I.J. Intelligent Systems and Applications, 2013, 12, 1-22
Published Online November 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2013.12.01

Prediction of Rainfall in India using Artificial Neural Network (ANN) Models

[1]Santosh Kumar Nanda, [2]Debi Prasad Tripathy, [3]Simanta Kumar Nayak, [4]Subhasis Mohapatra

[1]Centre of Research, Development and Consultancy, Eastern Academy of Science and Technology, Bhubaneswar, Odisha– 754001, India
[2]Professor, Department of Mining Engineering, National Institute of Technology, Rourkela, Odisha, India
[3]Department of Computer Science and Engineering, Eastern Academy of Science and Technology, Bhubaneswar, Odisha-754001, India
[4]Department of Information Tehnology, Eastern Academy of Science and Technology, Bhubaneswar, Odisha-754001, India
E-mail: santoshnanda@live.in; debi_tripathy@yahoo.co.in; simanta.nayak@eastodissa.ac.in; subhasis22@gmail.com

***Abstract*—** In this paper, ARIMA(1,1,1) model and Artificial Neural Network (ANN) models like Multi Layer Perceptron (MLP), Functional-link Artificial Neural Network (FLANN) and Legendre Polynomial Equation (LPE) were used to predict the time series data. MLP, FLANN and LPE gave very accurate results for complex time series model. All the Artificial Neural Network model results matched closely with the ARIMA(1,1,1) model with minimum Absolute Average Percentage Error(AAPE). Comparing the different ANN models for time series analysis, it was found that FLANN gives better prediction results as compared to ARIMA model with less Absolute Average Percentage Error (AAPE) for the measured rainfall data.

***Index Terms*—** Autoregressive Integrated Moving Average Model, ARIMA, Autocorrelation Function, FLANN, MLP, Legendre neural Network (LeNN)

I. Introduction

Rain is very important for life. All living beings need water to live. Rainfall is a major component of the water cycle and is responsible for depositing most of the fresh water on the Earth. It provides suitable conditions for many types of ecosystem as well as water for hydroelectric power plants and crop irrigation. The occurrence of extreme rainfall in a short time causes serious damage to economy and sometimes even loss of lives due to flood. Insufficient rainfall for long period causes drought. This can effect to economic growth of developing countries. Thus, rainfall estimation is very important because of its effects on human life, water resources and water usage. However, rainfall affected by the geographical and regional variations and features is very difficult to estimate.

Some Researchers have carried out rainfall estimation, Using Sigmoid Polynomial Higher Order Neural Network (SPHONN) Model [1] and that gives better rainfall estimation than Multiple Polynomial Higher Order Neural Network (M-HONN) model and Polynomial Higher Order Neural Network (PHONN) models. [1]

As the next step, the research will focus more on developing automatic higher order neural network models. Monthly Rainfall are estimated Using Data-Mining Process [2] of Isparta. The monthly rainfall of Senirkent, Uluborlu, Eğirdir, and Yalvaç stations were used to develop rainfall estimation models. When comparing the developed models output to measured values, multilinear regression model from data-mining process gave more appropriate results than the developed models. The input parameters of the best model were the rainfall values of Senirkent, Uluborlu, and Eğirdir stations. Consequently, it was shown that the data-mining process, produced a better solution than the traditional methods, can be used to complete the missing data in estimating rainfall. Various techniques used to identify patterns in time series data (such as smoothing, curve fitting techniques and auto-correlations). The authors proposed to introduce a general class of models that can be used to represent the time series data and predict data using autoregressive and moving average models. Models for time series data can have many forms and represent different stochastic processes. When modeling variations in the level of a process, three broad classes of practical importance are the autoregressive (AR) models, the integrated (I) models, and the moving average (MA) models. These three classes depend linearly on previous data points. Combinations of these ideas produce autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) models. [4]

II. Motivation

Many researchers had investigated the applicability of ARIMA model to find the estimation value of rainfall

in a specific area in particular period of time such as ARIMA Models for weekly rainfall in the semi-arid Sinjar District at Iraq [1-3].They collected weekly rainfall record spanning the period of 1990-2011 for four stations (Sinjar, Mosul, Rabeaa and Talafar) at Sinjar district of North Western Iraq to develop and test the models. The performance of the resulting successful ARIMA models was evaluated by using the data for the year 2011 through graphical comparison between the forecast and actually recorded data. The forecasted rainfall data showed very good agreement with the actual recorded data.

This gave an increasing confidence of the selected ARIMA models. The results achieved for rainfall forecasting will help to estimate hydraulic events such as runoff, so that water harvesting techniques can be used in planning the agricultural activities in that region. Predicted excess rain can be stored in reservoirs and used in a later stage, but there are so many disadvantages using ARIMA model so it can only be used when the time series is Gaussian. However, if the time series is not Gaussian, a transformation has to be applied before these models can be used, however, such transformation does not always work. Another disadvantage is that ARIMA models are non-static and cannot be used to reconstruct the missing data.

III. Present Work

In this research work the authors propose to develop a new approach based on the application of an ARIMA with other applications like Artificial Neural Network (ANN), Legendre Polynomial Equation, Functional Link Artificial Neural network (FLANN) and Multilayer Perceptron (MLPs) to estimate yearly rainfall.

IV. ARIMA Model

In time series analysis, the Box–Jenkins methodology, named after the statisticians George Box and Gwilym Jenkins, applies autoregressive moving average ARMA or ARIMA models to find the best fit of a time series to past values of this time series, in order to make forecasts. This approach possesses many appealing features. To identify a perfect ARIMA model for a particular time series data, Box and Jenkins (1976) [12] proposed a methodology that consists of four phases viz.

A. Model identification.

B. Estimation of model parameters.

C. Diagnostic checking for the identified model appropriateness for modelling.

D. Application of the model (i.e. forecasting).

Step A. In the identification stage, one uses the IDENTIFY statement to specify the response series and identify candidate ARIMA models for it. The IDENTIFY statement reads time series that are to be used in later statements, possibly differencing them, and computes autocorrelations, inverse autocorrelations, partial autocorrelations, and cross correlations. Stationary tests can be performed to determine if differencing is necessary. The analysis of the IDENTIFY statement output usually suggests one or more ARIMA models that could be fit.

Step B & C. In the estimation and diagnostic checking stage, one uses the ESTIMATE statement to specify the ARIMA model to fit to the variable specified in the previous IDENTIFY statement, and to estimate the parameters of that model. The ESTIMATE statement also produces diagnostic statistics to help one judge the adequacy of the model. Significance tests for parameter estimates indicate whether some terms in the model may be unnecessary.

Goodness-of-fit statistics aid in comparing this model to others. Tests for white noise residuals indicate whether the residual series contains additional information that might be utilized by a more complex model. If the diagnostic tests indicate problems with the model, one may try another model, and then repeat the estimation and diagnostic checking stage.

Step D. In the forecasting stage one uses the FORECAST statement to forecast future values of the time series and to generate confidence intervals for these forecasts from the ARIMA model produced by the preceding ESTIMATE statement.

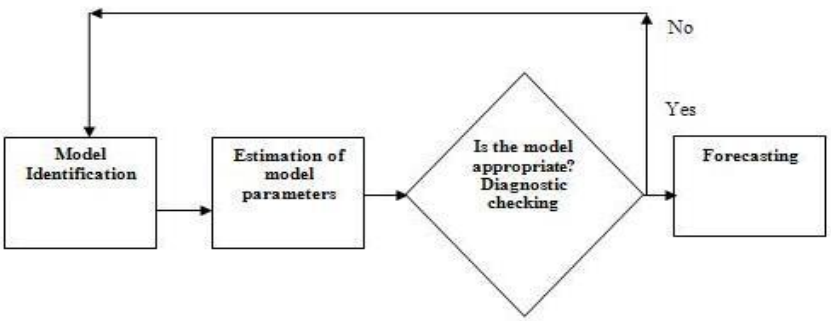

Fig. 1: Outline of Box-Jenkins Methodology

The most important analytical tools used with the time series analysis and forecasting are the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF). They measure the statistical relationships between observations in a single data series. Using ACF gives big advantage of measuring the amount of linear dependence between observations in a time series that are separated by a lag k. The PACF plot is used to decide how many auto regressive terms are necessary to expose one or more of the time lags where high correlations appear, seasonality of the series, trend either in the mean level or in the variance of the series [5]. In order to identify the model (step A), ACF and PACF have to be estimated. They are used not only to help guess the form of the model, but also to obtain approximate estimates of the parameters [6].

The next step is to estimate the parameters in the model (step B) using maximum likelihood estimation. Finding the parameters that maximize the probability of observations is the main goal of maximum likelihood. The next, is checking on the adequacy of the model for the series (step C). The assumption is the residual is a white noise process and that the process is stationary and independent.

The ARIMA model is an important forecasting tool, and is the basis of many fundamental ideas in time-series analysis. An autoregressive model of order p is conventionally classified as AR (p) and a moving average model with q terms is known as MA (q). A combined model that contains p autoregressive terms and q moving average terms is called ARMA (p,q). If the object series is differenced d times to achieve stationary, the model is classified as ARIMA (p, d, q), where the symbol "I" signifies "integrated". Thus, an ARIMA model is a combination of an autoregressive (AR) process and a moving average (MA) process applied to a non-stationary data series. So the general non-seasonal ARIMA (p, d, q) model is as:

- AR: p = order of the autoregressive part,
- I: d = degree of differencing involved
- MA: q = order of the moving average part.

The equation for the simplest ARIMA (p, d, q) model is as follows:

$$Y_t = C + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots\ldots + \phi_p Y_{t-p} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} \ldots\ldots - \theta_p e_{t-p} \quad (1)$$

V. ARIMA (0, 1, 0) = Random Walk

In the models mentioned earlier, it was encountered two strategies for eliminating autocorrelation in forecast errors. For example, suppose one initially fits the random-walk-with-growth model to the time series Y. The prediction equation for this model can be written as:

$$\hat{Y}(t) - Y(t-1) = \mu \quad (2)$$

Where the constant term (here denoted by "mu") is the average difference in Y. This can be considered as a degenerate regression model in which DIFF(Y) is the dependent variable and there are no independent variables other than the constant term. Since it includes (only) a nonseasonal difference and a constant term, it is classified as an "ARIMA(0,1,0) model with constant." Of course, the random walk without growth would be just an ARIMA(0,1,0) model *without* constant.[12]

