Dynamics of Networked Nonlinear Dynamical Systems

by

Garimella Ramamurthy

Report No: IIT/TR/2016/-1

Centre for Security, Theory and Algorithms
International Institute of Information Technology
Hyderabad - 500 032, INDIA
February 2016
NETWORK THEORY PROJECT REPORTS-2016

IIIT HYDERABAD

Guide: Dr. Ram Murthy Garimella
Abstract—This paper introduces a novel idea of using a Real Verhulst model as an activation function of a typical neural network. We are modelling Verhulst network with different number of nodes 1, 2 and 3 with and without self feedback. We could observe that it can be used as activation function for neural networks and its behaviour is varying based on graph structure.

I. INTRODUCTION

Time series prediction refers to the process by which the future values of a system are predicted based on the information obtained from the past and current data points. At present, there are a lot of methods for time series prediction, from traditional statistical method such as ARMA (Auto Regressive Moving Average) model to artificial intelligence based approaches, the core of these models is to establish a prediction model. Neural Network (NN) based models are widely used as an artificial intelligence based approach, back propagation (BP) being the most widely used technique for updating the parameters of the model. BP neural network is the most used neural network at present. It has unique approximation ability and simple structure, and it is a neural network with good performance. The BP learning process works in small iterative steps, and the network produces some output based on the current state of its synaptic weights (initially, the output will be random). This output is compared to the known-good output, and a mean-squared error signal is calculated. The error value is then propagated backwards through the network, and small changes are made to the weights in each layer. The weight changes are calculated to reduce the error signal for the case in question. The whole process is repeated for each of the example cases, then back to the first case again, and so on. However, not only are the statistical models not as accurate as the neural network-based approaches for nonlinear problems, they may be too complex to be used in predicting future values of a time series. One major criticism about the BP model is that it demands a great deal of training data and its application was inhibited largely by the slow convergence rate and over-prolonged training time, primarily the results of inappropriate sample pre-processing for a large initial sample domain. On the other hand, it is well known that selecting the number of neurons in hidden layer is also an important and tough problem because it affects the overall performance of neural networks. However, there is still no definite theory to settle it out.

As the neural network, the large amount of data that can be used to provide information, but also increase the difficulties of dealing with these data. Verhulst theory is an interdisciplinary scientific area that was first introduced in late 1830s by Pierre-Francois Verhulst. He studied various model Verhulst proposed the following differential equation for the population $P(t)$ at time $t$:

$$\frac{dP}{dt} = rP \left(1 - \frac{P}{K}\right)$$

When the population $P(t)$ is small compared to the parameter $K$, we get the approximate equation:

$$\frac{dP}{dt} = rP$$

whose solution is

$$P(t) = P(0)e^{rt}$$

i.e. exponential growth. The growth rate decreases as $P(t)$ gets closer to $K$. It would even become negative if $P(t)$ could exceed $K$.

Dividing the equation by $P^2$ and setting $p=1/P$, we get

$$\frac{dp}{dt} = -rp + \frac{r}{K}$$

With

$$q = \frac{p1}{K}$$

we get

$$\frac{dq}{dt} = rq$$

and

$$q(t) = q(0)e^{rt} = \left(\frac{1}{P(0)\frac{1}{K}}\right)e^{rt}$$

So we can deduce $p(t)$ and $P(t)$. Finally we get after rearrangement

$$P(t) = \frac{P(0)e^{rt}}{1 + \frac{P(0)e^{rt}}{K}}$$
We could see these observations in graph 4(a) and 4(b). The total population increases progressively from $P(0)$ at time $t = 0$ to the limit $K$, which is reached only when $t \to \infty$. Without giving the values he used for the unknown parameters $r$ and $K$, Verhulst compared his result with data concerning the population of France between 1817 and 1831, of Belgium between 1815 and 1833, of the county of Essex in England between 1811 and 1831, and of Russia between 1796 and 1827. The fit turned out to be pretty good.

II. MODELLING VERHULST NETWORK

A. Network Structure

This is similar to a neural network. The equations for neural network are:

\[ \text{net}_i = \sum_{j=1}^{n} W_{ij} x_j[n] \]

where $n$ is the total number of inputs.

\[ x_i[n+1] = f_i(\text{net}_i) \]

For a Verhulst network,

\[ f_i(x) = a_i x(1 - x) \]

where $a_i \in (0, 4)$

Therefore,

\[ x_i[n+1] = a_i \left( \sum_{j=1}^{n} W_{ij} x_j \right) (1 - \sum_{j=1}^{n} W_{ij} x_j) \]

B. Assumptions

- There are three nodes (inputs/outputs) in the network.
- The weights between any two nodes are symmetric, i.e., $W_{ij} = W_{ji}$.
- We have used parallel mode of computation.

