Time Series Forecasting

Chandini RadhaKrishnan[1], Chinmay Bolumbu[2], Vignesh Namasivayam[3]

Arizona State University

Abstract— **In this paper we present a solution to compute time series prediction using Nonlinear AutoRegressive neural network (narnet). This network is used to predict using a given dataset which has values arranged chronologically. This network was designed and built using MATLAB using the Neural Net Time Series toolbox. We trained our model using the Levenberg-Marquadt algorithm and predicted the future values of the give time series dataset.**

Keywords— **Time series prediction, Neural Net Time Series toolbox, Nonlinear Autoregression, Levenberg-Marquadt.**

I. INTRODUCTION

Time series forecasting is the use of a supervised model to predict future values based on previously observed values. It is a sequential data with temporal changes. Time series can be applied to real-valued, continuous and discrete data sets. Time series forecasting comprises various methods for analyzing time series data to extract meaningful statistics and other significant characteristics of the data. Time series are widely used for non-stationary data like weather forecasting, stock market prediction, signal processing, earthquake prediction and so on.

Time series prediction is very interesting and challenging if the dataset is just a column of random values. To calculate time series predictions for these types of dataset can be done using different algorithms, one of the most common and well predicted outputs can be calculated by making delays of the given dataset and training it to predict values based on the previous pattern of values given.

The time series problem in our case was to predict 30 points from 276-305 using linear regression methods with the given dataset which had a column of 275 data points that were arranged chronologically as in Fig1.

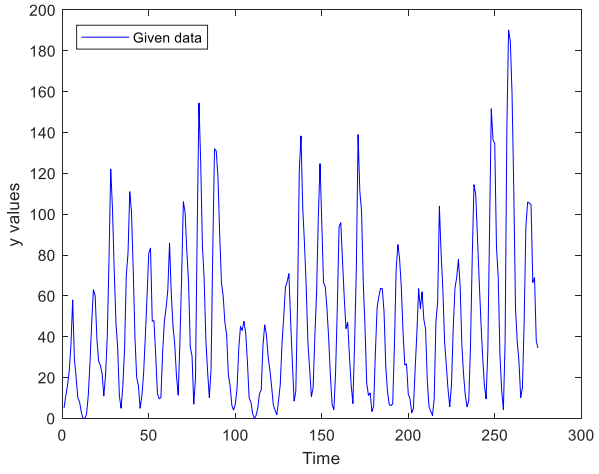

Fig.1 Time vs Given data points.

II. EVALUATION APPROACHES

Before choosing the Nonlinear AutoRegressive Neural Network (narnet) approach using the NTS toolbox, we tried two other different approaches to predict the time series data points.

A. Linear Regression

The first method that was tried to implement was using the linear regression model[3] with the parameters as theta_0 and theta_1 (i.e. straight line) using the hypothesis function as

$$h(\theta) = \theta_0 + \theta_1 x$$

The cost function calculated for this is given by

$$J = (1 / (2 * m)) * \text{sum}(((x*\theta)-y)^2)$$

where, theta - parameters
x - features
h - predictions
y - actual values
m - number of training samples

Based on this hypothesis[3] and the cost function, we did gradient descent and arrived at the parameter values 0.017084 and 0.270464. which helped us plotting the regression line for the given data.

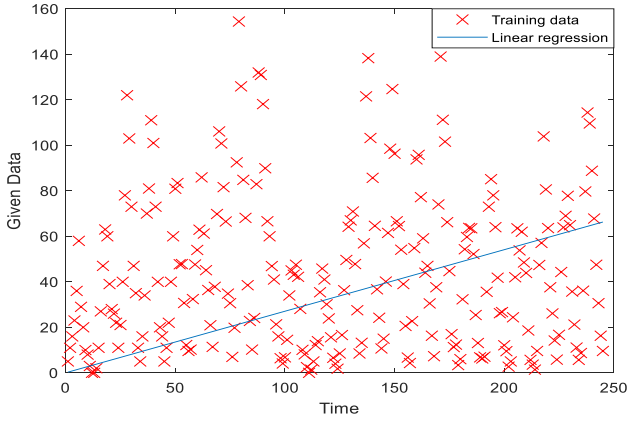

Fig.2 Linear Regression plot for the given data

We were able to analyze that this method is not well suited to solve our problem as the data values are scattered around, slope of regression model was high, and the values were constantly increasing for this model which gave us poor predictions.