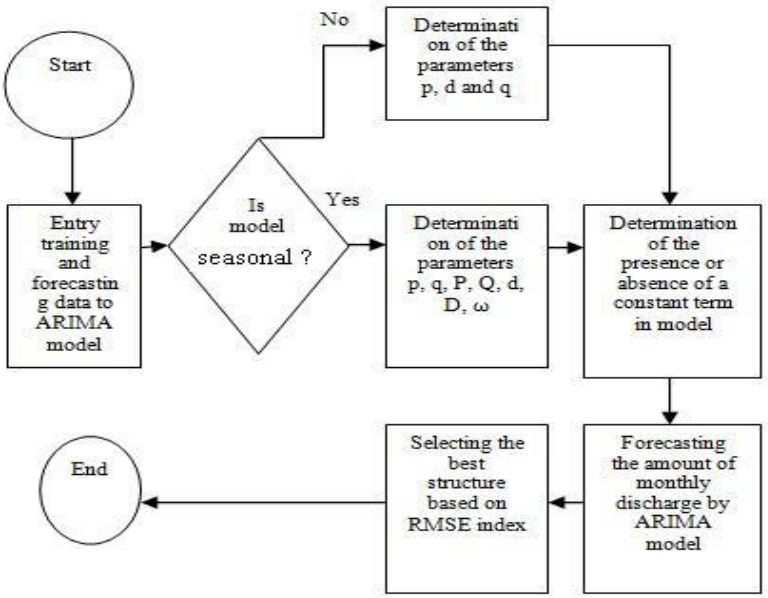

Fig. 2: ARIMA (p,d,q) flowchart

VI. ARIMA (1, 1, 0) = Differenced First-Order Autoregressive Model

If the errors of the random walk model are auto correlated, perhaps the problem can be fixed by adding one lag of the dependent variable to the prediction equation i.e., by regressing DIFF(Y) on itself lagged by one period. This would yield the following prediction equation:-

$$\hat{Y}(t) - Y(t-1) = \mu + \Phi\,(Y(t-1) - Y(t-2)) \quad (3)$$

which can be rearranged to:-

$$\hat{Y}(t) = \mu + Y(t-1) + \Phi\,(Y(t-1) - Y(t-2)) \quad (4)$$

This is a first-order autoregressive, or "AR(1)", model with one order of nonseasonal differencing and a constant term i.e., an "ARIMA(1,1,0) model with constant." Here, the constant term is denoted by "mu" and the autoregressive coefficient is denoted by "phi", in keeping with the terminology for ARIMA models popularized by Box and Jenkins. (In the output of the Forecasting procedure in Statgraphics, this coefficient is simply denoted as the AR(1) coefficient.[4]

VII. ARIMA (0, 1, 1) without Constant = Simple Exponential Smoothing

Another strategy for correcting autocorrelated errors in a random walk model is suggested by the simple exponential smoothing model. Recall that for some nonstationary time series (e.g., one that exhibits noisy fluctuations around a slowly-varying mean), the random walk model does not perform as well as a moving average of past values. In other words, rather than taking the most recent observation as the forecast of the next observation, it is better to use an *average* of the last few observations in order to filter out the noise and more accurately estimate the local mean. The simple exponential smoothing model uses an *exponentially weighted moving average* of past values to achieve this effect. The prediction equation for the simple exponential smoothing model can be written in a number of mathematically equivalent ways, one of which is:

$$\hat{Y}(t) = Y(t-1) - \theta\, e(t-1) \quad (5)$$

Where (t-1) denotes the error at period t-1. Note that this resembles the prediction equation for the ARIMA (1,1,0) model, except that instead of a multiple of the lagged difference it includes *a multiple of the lagged forecast error*. (It also does not include a constant term yet.) The coefficient of the lagged forecast error is denoted by the Greek letter "theta" (again following Box and Jenkins) and it is conventionally written with a *negative* sign for reasons of mathematical symmetry. "Theta" in this equation corresponds to the quantity "1-minus-alpha" in the exponential smoothing formulas. When a lagged forecast error is included in the prediction equation as shown above, it is referred to as a "moving average" (MA) term. The simple exponential smoothing model is therefore a first-order moving average ("MA(1)") model with one order of nonseasonal differencing and no constant term i.e., an "ARIMA(0,1,1) model without constant."

This means that in Statgraphics (or any other statistical software that supports ARIMA models) one can actually fit a simple exponential smoothing by specifying it as an ARIMA(0,1,1) model without constant, and the estimated MA(1) coefficient corresponds to "1-minus-alpha" in the SES formula.

VIII. ARIMA (0, 1, 1) with Constant = Simple Exponential Smoothing with Growth

By implementing the SES model as an ARIMA model, you actually gain some flexibility. First of all, the estimated MA(1) coefficient is allowed to be *negative*: this corresponds to a smoothing factor larger than 1 in an SES model, which is usually not allowed by the SES model-fitting procedure. Second, you have the option of including a constant term in the ARIMA model if you wish in order to estimate an average non-zero trend. The ARIMA(0,1,1) model *with* constant has the prediction equation:

$$\hat{Y}(t) = \mu + Y(t-1) - \theta\, e(t-1) \quad (6)$$

The one-period-ahead forecasts from this model are qualitatively similar to those of the SES model, except that the trajectory of the long-term forecasts is typically a sloping line (whose slope is equal to mu) rather than a horizontal line.

IX. ARIMA(0,2,1) Or (0,2,2) without Constant = Linear Exponential Smoothing

Linear exponential smoothing models are ARIMA models which use two nonseasonal differences in conjunction with MA terms. The second difference of a series Y is not simply the difference between Y and itself lagged by two periods, but rather it is the *first difference of the first difference* i.e., the change-in-the-change of Y at period t. Thus, the second difference of Y at period t is equal to –:

$$(Y(t)-Y(t-1)) - (Y(t-1)-Y(t-2))$$
$$= Y(t) - 2Y(t-1) + Y(t-2) \quad (7)$$

A second difference of a discrete function is analogous to a second derivative of a continuous function: it measures the "acceleration" or "curvature" in the function at a given point in time. The ARIMA(0,2,2) model without constant predicts that the

second difference of the series equals a linear function of the last two forecast errors:

$$\hat{Y}(t) - 2Y(t-1) + Y(t-2) = -\theta_1 e(t-1) - \theta_2 e(t-2) \quad (8)$$

which can be rearranged as:

$$\hat{Y}(t) = 2Y(t-1) - Y(t-2) - \theta_1 e(t-1) - \theta_2 e(t-2) \quad (9)$$

Where theta-1 and theta-2 are the MA(1) and MA(2) coefficients. This is essentially the same as Brown's linear exponential smoothing model, with the MA(1) coefficient corresponding to the quantity 2*(1-alpha) in the LES model. To see this connection, recall that forecasting equation for the LES model is:

$$\hat{Y}(t) = 2Y(t-1) - Y(t-2) - 2(1-\alpha)e(t-1) + (1-\alpha)^2 e(t-2) \quad (10)$$

Upon comparing terms, we see that the MA(1) coefficient corresponds to the quantity 2*(1-alpha) and the MA(2) coefficient corresponds to the quantity -(1-alpha)^2 (i.e., "minus (1-alpha) squared"). If alpha is larger than 0.7, the corresponding MA(2) term would be less than 0.09, which might not be significantly different from zero, in which case an ARIMA(0,2,1) model probably would be identified.

X. A "Mixed" Model - ARIMA(1,1,1)

The features of autoregressive and moving average models can be "mixed" in the same model. For example, an ARIMA(1,1,1) model with constant would have the prediction equation:

$$\hat{Y}(t) = \mu + Y(t-1) + \varphi(Y(t-1) - Y(t-2)) - \theta e(t-1) \quad (11)$$

Normally, the authors plan to stick to "unmixed" models with either only-AR or only-MA terms, because including both kinds of terms in the same model sometimes leads to over fitting of the data and non-uniqueness of the coefficients.

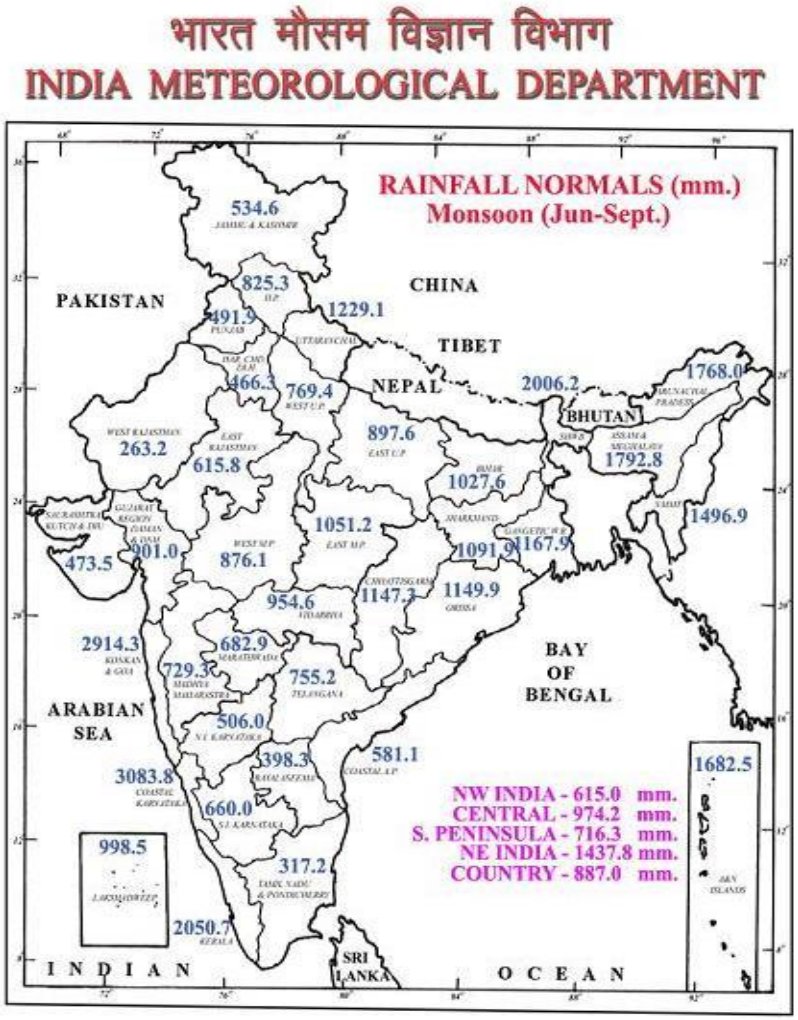

Fig. 3: Rainfall over INDIA in (June-Sept (2012)) [7]

XI. Results And Discussion

The data was chosen as a sample of calculations followed by Fig. 4 as shown in table1.

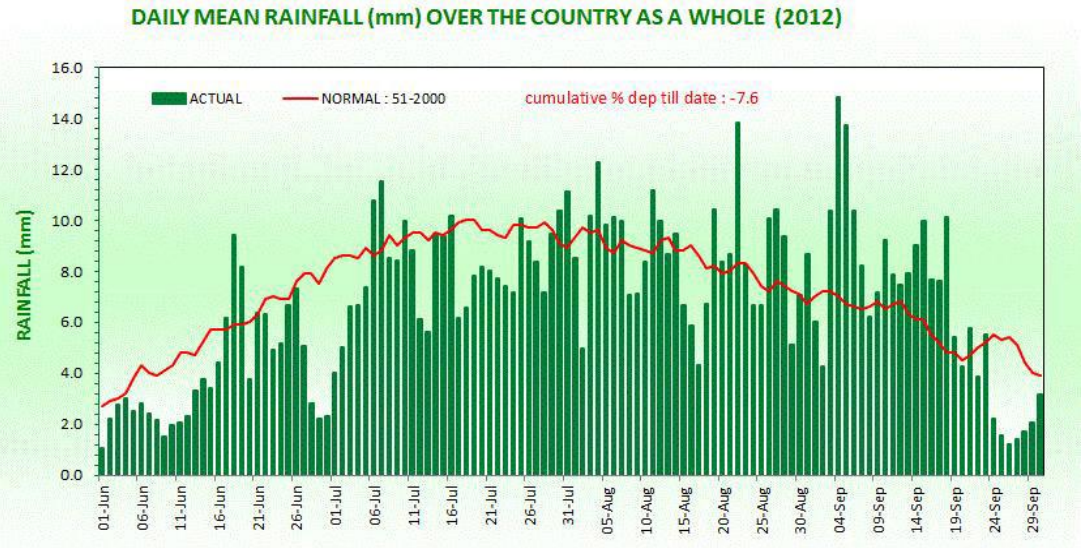

Fig. 4: DAILY MEAN RAINFALL (mm) OVERTHE COUNTRY AS WHOLE (Jun-sep-2012) [8]

Table 1: Rainfall data (June-Sept 2012)

