Application of Wavelet Neural Network in Mixed Design of Concrete

CE F376: Design Project

by

Sreenivas Kanaparthy

2015B2A20802H

Project Work work carried out under guidance of

Prof. Dr. A Vasan

D117, Professor, Department of Civil Engineering

Associate Dean, Academic - Undergraduate Studies

Birla Institute of Technology & Science, Pilani – Hyderabad Campus

Jawahar Nagar, Hyderabad 500 078, Telangana, India

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE PILANI (Hyderabad)

November 2018

CERTIFICATE

This is to certify that the Project Work entitled Application of Wavelet Neural Network in Mixed Design of Concrete and submitted by Sreenivas Kanaparthy having ID-No. 2015B2A20802H

for the partial fulfillment of the requirements of B.S. Civil degree of BITS, embodies the bonafide work done by him under my supervision.

_____

Signature of Professor
Prof. Dr. A. Vasan,
Department of Civil Engineering,
Bits Pilani Hyderabad Campus

Place : _____

Date : _____

ACKNOWLEDGEMENTS

I am very much grateful to the Director and Head of the Department of Civil Engineering, Birla Institute of Technology and Science, Hyderabad for giving me an opportunity to work for the project and also for using all the facilities of the institute under the supervision and guidance of highly respected Prof. Dr. A Vasan, Professor, Department of Civil Engineering, Associate Dean.

This project opportunity in the field of Neural Networks was a great chance for me to learn more about the practical problems in implementation of Neural Networks and the difficulty of their real world applications. Throughout the project Prof. Dr. A Vasan, had lent his support and valuable suggestions, which helped me improve the project, and grow simultaneously.

Through this acknowledgement I want to express my sincere gratitude towards all those people who gave their view and suggestions for helping in the completion of this project which has been a commensurable learning experience

Sreenivas Kanaparthy

ABSTRACT

Neural Networks are used in different forms to find optimal solutions of many kinds of problem statements. The attempt to train a Neural Network is to build a set of algorithms which mimic the functionality of a real human being's thinking. In this project we try to model a function by using the given labelled data. We randomly initialise the coefficients of the function and make the Neural Network learn these.

To exactly understand what is done in the project a little insight about a general Neural Network is to be introduced. A Neural Network consists of a input layer, hidden layer and an output layer. The hidden nodes, which comprise of the hidden layer, have two parts in them, one is the preactivation, i.e. the products of corresponding weights and their summation over all the nodes, the other part is the activation, where you apply a certain activation function to the output of the preactivation layer.

In this project instead of using the traditional hidden node we have incorporated a wavelet function which takes in the output of the prectivation and dilates it, translates it and then provides the output to the final output node.

List of Figures and Tables

Table of Contents

Chapter 1 : Dissertation on Wavelet Neural Networks and their application in the study of dynamical systems

1.1 Introduction

The main aim inn the given dissertation is to study the topic of wavelet neural networks and observe their applications in the dynamic systems. I have taken in insights about the basics and their mechanisms from this.

1.2 Wavelets

Wavelets are a class of functions used to localise a given function in both position and scaling. They are used in applications such as signal processing and time series analysis.

Wavelets form the basis of the wavelet transform which “cuts up data of functions or operators into different frequency components, and then studies each component with a resolution matched to its scale”[1]

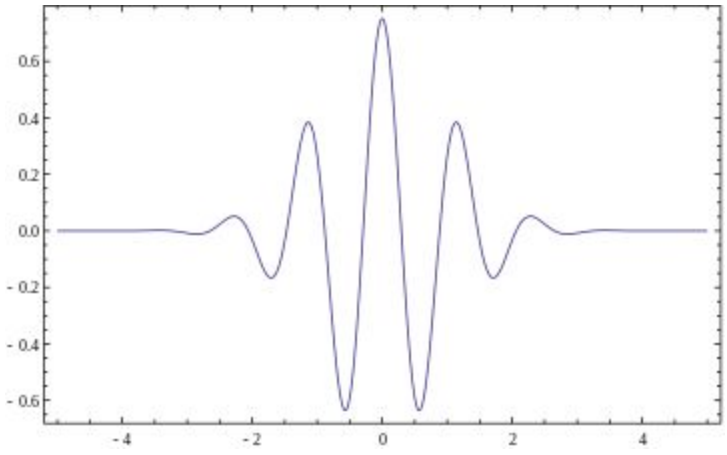

Figure 1 : A typical wavelet

In the context of signal processing, the wavelet transform depends upon two variables: scale (or frequency) and time.

There are two main types of wavelet transforms:

1. Continuous (CWT)
2. Discrete(DWT)

Continuous Wave Transform is designed to work with functions defined over the whole real axis.

Discrete Wave Transform deals with functions that are defined over a range of integers (usually t = 0, 1,... ,N − 1, where N denotes the number of values in the time series).