B. *Curve Fitting*

To better understand the model, we made use of the curve fitting tool in MATLAB for determining the coefficients of polynomials[4] that fit the given data based on the order of features.

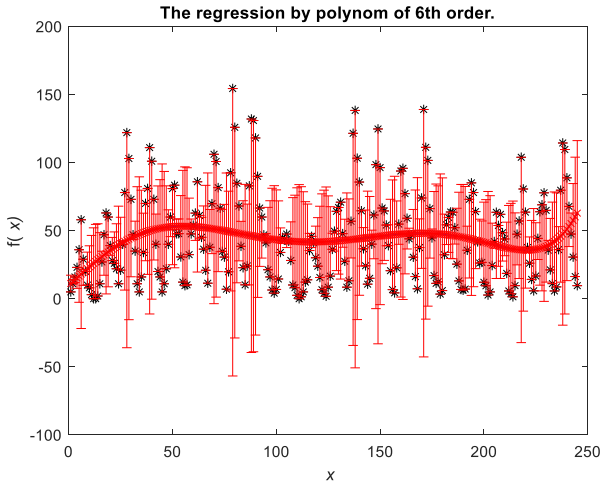

Fig. 3 Curve Fitting with order 6 polynomial

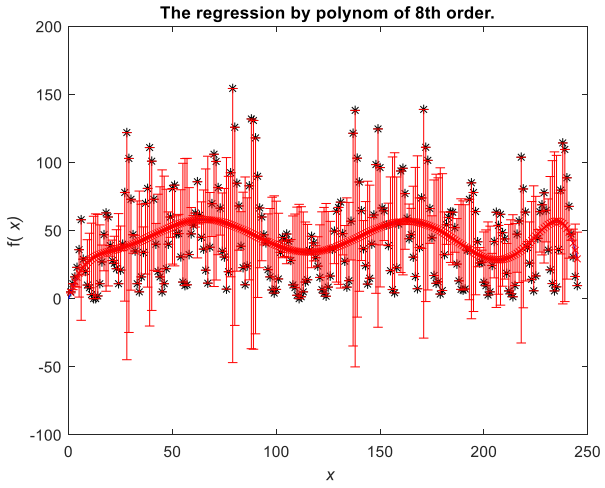

Fig. 4 Curve Fitting with order 8 polynomial

Based on the above two figures we can see that the order 6 polynomial curve has its predictions increasing exponentially and order 8 polynomial curve has predictions that are exponentially decreasing. This led to a conclusion that for the kind of data values we have, we cannot predict the future data points by having polynomial features[5] using the curve fitting tool. This led to our final selection of narnet using NTS Toolbox.

III. EXPERIMENTAL SETUP AND PROCEDURE

MATLAB R2018a was used to predict the time series throughout. The reason behind choosing MATLAB was because of the powerful nature of the software to do computations and its GUI features which makes it easier to the user. The project was done in three main stages where we split the given data into three parts such as for training the model, validating the model and testing the model.

A. *Training Data*

The given data was split into 89% (i.e. 245 data points) for training purpose. We trained the model with the NTS Toolbox in MATLAB using Nonlinear Autoregressive (NAR)[2] as it can predict y(t) given d past values of y(t) [1]. This NAR model is constructed by setting up the target values initially, later the percentage of training, validation and testing is set. After this we create our Neural Network based on the number of Delays (d) and number of Hidden Layers. In our case the architecture of the neural network was set to 4 Hidden layers and 25 delays. We used the Levenberg-Marquadt algorithm for training out network because its takes less time to compute and Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples [1].

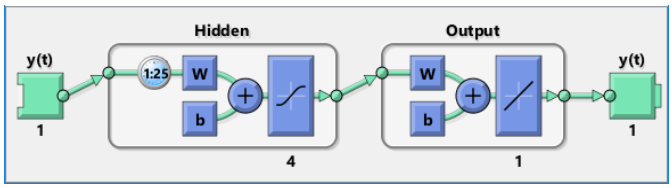

Fig. 5 Neural Network Model

B. *Validation Data*

The validation was done for 15% of the 245 training data points. With the validation we could determine the required number of hidden layers and the number of delays that are needed to achieve a lower Mean Square Error (MSE). Using this MSE we can calculate the Root Mean Square Error (RMSE) which gives us the distance between the actual value and the predicted value. This part of the experiment helps the model to study the data better and create predictions with low RMSE, such that the predicted data is close to the given data.