III. NUMERICAL EXPERIMENTS

A. Experiment 1

\[ x_1 = 0.7, x_2 = 0.5, x_3 = 0.4 \]

\[ W_{11} = W_{22} = W_{33} = 0 \]

(No feedback from $x_i[n]$ to $x_i[n+1]$)

\[ M = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0.5 & 0 \end{bmatrix} \]

$a_1$ is kept constant while $a_2$ and $a_3$ are varied from 2.8 to 4. The cycle length of the outputs for different values of $a_2$ and $a_3$ have been marked with different colors. $a_1$ is kept constant at 2.8, 3.2 and 3.6. The results are shown in figure 1(a), 1(b), 1(c) respectively.

B. Experiment 2

\[ x_1 = 0.7, x_2 = 0.5, x_3 = 0.4 \]

\[ W_{11} = W_{22} = W_{33} = 0.4 \]

(No feedback from $x_i[n]$ to $x_i[n+1]$)

\[ M = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{bmatrix} \]

A1 is kept constant while $a_2$ and $a_3$ are varied from 2.8 to 4. The cycle length of the outputs for different values of $a_2$ and $a_3$ have been marked with different colors. The results are shown in figure 2(a), 2(b), 2(c) respectively.

C. Experiment 3

Experiment 1 and Experiment 2 are repeated but with different

\[ x_1 = 0.4, x_2 = 0.4x_3 = 0.4 \]

. The results are shown for $a_1 = 2.8$ in figure 3(a), 3(b).

IV. RESULTS/OBSERVATIONS

The colour coding is as follows:

- cycle length 1 - green
- cycle length 2 - yellow
- cycle length 4 - orange
- cycle length 8 - red
- cycle length 16 - blue

The results for all experiments are shown in figures 1-3.

The boundary between points of different cycle length (denoted by different colours in the graph) is the region of bifurcation.

1) As $a_1$ is increased, the boundaries of bifurcation (change in cycle length) move closer to origin.
2) If we change $x_1$, $x_2$, $x_3$ without changing the weights or $a_i$'s, then there is no change in the bifurcation patterns.
3) A positive weight between $x_i$ and $x_j$ would necessarily mean that the cycles in $x_i$ and $x_j$ are the same because there is a feedback of $W_{ij}$ from $x_i[n]$ to $x_j[n+1]$ and a feedback of $W_{ij}$ from $x_i[n]$ to $x_i[n+1]$.
4) The introduction of self feedback straightens the boundaries of bifurcation.
V. CONCLUSIONS

We could see that initially there were repetitions, then cycles, with increase in size and then it goes on to chaos.

VI. ACKNOWLEDGMENT

We would like to thank our professor G. Ramamurthy to provide us opportunity to work on Real Verhulst Network and guiding us through out.

REFERENCES

SUMMARY:
The logistic map is a polynomial mapping or recurrence relation of degree 2. The chaotic behavior can arise from very simple non-linear dynamical equations. The logistic equation was first created by Pierre Francois Verhulst. Mathematically, the logistic map is written as:
\[ x(n + 1) = a x(n)(1 - x(n)) \]
where \( x(n) \) is a number between zero and one that represents the ratio of existing population to the maximum possible population. The values of interest for the parameter \( a \) are those in the interval \((0, 4]\). This nonlinear difference equation is intended to capture two effects:
- reproduction where the population will increase at a rate proportional to the current population when the population size is small.
- starvation (density-dependent mortality) where the growth rate will decrease at a rate proportional to the value obtained by taking the theoretical "carrying capacity" of the environment less the current population.

However, as a demographic model the logistic map has the pathological problem that some initial conditions and parameter values lead to negative population sizes.

In this project, we considered a neural network with three neurons:
- \( V = [V_1 \ V_2 \ V_3] \) define the state of the neurons 1, 2 and 3 respectively where \( V_i \in \mathbb{C} \).
- \( W \) is a 3x3 symmetric weight matrix where \( W_{ij} \in \mathbb{R} \).
- \( NET = W * V(n) \) where \( V(n) \) defines the state of the neural network at time \( n \).

