be critically exploited during the implementation of Fractal Image Compression algorithm[2][3].
Fractal dimension is a measure of how a detail in the pattern changes with scale that it is measured. The term was coined by B. Mandlebrot in 1975. Hausdorff dimension is also famously considered as one of the best approximators to the Fractal dimension of a structure. Hausdorff it as

$$D = \frac{\log N}{\log r}$$

where N is the is the no. of segments of the fractal after the iterative map and r is the ratio at which the initial segment was cut. [4].

2.2 Iterated Function Systems

Iterated Function System is the set of mappings developed by repeated application of contractive maps on any starting structure on a complete metric space. They are the primary method of developing fractals in mathematics.
As explained in the previous section, the final structures are always self similar.

Hutchinson(1981)[5] showed that the above systems result in a unique, closed and bounded set of mappings for a given metric space $\mathbb{R}^n$. Note for the uniqueness of the attractor, the tranformations have to be affine and contractive.
Using this information, Y Fisher(1992)[2] and M Barnsley(1994)[3] developed an algorithm for storing the contractive affine maps of an image in special file and then using that file when required for reconstruction of the image by repeated application of the transformations on a random starting image to retrieve the final image.

2.3 Contractive Mapping Fixed Point Theorem and Collage's theorem

A map/transformation is contractive iff on iterative application on any starting structure, we converge to a unique final structure.
The contractive mapping fixed point theorem states that for a transformation to be contractive, the distance between any two points in the metric space of the structure should reduce after the application of transform. The space should be metric and complete for an attractor to exist.
The Collage's theorem extends the above theorem to images stating that if we develop a set of contractive transforms for an image on itself, the application of these set of transforms on any starting image will converge to the image we initially found the transforms for.
The clarity of the image will depend upon the no. of iterations the decompression algorithm is run for and the clarity at which the initial contractive mappings were stored.

3 Fractal Image Compression and Decompression

Data: Binary Portable GrayMap M
Result: A set of contractive transformations
−dimensionOfRangeBlock=X;
−dimensionOfDomainBlock=Y(Y>X for contractive map);
−Store the above dimensions in the output file(to be used for decompression);
−Downsample the domain blocks to the size of range blocks;
forall the *RangeBlocks* **do**
  −Find the domain block and the transformation(out of a set of 24 predefined transformations) which leads to the minimum error;
  −Store the domain block and the transformation as the transformation for that range block.
end

Algorithm 1: Fractal Image Compression

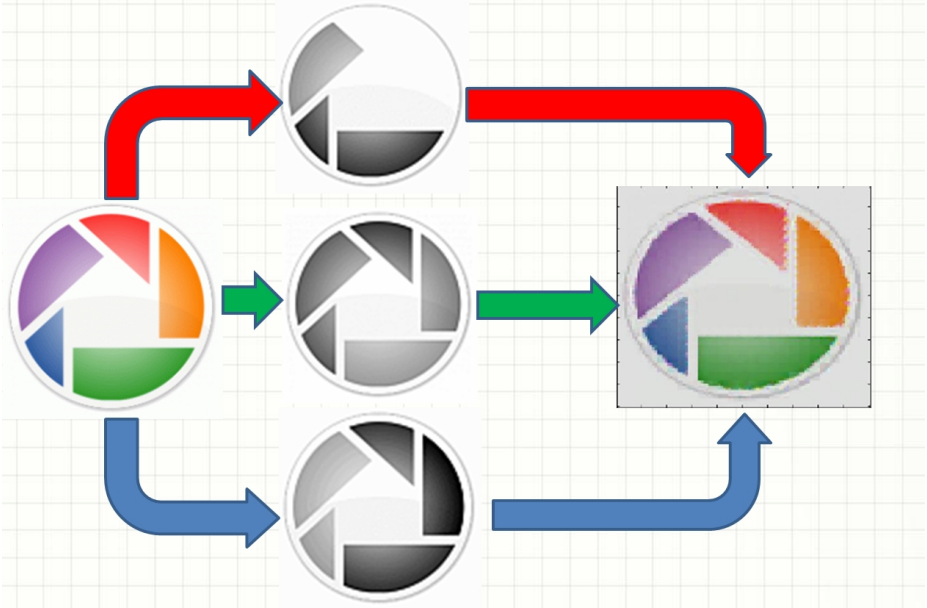
The above is the fractal compression algorithm abstracted out of Y Fisher(1992)[2] and M Barnsley(1994)[3] papers that we will be using for the rest of our results.