C. *Testing Data*

The remaining 15% of the 245 data points were used for testing purposes. The neural network model was trained for 8 iterations and 100 epochs values until a minimum value was reached. Based on the generated data points the RMSE is as:

$$\text{RMSE} = (\text{Mean}\ (\text{Predicted Values} - \text{Actual Values})^2)^{1/2}$$

The predicted data was compared with actual data and the minimum RMSE value that we could reach was 33.2779 using our train data and test data.

IV. RESULTS

We trained, validated and tested our model built using the narnet approach[2] in NTS Toolbox. Following were the steps we executed to obtain the results:

1. We plotted the given data and our data for training as in Fig. 6.

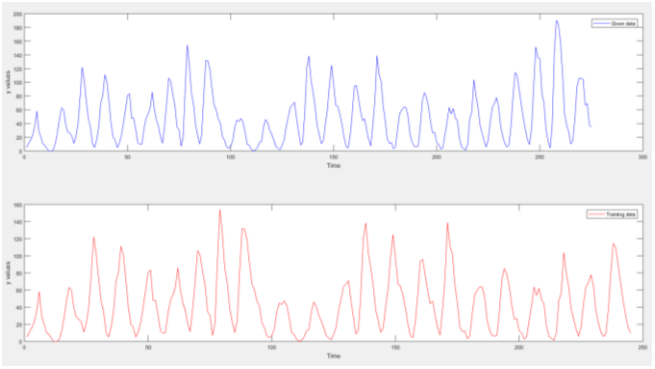

Fig. 6 Given Data and Training Data

2. This was followed by training the data using narnet in the NTS Toolbox.

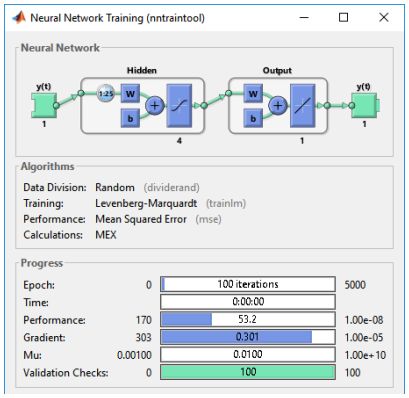

Fig. 7 NTS Tool Interface

3. Following results were obtained after training the neural network:

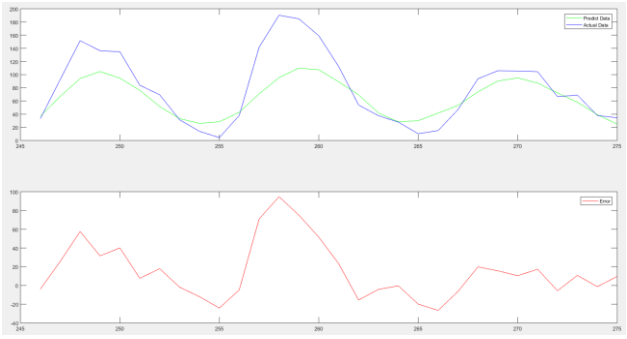

Fig. 8 Plot comparing Actual (Blue) and Predicted data (Green) along with Error (Red)

