International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 3; Issue: 1; July -2014

A Simple Approach on Network Configuration for Predicting Rainfall in High Resolution

Dr. Rakesh Kumar
Professor ,Dept. of MBA
SCIT, Pune,
Maharastra -411057
Email : meetrktit@gmail.com

Abstract: This paper is entitled "A Simple Approach on Network Configuration for Predicting Rainfall in High Resolution" aims at implementing the algorithm that can be used for the job of predicting patterns, if it exists in a data set. The algorithm that is we are used here is popularly known as "Back-propagation Algorithm" and the Concept of supervised learning. The implementation here provides the user with the flexibility of deciding not only the number of computational units (Neurons) in the input layer but also the number hidden layers required and the number of hidden neurons in each hidden layer. The implementation does not know about the data and its representational context and thus may require a few changes before application of the data set. The sensitivity of each of these variations to large and small changes in the values of the configurations was studied in detailed, so as to obtain a deep insight into their working. Data sets of increasing complexity were used as the networks were being trained to adjust to these sets. Thus the different data sets were as the networks were being trained to adjust to these sets. Thus the different data sets were used to test the accuracy and strongly of these methods. The data sets varied starting from a sine wave to the rainfall data.

Keywords: Neural Network, Back-Propagation, Network Configuration. Resolution Control

1. Introduction

The rainfall occurrence in a region may be defined as the expected amount of rainfall given in units in particular region within a certain period of time. The final result of the rainfall occurrence analysis is the determination of the probability of occurrence of natural phenomenon such as intensity, magnitude, etc. The observations from the past rainfall records are used to model the future activity. New methods are, therefore, to be introduced to overcome the limitations of conventional methods. Artificial neural network (ANN) approach has recently been touted as having enormous potential for a variety of problems in various fields such as image processing and signal processing, civil, and electrical and mechanical engineering.

This is primarily due to the fact that this approach does not depend upon any assumptions about distribution of data, has the ability to handle data obtained at different levels of precision, and had rapid data processing capability. In the present context, this approach has been used to evaluate the rainfall occurrence. In this

paper, i used the application of neural networks to model dynamic processes can be explored in different levels of Application that can be used and analyzed for predicting the data and its operational values..

2. Objective

The objective of this paper is to develop a feed forward artificial neural network to predict the pattern of occurrence of the events observed in the above mentioned data. In order to achieve this the following distinct phases were followed:

(i) Prediction of synthetic data

In this phase, a neural network was developed for the prediction of synthetic data namely, the sine data.

(ii) Analysis of rainfall data

This includes the final part of the paper with the prediction of the pattern of the pattern of natural data. This is accompanied with extensive analysis of the results and the evaluation of other component factors such as iterations allowed, learning rate on the efficiency of the network.

Weighting Factors:

A neuron usually receives any simultaneous input. Each input has its own relative weights, which gives the input the impact it needs on the processing element's summation function. These weights perform the same type of the function, as do the varying synaptic strengths of biological neurons. In both the cases, some input are made more important then others, so that they have greater effect on the processing element as they combine to produce a neural response. Weights are adoptive coefficients with in the network that determine the intensity of the input signal as registered by the artificial neuron. They are measure of input connection strength. These strengths can be modified in response to various training sets and according to a network specific topology or through its learning rules.

Summation Function:

The first step in processing element's operation is to compute weighted sum of all the inputs. Mathematically, the inputs and the corresponding weights are vectors which can be represented as (i_1, i_2, i_3 ... i_n) and (w_1, w_2, w_3 ... w_n).

$$\text{Weighted sum} = \sum_{i=1}^{n} x_i * w_i$$

$x_i \rightarrow i^{th}$ input

$w_i \rightarrow$ weight of the i^{th} input

This simplistic summation function is found by multiplying each component of the *i* vector by the corresponding component of the w vector and then adding of all the products. The result is a single number and not multi-element vector.