Day	June	July	Aug	Sep
1	1	4	8.6	6.3
2	2.3	5.1	5	4.4
3	2.8	6.6	10.2	10.5
4	3.1	6.9	12.3	14.6
5	2.6	7.5	9.8	13.8
6	2.8	10.8	10.2	10.5
7	2.5	10.6	10	8.2
8	2.3	8.6	7	6.5
9	1.5	8.5	7.2	7.2
10	1.9	10	8.4	9.3
11	2	8.8	11	7.9
12	2.4	6.2	10	7.5
13	3.4	5.7	8.6	8
14	3.8	9.4	9.4	8.6
15	3.5	9.3	6.3	10
16	4.5	10.2	6	7.5
17	6.2	6.2	4.5	7.6
18	9.4	6.5	6.8	10.3
19	8.2	7.8	10.6	5.5
20	3.8	8.2	8.4	4.4
21	6.4	8	8.6	5.9
22	6.3	7.8	13.8	3.9
23	5	7.5	8.4	5.6
24	5.2	7.2	6.2	2.4
25	6.6	10.1	6.2	1.6
26	7.3	9.2	10.2	1.3
27	5	8.4	10.6	1.5
28	2.9	7.2	9.8	1.8
29	2.3	9.5	5.2	2.2
30	2.4	10.4	7.2	3.3
31		11.2		

XII. Detail Analysis Of ARIMA(1,1,1) Model

$$Y(t) = \mu + Y(t-1) + \emptyset(Y(t-1)) - \theta(\varepsilon(t-1)) \quad (12)$$

ARIMA(1,1,1) EQUATION:

$Y(t) = \mu + Y(t-1) + \emptyset(Y(t-1) - Y(t-2)) - \theta(\varepsilon(t-1))$

Parameter Values

$\mu = 0.03216, \emptyset = 0.5486, \theta = 0.9585$

$\emptyset$ is the autocorrelation coefficient

θ is the exponential smoothing

$\varepsilon = 2.718$ is the error which was calculated from the previous value.

Table 2: Predicted data using ARIMA(1,1,1) Model

SL. NO.	T	ACTUAL Y(T)	PREDICTED Y'(T)	ERROR (E)
1	1	6.3	6.3	0
2	2	4.4	9.7	5.3
3	3	10.5	1.7	8.7
4	4	14.6	8	6.6
5	5	13.8	16.18	-2.4
6	6	10.5	19.1	-8.6
7	7	8.2	8	0.2
8	8	6.5	12.5	-6
9	9	7.2	4.3	2.9
10	10	9.3	8.4	0.9
11	11	7.9	13.57	-5.67
12	12	7.5	6.82	0.68
13	13	8	11	-3
14	14	8.6	10.34	-1.74
15	15	10	11.68	-1.68
16	16	7.5	13.9	-6.4
17	17	7.6	5.5	2.1
18	18	10.3	9.78	0.52
19	19	5.5	15.4	-9.9
20	20	4.4	1	3.4
21	21	5.9	3.58	2.32
22	22	3.9	6.94	-3.04
23	23	5.6	3.15	2.45
24	24	2.4	6.3	-3.9
25	25	1.6	0.01	1.59
26	26	1.3	1	0.3
27	27	1.5	1.11	0.39
28	28	1.8	1.98	-0.18
29	29	2.2	2.64	-0.4
30	30	3.3	3.05	0.25
Absolute value of Average % of Error				47.7

Table 3: ARIMA Model: C1 estimates at each iteration

ITERATION	SSE	PARAMETERS		
0	361.973	0.1	0.1	0.152
1	354.918	0.047	0.153	0.058
2	350.117	0.196	0.303	0.049
3	344.932	0.342	0.453	0.041
4	339.027	0.485	0.603	0.033
5	331.42	0.621	0.753	0.025
6	319.289	0.738	0.903	0.019
7	295.975	0.588	0.974	0.035
8	294.094	0.567	0.969	0.028
9	293.744	0.558	0.965	0.03
10	293.605	0.553	0.962	0.031
11	293.564	0.55	0.96	0.032
12	293.564	0.55	0.96	0.032
13	293.564	0.549	0.959	0.032
14	293.564	0.549	0.958	0.032

Unable to reduce sum of squares any further

Final Estimates of Parameters

Type	Coef	SE Coef	T	P
AR 1	0.5486	0.0988	5.55	0
MA 1	0.9585	0.0467	20.51	0
Constant	0.03216	0.01453	2.21	0.029

Differencing: 1 regular difference

Number of observations: Original series 91, after differencing 90

Residuals:

SS = 291.059 (backforecasts excluded)

MS = 3.346

DF = 87

Table 4: Modified Box-Pierce (Ljung-Box) Chi-Square statistic

Lag	12	24	36	48
Chi-Square	10.7	30.2	45.8	74
DF	9	21	33	45
P-Value	0.298	0.087	0.069	0.004

Period	Forecast	Lower	Upper Actual
92	8.3439	4.7582	11.9296
93	9.0036	4.8402	13.167
94	9.3977	5.0332	13.7621
95	9.646	5.1962	14.0957
96	9.8143	5.321	14.3077
97	9.9389	5.4187	14.459
98	10.0393	5.4999	14.5788
99	10.1266	5.5714	14.6818
100	10.2067	5.6375	14.7758
101	10.2827	5.7006	14.8648
102	10.3566	5.7621	14.9511
103	10.4293	5.8227	15.036
104	10.5014	5.8828	15.12
105	10.573	5.9426	15.2035
106	10.6445	6.0023	15.2868
107	10.7159	6.062	15.3699
108	10.7872	6.1216	15.4529
109	10.8585	6.1812	15.5358
110	10.9298	6.2409	15.6187
111	11.001	6.3006	15.7015
112	11.0723	6.3603	15.7843
113	11.1435	6.42	15.8671
114	11.2148	6.4798	15.9498
115	11.286	6.5396	16.0325
116	11.3573	6.5994	16.1152
117	11.4285	6.6592	16.1979
118	11.4998	6.7191	16.2805
119	11.571	6.779	16.3631
120	11.6423	6.8389	16.4457
121	11.7135	6.8988	16.5282

Forecasts from period 91

95% Limits

The first step in the application of the methodology is to cheek whether the time series (monthly rainfall) is stationary and has seasonality. The monthly rainfall data (Fig. 5) shows that there is a seasonal cycle of the series and it is not stationary. The entire ARIMA Model is developed by using Matlab 16.

The plots of ACF and PACF of the original data (Fig. 6 & 7) show that the rainfall data is not stationary.

A stationary time series has a constant mean and has no trend over time. However it could satisfy stationary in variance by having lag transformation and satisfy stationary in the mean by having differencing of the original data in order to fit an ARIMA model. The Autocorrelation Function for monthly Rainfall and the Partial Autocorrelation Function for monthly Rainfall are shown in Fig.7.

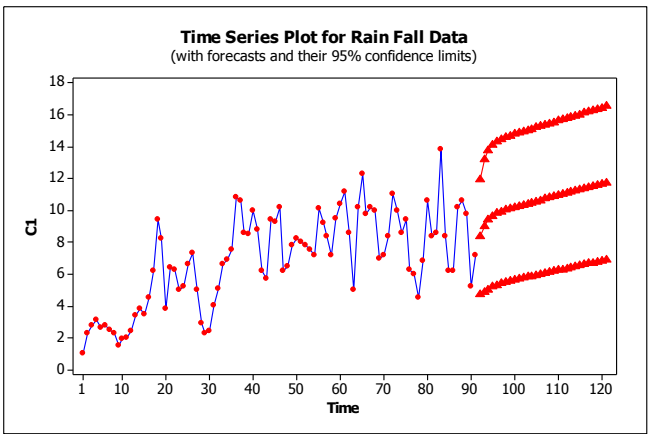

Fig. 5: Time series rainfall data for the period (Jun-Sep in 2012)

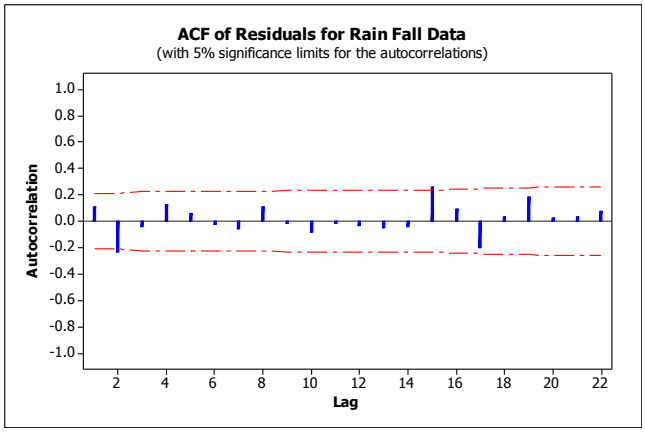

Fig. 6: ACF for monthly rainfall data

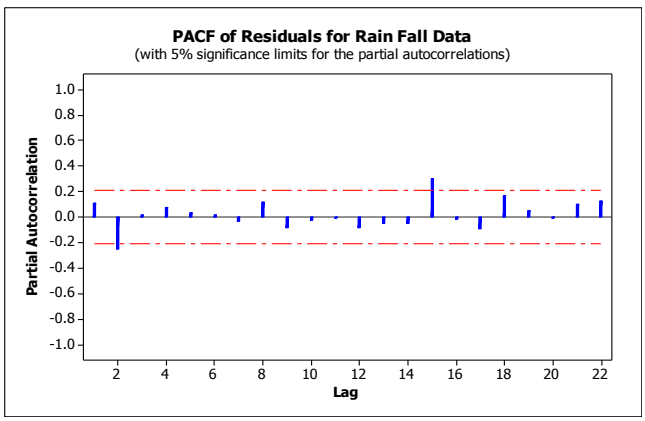

Fig. 7: PACF for monthly rainfall data

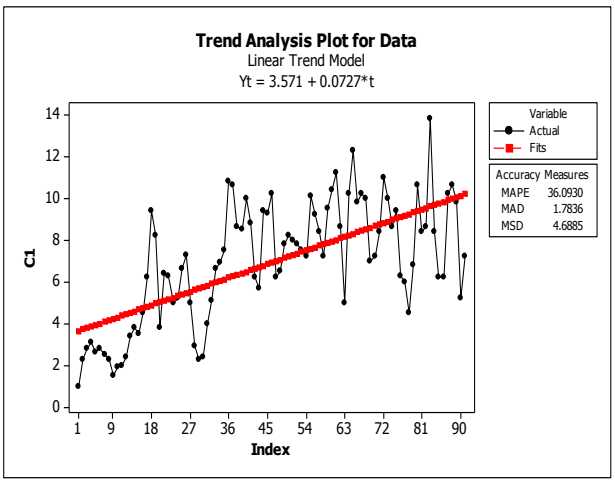

Fig. 8: Represents Trend Analysis of the rainfall data

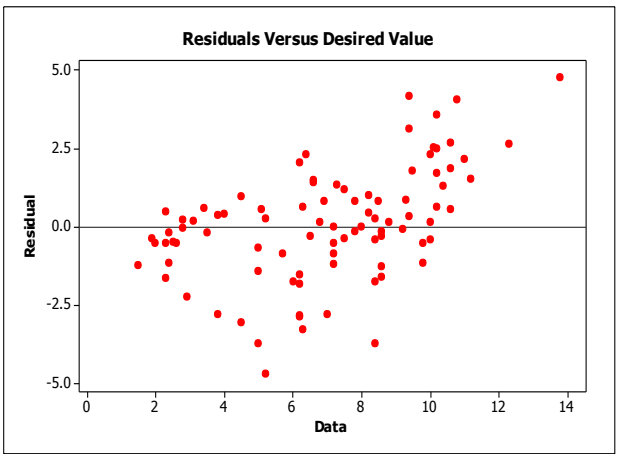

Fig. 9: Represents Residuals associated with ARIMA Model

XIII. Artificial Neural Network (ANN)

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. A neural network can be trained to perform a particular function by adjusting the values of the connections (weights) between the elements. Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown in Figure 10.