The equations corresponding to the Verhulst network are as follows:
\[ V_j(n + 1) = a[(NET(n))(1 - NET(n))] \]
\[ NET(n) = W_{j1} V_1(n) + W_{j2} V_2(n) + W_{j3} V_3(n) \]

The network is called as Complex Verhulst Network when \( a \) takes complex value with its magnitude lying in a specific range.
Verhulst Network with three nodes:

NUMERICAL EXPERIMENTS:

Bifurcation diagram for real values of $a$:
RESULTS AND OBSERVATIONS:
Few of the plots from our experiments are shown below:
- Real inputs, Real weights
- Complex inputs, Real weights

- Complex inputs, Complex weights
• Complex inputs, Real weights, Complex A values - plot of X values

REFERENCES:
Modulo Neuron Dynamics

Abstract—In this research paper, a novel model of real valued neuron called modulo neuron is proposed. One main advantage is to increase the state space size without increasing the number of neurons. State space representation of a non-linear dynamical system associated with a linear congruential sequence is discussed briefly. Based on the periodicity of such sequence, it is inferred that the dynamical system exhibits cycles in the state space. The cycle length is determined and analysed for various values of associated primes.

Keywords—Modulo Neuron, Hopfield Network, State Space

I. INTRODUCTION

Researchers became interested in utilizing homogeneous, nonlinear dynamical systems to model a collection of neurons. One such effort resulted in the Hopfield neural network that acts as an associative memory.

Traditionally, artificial neural networks are based on the McCulloch-Pitts neuronal model. These are aptly suited for processing binary valued vectors. But there are many applications such as processing gray scale images where nonbinary neuronal state values are necessary and useful. Thus, in literature, neural networks based on multivalued neurons are proposed and studied.

II. MODULO NEURON

A. Mathematical Constructs of the Modulo Neuron

Generic Equation:

\[ y = (\sum_{j=1}^{N} w_j x_j - T) \mod p \]

More precisely, it can be stated as:

\[ v_i(n+1) = [(\sum_{j=1}^{N} M_{ij} v_j(n)) - T_i] \mod p \]

Where \( T_i \) is the threshold at the \( (i)_{th} \) neuron and \( M_{ij} \) is the \( (i,j)_{th} \) entry of synaptic weight matrix \( M \).

Here, \( V, M, T, p \) are bounded Integer values. \( P \) is usually taken to be a prime number.

Taking negative values into consideration, we can express the net as:

\[ y = f(Net) = \{(Net) \mod p' \} \text{ if } Net \geq 0 \]

\[ y = f(Net) = \{(-Net) \mod p' \} \text{ if } Net < 0 \]

B. Interpretations of Modulo Neuron Model

Each of the neurons is in one of the \( p \) possible states i.e. \( 0,1,2,..., p-1 \) where \( p \) could be a prime number. Also, the initial state value lies on the non-symmetric bounded lattice. Thus the state vector of all \( N \) neurons lies on the bounded (non-symmetric) lattice at any time.

Based on the set of states at which the above state updation takes place at any time, the neural network operates in the following modes:

1) Serial Mode: The state updation in takes place exactly at one of the nodes.
2) Fully Parallel Mode: The state updation in takes place at all the \( N \) nodes
3) Other Parallel Modes: The set of nodes at which the state updation in takes place is strictly larger than one and strictly smaller than \( N \).

There are certain distinguished states in the state space of the neural network (represented by \( (M, T) \) where \( M \) is the synaptic weight matrix and \( T \) is the threshold vector) called stable states.

C. COMPLEX MODULO NEURON

This modulo neuron network can also be used in analysis of models in the complex domain. In such a case the complex modulo neuron dynamics can be expressed as:

\[ y = (\sum_{j=1}^{N} w_j x_j - T) \mod p \]

\[ y = (\text{real part of net}) \mod p + j(\text{imaginary part of net}) \mod p \]

D. Linear Congruential Sequence

Interesting non-linear, discrete time sequences which are periodic were utilized for random number generation

\[ x(n+1) = (a x(n) + c) \mod m \]
Such a sequence is called a linear congruential sequence. It is easy to show that the sequence is always periodic. Most useful sequences are the ones with long period. Based on the theory of finite fields, it is well known that, when “m=p” is a prime number, multipliers $a_1, a_2, \ldots, a_k$ can be chosen such that the sequence $x(n+k) = (a_1 x(n+k-1) + \ldots + a_k x(n)) \mod m'$ has a period length $p^k - 1$. It should be noted that the initial values i.e. $x(0), x(1), \ldots x(k-1)$ can be chosen arbitrarily, but not all zero.