1.2.1 Characteristics of Wavelets

A wavelet is a 'small wave' function, usually denoted ψ(·).
For a function ψ(·), defined over the real axis (−∞,∞), to be characterised as a wavelet it must be able to satisfy the corresponding properties :

1. The integral of ψ(·) is zero, ie the total area is zero, top and bottom parts are equal:

$$\int_{-\infty}^{\infty} \psi(u)du = 0 \qquad \text{Equation(1)}$$

2. The integral of the square of ψ(·) is unity, unity area when magnitude is considered:

$$\int_{-\infty}^{\infty} \psi(u)^2 du = 1 \qquad \text{Equation(2)}$$

3. Admissibility Condition, should be finite:

$$C_\psi \equiv \int_0^{\infty} \frac{|\psi(f)|^2}{f}\, df \qquad \text{satisfies } 0 < C\psi < \infty \text{ (1.2.3)}$$

Equation (1) tells us that any excursions the wavelet function ψ makes above zero, must be cancelled out by excursions below zero. Clearly the line ψ(u) = 0 will satisfy this, but equation (2) tells us that ψ must make some finite excursions away from zero.
If the admissibility condition is also satisfied then the signal to be analysed can be reconstructed from its continuous wavelet transform.[1] Basically it all adds up to localising the output.

A function ψ(.) is also known as the mother wavelet and you can form a family of wavelets by using translation and dilation versions of the same mother wavelet. This can be represented in the form of an equation, as shown below:

$$\psi(u) = \frac{1}{\sqrt{\lambda}} \psi\left(\frac{u-t}{\lambda}\right)$$

1.3 Neural Networks

An Artificial Neural Network (ANN) is a highly parallel distributed network of connected processing units called neurons. It is motivated by the human cognitive process:the human brain is a highly complex, nonlinear and parallel computer. The network has a series of external inputs and outputs which take or supply information to the surrounding environment. [1]

Inter-neuron connections are called synapses which have associated synaptic weights. These weights are used to store knowledge which is acquired from the environment. Learning is achieved by adjusting these weights in accordance with a learning algorithm. It is also possible for neurons to evolve by modifying their own topology, which is motivated by the fact that neurons in the human brain can die and new synapses can grow.[2]

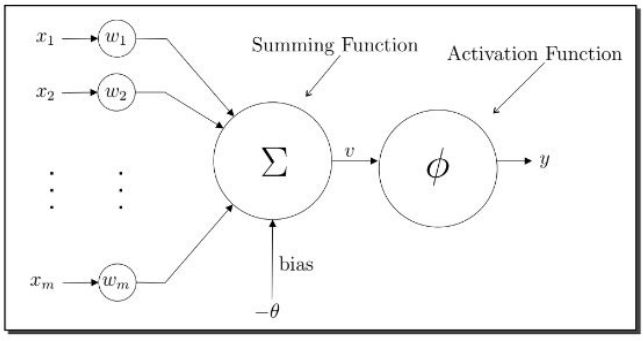

Figure 2 : A typical Neuron

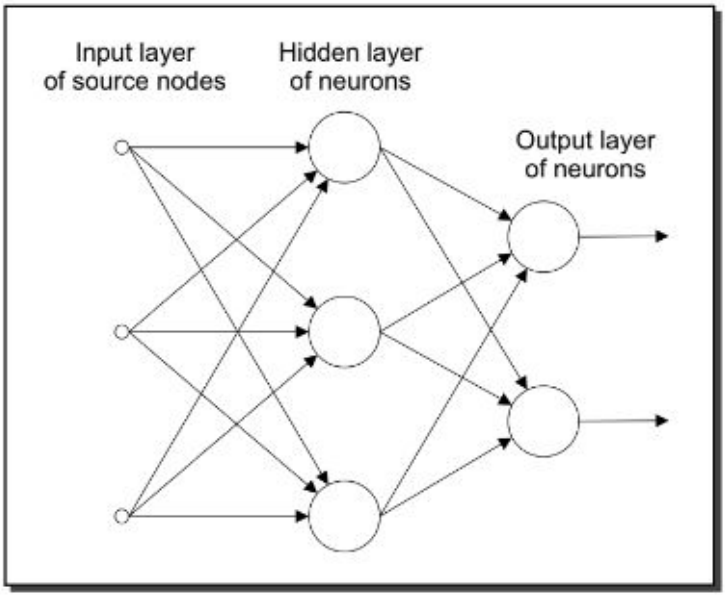

Figure 3 : A multilayer feed forward neural network

One of the primary aims of an ANN is to generalise its acquired knowledge to similar but unseen input patterns. Two other advantages of biological neural systems are the relative speed with which they perform computations and their robustness in the face of environmental and/or internal degradation. Thus damage to a part of an ANN usually has little impact on its computational capacity as a whole. This also means that ANNs are able to cope with the corruption of incoming signals.[1]

1.4 Wavelet Neural Network

1.4.1 Introduction

Wavelet neural networks combine the theory of wavelets and neural networks into one.A wavelet neural network generally consists of a feed-forward neural network, with one hidden layer, whose activation functions are drawn from an orthonormal wavelet family.

