FRACTAL CODING USING NEURAL NETWORKS

Rashad A. Al-Jawfi (PhD)
Department of Mathematics , Faculty of Science, Ibb University, Yemen
algofi@yemen.net.ye

Abstract: In this paper we attempt to form a neural network to coding fractal image. Our approach to this problem consists of finding an error function which will be minimized when the network coded attractor is equal to the desired attractor. First, we start with a given iterated function system attractor; with a random set of weights of the network. Second, we compare the consequent images using this neural network with the original image. On the basis of the result of this comparison we can update the weight functions and the code of the iterated function system (IFS). A common metric or error function used to compare between the two image fractal attractors is the Hausdorff distance. The error function gives us good means to measure the difference between the two images. The distance is calculated by finding the farthest point on each set relative to the other set and returning the maximum of these two distances.

Key words: image coding, fractal, neural networks, inverse problem

1. Introduction

The basic concept of fractals is that they contain a large degree of self-similarity. This means that fractals usually contain little copies of themselves, buried deep within the original. On the other hand, neural networks have been hailed as the paradigm of choice for problems which require "Human Like" perception. A network could be performing its function perfectly, responding correctly to every input that is given. However, its internal workings could still be construed as a black box, leaving its user without knowledge of what is happening internally.

In this paper, we are interested in different ways to tease neural networks open, to analyze what they are representing and how they are "thinking". In particular, we present a novel algorithm to introduce the code of the iterated function system, which generates a fractal image. Its feature being that it is exactly fully describing a network's function, concisely, not an incremental collection of approximations and direct mapping of a network's input to its output.

2. The inverse problem of fractals

2.1 Theory and example

In several mathematical fields, many problems have inverses: for example, integration, in a certain sense, is an inverse problem for differentiation; the problem of determining the forces under the action of which a particle moves along a given curve is another example of an inverse problem in dynamics of particles, etc. In an analogous way, the problem of generating fractals by the use of IFS calls for an inverse problem, namely: For a given set in R^n, construct a suitable IFS whose attractor is the given set (to a certain desired degree of accuracy)[1].

The tackling of this inverse problem, as it stands, is difficult, if it is not impossible. However, if the given set is self-similar, then the required construction is almost straightforward. The IFS can be found easily by making mathematical translation of the property of self-similarity.

For determining the mathematical translation of self-similarity, let us consider the adjoining figure.

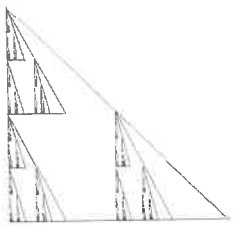


Fig. 2.1

It is clear that, the big triangle has been divided into three congruent reduced triangles, of unequal sides. The process of adjunction of the small triangles is shown in the fig. 2.2.

For $i = 1,2,3$, f_i are of the form

$$F_i\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix}.$$

Each of them is defined by transforming the points, A into A', B into B' and C into C'.

Suppose $A = (0,0), B = (1,0), C = (0,1)$ also $A' = (0,0)$, $B' = (.25,0)$, $C' = (0,.5)$ for the first small triangle and $A' = (0.5,0)$,

$B'=(0.75,0)$, $C'=(0.75,.5)$ for the second and $A'=(0,0.5)$, $B'=(.25,0.5)$, $C'=(0,1)$ for the third. Then, for evaluating the map F_1, we have to solve the following two linear systems:

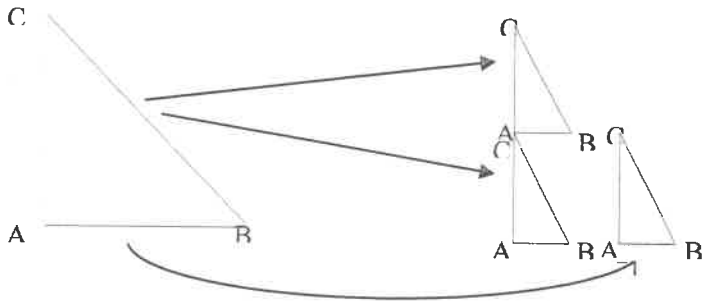


Fig. 2.2

$$\begin{matrix} 0a+0b+e=0 \\ 1a+0b+e=.25 \\ 0a+1b+e=0 \end{matrix} \quad \text{and} \quad \begin{matrix} 0c+0d+f=0 \\ 1c+0d+f=0 \\ 0c+1d+f=.5 \end{matrix}$$

These equalities imply

$a_1=.25$, $b_1=0$, $e_1=0$, $c_1=0$, $d_1=.5$ and $f_1=0$.