For an arbitrary prime number $p$, the modulo nonlinear dynamical system is always periodic with an integer cycle length for many interesting values of $k$ (dimension of the state transition matrix). The periodicity can be explicitly predicted in some cases.

1) **CASE A:** $l = (p^k - 1)$ is the period and it is divisible by ‘k’. In this case we have CYCLE LENGTH = (l)
2) **CASE B:** $l = (p^k - 1)$ is the period and it is not divisible by ‘k’. In this case we have CYCLE LENGTH = (k)(l)

A state $V(n)$ is called a stable state if and only if

\[ V(n) = [M V(n)-T] \mod 'p' \]

Fibonacci Sequence Modulo prime based Neural Network
\[ x(n+2) = (x(n) + x(n+1)) \mod 'p' \]

We now define the state vector associated dynamical system

\[ Y(n) = \begin{bmatrix} x(n) \\ x(n+1) \end{bmatrix} \]

with

\[ Y(0) = \begin{bmatrix} x(0) \\ x(1) \end{bmatrix} \]

State space description : $Y(n+1) = (A Y(n)) \mod 'p'$, where

\[ A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \]

Example : Let $p = 3$ Cycle Length = $p^2 - 1 = 3^2 - 1 = 8$

**Observations and Results**

A graph of number of cycles with the corresponding prime number is shown for various dimensionality’s. We observe that there is no definitive pattern but a strong correlation of number of cycles with the magnitude of the prime number as can be seen from the graph.
III. Conclusion

It could be proved that there is no loss of generality in assuming the threshold value at each neuron is zero.

The advantages of above neuronal model is the following

1) The state space size increases without increasing the number of neurons
2) The number of patterns that can be stored in the associative memory increases
3) we are ensuring quantization of inputs, outputs necessary in some and synaptic weights. This type of quantization can be expected in biological neurons.
4) Feedforward neural networks based on multi-state neurons enable fine partition of pattern space leading to finer classification.
5) Multi-state neurons (rather than binary neurons) are applications

It seems that a general theorem capturing the dynamics is not possible but as can be seen from the results a pattern may be thought of being existent but is definitely inconsistent.

REFERENCES

[1] Garimella Rama Murthy and Moncef Gabbouj Linear Congruential Sequences : Feedback and Recurrent Neural Network, Tampere University of Technology, Finland

Generalized Mandelbrot Set Dynamics

Agnisnato Datta
201230208

S Phani Teja
201230109

Prithvi Reddy
201230198

Akash Kant
201230194

Shubham Chhabra
201230206

Nihal Randhawa
201230184

Jeevan Chowdary Mandadapu
201230190

Nimish Sehgal
201231009

Jeevan Chowdary Mandadapu
201230190

Prithvi Reddy
201230198

Aishwarya Jain
201231103

S Phani Teja
201230198

Uhith Lagisetti
201230200

Abstract—The general Mandelbrot sets are kinds of mathematical dictionary or atlas that map out the behavior of the Julia sets for different values of a complex constant, $c$. In this paper we investigate the general Mandelbrot sets generated using the function $z(n+1) = az(n)^2 - bz(n) + c$ for different values of $a$ and $b$. We use the escaping time algorithm to construct the general Mandelbrot set image which present a better understanding of the structure of the Mandelbrot sets. Some interesting results obtained are included in this project.

I. INTRODUCTION

The generalized Mandelbrot set is the set of complex numbers $c$ for which the function $z(n+1) = az(n)^2 - bz(n) + c$ does not diverge when iterated from $z(0) = 0$. In other words, it consists of those $c$ values for which $z(1), z(2), z(3), ...$ is bounded given $z(0) = 0$.

$$Z(n+1) = az(n)^2 - bz(n) + c$$

$$c \in M \iff \lim_{n \to \infty} |z(n+1)| \leq 2$$

where $M = \text{Generalized Mandelbrot set}$.

In set builder form, it can be represented as:

$$M = \{c \in C : Z(n) \text{ is bounded } \forall \ n \in N\}$$

where $C = \text{Set of complex numbers}$

$N = \text{set of natural numbers}$

Mandelbrot set images are made by sampling complex numbers. Treating the real and imaginary parts of each number as image coordinates, pixels are colored according to how rapidly the sequence diverges, if at all.