One applications of wavelet neural networks is that of function estimation. Given a series of observed values of a function, a wavelet network can be trained to learn the composition of that function, and hence calculate an expected value for a given input.

The structure of a wavelet neural network is very similar to that of a (1+ 1/2) layer neural network. That is, a feed-forward neural network, taking one or more inputs, with one hidden layer and

whose output layer consists of one or more linear combiners or summers (see Figure below). The hidden layer consists of neurons, whose activation functions are drawn from a wavelet basis. These wavelet neurons are usually referred to as **wavelons.**

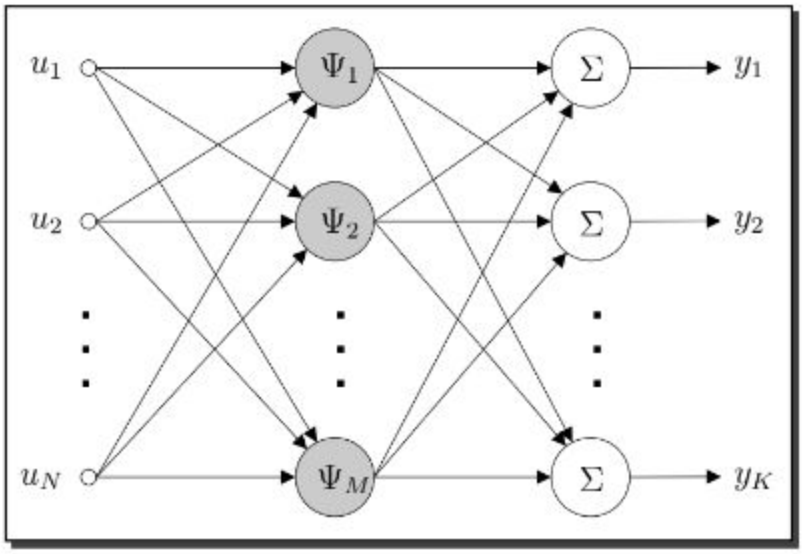

Figure 4 : Structure of a Wavelet Neural Network

There are two main approaches to creating wavelet neural networks:

- In the first the wavelet and the neural network processing are performed separately. The input signal is first decomposed using some wavelet basis by the neurons in the hidden layer. The wavelet coefficients are then output to one or more summers whose input weights are modified in accordance with some learning algorithm. This type of wavelet neural network is usually referred to as a **wavenet.**
- The second type combines the two theories. In this case the translation and dilation of the wavelets along with the summer weights are modified in accordance with some learning algorithm.This type of wavelet neural network is usually referred to as a **wavelet network.**

1.4.2 One Dimensional Wavelet Neural Network

The simplest form of wavelet neural network is one with a single input and a single output. The hidden layer of neurons consist of wavelons, whose input parameters (possibly fixed) include the wavelet dilation and translation coefficients. These **wavelons** produce a non-zero output when the input lies within a small area of the input domain.

The output of a wavelet neural network is a linear weighted combination of the wavelet activation functions.

$\Psi_{\lambda,t(u)} = \Psi(\frac{u-t}{\lambda})$ λ - dilation parameter, t - translation parameters

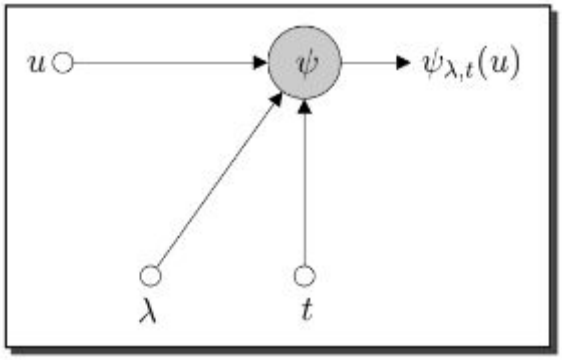

Figure 5 : Wavelet neuron (wavelon)

1.4.3 Learning Algorithm

The parameters $\bar{y}$, w_i's, t_i's and λ_i's should be formed into one vector θ. Now yθ(u) refers to the wavelet network, with parameter vector θ.

$$y_\theta(u) = \sum_{i=1}^{M} w_i \psi\left(\frac{u - t_i}{\lambda_i}\right) + \bar{y}$$

The objective function to be minimised is then :

$$C(\theta) = \frac{1}{2}\mathrm{E}\{(y_\theta(u) - f(u))^2\}.$$

The minimisation is performed using a stochastic gradient algorithm. This recursively modifies θ, after each sample pair $\{u_k, f(u_k)\}$, in the opposite direction of the gradient of :

$$c(\theta, u_k, f(u_k)) = \frac{1}{2}(y_\theta(u_k) - f(u_k))^2.$$

The gradient for each parameter of θ can be found by calculating the partial derivatives of $c(\theta, u_k, f(u_k))$ as follows:

$$\frac{\partial c}{\partial \bar{y}} = e_k$$
$$\frac{\partial c}{\partial w_i} = e_k \psi(z_{ki})$$
$$\frac{\partial c}{\partial t_i} = -e_k w_i \frac{1}{\lambda_i} \psi'(z_{ki})$$
$$\frac{\partial c}{\partial \lambda_i} = -e_k w_i \left(\frac{u_k - t_i}{\lambda_i^2} \right) \psi'(z_{ki})$$

Where $e_k = y_0(u_k) - f(u_k),\ z_{ki} = \frac{u_k - t_i}{\lambda_i}$ and $\psi'(z) = \frac{d\psi(z)}{dz}$.