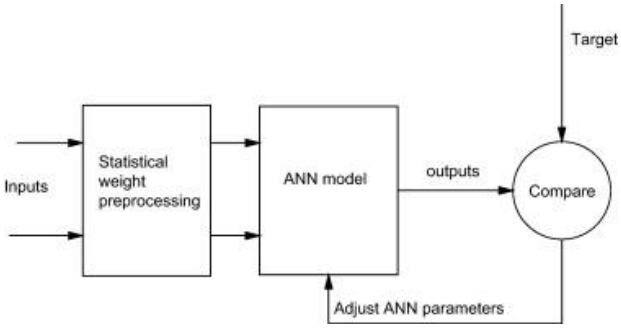

Fig. 10: Basic principle of artificial neural networks

Here, the network is adjusted, based on a comparison of the output and the target, until the sum of square differences between the target and output values becomes the minimum. Typically, many such input/target output pairs are used to train a network. Batch training of a network proceeds by making weight and bias changes based on an entire set (batch) of input vectors. Incremental training changes the weights and biases of a network as needed after presentation of each individual input vector. Neural networks have been trained to perform complex functions in various fields of application including pattern recognition, identification, classification, speech, vision, and control systems.

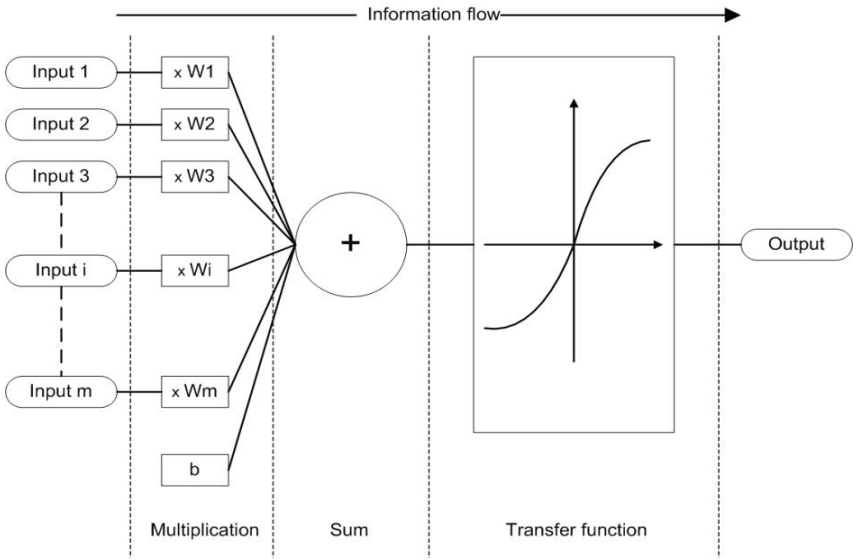

Fig. 11: Working principle of an artificial neuron

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model (function).Such a model has three simple sets of rules: multiplication, summation and activation. At the entrance of artificial neuron the inputs are weighted what means that every input value is multiplied with individual weight. In the middle section of artificial neuron is sum function that sums all weighted inputs and bias.

At the exit of artificial neuron the sum of previously weighted inputs and bias is passing through activation function that is also called transfer function (Fig.11).

Although the working principles and simple set of rules of artificial neuron looks like nothing special the full potential and calculation power of these models come to life when interconnected into artificial neural networks (Fig.12). These artificial neural networks use simple fact that complexity can be grown out of merely few basic and simple rules.

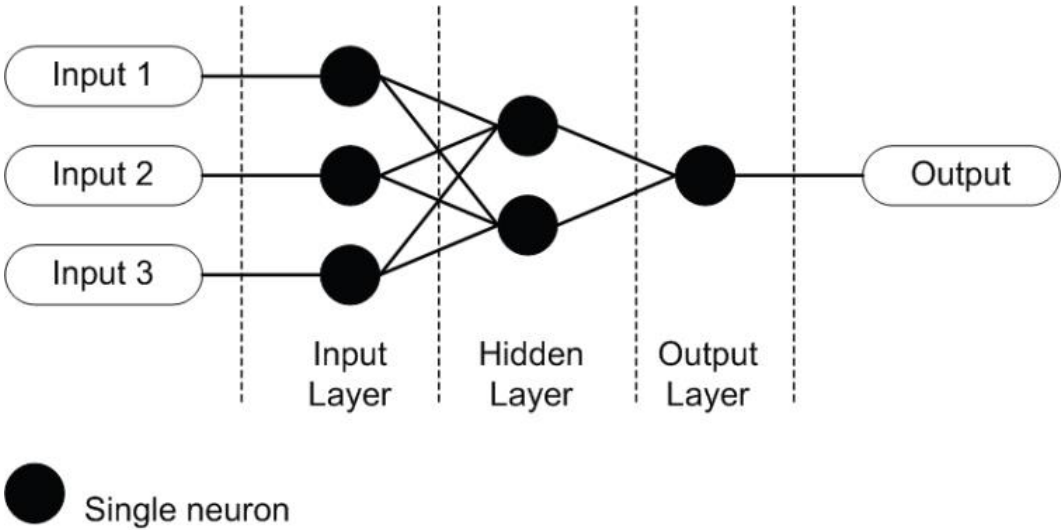

Fig. 12: Example of simple Artificial Neural Network

In order to fully harvest the benefits of mathematical complexity that can be achieved through interconnection of individual artificial neurons and not just making system complex and unmanageable we usually do not interconnect these artificial neurons randomly. In the past, researchers have come up with several "standardized" topographies of artificial neural networks. These predefined topographies can help us with easier, faster and more efficient problem solving. Different types of artificial neural network topographies are suited for solving different types of problems. After determining the type of given problem we need to decide for topology of artificial neural network we are going to use and then fine-tune it. One needs to fine-tune the topology itself and its parameters. Fine tuned topology of artificial neural network does not mean that one can start using our artificial neural network, it is only a precondition. Before one can use artificial neural network one need to analyze it solving the type of given problem. Just as biological neural networks can learn their behavior/responses on the basis of inputs that they get from their environment the artificial neural networks can do the same. There are three major learning paradigms: supervised learning, unsupervised learning and reinforcement learning. Learning paradigms are different in their principles they all have one thing in common; on the basis of "learning data" and "learning rules" (chosen cost function) artificial neural network is trying to achieve proper output response in accordance to input signals. After choosing topology of an artificial neural network, fine-tuning of the topology and when artificial neural network has learnt a proper behavior one can start using it for solving a given problem. Artificial neural networks have been in use for some time now and one can find them working in areas such as process control, chemistry, gaming, radar systems, automotive industry, space industry, astronomy, genetics, banking, fraud detection, etc. and solving of problems like function approximation, regression analysis, time series prediction classification, pattern recognition, decision making, data processing, filtering, clustering, etc.[9]

XIV. Types of Activation Functions in ANN

There are a number of activation functions that can be used in ANNs such as sigmoid, threshold, linear etc. An activation function is defined by $\Phi(v)$ and defines the output of a neuron in terms of its input v. There are three types of activation functions.

1. Threshold function an example of which is

$$\phi(\upsilon)=\begin{cases}1 & if\ \upsilon\geq 0\\ 0 & if\ \upsilon<0\end{cases} \quad (13)$$

This function is also termed the *Heaviside* function.

2. Piecewise Linear

$$\phi(\upsilon)=\begin{cases}1 & if\ \upsilon\geq +1/2\\ V & if\ \ 1/2>\upsilon>-1/2\\ 0 & if\ \ \upsilon\leq -1/2\end{cases} \quad (14)$$

3. Sigmoid Examples include

3.1 Logistic function whose domain is [0,1]

$$\phi(\upsilon)=\frac{1}{1+\exp(-\alpha\upsilon)} \quad (15)$$

3.2 The hyperbolic tangent whose domain is [-1,-1]

$$\varphi(\upsilon)=\frac{1-\exp(-\alpha\upsilon)}{1+\exp(-\alpha\upsilon)}=\tanh(\alpha\upsilon/2) \quad (16)$$

3.3 Algebraic sigmoid function whose domain is [-1,-1]

$$\varphi(\upsilon)=\frac{\upsilon}{\sqrt{1++\upsilon^2}} \quad (17)$$

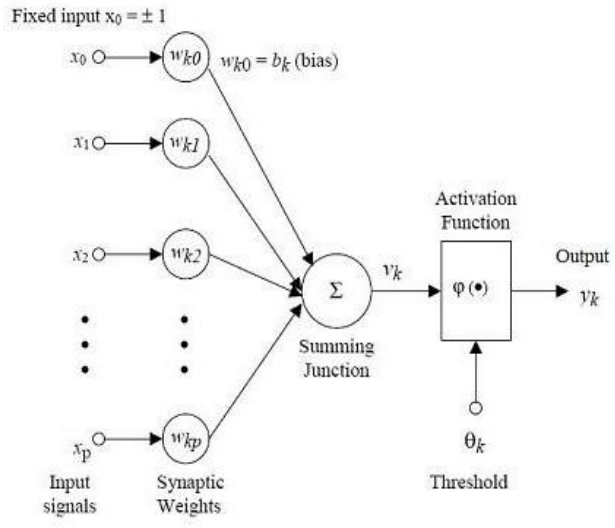

Fig. 13: Working principle of Activation Function

The Back-Propagation Algorithm

The Back-propagation algorithm [10] is used in layered feed-forward ANNs. This means that the artificial neurons are organized in layers, and send their signals "forward", and then the errors are propagated backwards. The network receives inputs by neurons in the input layer, and the output of the network is given by the neurons on an output layer. There may be one or more intermediate hidden layers as shown in (Fig.12). The Back-propagation algorithm uses supervised learning, which means that the algorithm is provided with examples of the inputs and outputs that the network is expected to compute, and then the error (difference between actual and expected results) is calculated. The idea of the Back-propagation algorithm is to reduce this error, until the ANN learns the training data. The training begins with random weights and the goal is to adjust them so that the error will be minimal.

XV. Multi Layer Perceptron (MLP)

An MLP is a network of simple neurons called perceptron. The perceptron computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights and then possibly putting the output through some nonlinear activation function. Mathematically this can be written as

$$y=\varphi\left(\sum_{i=1}^{n}\omega_i x_i+b\right)=\varphi(W^T x+b) \quad (18)$$

Where W denotes the vector of weights, X is the vector of inputs; b is the bias and α is the activation function. A signal-flow graph of this operation is shown in Fig. 14.

The original Rosenblatt's perceptron used a Heaviside step function as the activation function α. Nowadays, and especially in multilayer networks, the activation function is often chosen to be the logistic sigmoid $1/(1+e^{-x})$ or the hyperbolic tangent tanh(x).

They are related by $(\tanh(x)+1)/2=1/(1+e^{-2x})$ .These functions are used because they are mathematically convenient and are close to linear near origin while saturating rather quickly when getting away from the origin. This allows MLP networks to model well both strongly and mildly nonlinear mappings.