It takes a square portable graymap as input and outputs a set of contractive transformations. Obviously the above is a minimalistic algorithm customised for our purpose

Data: −A set of contractive transformations
− Dimensions of range and domain blocks used during compression
Result: A grayscale image
−dimensionOfFinalImage=X;
−Create an image of size X*X with each pixel randomly assigned from 0-255;
for *# of predetermined iterations* **do**
  −Find the domain block size in the new sized image such that the # of domain blocks remains the same;
  −Downsample it by the ratio of $\frac{dimension(RangeBlocks)}{dimension(DomainBlocks)}$;
  − Apply the contractive transformation stored and copy the result to a new image(which will become the initial image for the next iteration)
end
− Output the new image at the end of the last iteration as the result image

Algorithm 2: Fractal Image Decompression

Fig. 1. Extending the above algorithm to compress and decompress colored images



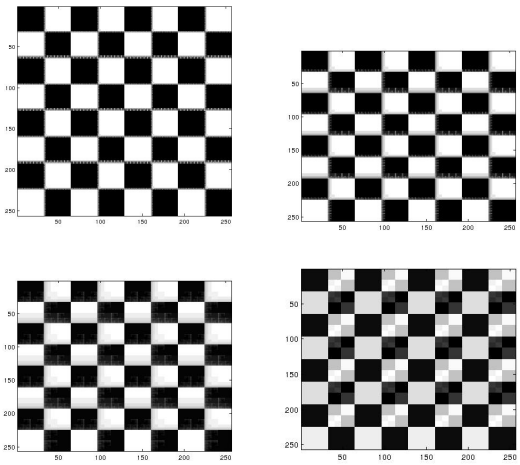


Fig. 2. The outputs for checkered board of 8 × 8 for range block sizes 4 pixels, 8 pixels, 16 pixels and 32 pixels respectively

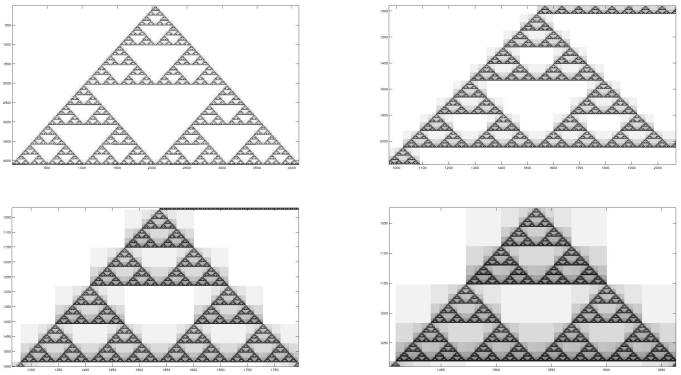


Fig. 3. The output for Sierpinski's triangle for range block size 4 pixels, the observations on zooming in on each particular 1 out of the three big triangles