To implement this algorithm, a learning rate value and the number of learning iterations need to be chosen. The learning rate $\gamma \in (0, 1]$ determines how fast the algorithm attempts to converge.

The gradients for each parameter are multiplied by γ before being used to modify that parameter. The learning iterations determine how many times the training data should be fed through the learning process. The larger this value is, the closer the convergence of the network to the function should be, but the computation time will increase.

1.4.4 Constraints on Adjustable Parameters

Zhang and Benveniste proposed to set constraints on the adjustable parameters to help prevent the stochastic gradient algorithm from diverging. If we let $f : D \rightarrow R$ be the function we are trying to approximate, where $D \in R$ is the domain of interest, then:

(1) The wavelets should be kept in or near the domain D. To achieve this, choose another domain E such that $D \subset E \subset R$. Then require:

$$t_i \in E \quad \forall i$$

(2) The wavelets should not be compressed beyond a certain limit, so we select $\varrho > 0$ and require:

$$\lambda i > \varrho \quad \forall i$$

<u>Initialising the Adjustable Parameters for the Wavelet Network</u>

Before we can run the network, the adjustable parameters $\bar{y}$, w_i, t_i and λ_i need to be initialised.

- $\bar{y}$ should be estimated by taking the average of the available observations.

- The w_i's should be set to zero.
- To set the t_i's and λ_i's select a point p within the interval [a,b] and set $t_1 = p$ and $\lambda_1 = E(b - a)$, where $E > 0$ is typically set to 0.5.

Initialising the Parameters for the Wavenet

Before we can run the network for this purpose, we need to initialise the (adjustable and non-adjustable) parameters w_i, t_i and λ_i

- The w_i's should be set to zero.
- To set the t_i's and λ_i's we first need to choose a resolution level L. The λ_i's are then set to 2L. The t_i's are set to intervals of 2^{-L} and all satisfy $t_i \in E$, where E is a domain satisfying $D \subset E \subset R$.

Chapter 2 : Implementation on Database

2.1 Database and its classes

The database consists of 6 input classes namely Cement, Blast Furnace Slag, Fly Ash, Water, Superplasticizer,Coarse Aggregate, Fine Aggregate, and one output class named, Concrete compressive strength. The total number of data points for each class are 283.