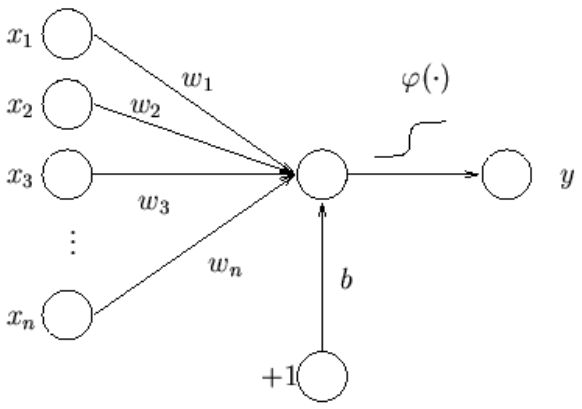

Fig. 14: Signal-flow graph of the perceptron

A single perceptron is not very useful because of its limited mapping ability. No matter what activation function is used, the perceptron is only able to represent an oriented ridge-like function. The perceptron can, however, be used as building blocks of a larger, much more practical structure. A typical multilayer perceptron (MLP) network consists of a set of source nodes forming the *input layer*, one or more *hidden layers* of computation nodes, and an *output layer* of nodes. The input signal propagates through the network layer-by-layer. The signal-flow of such a network with one hidden layer is shown in Figure 14.

The computations performed by such a feed forward network with a single hidden layer with nonlinear activation functions and a linear output layer can be written mathematically as

$$X = f(s) = B\,\varphi\,(As + a) + b \qquad (19)$$

Where S is a vector of inputs and X a vector of outputs. A is the matrix of weights of the first layer, a is the bias vector of the first layer. B and b are, respectively, the weight matrix and the bias vector of the second layer. The function φ denotes an element wise nonlinearity. The generalization of the model to more hidden layers is obvious.

While single-layer networks composed of parallel perceptron are rather limited in what kind of mappings they can represent, the power of an MLP network with only one hidden layer is surprisingly large. MLP networks are typically used in *supervised learning* problems. This means that there is a training set of input-output pairs and the network must learn to model the dependency between them. The training here means adapting all the weights and biases (A,B,a and b) in Equation (19) to their optimal values for the given pairs (S(t),X(t)). The criterion to be optimised is typically the squared reconstruction error $\sum_t \| f(s(t)) - x(t) \|^2$.

The supervised learning problem of the MLP can be solved with the *back-propagation algorithm*. The algorithm consists of two steps. In the *forward pass*, the predicted outputs corresponding to the given inputs are evaluated as in Equation (18). In the *backward pass*, partial derivatives of the cost function with respect to the different parameters are propagated back through the network. The chain rule of differentiation gives very similar computational rules for the backward pass as the ones in the forward pass. The network weights can then be adapted using any gradient-based optimisation algorithm. The whole process is iterated until the weights have converged.

The MLP network can also be used for unsupervised learning by using the so called *auto-associative* structure. This is done by setting the same values for both the inputs and the outputs of the network. The extracted sources emerge from the values of the hidden neurons. This approach is computationally rather intensive. The MLP network has to have at least three hidden layers for any reasonable representation and training such a network is a time consuming process.

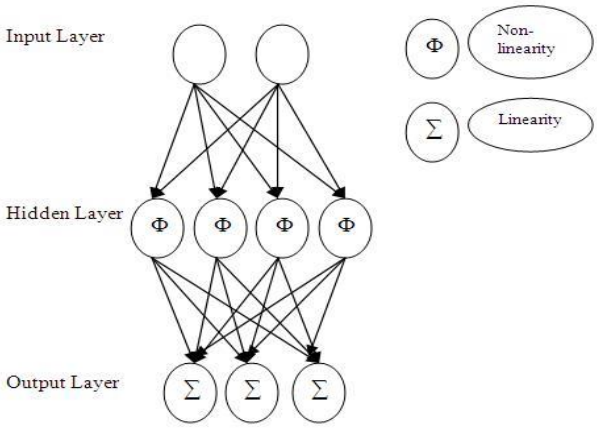

Fig. 15: Signal-flow graph of an MLP

XVI. Functional-Link Artificial Neural Network (FLANN)

Neural Network (NN) represents an important paradigm for classifying patterns or approximating complex non-linear process dynamics. These properties clearly indicate that NN exhibit some intelligent behavior, and are good candidate models for non-linear processes, for which no perfect mathematical model is available. Neural networks have been a powerful tool for there applications for more than last two decades [11-15]. Multilayer Perceptron (MLP), Radial Basis Function (RBF), Support vector machine (SVM) etc. are the types of Neural Network Model, where these models have better prediction competence with high computational cost. Generally, these models have high computational cost due to the availability of hidden layer. To minimize the computational cost, structures like, functional link artificial neural network (FLANN) [16] and Legendre Neural Network (LeNN) [17-18] were proposed. Three types of functional based artificial neural networks were applied to estimate rain fall. These included: Multi layer Perceptron (MLP), Functional Link Artificial Neural Network (FLANN) and Legendre Neural Network (LeNN).In general the functional link based neural network models were single-layer ANN structure possessing higher rate of convergence and lesser computational load than those of an MLP structure. The mathematical expression and computational calculation is evaluated as per MLP.

J. C Patra (2008) originally proposed Functional link artificial neural network (FLANN) and it is a novel single layer ANN structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries .In FLANN, the hidden layers are removed. Further, the FLANN structure

offers less computational complexity and higher convergence speed than those of an MLP because of its single layer structure. The FLANN structure is depicted in Fig. 16. Here, the functional expansion block make use of a functional model comprising of a subset of orthogonal sin and cos basis functions and the original pattern along with its outer products. For example, considering a two-dimensional input pattern X = [x1x2]T . The enhanced pattern is obtained by using the trigonometric functions as X*= [x1 cos (πx1) sin(πx1) . . . x2 cos(πx2) sin(πx2) . . . x1x2]T which is then used by the network for the equalization purpose. The BP algorithm, which is used to train the network, becomes very simple because of absence of any hidden layer.

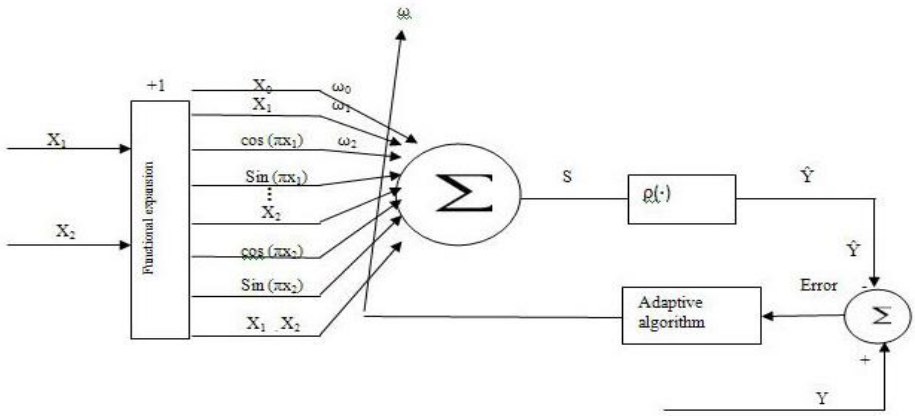

Fig. 16: FLANN structure

XVII. Legendre Polynomial Equation in ANN

Structure of the Legendre Neural Network [16-18] (LeNN) (Fig.17) is similar to FLANN.

In contrast to FLANN, in which trigonometric functions are used in the functional expansion, LeNN uses Legendre orthogonal functions. LeNN offers faster training compared to FLANN.

The performance of this model may vary from problem to problem. The Legendre polynomials are denoted by Ln(X), where n is the order and -1<x<1 is the argument of the polynomial. The zero and the first order Legendre polynomials are, respectively, given by $L_0(x) = 1$ and $L_1(x) = x$. The higher order polynomials are given by: $L_2(x) = 1/2(3x^2-1)$, $L_3(x) = 1/2\ (5x^3 - 3x)$ etc.

Polynomials are generated by using the following mathematical expression

$$L_{n+1}(x) = 1/n+1[(2n + 1)\ x\ L_n(x) - nL_{n-1}(x)] \quad (20)$$

Similar to FLANN, the two dimensional input pattern $X=[x_1\ x_2]T$ is enhanced to a seven dimensional pattern by Legendre functional expansion

$$X_e = [1, L_1(x_1),\ L_2(x_1),\ L_3(x_1),\ L_1(x_2), L_2(x_2),\ L_3(x_2)] \quad (21)$$

For Legendre neural network, the training is carried out in the same manner as FLANN and PPN. In all models, supervised learning is used. As in normal Artificial Neural Network techniques, presence of hidden layers increases the complexity in the real time system, therefore, FLANN and LeNN is suitably used due to less computational cost.

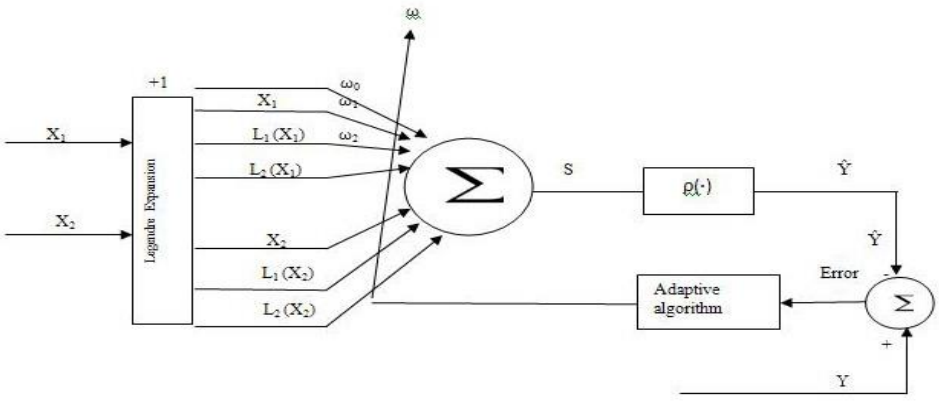

Fig. 17: Legendre Neural Network Structure

XVIII. Development of ANN Based Rainfall Estimation Model

The artificial neural network-based rainfall estimation models consist of three input parameters: Initialize the input data Ø, θ, Y_{t-1} and the desired output patterns are: $d(t)$, $d2(t)$, $d3(t) \cdots dn(t) \in R$. During training period, the desired network output was calculated with ARIMA(1,1,1) model. Since the procedures of these three models were similar; therefore, one algorithm is presented here to emphasize the development of functional-based artificial neural network-based rainfall estimation models. Fig.18 graphically represent the algorithm of functional-based artificial neural network based rainfall estimation models.

Step-1: Start

Step- 2: Initialize the input data are Ø, θ, Y_{t-1}

where Ø =0.5486, θ =0.9585, μ =0.03216

$$X_1 = [Y_{t-1}, \theta, Ø, \mu, \varepsilon(t-1), \varepsilon(t-2)]$$

$$Y(t) = \mu + Y(t-1) + Ø(Y(t-1)) - \theta(\varepsilon(t-1)) \quad (22)$$

Step-3: Initialize the weighting parameters wi, for i = 1, 2, . . . l, where "i" is the number of functional elements.

Step-4: (Produce functional blocks). For FLANN the functional block was made as follows:

$$Xi = [1, x_1, \sin(\pi x_1), \cos(\pi x_1), x_2, \sin(\pi x_2), \cos(\pi x_2) \ldots] \quad (23)$$

For MLP the functional block was made as follows by initialize the weighting parameters

$w^m_{i,j}$,for i=1,2,…n. and m=2 (number of layers)

$$w^m_{i,j} \longleftarrow w^m_{i,j}(0) \quad (24)$$

For LeNN the functional block was made as follows:

$$Xi = [1, x_1, L1(x_1), x_2, L1(x2), \ldots], \quad (25)$$

Where $L1(x) = 1/2(3x^2-1)$, $L2(x) = 1/2(5x^3 - 3x)$, and so forth.