Transfer Function :

A transfer function is a mathematical representation, in terms of spatial or temporal frequency, of the relation between the input and output of a (linear time-invariant) system. With optical imaging devices, for example, it is the

Fourier transform (hence a function of spatial frequency) of the point spread function i.e. the intensity distribution caused by a point object in the field of view. Some of the popular transfer functions are:

1. Sigmoid
2. Gaussian
3. Hyperbolic Tangent
4. Secant Transfer function

Sigmoid, Gaussian, Hyperbolic Tangent and Secant Transfer function normalize the output signal generated by each node. A node's transfer function serves the purpose of controlling the output signal strength for the node (except for the input layer, which uses the inputs themselves). These functions set the output signal strength between 0.0 and 1.0. The input to the transfer functions is the dot product of the entire node's input signals and the node's weight vector. The following figure shows the behavior of Sigmoid function.

This Sigmoid function is the most widely used function for back-propagation neural networks. The Sigmoid function is represented by the mathematical relationship

$$\varphi(x) = \frac{1}{1+e^{-ax}}$$

The Sigmoid function acts as on an output gate that can be opened (1) or closed (0). Since the function is continuous, it is also possible for the gate to be partially opened (i.e. somewhere between 0 and 1). Models incorporated sigmoid transfer functions often help generalized learning characteristics and yield models with improved accuracy. use of sigmoid transfer functions can also lead to longer training times The objective of System Design is the specification of modules and the ways these modules are to be integrated to form a complete software module fulfilling its design objectives in the System Design, high-level abstraction of the whole module

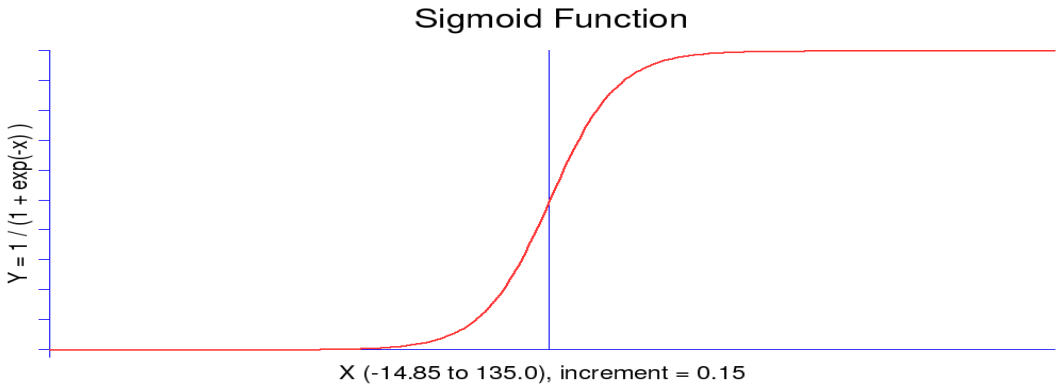


Figure 1: Sigmoid Function identification

is provided using diagram called Data Flow Diagram.

3. Implementation

As we described previously the different parameters have to be taken care of before starting implementation. The different parameters include:

1. The learning rate
2. The error tolerance
3. The momentum factor

These are defined using the statics statements and can be appropriately set for controlling the networks behavior. There are two functions that effectively control the training and prediction parts of the program.

1) HDR_PTR train_NN (HDR_PTR hp, FILE *fp)

a) This function starts by accepting the address of the first neuron in the network and the address of the Input file. Then it calls the function

2) HDR_PTR feed_network(HDR_PTR HP, FILE *fp)

a) This function reads data from the input file and initializes the network.

b) HDR_PTR back_propagation(HDR_PTR HP, float *targets, int propagation)

c) This function back propagates through the network to adjust the weights until the desired target is achieved.

3) HDR_PTR use_NN(HDR_PTR hp, FILE *fp)

This function retrieves the weights stored after successful training and makes prediction of data which were not used in training.

For an online transaction processing work load, performance is typically measured in Transactions per second (TPS) and Responsetime.

4. System Design

Train the Network:

The Training Session is basically divided into two parts, i.e. Output layer training and Hidden layer training.