2.2 Code

```
clear all
close all
% intitate of data
P=283 % number of sample
m=6    %number of input node
n=400;     %number of hidden node
N = 1     %number of output node


% a(n) b(n) scale and shifting parameters matrix
% x(P,n) input matrix of P sample
% net(P,n) output of hidden node
% y(P,N) output of the network
% d(P,N) ideal output of the network
% phi(P,n) output of hidden node wavelet function
% W(N,n) weight between the output and the hidden
% WW(n,m) weight value between hidden and input node

x = csvread('/home/svshivapuja/Downloads/wnn_data.csv',1,0,[1 0 283 6])
d= csvread('/home/svshivapuja/Downloads/wnn_data.csv',1,7)
W = rand(N,n)
WW = rand(n,m)
a = ones(1,n)

for j=1:n
b(j)=j*P/n;
a(j)=10*j*P/n;
end

% EW (N,n) gradient of W
% EWW(n,m) gradient of WW
% Ea(n) gradient of a
% Eb(n) gradient of b

epoch = 1;
epo = 20000;
error =0.5;
err =0.00000001;
delta=0.03456;
lin=0.8;
```

```
41
42 - while (error >= err & epoch<=epo)
43
44   % w is the middle variant
45   % calculation of net input
46 -     for p = 1: P
47 -         for j=1:n
48 -             u=0;
49 -             for k =1:m
50 -                 u = u + WW(j,k)*x(p,k);
51 -             end
52 -             net(p,j)=u;
53 -         end
54 -     end
55
56   % calculation of mexican hat wavelet output
57
58 -     for p = 1:P
59 -         for j= 1:n
60 -             u = net (p,j);
61 -             u = (u-b(j))/a(j);
62               %phi(p,j) = cos(1.75*u)*exp(-u*u/2);
63               %mexican hat wavelet
64 -             phi(p,j)= (1-u^2) * exp(-u*u/2);
65 -         end
66 -     end
67
68   % calculation of output
69 -     for p = 1 : P
70 -         for i = 1: N
71 -             for j = 1:n
72 -                 u = net (p,j);
73 -                 u = (u-b(j))/a(j);
74 -                 u = u + W(i,j)*phi(p,j);
75                   %y(p,i) = delta*abs(u);
76 -             end
77 -             y(p,i) = delta*abs(u);
78 -         end
79 -     end
80
```

```
80
81      % calculation of error of output
82
83 -        u = 0;
84 -        for p = 1 : P
85 -            for i = 1: N
86 -                u = u +(d(p,i)- y(p,i))^2;
87 -            end
88 -        end
89
90 -        error = u;
91
92 -        for i = 1 : N
93 -            for j = 1: n
94 -                u = 0;
95 -                for p = 1:P
96 -                    u= u + (d(p,i)-y(p,i))*phi(p,j);
97 -                end
98 -                Ew(i,j) = u;
99 -            end
100 -       end
101
102 -       for j = 1 : n
103 -           for k = 1: m
104 -               u = 0;
105 -               for  p = 1:P
106 -                   for i = 1:N
107 -                       u = u + (d(p,i) - y(p,i))*W(i,j)*phi(p,j)*x(p,k)/a(j);
108 -                   end
109 -               end
110 -               EWW (j,k) = u;
111 -           end
112 -       end
113
114 -       for j = 1 : n
115 -           u = 0;
116 -           for p = 1:P
117 -               for  i = 1:N
118 -                   u=  u + (d(p,i) - y(p,i))*W(i,j)*phi(p,j)/a(j);
119 -               end
120 -           end
121 -           Eb(j) = u;
122 -       end
```

```
123
124
125 -       for j = 1 : n
126 -           u = 0;
127 -           for p = 1:P
128 -               for  i = 1:N
129 -                   u= u + (d(p,i) - y(p,i))*W(i,j)*phi(p,j)*x(p,k)/a(j);
130 -               end
131 -           end
132 -           Ea(j) = u;
133 -       end
134
135       % adjust of weight value
136
137 -       WW = WW - lin*EWW;
138 -       W = W - lin*EW;
139 -       a = a - lin*Ea;
140 -       b = b - lin*Eb;
141 -       epoch = epoch +1;
142 -       epoch
143 -   end
144
145 -   plot (x,d,'-*',x,y,'-o')
146 -   y
147 -   d
148 -   mse(d-y)
```

2.3 Analysis of Results

2.3.1 Results

The entire code is run on four different types of wavelets and their results have been tabulated, the four wavelets taken into consideration are shown in the following figures :