The Mandelbrot set has become popular outside mathematics both for its aesthetic appeal and as an example of a complex structure arising from the application of simple rules.

II. NUMERICAL EXPERIMENTS AND RESULTS

For experiments we have considered different combinations of $a$ and $b$ values. We have divided the complete problem into three sets: 1) $a \neq 0, b = 0$ 2) $a = |b|$ 3) $a \neq b$. The enclosed region around the bright boundary forms the Mandelbrot set for the given $a$ and $b$ parameters.

1) $Case : a \neq 0, b = 0$: Here four different values of $a$ are considered. Fig. 1 shows the Mandelbrot set for the following values of $a$.

- $a = 0.5$
- $a = 1$
- $a = 1.5$
- $a = 2$

![Fig. 1: Mandelbrot set for a=0.5 (left top), a=1 (right top), a=1.5 (left bottom), a=2 (right bottom)](image)

Observations:

With increasing $a$ the bounded region shrinks, which means that $z$ for most of the $c$ values blows up. We observe that as the initial circles also vary with $a$, most of the values blow up in first few iterations itself. Hence we can say that with increasing $a$, $z$ blows up at a faster rate. In this case the shape of the bounded region almost remains the same.

2) $Case : a = b$: The following values of $a$ and $b$ have been considered for generating Mandelbrot sets.

- $a = b = 0.5$
- $a = b = 1$
- $a = b = 1.5$
- $a = b = 2$. 
Fig. 2: Mandelbrot set for $a = b = 0.5$ (left top), $a = b = 1$ (right top), $a = b = 1.5$ (left bottom), $a = b = 2$ (right bottom)

Fig. 2 shows the Mandelbrot sets for the given cases.

**Observations:**

As the value of $a$, $b$ increases the Mandelbrot set becomes smaller and smaller. The shape of the converging region changes rapidly. And as we increase the values of $a$ and $b$, the converging region shifts towards right implying that Mandelbrot set is confined to the values of $c$ having positive real part.

Fig. 3: Mandelbrot set for $a = 1$, $b = 0.5$ (left top), $a = 1.5$, $b = 0.5$ (right top), $a = 1$, $b = 1.5$ (left bottom), $a = 1$, $b = 2$ (right bottom)

3) Case : $a \neq b$: Different combinations of $a$ and $b$ are taken in experimental observations. Fig. 3 shows the following four cases.

- $a = 1, b = 0.5$
- $a = 1.5, b = 0.5$
- $a = 1, b = 1.5$
- $a = 1, b = 2$

**Observations:**

As $a$ is varied keeping $b$ constant, it only affects the size of Mandelbrot set. But when $b$ is varied keeping a constant, the shape changes drastically and the complete set shifts towards right of the complex plane. The brighter part around the boundary becomes wider in the latter case implying that we need more number of iterations to decide the boundary.

4) Special cases: We have considered the following two special cases for our experimental analysis.

- $a = 1, b = -1$
- $a = -1, b = 1$

Fig. 4: Mandelbrot set for $a = 1$, $b = -1$ (left), $a = -1$, $b = 1$ (right)

**Observations:**

We observe that the direction of the graph is decided by the sign of $a$ irrespective of sign or value of $b$. Also we see that $b$ effects the shape and place of the bounded region in the complex plane.

**III. Conclusion**

We find that general Mandelbrot sets are symmetrical by reflection in the real axis and are bounded. We also find that the orientation of the graph depends on the sign of $a$ and the size of the graph decreases as the value of $a$ increases. And we also observe that the shape of the graph becomes more ragged around the boundary of the converging region as the value of $b$ increases and the graph shifts towards either the right side or left depending on the sign of $b$. A video on the formation of bounded regions is present here: MandelbrotSet.

**REFERENCES**

1. Summary:

Mandelbrot Set:
The Mandelbrot set is the set of complex numbers $c$ for which the function $f_c(z) = z^2 + c$ does not diverge when iterated from $z = C$, i.e., for which the sequence $f_c(0), f_c(f_c(0)), \ldots$ remains bounded in absolute value.

More precisely, the Mandelbrot set is the set of values of $c$ in the complex plane for which the orbit of 0 under iteration of the quadratic map

\[ z_{n+1} = z_n^2 + c \]

Initial image of a Mandelbrot set zoom sequence

Neural networks based on mandelbrot set:
In principle we can transpose the fields of trajectories of any analytic function (in our case the mandelbrot set) and interpret as symbolic representations of neurons. An iteration of a fractal algorithm will move us from $z_n$ to $z_{(n+1)}$. Similarly, a neuron at any state $z_n$ in a network could send its activity along its axon (following the respective trajectory) to the point $z_{(n+1)}$ of the neural network.
Here we have designed a neural network with 3 neurons.