Step-5: (Calculation of the system outputs). For functional based neural network models, the output was calculated as follows:

$$O_i = \sum_{i=1}^{N} w_i * X_i \quad (26)$$

For MLP Calculate the output of the system

$$a^m_{i,j} = (w^1_{i,j})^T * X_k \text{ and } y_i = (w^2_{i,j})^T * a^m_j \quad (27)$$

Step-6: (Calculation of the output error). The error was calculated as

$$ei = di - Oi$$

Where d_j was the desired output and o_j was the estimated output of the system.

The error was calculated by a MLP as $e_j = d_j - y_j$

Where d_j was the desired output and y_j was the estimated output of the system

Step-7: (Updating the weight vectors). The weight matrixes were updated next using the following relationship:

$$wi(k+1) = wi(k) + \alpha ei(k)Xi(k), \quad (28)$$

where k was the time index and α was the momentum parameter.

Step-8: If error $\leq \varepsilon(0.01)$ then go to Step 9 otherwise, go to Step 3.

Step-9: After the, learning was complete, the weights were fixed, and the network can be used for testing.

Step-10: End

XIX. Simulation Result and Discussion

The proposed system models for rainfall estimation were validated using simulation studies. The studies were carried out by using MATLAB simulation environment. For validation of the models, the rainfall data was collected from India meteorological department (IMD) over of time period June to September 2012(India). The entire simulation was based on the algorithm of Functional-link Artificial Neural Network (FLANN), Multi Layer Perceptron (MLP) & Legendry polynomial equation(LPE) as per flow-chart given in Fig.18.

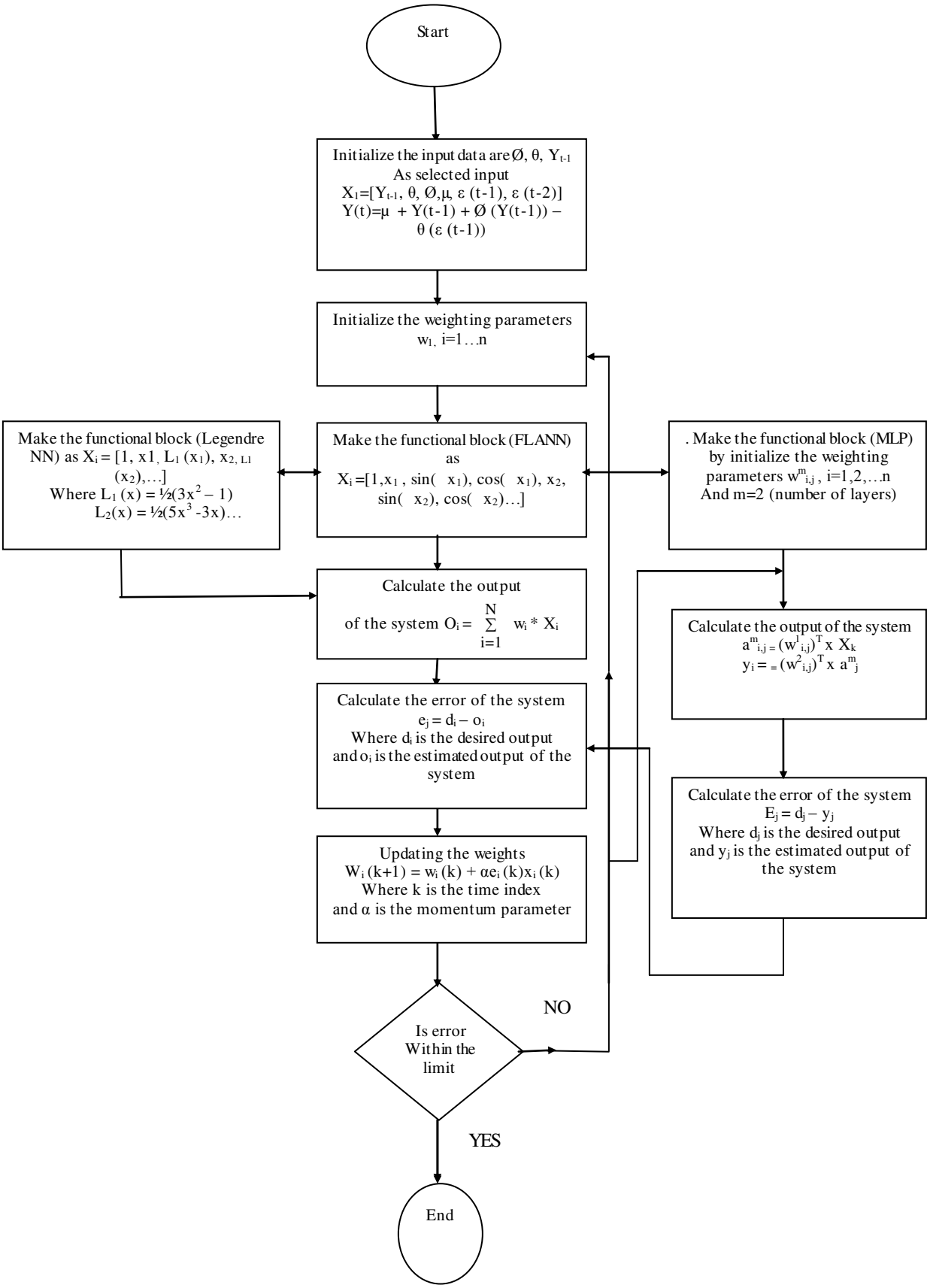

Fig.18: Systematic algorithm based neural network based Rainfall Estimation Model

XX. Algorithm of Multi Layer Perceptron (MLP)

Step-1: Start

Step-2: Randomly select the initial values of the weight vectors w^m_{ij}, for i = 1, 2…n_1, and m=2(number of layers).

Step-3: Initialization all the weights $w^m_{i,j}$ were initialized to random number and given as $w^m_{i,j}(0)$

$$w^m_{i,j} \leftarrow - w^m_{i,j}(0).$$

Step-4: Calculate the output of the system

$$a^m_{i,j} = (w^1_{i,j})^T \text{ x } X_k$$

$$y_i = = (w^2_{i,j})^T \text{ x } a^m_j$$

Where $a^m_{i,j}$ was the output of the hidden layer(m=1) and y_i is the output layer. $w^1_{i,j}$ is the weight associated for hidden layer and $w^2_{i,j}$ is the weight associated for output layer. In the model, the weight associated with the hidden layer were basically n_i X 2 matrix as two inputs were selected and the weight associated with the out layer is 1 x n_i as this model is a MISO(Multi input and single output) system.

Step-5: Calculate the error of the error.

The error was calculated as $e_j = d_j - y_j$ .It may be seen that the network procedure a scalar output.

Step-6: Calculate of the derivative of network output of each layer for hidden layer (m=1).

The derivative of activation function of hidden layer can be represented as

$$f^1(n^1) = d/dn(1/1+\exp^{-n})$$

$$= (1- 1/1+\exp^{-n}) \text{ x } (1/1+\exp^{-n}) = (1-a^m_{i,j})(a^m_{i,j})$$

For output layer (m=2)

The derivative of activation function of output layer can be represented as

$$f^2(n^2) = d/dn(n)=1$$

Where n was the output of each neuron of hidden and output layer.

Step-7: Back-propagation of error by sensitivities at each layer was calculated as follows:

For output layer (m=2)

$$S^2_j = -2F^2(n^2)(d_j - y_j) = -2\ f^2(n^2) \text{ x } (e_j)$$

For hidden layer (m=1)

Step-8: Updating the weight vectors

The weight matrices were updated next using the following relationship

$$w^2_{i,j}(new) = w^2_{i,j}(old) + ns^2_j(a^1_{ij})^T$$

$$w^1_{i,j}(new) = w^1_{i,j}(old) + ns^1_j$$

Where η was the momentum parameter or tuning parameter of the system

Step-9: If error <= ε (0.01) then go to step 10, otherwise go to step 3

Step-10: After the learning was completed, the weights were fixed and network can be used for testing.

XXI. Algorithm of Functional-Link Artificial Neural Network (FLANN)

Step-1: start

Step 2: Initialize the inputs data are Ø, θ, Y_{t-1}.

Step-3: Randomly select the initial values of the weight vectors w_i, for i = 1, 2…n.

Step-4: Make the functional block (FLANN) as

$$X_i = [1, x_1, \sin(x_1), \cos(x_1), x_2, \sin(x_2), \cos(x_2)\ldots]$$

Step-5: Calculate the output of the system

$$O_i = \sum_{i=1}^{N} w_i * X_i$$

Step-6: Calculation of the output error. The error was calculated as

$$ei = di - Oi$$

Where d_i was the desired output Where *Oi* estimated output of the system.

Step-7: Updating the weight vectors. The weight matrixes were updated next using the followingrelationship:

$$wi(k+1) = wi(k) + \alpha ei(k)Xi(k), \quad (29)$$

Where k was the time index and α is the momentum parameter.

Step-8: If error <= ε (0.01) then go to step 9, other wise go to step 2.

Step-9: After the, learning was completed, the weights were fixed, and the network can be used for testing.

Step-10: End

XXII. Algorithm of Legendre Polynomial Equation (LPE)

Step-1: Start

Step 2: Initialize the inputs data are Ø, θ, Y_{t-1}.

Step-3: Randomly select the initial values of the weight vectors w_i, for $i = 1, 2 \ldots n$.

Step-4: For LPE the functional block is made as follows:

$$Xi = [1, x_1, L_1(x_1), x_2, L_1(x_2), \ldots] \quad (30)$$

Where $L_1(x) = 1/2(3x^2-1)$, $L_2(x) = 1/2\ (5x^3 - 3x)$, and so forth.

Step-5: Calculate the output of the system

$$O_i = \sum_{i=1}^{N} w_i * X_i$$

Step-6: Calculation of the output error. The error was calculated as

$$ei = di - Oi$$

Where d_i was the desired output Where Oi estimated output of the system.

Step-7: Updating the weight vectors. The weight matrixes were updated next using the following relationship:

$$wi(k+1) = wi(k) + \alpha ei(k)Xi(k), \quad (31)$$

where k was the time index and α is the momentum parameter.

Step-8: If error <= ε (0.01) then go to step 9, otherwise go to step 2.

Step-9: After the, learning is complete, the weights were fixed, and the network can be used for testing.

Step-10: End

In this problem, the system is a MISO (multi input and output system) system. The system architectures of these proposed Artificial Neural Network models are same, whereas only the input pattern or functional blocks are different. To design these models, total number of 121 dataset were selected. Out of 121, 91 dataset were selected for training process and 30 data were selected for testing process. In this proposed systems, both iteration and epoch based training methods were applied.

In the proposed work, supervised learning methods were applied. From the simulation results Mean Square Error Plots of entire systemwas derived and plotted.