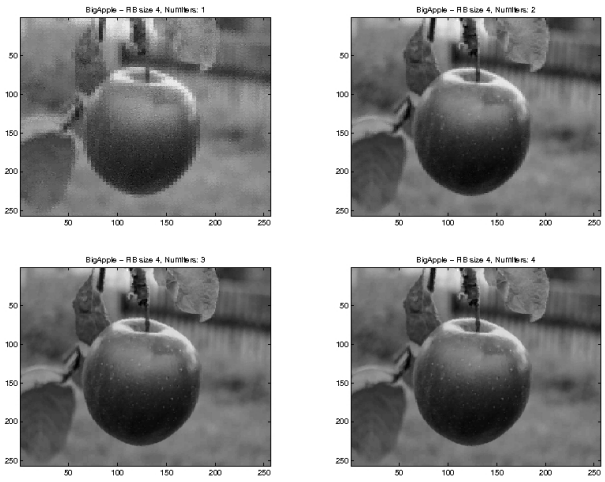


Fig. 4. The output for an apple's image with range block size 4

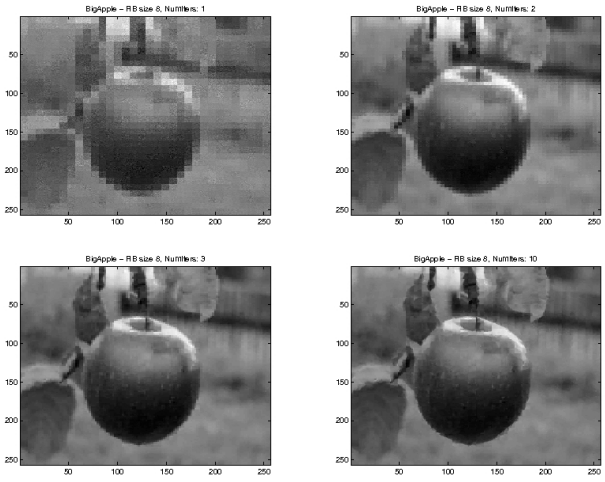


Fig. 5. The output for an apple's image with range block size 8

4 Stacked Datasets

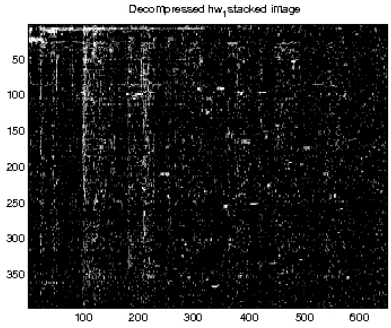
In the next section, we run the image compression algorithm on datasets by interpreting the 0-1 feature-attribute matrix as a black and white image.
Stacking(row wise) is done by fixing a first row and then successively finding rows that are similar to the fixed rows and adding the most similar row to the set of fixed rows. A weighted version is used where the similarity to the last added row is given more weight. Similarly, the columns are stacked by looking at the similarities across the columns.
Given the dataset, stacking of the rows and columns is first done to collect together local clusters of similar regions so as to facilitate identification of self-similar parts.

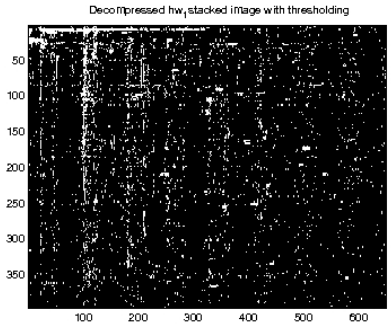
5 Results

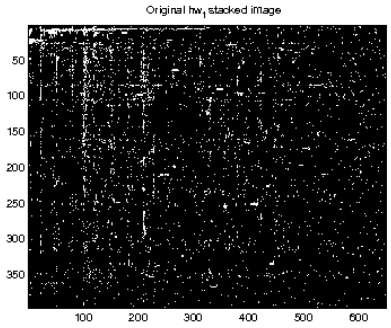
We ran the experiment range block size = 2 pixel×2 pixel. We use a custom column stacked datasets which are 0-1 matrices. We then implement our compression and decompression algorithm on this image after adding padding bits of 0 to make it a square matrix as required by our algorithm mentioned above. The results are then thresholded at the value such that the output error is minimised. One of the interesting observations was that the thresholds occured exactly just above 0.5 or just below for all of the datasets(may be conincidence).

Dataset	**Dimension of the stacked data**	**Errors**	**Error Percentage**
HW1	396×650	9864	3.83%
Lemm1	578×1000	23689	4.13%
Relpol1	400×1000	13168	3.29%