- $V = [V_1, V_2, V_3]$ define the state of the neurons 1, 2 and 3 respectively where $V_i \in \mathbb{C}$.
- $W$ is a 3x3 symmetric weight matrix where $W_{ij} \in \mathbb{R}$
- Net = $W^*V[n]$ where $V[n]$ defines the state of the neural network at time $n$.
- $V_j[n+1] = \text{Net}_j^2 + c$ where $c \in \mathbb{C}$

2. Numerical Experiments:

Experiments were performed by varying the values of weights $w$, Constant factor $c$ and the initial state of the neural network $V$ to check if the output after 1000 iterations converged or blew up. The results are presented in the table as shown:

<table>
<thead>
<tr>
<th>Weights (real)</th>
<th>Constant Factor (real and imaginary)</th>
<th>Initial State</th>
<th>Converge/Blow up</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>0-0.1</td>
<td>0-0.1</td>
<td>converge</td>
</tr>
<tr>
<td>0-1</td>
<td>0-0.1</td>
<td>0.1-0.5</td>
<td>Blow up</td>
</tr>
<tr>
<td>0-1</td>
<td>0-0.1</td>
<td>0.1-0.2</td>
<td>converge</td>
</tr>
<tr>
<td>0-1</td>
<td>0-0.1</td>
<td>0.2-0.3</td>
<td>After 12 iterations blows up</td>
</tr>
<tr>
<td>0-0.1</td>
<td>0-0.1</td>
<td>0.2-0.3</td>
<td>Converge</td>
</tr>
<tr>
<td>0-0.1</td>
<td>0-0.1</td>
<td>0.5-1</td>
<td>Converge</td>
</tr>
<tr>
<td>0-0.1</td>
<td>0-0.1</td>
<td>5-10</td>
<td>Converge</td>
</tr>
<tr>
<td>0-0.1</td>
<td>0-0.1</td>
<td>10-20</td>
<td>Converge</td>
</tr>
<tr>
<td>0-0.1</td>
<td>0-0.1</td>
<td>30-45</td>
<td>Converge</td>
</tr>
</tbody>
</table>
### 3. Results and Observations:

The graphs below are plotted with iteration on x-axis and V₁ corresponding to that iteration. In case of convergence while most graphs looked as follows:

<table>
<thead>
<tr>
<th>0-0.1</th>
<th>0-0.1</th>
<th>45-50</th>
<th>Diverge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>0-0.1</td>
<td>0-0.1</td>
<td>Diverge</td>
</tr>
<tr>
<td>1-2</td>
<td>0-0.1</td>
<td>0-0.01</td>
<td>Diverge</td>
</tr>
<tr>
<td>1-2</td>
<td>0-0.1</td>
<td>0-0.001</td>
<td>Diverge</td>
</tr>
<tr>
<td>0.5-1</td>
<td>0-0.1</td>
<td>0-0.1</td>
<td>Converge</td>
</tr>
<tr>
<td>0.5-1</td>
<td>0-0.1</td>
<td>0-5-1</td>
<td>Diverge</td>
</tr>
<tr>
<td>0-1</td>
<td>0-0.1</td>
<td>0-0.1</td>
<td>Converge</td>
</tr>
<tr>
<td>0-1</td>
<td>0.1-0.2</td>
<td>0-0.1</td>
<td>Converge</td>
</tr>
<tr>
<td>0-1</td>
<td>0.2-0.3</td>
<td>0-0.1</td>
<td>Diverge</td>
</tr>
</tbody>
</table>

Some were different and looked as follows:
Under the above conducted experiments no cycles were found.

The range of value of the weights which were used to calculate and update the network states affected the output of the network and whether it converges or diverges. The effect was more as compared to the constant value or the value of initial states

4. References:
- [https://en.wikipedia.org/wiki/Mandelbrot_set](https://en.wikipedia.org/wiki/Mandelbrot_set)
- [http://www.fractal.org/Life-Science-Technology/Publications/Fractal-Neural-Networks.htm](http://www.fractal.org/Life-Science-Technology/Publications/Fractal-Neural-Networks.htm)