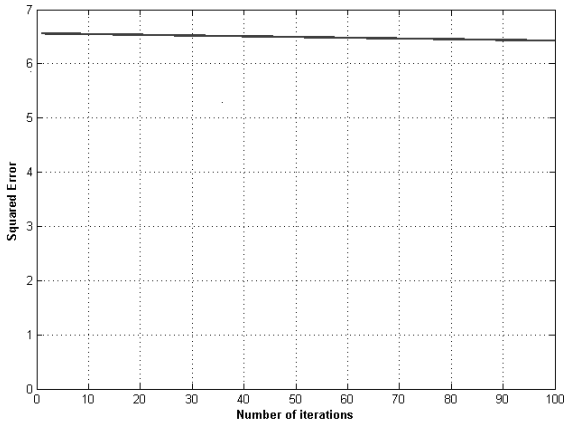

Fig. 19: Represents Mean Square Error for MLP

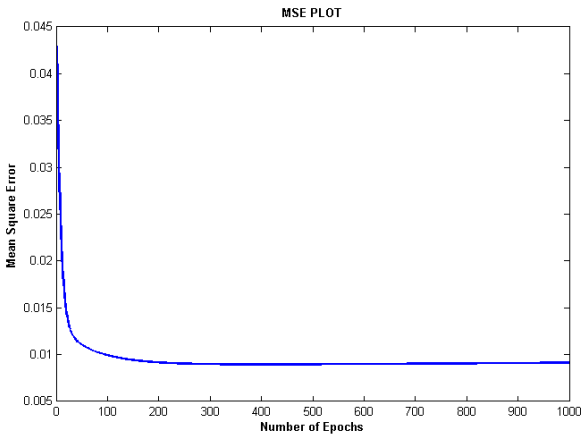

Fig. 20: Mean Square Error for FLANN

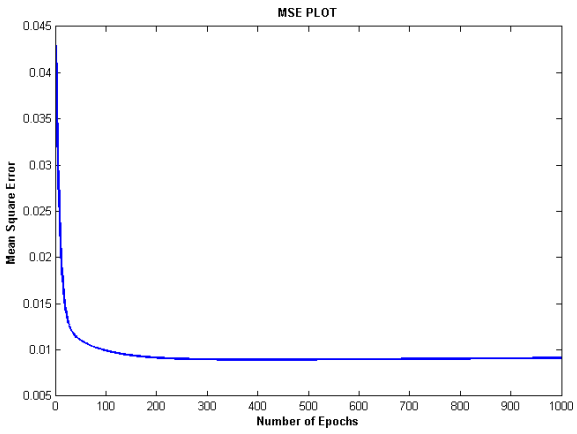

Fig. 21: Mean Square Error for LPE

Similarly Fig.20 & 21 represents Mean Square Error for FLANN and LPE respectively. Including this, performance studies of the proposed model for testing sample (30 samples) was made. After a good training result, having high convergence MSE plot was the updated weight associated with the error remains fixed and the model was freezed for its testing performance. The system was called a good system if its testing performance is within its error limit.

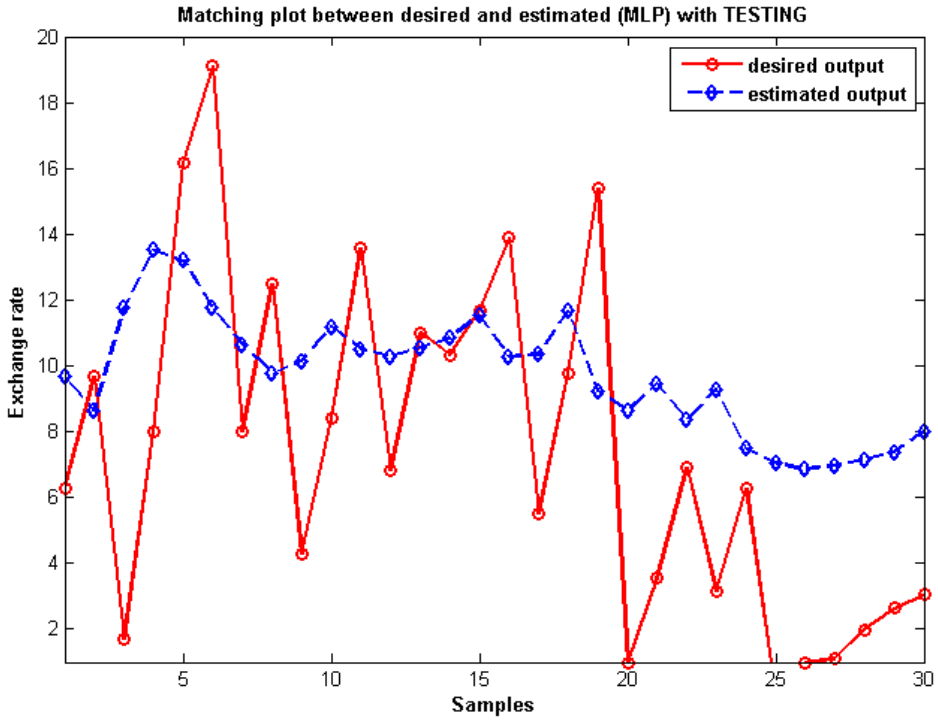

Fig. 22: The testing performance of the MLP system for 30 number of testing samples

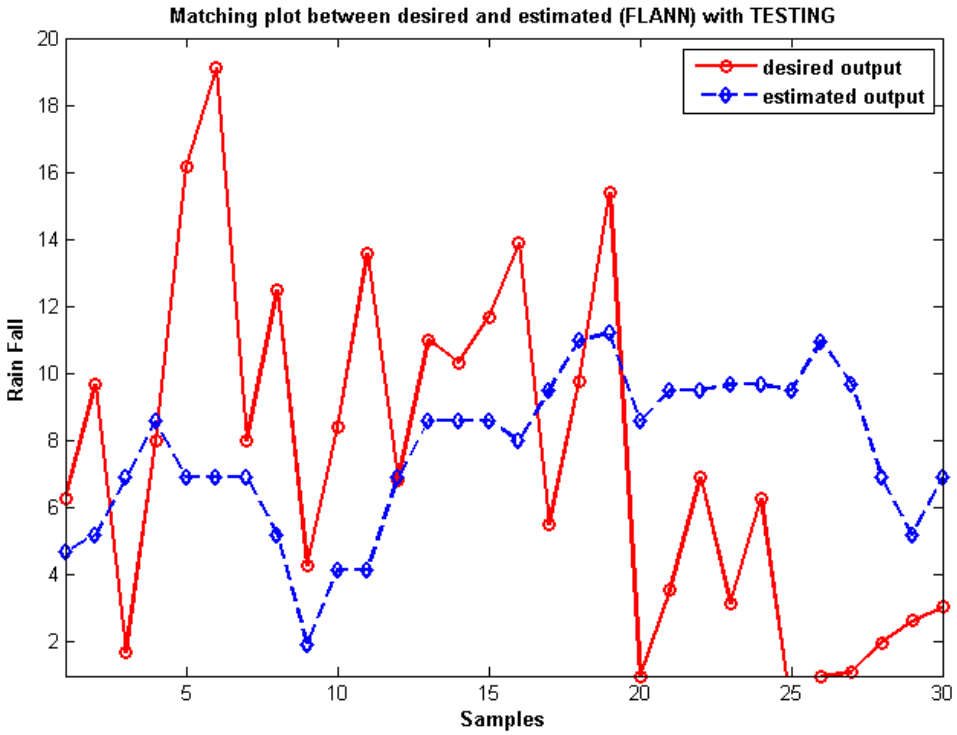

Fig. 23: The testing performance of the FLANN system for 30 number of testing samples

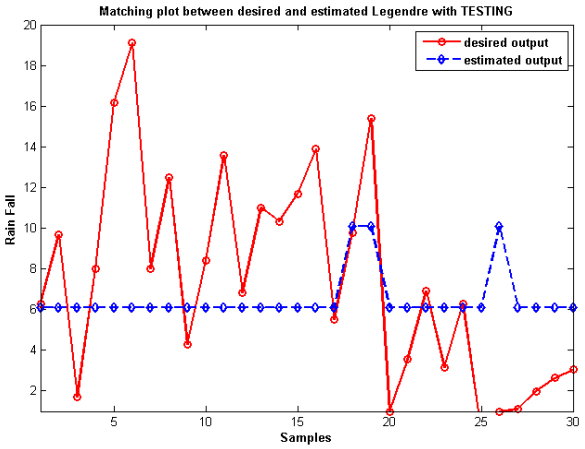

Fig. 24: represents the testing performance of the Legendre system for 30 number of testing samples

The Absolute Average Percentage Error (AAPE) was used as the performance index and was calculated as follows:

$$\text{Average Percentage Error} = \frac{\sum_{i=0}^{n=30} (\text{data})_i - (\text{Estimation})_i}{\sum_{i=0}^{N=30} (\text{data})_i} \times 100 \; ; \qquad (32)$$

Table 3: Absolute Error Analysis of MLP, FLANN & LPE over Actual Rainfall data (Sept-2012) ERROR PREDICTION TABLE

SL.NO	t	Actual Y(t)	ARIMA Error	MLP Error	FLANN Error	Legendre Polynomial Error
1	1	6.3	0	9.6682	4.6963	6.0814
2	2	4.4	5.3	8.6335	5.204	6.0814
3	3	10.5	8.7	11.7657	6.9045	6.0814
4	4	14.6	6.6	13.5263	8.5951	6.0814
5	5	13.8	-2.4	13.2059	6.9045	6.0814
6	6	10.5	-8.6	11.7657	6.9045	6.0814
7	7	8.2	0.2	10.6519	6.9045	6.0814
8	8	6.5	-6	9.7743	5.204	6.0814
9	9	7.2	2.9	10.141	1.9123	6.0814
10	10	9.3	0.9	11.1955	4.1676	6.0814
11	11	7.9	-5.67	10.5002	4.1676	6.0814
12	12	7.5	0.68	10.2959	6.9045	6.0814
13	13	8	-3	10.5509	8.5951	6.0814
14	14	8.6	-1.74	10.8518	8.5951	6.0814
15	15	10	-1.68	11.5311	8.5951	6.0814
16	16	7.5	-6.4	10.2959	7.9936	6.0814
17	17	7.6	2.1	10.3472	9.5055	6.0814
18	18	10.3	0.52	11.6724	10.9826	10.1277
19	19	5.5	-9.9	9.2384	11.2159	10.1277
20	20	4.4	3.4	8.6335	8.5951	6.0814
21	21	5.9	2.32	9.4544	9.5055	6.0814
22	22	3.9	-3.04	8.3537	9.5055	6.0814
23	23	5.6	2.45	9.2926	9.6794	6.0814
24	24	2.4	-3.9	7.4982	9.6794	6.0814
25	25	1.6	1.59	7.0334	9.5055	6.0814
26	26	1.3	0.3	6.8578	10.9698	10.1277
27	27	1.5	0.39	6.975	9.6794	6.0814
28	28	1.8	-0.18	7.1501	6.9045	6.0814
29	29	2.2	-0.4	7.3825	5.204	6.0814
30	30	3.3	0.25	8.0143	6.9045	6.0814
Absolute value of Average % of Error			47.7	31.0983= 14.92db	3.2093 = 4.94 db	12.716 =11.04 db

Table.3 summarized the results for Rainfall Estimation by proposed models and compared it with real time data. The Absolute Average Percentage Error was then represented in decibel to determine the exact performance between the proposed model. It was found that for Multilayer Perceptron (MLP) Absolute Average Percentage error (AAPE) is 31.0983 and corresponding 14.9db. Similarly for FLANN Absolute Average Percentage error(AAPE) was 3.2 and corresponding 4.94db and for Legendry Polynomial Equation(LPE) Absolute Average Percentage error(AAPE) was 12.716 and corresponding 11.04db.From the simulation result it was observed that the FLANN model had lower AAPE in db than the other two models.

XXIII. Conclusion

In this study, rainfall estimation was predicted using a complex statistical model ARIMA(1,1,1) and three different kind of Artificial Neural Network (ANNs) models, MLP, FLANN and LPE. The best statistical model for time series model was ARIMA and ARIMA(1,1,1) model was used for analysis for Rainfall Estimation data. Artificial Neural Network models viz MLP, FLANN and LPE were successfully applied for the complex time series models. Comparing the different ANN models for time series analysis, it was found that FLANN gives very close and better prediction result as compared to ARIMA model with less Absolute Average Percentage Error (AAPE).

References

[1] Ming Zhang and Jessica Crane. Rainfall Estimation Using *SPHONN* Model. May 15-17, Pittsburgh, USA. pp. 1-5. 2000.