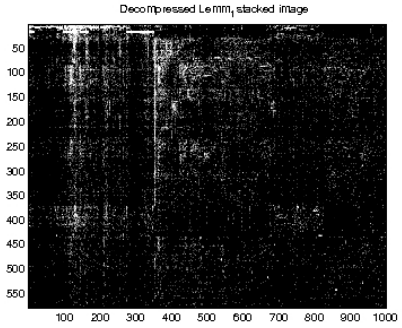
5.1 HW1 dataset

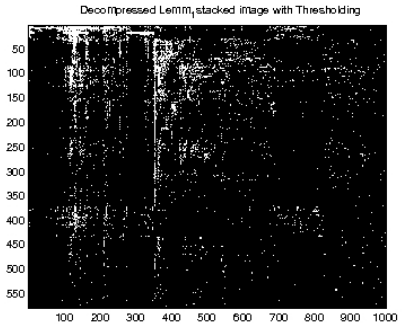


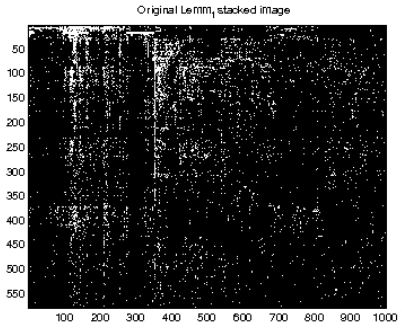




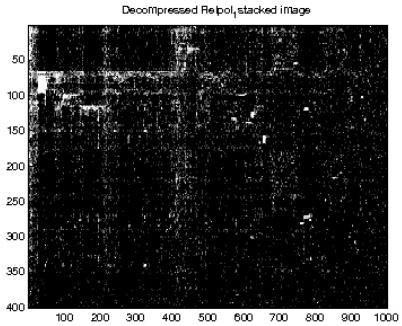
5.2 Lemm1 dataset

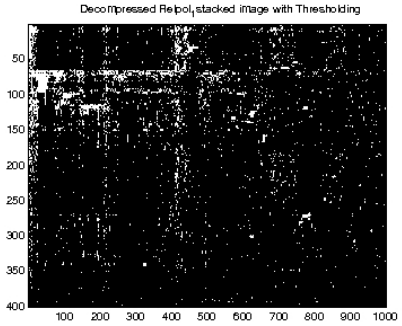


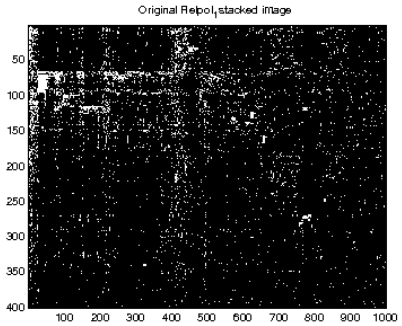




5.3 Relpol1 dataset







6 Conclusion and Future Work

For the above datasets, we can conclude that fractal image compression and decompression only induces 4% error into the original dataset after thresholding at a custom threshold.
Also as we can observe from our testing of fractal compression and decompression, the original algorithm by Y Fisher(1992)[2] and M Barnsley(1994)[3] can be extended for color images by passing each color as a grayscale map and then reconstructing it as an union of the resulting maps.
Particularly the results of checkered board and Sierpinski's triangle are really important as we know they are highly self-similar images, As you can observe, even on zooming in we can see that the core part of the Sierpinski's triangle looks similar to the original image.
The checkered board as expected gave the best results since we were considering square blocks for self similarity
We will be continuing this project by finding clusters in the resulting stacked data and the original stacked data and finding out how many of the members change clusters after the clustering and this will be a good measure of self similarity of data set. This we will be comparing with the existing measures of self similarity to see if any correlation can be found.

7 Acknowledgments

The authors would like to thank KVS Dileep for his continual and valuable inputs for the project. We would also like to thank the TAs Abhijit and Shubranshu for their insights in the project. We would also like to thank Prof. Sutanu Chakraborti for helping us direct this project from a research perspective.

References

1. Using fractal simesions to cluster datasets(2003) ftp://ftp10.us.freebsd.org/users/azhang/disc/disc01/cd1/out/papers/kdd/p260-barbara.pdf, Daniel Barbara and Ping Chen

2. Fractal Image Compression , http://isites.harvard.edu/fs/docs/icb.topic1386881.files/Imaging%20and%20Inverse%20Problems/1992_Fisher.pdf, Yuval Fisher

3. Fractal Image Compression , https://karczmarczuk.users.greyc.fr/matrs/Dess/RADI/Refs/barnsley.pdf ,Michael F. Barnsley

4. Fractals and Geometry of Nature , http://users.math.yale.edu/ bbm3/web_pdfs/encyclopediaBritannica.pdf, Benoit Mandlebrot

5. J. E. Hutchinson, Fractals and self-similarity , Indiana Univ. Math. J. 30(1981) 713747