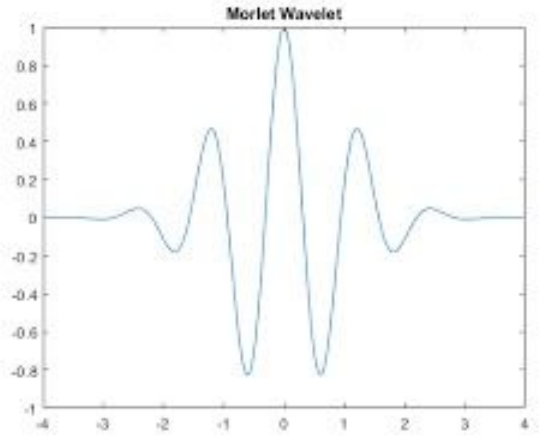

Figure 6 : Morlet Wavelet

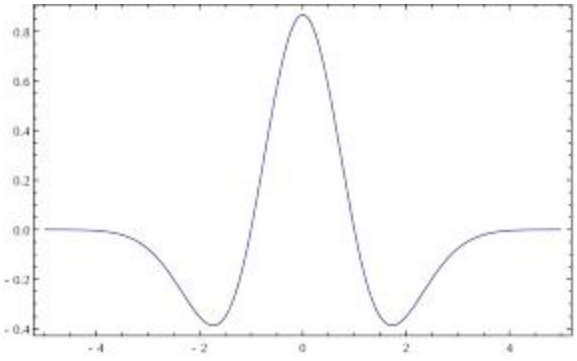

Figure 7 : Mexican Hat Wavelet

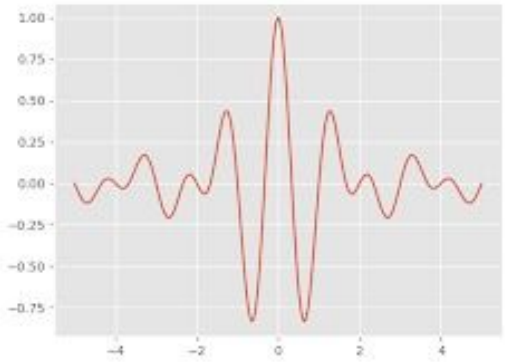

Figure 8 : Shannon Wavelet

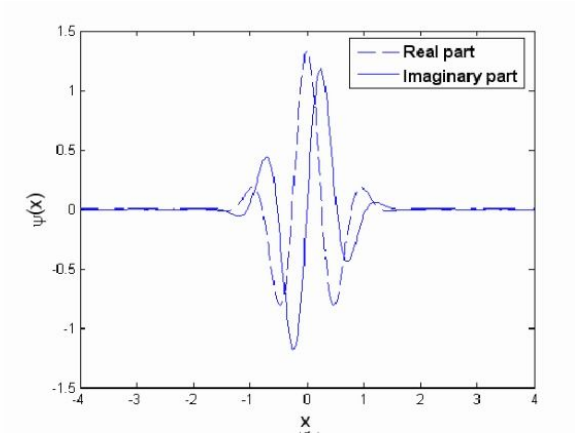

Figure 9 : Modified Morlet Wavelet

Wavelet	Morlet	Mexican Hat	Shannon (Real)	Modified Morelet
Error probability	0.8386	0.6866	0.7692	0.8339

Table 1 : Output Summary

The graphs of the outputs are as follows, the * points in th graphs indicate the expected values and o- marks in the graph are points of predicted values. Different colours are assigned to the six different classes.