Initially the input data 'data' is read from the file and assigned to the Input Neurons as follows:-

Input $_{i, j}$ = data (for i = 1, i.e. in the case of Input Layer

(Where i denote the layer, j denotes the current neuron)

For Hidden and Output layer

Input $_{i, j, h}$ = Output $_{i-1, h}$

(where *h* = 1... the number of neurons in the immediate upper layer 'i-1')

The next consecutive data are stored in the target array as follows:-

Targets $_{k}$ = data

(where *k*=1...number of output)

Then the weighted sum is calculated

and output is determined as follows:-

$wt\_sum_{i,j} = \Sigma (Input_{i,j} * Weight_{i,j})$

$Output_{i,j} = Sigmoid (wt\_sum_{i,j}$

(where $sigmoid(x) = 1/(1+e^{-x})$)

Output Layer Training:

Calculate the Error as follows:-

$Error_{i,j} = Targets_{j} - Output_{i,j}$

Now adjust the weights to minimize the error as follows

$\Delta wt_{i,j,h} = \alpha * \Delta wt_{i,j,h} + Error_{i,j} * \eta * derv\_sigmoid (wt\_sum_{i,j}) * Output_{i,j}$

$Weight_{i,j,h} = Weight_{i,j,h} + \Delta wt_{i,j,h}$

(where i denote the output layer, h = 1... no. of neurons in the immediate upper layer, $derv\_sigmoid(x) = e^{-x}/(1+e^{-x})^{2)}$

Find the relative error as follows

$REL\_ERROR = ABS (Error_{i,j}) / Output_{i,j}$

The above steps are to be carried out until the REL_ERROR is less than a predefined 'Err_Tol'.

Hidden Layer Training:

Calculate the local gradient for each neuron in the hidden layers

$locGrad_{i,j} = locGrad_{i,j} + Error_{i+1,j} * derv\_sigmoid (wt\_sum_{i+1,j}) * Weight_{i+1,j}$

$Error_{i,j} = locGrad * derv\_sigmoid (Wt\_sum_{i,j})$

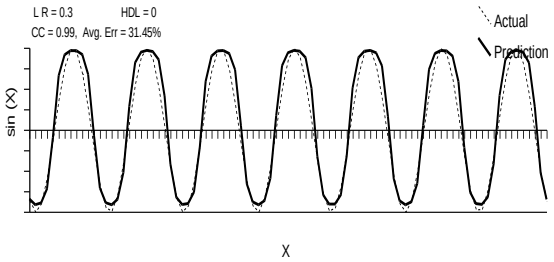
Following segment computes the change in weights for each I/P connection and adjusts the weights accordingly.

$\Delta wt_{i,j,h} = \Sigma * Error_{i,j} * Output_{i-1,j}$

$Weight_{i,j,h} = Weight_{i,j,h} + \Delta wt_{i,j,h}$;

(where h = 1 ... no. of neurons in the immediate upper layer, j = 1... no. of neurons in the i^{th} layer)

Now with the adjusted weights



is calculated in the following manner

$Output_{i,j} = sigmoid (\Sigma (Input_{i,j,h} * Weight_{i,j,h)})$

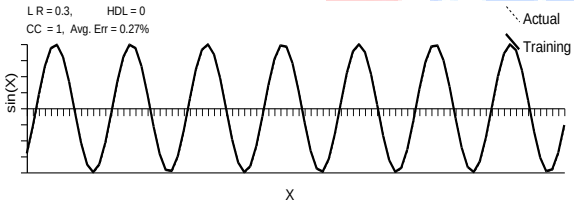
(where h=1 ... number of neurons in the previous layer)

The Output layer training and Hidden layer training are to be repeated while the number of iteration is lesser than a predefined value PROPAGATION. When the training is successful the current weights of the neurons are stored are saved in a file.

5. Testing with Sine Curve:

This stage is a simple stage. The network is trained with sine wave (A sin ωt). The sine wave consists of values with an interval of 18 degrees. The network is trained up to 180 degrees and is required to predict any of the following cycles with minimum error.The network is fed with 10 inputs of 10 consecutive values and is required to predict the 11^{th} values in succession of the same interval. The sigmoid function was used as the transfer function.The network configuration is kept as given in the following table and its behaviour is tested by changing the network parameters (Learning Rate, No. of Hidden Layers, etc). The parameters were set as follows:

Training Mode:



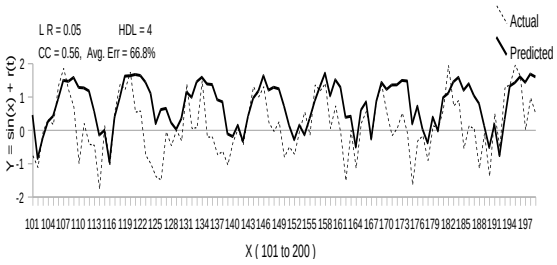
Prediction Mode:

6. Testing with Sine Curve with Random Noise:

A random noise between (-1, 1) is added to the values of A sin ωt and then it is trained with different combination of network parameters. After each set of training, the values of the same (series generated by adding the random noise to the sine wave) are predicted and the results are shown in the following graphs.