Therefore, $F_1\begin{bmatrix} x \\ y \end{bmatrix}=\begin{bmatrix} .25 & 0 \\ 0 & .5 \end{bmatrix}\cdot\begin{bmatrix} x \\ y \end{bmatrix}+\begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Similarly, we can evaluate the coefficient of F_2 and F_3.

2.2 (Collage theorem) [2]

Let (X,d) be a complete metric space, let $H=\bigcup_{m=1}^{n} F_m$ be an IFS with contraction factor r, and fixed point T_0. Let T be a closed subset of X. Let $\varepsilon>0$ be any positive number, and suppose that the $\{f_i\}$ are chosen such as

$d(T,H(T))<\varepsilon$

Then

3. Attractor Coding $\forall x \in X$

3.1 Design of neural network

The Hopfield network uses the fixed points of the network dynamics to represent memory elements. Networks studied by Pollack [8], and Giles [5] use the current activation of the network as a state in a state machine while using the dynamics of the network which is treated as an Iterated Function system that is coding for its fractal attractor[2]. Melnik [6] applies one of the transforms on a random point for a number of steps until it converges.

There is still no general algorithm for fractals image coding, the problem we want to solve in this paper, which is given a fractal attractor, is to find a set of weights for the neural network which will approximate the attractor.

A neural network consists of two input units and two output units for all transforms (IFS), representing scalar function.

. The transform is selected randomly, and all input neurons receive a coordinate of each point of fractal image, one neuron for x coordinate and the other for y coordinate for each transform. And return as x and y output, consists of sigmoid function [6,7] with a bias.

The equations of x and y output are as fellows:

$$Y_{out} = (1 + e^{-WY_{in}})^{-1}$$

where

$$WX_{in} = w_{xx} \cdot X + w_{xy} \cdot Y + w_x$$

and

$$WY_{in} = w_{yx} \cdot X + w_{yy} \cdot Y + w_y$$

where w_{ij} is the weight function from i input to j output neuron and w_i is the bias of i input neuron.

At the end of this operation for a large number of points with random iterations, we get an image. Of course, this image is different in general with the image we want to find the iterated function system (IFS) of it.

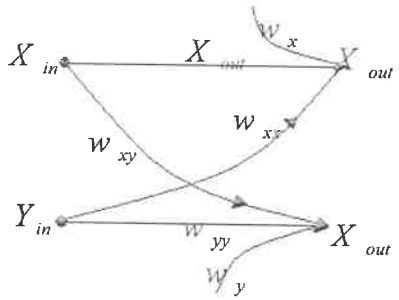


Fig. 3.1

Then, we must update the weight functions of the neural network w_{ij} to get better approximation to the target image.

This change of the weight function is depending with the measure of the difference between the two images. This difference is known as error function, which must minimize with every update of weight functions.

The error function, used to compare fractals attractors, is the hausdorff distance[2,4,5].

The distance between the two images A and B is calculated as follows: We first calculate the distance between the element $a \in A$ and the set B, which is the smallest distance between a and each element $b \in B$.

$$d(a,B) = \min\{\| b - a \|; b \in B\}$$

Then the distance between A and B is the largest distance for each element $a \in A$.

$$d(A,B) = \max\{ d(a,B); a \in A\}$$

Also the distance between B and A is equal to

$$d(B,A) = \max\{ d(b,A); b \in B\}.$$

And then the distance between the two sets is the largest of the two distances $d(A,B)$ and $d(B,A)$:

$$H(A,B) = \max\{d(A,B), d(B,A)\}.$$

Our error function is defined as:

$$E(T, A) = \sum_i d(T_i(x, y), A) + \sum_{x,y} d((x, y), T(A))$$

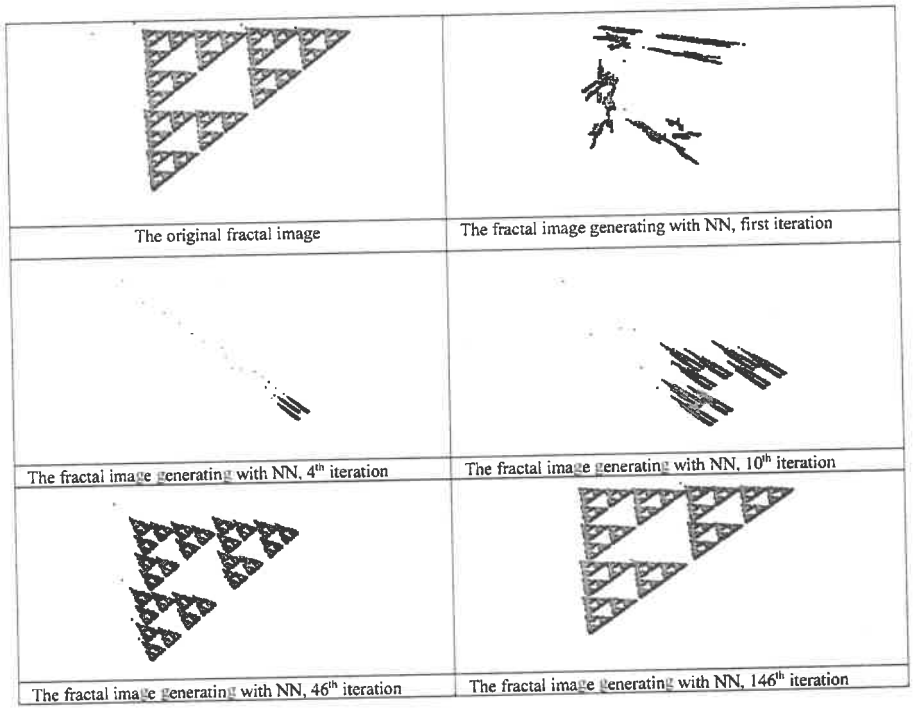
where $T_i(x, y)$ is the image of the point (x, y) with respect to the transform T_i. And $T(A)$ is the all image of A.

The value of the error function with respect to the iteration of neural network is shown in table (3.1).

Table (3.1) The values of error function

iteration	E	Iteration	E
1	1.30555555389987	120	0.348059498735117
18	0.898381268305497	135	0.347548076005266
39	0.464376684618578	145	0.344510802955861
50	0.459844044755687		

And fig (3.2) shows the original image and some images generating with the neural network for some iterations.



The original fractal image

The fractal image generating with NN, first iteration

The fractal image generating with NN, 4th iteration

The fractal image generating with NN, 10th iteration

The fractal image generating with NN, 46th iteration

The fractal image generating with NN, 146th iteration

Fig (3.2)

3.2 The procedure of fractals image coding

1. Input: fractal image, random weights, n the number of (IFS), ε and μ.
2. Compute the error function $E(A, T(A))$.
3. If $E > \varepsilon$ then
   - Compute $\frac{\partial E}{\partial W_{ij}}$, $\Delta W_{ij} = -\mu * \frac{\partial E}{\partial W_{ij}}$
   - Update the weights $W_{ij} = W_{ij} + \Delta W_{ij}$
   - Go to 2
4. Else: stop

Future Work

This paper focused on the inverse problem of fractals with related to iterated function systems for 2-dimension linear fractals. Solving the inverse problem of 3-dimension fractals remains an open problem, as does the same problem of non-linear iterated function systems.

References

1. Barnsley, M. F., "Fractals Everywhere", Academic Press, New York, (1988).
2. Barnsley, M. F., Ervin, V.,Hardin, D., and Lancaster, J., "Solution of an inverse problem for fractals and other sets", Proceedings of the National Academy of Science 83, pp. 1975-1977, April (1986).
3. Bruno, F., "Solving the inverse problem for function/image approximation systems II. Algorithm and computations", Fractals ,vol.2 No.3, pp.335-346, (1994).
4. Fausett, L., "Fundamentals of Neural Networks", Pretice-Hall.Inc., (1994).
5. Giles, C.L., Miller, C.B., Chen, D., Chen, H.H., Sun, G.Z., and Lee, Y.C., "Learning and extracting finite state automata with second-order recurrent neural networks", Neural Computation, vol. 4, no. 3, pp. 393-405, (1992).
6. Hutchinson J.E., "Fractals And Self-similarity", Indiana University Mathematics Journal, vol. 30, No. 4, July-August, pp.713-747, (1981).
7. Melnik, O., "Representation of information in neural networks", Ph.D. thesis, Brandies University, (2000).
8. Pollack, J. B., "The induction of dynamical recognizers", Machine Learning, vol. 7, pp. 227-252, (1991).