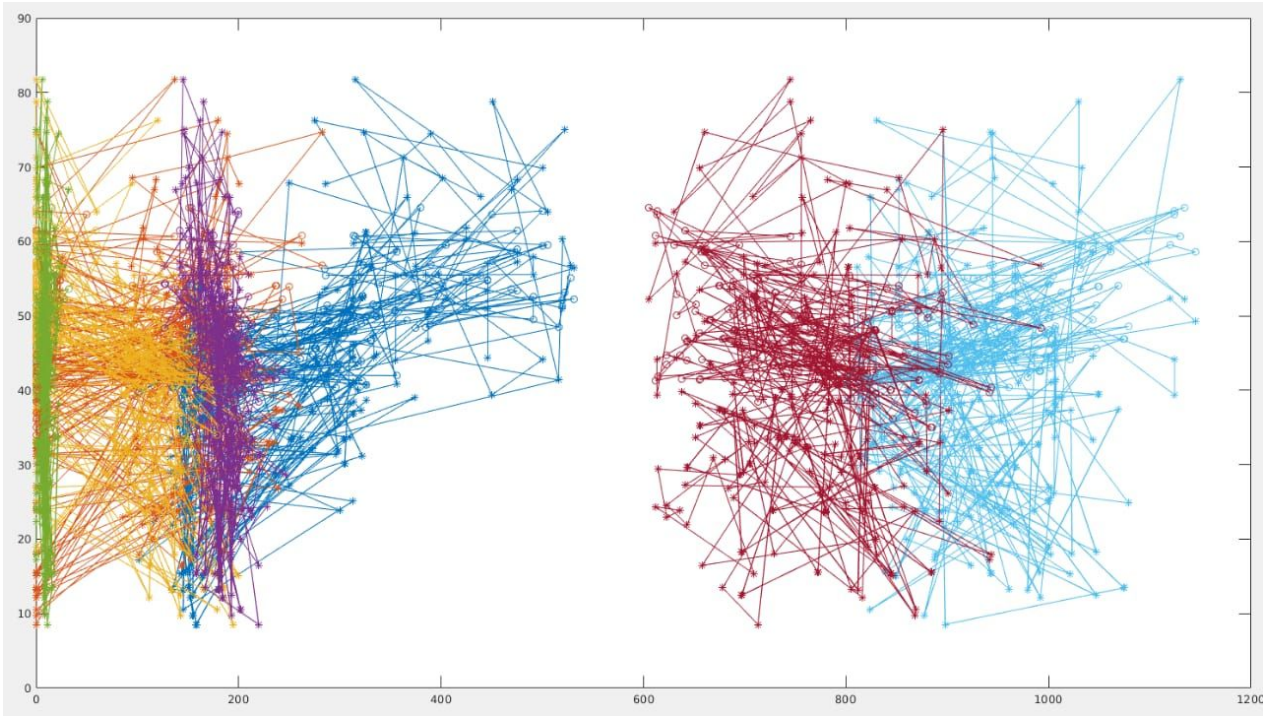

Figure 10 : Output of Morlet Wavelet

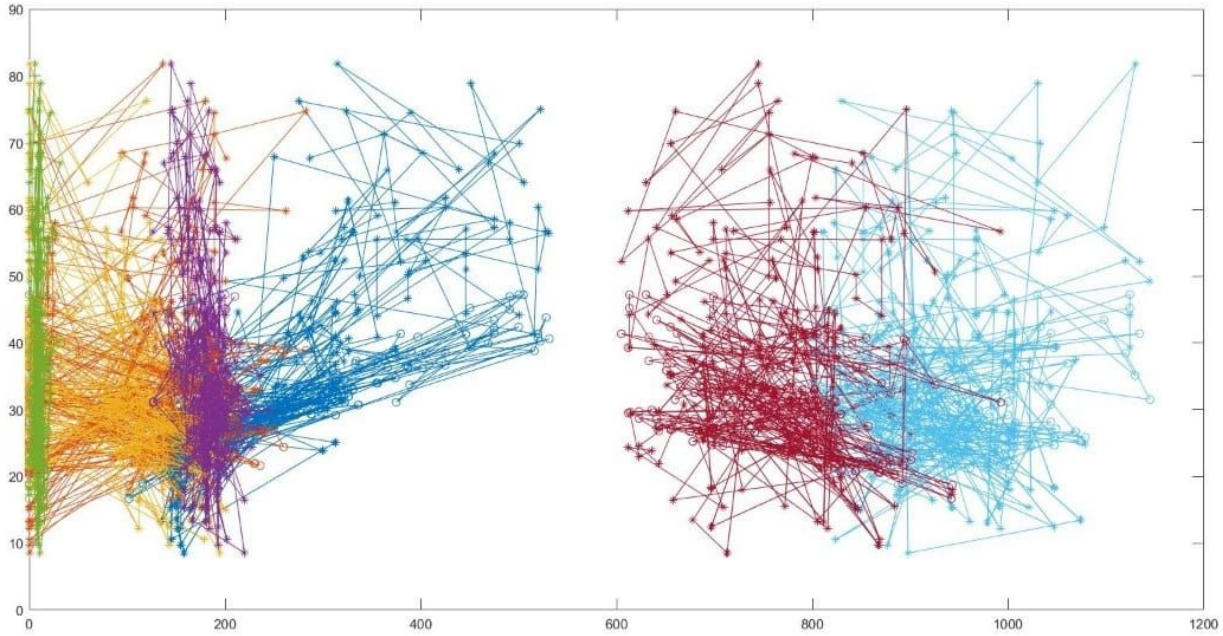

Figure 11 : Output of Mexican Hat Wavelet

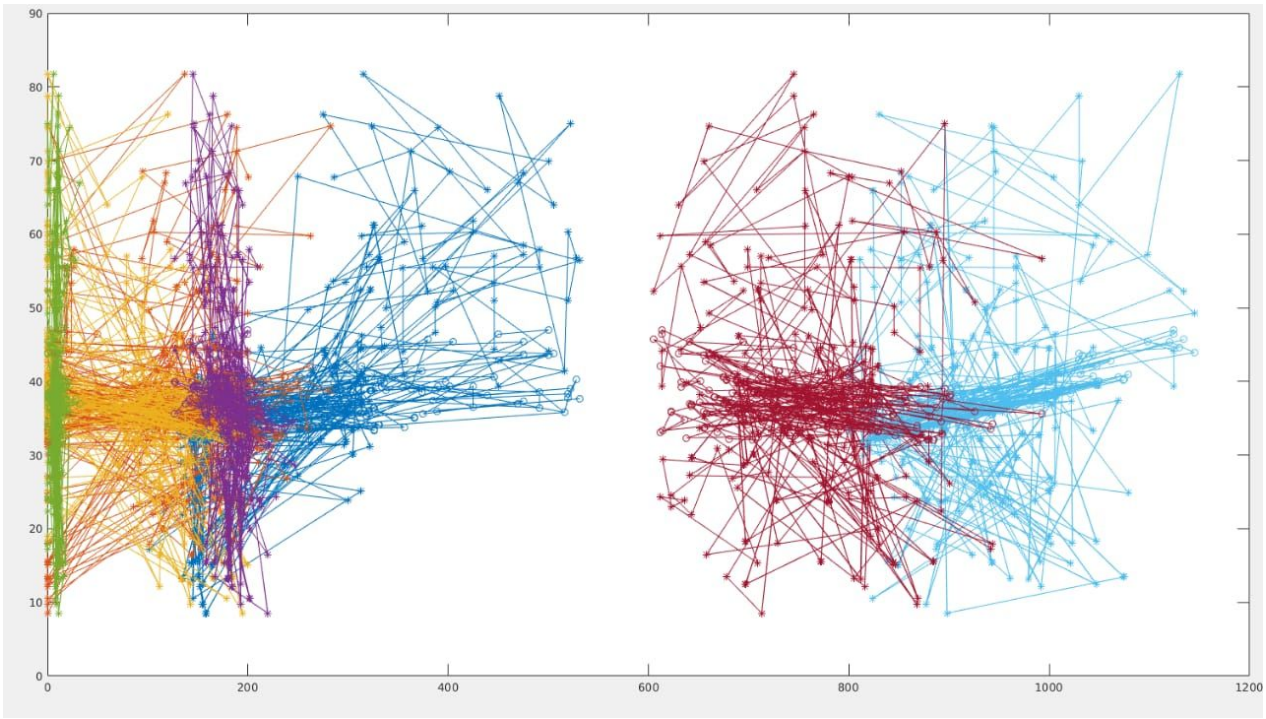

Figure 12 : Output of Shannon (Real) Wavelet

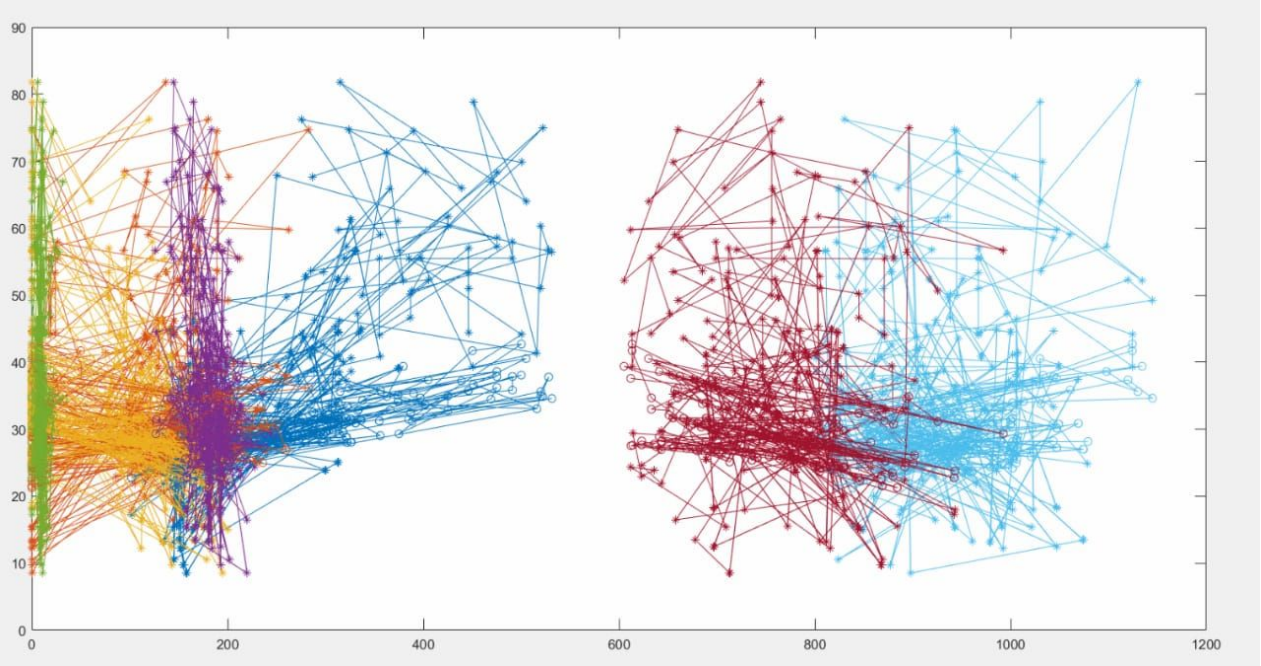

Figure 13 : Output of Modified Morlet

2.3.2 Drawbacks

The main problem encountered while training is the initialization of the weights and the extent of the learning parameter 'lin', the scaling factor 'delta', initialising and working with the translation and dilation parameters a_j's and b_j's is also a tough task.

There is no optimal way to initialise and set the stationary parameters which leads to much difficulty to decide which direction to proceed with the code.

Chapter 3 : Conclusions and Future Scope

3.1 Conclusions

The implementation of function learning can be achieved using wavelet neural networks. Fine tuning of the parameters can help us drive the loss to about the range of micro units.

Also, more data will give us more insight to help the network learn better, the more the number of data points the better will be the prediction model and its results.

To test this model, we need to just take the matrices after learning is done and use it to predict the output by giving in the inputs.