Training Mode:

Prediction Mode :



Year	*Correlation-Coefficient*	*Phase (%)*	*Average Absolute Error*	*Average Relative Error*	*Stand. Dev for Observations*	*Stand. Dev for Prediction*
2007	0.64	58.82	1.9	279.35	2.63	2.08
2008	0.70	56.86	2.3	254.46	2.91	3.96
2009	0.70	58.17	1.83	330.72	2.64	2.28
2010	0.62	50.98	2.09	266.61	2.6	1.7
2011	0.65	57.52	1.89	295.91	2.53	1.99
2012	0.71	53.59	2.27	338.4	2.94	3.77
2013	0.62	58.82	1.88	200.19	2.56	1.97
MEAN	**0.66**	**56.39**	**2.02**	**280.8**	**2.68**	**2.53**

Table : 1 - Depicting the Details of the Year Wise Prediction of Overall India

AREA				CC	Phase (%)	Abs Avg Error	Rel Avg Error	Std Dev (Obs)	Std Dev (Pred)
Lat 1	Lat 2	Lon 1	Lon2						
10.5	25.5	76.5	85.5	0.31	55.05	2.28	241.47	2.87	1.97
12.5	23.5	78.5	83.5	0.26	62.09	4.15	336	4.09	1.63
16.5	19.5	80.5	81.5	0.23	53.96	7.18	378.13	8.06	2.19
18.5	19.5	80.5	81.5	0.21	61.87	8.48	403.84	10.9	3.7
14.5	21.5	80.5	81.5	0.25	58.09	6.1	389.84	6.48	2.2

Table 2 : Summary Table for rainfall anomaly prediction in average of five areas

7. Results:

From the graphs we have seen that performance of the neural network is different for different combination of network parameters. It performs better with 0 hidden layers. Whereas its performance suffers with 1 or more hidden layers. Also hidden layers add extra computational overhead to the performance of the neural network. The learning rate (η) plays a significant role in its performance. When the neural network is trained slowly i.e. with lower values of η then the predictions are more accurate than those done with higher values of η.

From the graphs we have seen that it performs better with multiple numbers of hidden layers. Whereas its performance suffers with lesser number of hidden layers. Although hidden layers add extra computational overhead to the performance of the neural network but it is necessary to add hidden layer due the accuracy reasons. So we have to choose an optimum set of network configuration in which the computational overheads are less and the accuracy of the results is also more.

8. Conclusion and Future Enhancement :

In this result the application of Neural Network in Rainfall Assessment was explored. The data were studied in depth. Various configurations of the network were studied and the effect of these changes on the final prediction were carefully noted and systematically analyzed. The input parameters used for the simulation as well as the results obtained were carefully tabulated, analyzed, and later used with modification for designing the next set of simulation. The result of the present study demonstrated the potential for ANN development in the field of rainfall forecasting. However, it was observed that the ANN could be sensitive to the choice of input data. Through the paper several advantages of the Artificial Neural Network were realized. Perhaps the dominant advantage is adaptability. This adaptability is also coupled to ANN's ability to generalize, capture complexity, particularly the nonlinearity of the rainfall data.

9. References :

1. Simon Haykin, "Neural Network A Comprehensive Foundation", Second Edition, Pearson Education India, 1999
2. Kevin L.Priddy and Paul E.Keller, "An Introduction to Neural Networks".
3. Christopher M. Bishop, "Neural Networks for Pattern Recognition", Oxford University Press
4. Goswami and Srividya, 1997: Cognitive Network for advanced forecasting of all India Seasonal Rainfall (June - September).
5. Navone et al, 2001: Prediction of seasonal rainfall.
6. http://www.tek271.com/articles/neurelNet/IntoToNeuralNets.html
7. http://www.doc.ic.ac.uk./~nd/surprise_96/journal/vol4/cs11/report.html
8. http://www.learnartificialneuralnetworks.com/
9. http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,57545,00.html
10. https://www.dacs.dtic.mil/techs/neural/neural_ToC.php