[2] Ozlem Terzi. Monthly Rainfall Estimation Using Data-Mining Process. July 2012 Academic Editor: Tzung P. Hong.

[3] D.I.F. Grimes, E. Pardo-Igu´zquiza and R. Bonifacio TAMSAT.Optimal areal rainfall estimation using raingauges and satellite data. July 1999.

[4] Saleh Zakaria, Nadhir Al-Ansari, Sven Knutsson and Thafer Al-Badrany ARIMA Models for weekly rainfall in the semi-arid Sinjar District at Iraq vol. I, no. 4, 2012 25-ISSN: 1792-9040(print), 1792-9660 (online) International Scientific Press, 2011.

[5] Dayhoff E.J. Neural Network Architecture – An Introduction Van Norstand Reilold, New York, 1990.

[6] M. T.Hagan, H.B.Demuth and M.Beale, Neural Network Design Thomson Asia Pte. Ltd , Singapore,2002

[7] http://www.imd.gov.in/section/hydro/Monsoon.jpg

[8] http://www.imdpune.gov.in/mons_monitor/all-India.gif

[9] Girish Kumar Jha, Artificial Neural Networks, Indian Agriultural Research Institute, PUSA, New Delhi.

[10] Bose N.K. and Liang P., Neural Network Fundamentals with Graphs, Algorithms, Applications, TMH Publishing Company Ltd, 1998.

[11] Montgomery, D.C and L.A. Johnson, Forecasting and Time Series Analysis. McGraw-Hill Book Company, http://www.abebooks.com/Forecasting-Time Series-Analysis Montgomerouglas /1323032148/bd, McGraw-Hill, (1967).

[12] Pankratz, A., Forecasting With Univariate Box-Jenkins Models Concepts and Cases.John Wiley & Sons, Inc. New York, ISBN0-471-09023-9, pp: 414.

[13] B. Widrow, S.D. Sterns, Adaptive Signal Processing, Prentice-Hall, Inc. Engle-wood Cliffs, New Jersey, 1985.

[14] S. Haykin, Adaptive Filter Theory, Pearson Education Asia, 4th edition, 2002.

[15] S. Haykin, Neural Networks: A comprehensive foundation, Pearson Education Asia, 2nd Edition, 2002.

[16] J. C. Patra, W. C. Chin, P. K. Meher and G. Chakraborty, Legendre-FLANN-based nonlinear channel equalization in wireless communication system, in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 1826–1831, Singapore, October, 2008.

[17] S.K. Nanda, S.Panda, P Raj Sekhar Subudhi and R. K. Das A Novel Application of Artificial Neural Network for the Solution of Inverse Kinematics Controls of Robotic Manipulators, International Journal of Intelligent Systems and Applications, (IJISA), Hongkong, Vol.4, Issue 9, pp-81-91, 2012.

[18] S.K.Nanda and D.P.Tripathy, Application of Functional Link Artificial Neural Network for Noise Prediction in Mining Industry, International Journal of Advances of Fuzzy Logic System, USA, 2011, http://dx.doi.org/10.1155/2011/831261.

[19] Richard O. Duda, Peter E. Hart and David G. Stork, Pattern Classification, 2nd edition, JohnWiley & Sons, INC.2001.

[20] Y.H. Pao, Adaptive Pattern Recognition and Neural Networks, Addison Wesley, Reading, Massachusetts, 1989.

[21] Y.H. Pao, S. M. Phillips,D. J. Sobajic, Neural-net computing and intelligent control systems,Int. J. Conr. , vol. 56, no.2, pp.263-289, 1992

[22] Y.H. Pao, G.H. Park and D.J. Sobjic, Learning and Generalization Characteristics of the Random Vector function, Neuro Computation, vol.6, pp.163-180, 1994.

[23] Fu L.M., Hsu H.H. and Principe J.C., Incremental Back Propagation Learning Networks, IEEE Trans. on Neural Network, vol.7, no.3, pp.757-762, 1996.

[24] Zhao Q and Higuchi T., Evolutionary Learning of Nearest-Neighbor MLP, IEEE Trans. on Neural Network, vol.7, no.3, pp.762-768, 1996.

[25] J.-S.R. Jang, C.-T. Sun and E. Mizutan, Neuro-Fuzzy and Soft Computing, Prentice Hall of IndiaPrivate Limited, New Delhi, 2005.

[26] S. Haykin, Adaptive Filter Theory, Pearson Education, Inc., New Delhi, 2002.

[27] S.Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, Reading, MA, 1994.

[28] M. Hagan, H. Demuth and M. Beale, Neural Network Design, Thomson Learning, New Delhi, 2003.

[29] K. S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Network., 1, 4–27, 1990.

[30] Z. Xiang, G. Bi and T.-L. Ngoc, Polynomial perceptrons and their applications to fading channel equalization and co-channel interference suppression, IEEE Trans. Signal Processing, 42, Pp 470–2479, 1994.

[31] J.C. Patra and R.N. Pal, Functional link neural network-based adaptive equalization of nonlinear channels with QAM signal, In: Proceedings of IEEE International Conference on Systems, Man, Cybernetics (SMC1995), Vancouver, BC, Canada, October, pp. 2081–2086, 1995.

[32] J.-S. R. Jang, C.-T. Sun and E. Mizutan Neuro-Fuzzy and Soft Computing, Prentice Hall of India Private Limited, New Delhi, India, 2005.

[33] S. Haykin, Adaptive Filter Theory, Pearson Education, Inc., New Delhi, India, 2002.

[34] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, USA, 1994.

[35] M. Hagan, H. Demuth and M. Beale, Neural Network Design, Thomson Learning, New Delhi, India, 2003.

[36] J. C. Patra and R. N. Pal, Functional link artificial neural network-based adaptive channel equalization of nonlinear channels with QAM signal, In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 2081–2086, 1995.

[37] J. C. Patra and R. N. Pal, A functional link artificial neural network for adaptive channel equalization, Signal Processing, vol. 43, no. 2, pp. 181–195, 1995.

[38] J. C. Patra, R. N. Pal, R. Baliarsingh and G. Panda, Nonlinear channel equalization for QAM signal constellation using artificial neural networks, IEEE Transactions on Systems,Man, and Cybernetics, Part B, vol. 29, no. 2, pp. 262–271, 1999.

Authors' Profiles

Dr. Santosh Kumar Nanda served as Director of Centre of Research, Development and Consultancy of Eastern Academy of Science & Technology (EAST). Completed his PhD from National Institute of Technology, Rourkela. His research interests are Soft Computing, Artificial Intelligence, Image Processing, Prediction of machinery noise and vibration, Noise and vibration control, Mathematical modelling, Pattern Recognition. He has more than 50 research articles in reputed International Journals and International conferences etc. He is now **Editor in Chief** of International Journal of Logic and Computation (IJLP, CSC Journal Press, and Malaysia). He is now Editor of International Journal of Computer Application (IJCA), International Journal of Advancements in Computing Technology (IJACT), International Journal of Computer Applications (IJCA, Acta Press). International Journal of Open Problems in Computer Science and Mathematics (IJOPCM). Recently his name was selected for **Editor in Chief** for Asian Journal of Information Management, USA. In addition to this, his name was also selected for **Regional Editor** of International Journal of Applied Science, **Regional Editor** of International Journal of Artificial Intelligence, **Regional Editor** of International Research Journal of Information Technology and Regional Editor of Trends in Applied Science Research. Recently He is selected as Associate Editor of ***International Journal of Intelligent Systems and Applications***, Hong Kong. He is now associate editor of ***International Journal of Intelligent System, Hong Kong***. He is now editor-in-chief of prestigious journal **International Journal of Engineering and Manufacturing(IJEM), Hongkong**. He is also International Program Committee of 17th Online World Conference on Soft Computing in Industrial Applications (WSC17) 2012,16th Online World Conference on Soft Computing in Industrial Applications (WSC16) 2011 and 15th Online World Conference on Soft Computing in Industrial Applications (WSC15) 2010. He is now selected as International Program Committee of 2013 World Conference on Information Systems and Technologies (WorldCIST'13) He is also acting as reviewers in many reputed International Journals. Currently he is an Individual Member in International Rough Set Society

and Member of International Association of Engineers (IAENG). Currently his name was selected for Marquis Who's and Who 2011, 2012 and 2013.

Prof. Debi Prasad Tripathy (1966) is currently working as Professor and Former Head in the Department of Mining Engineering at National Institute of Technology, Rourkela, Orissa, India. He did his B.E. (Min.) from VNIT, Nagpur; M.Tech. (Mine Planning) from I.T., B.H.U., Varanasi; Ph.D. (Mine Environment) from Indian School of Mines, Dhanbad. He also did PG. Dip. in Ecology and Environment from IIEE, New Delhi; Master of Financial Management (Pondicherry University); PG. Dip. in PM&IR (Annamalai University). He has attended an International Training program on World Bank aided EMCBTAP: Mining Sub-component, UNSW, Sydney, June 14-25, 2004, Australia. He has undertaken a specialized training on "Disposal and Utilization of Coal Combustion Residues" at Southern Illinois University, USA in June 2006. His current areas of teaching and research are: Mine Environment and Safety Engineering, Mine Planning, Computer Applications in Mining and Mine Management.

He has published five books namely: Noise Pollution, 1999; Dictionary on Environmental Science and Engineering, 2000, Dictionary of Earth Science, Dictionary of Quotations and Environmental Pollution Research, 2002 published by M/s APH Publishers, New Delhi. He has published over 120 technical papers in International/ National Journals and Conferences/Seminars on Mine Environment and other areas. He has organized/member of several International/National Seminars/ Conferences on Environment and edited a number of Proceedings. He has been awarded with Deokhas Prize from Nagpur University for academic excellence, honored with "MSPI Outstanding Personalities Award-1998", New Delhi; UWA Life time Achievement Award", Chennai. He has been honored with Gold medals and citations for Best Paper from I.E. (I), OSC, and Bhubaneswar in 2001, 2002, 2003, 2005, 2006, 2007, 2008 and 2009. In recognition of his research contributions, he has been selected as member, Editorial Board of Journal of Pollution Research (EnviroMedia), India; International Journal, JGMR; IJ of Earth Science Engg., and Peer Reviewer of several International Journals. He has visited Australia, USA, Canada, Papua New Guinea, Dubai, South Africa, Botswana, Singapore and Thailand on various academic assignments. He is a Fellow of ISRMTT, IE (I), UWA, and member of other prestigious professional societies like MEAI, ISTE, IMJRF, IAEM, MMGI and ASEG.

Prof. Simanata Kumar Nayak is presently working as Professor in Eastern Academy of Science and Technology, Bhubaneswar, Odisha-754001. He is continuing his PhD at Utkal, University in Computational Drug Design. He received his Master degree from N.I.T Rourkela. His specialization is in Computer Graphics, Computer Networking, Artificial Intelligence, and Computational Intelligence. He has several publications in International journals and reputed International Conference. He has supervised more than five PG Thesis and handling B tech projects also. His research interests are computational drug design, neural network and fuzzy logic.

Subhasis Mohapatra is presently working as Asst. Professor in Eastern Academy of Science and Technology, Bhubaneswar, Odisha-754001. He received his Master degree from Biju Pattniak University. His specialization is in Computational Intelligence, Time Series Prediction. He has supervised more than thirty B tech projects. His Computational Intelligence, Time Series Prediction.

How to cite this paper: Santosh Kumar Nanda, Debi Prasad Tripathy, Simanta Kumar Nayak, Subhasis Mohapatra,"Prediction of Rainfall in India using Artificial Neural Network (ANN) Models", International Journal of Intelligent Systems and Applications(IJISA), vol.5, no.12, pp.1-22, 2013. DOI: 10.5815/ijisa.2013.12.01