3.2 Future Scope

The problem statement can be extended to any number variable function prediction as required. As mentioned earlier, if more data points are provided the performance of the model can be improved.

References

[1] : Wavelet Neural Networks and their application in the study of dynamical systems, David Veitch, *Department of Mathematics, University of York, UK, August 2005.* (https://pdfs.semanticscholar.org/0c8b/e141c9092ed389b9931ac09ec2e852d437c6.pdf)

[2] : Wavelet neural networks for function learning, Jun Zhang, G.G. Walter, Y. Miao, Wan Ngai Wayne Lee. Published in: *IEEE Transactions on Signal Processing (Volume: 43 , Issue: 6 , Jun 1995).*

[3] : https://en.wikipedia.org/wiki/Wavelet

[4] : Function learning using Wavelet Neural Network, Shashidhara H.L, Sumit Lohani, Vikram M. Gadre, *Department of Electrical Engineering, Indian Institute of technology Bombay*

[5] : A Wavelet Neural Network for SAR Image Segmentation, Xian-Bin Wen, Hua Zhang and Fa-Yu Wang, *Journal, Published: 22 September 2009.*

[6] : A Wavelet Neural Network for Function Approximation and Network Optimisation, Kunikazu Kobayashi and Toyoshi Torioka,*Department of Computer Science and Systems Engineering,Faculty of Engineering, Yamaguchi University*