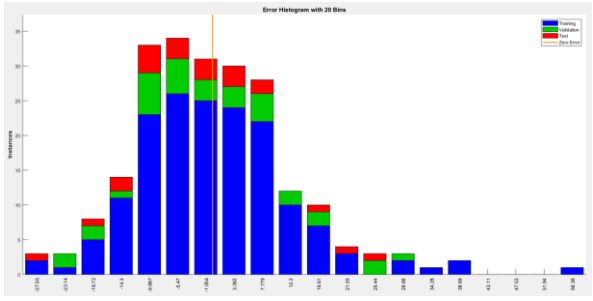

Fig. 9 Error Histogram

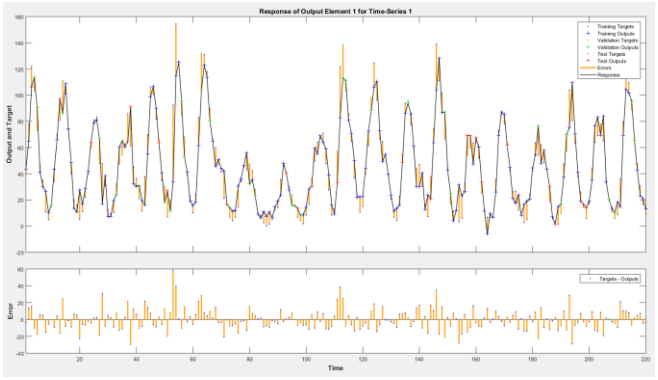

Fig. 10 Time Response plot with info about MSE

4. The below figure shows the predicted values (time units = 246 to 275) generated by the trained model using the Nonlinear AutoRegressive Neural Network.

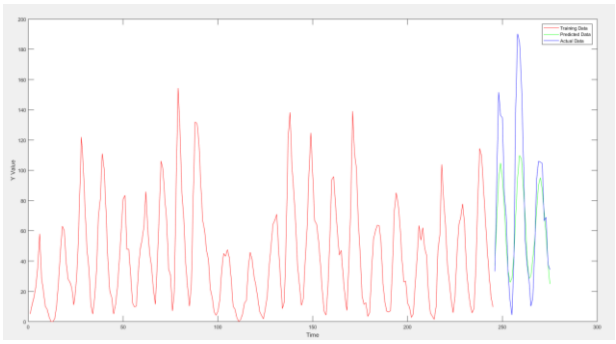

Fig. 11 Predicted values by the trained model (246 to 275 time units)

V. Conclusion

Using the NAR Neural Network, we trained a Regression model for Time Series Forecasting. From the Actual given data (1 – 275 time units), Data between the time units 1 to 245 were selected as Training Data, data between 246 – 275 were selected as Test data. The model was trained using the Training Data which predicted data between time units 246 to 275. The values predicted by the model were compared with the 30 test data points (246 – 275 time units) for validation. Root Mean Square Error was calculated between the predicted data and the test data to evaluate the effectiveness of our model and it was achieved as 33.2779.

Now based on the above trained model, we changed our training data to be the actual given data points and we predicted the next 30 data points (i.e. 276-305). We achieved our time series predictions chronologically as shown in Fig 12.

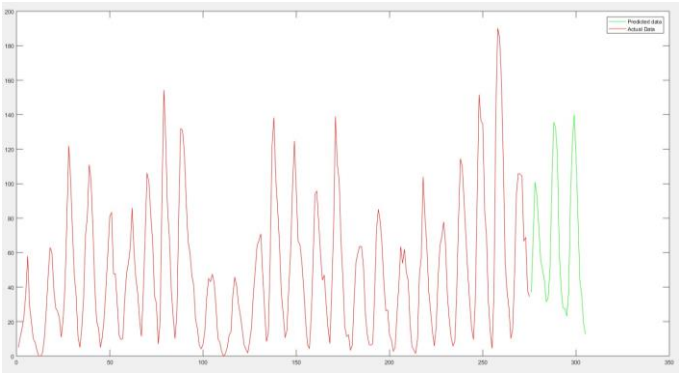

Fig. 12 Predicted values for n=30 data points (i.e. 276-305)

REFERENCES

[1] MathWorks,"Neural Time Series Toolbox (ntstool)", available online at:https://www.mathworks.com/help/deeplearning/ref/ntstool.html?searchHighlight=ntstool&s_tid=doc_srchtitle

[2] MathWorks, "Nonlinear autoregressive neural network(narnet)" available online at : https://www.mathworks.com/help/deeplearning/ref/narnet.html?s_tid=doc_ta

[3] Ng, Andrew. "CS229 Lecture notes." CS229 Lecture notes 1.1 (2000): 1-3..

[4] MathWorks, "Curve Fitting Toolbox", available online at: https://www.mathworks.com/help/curvefit/interactive-and-programmatic-curve-fitting-environments.html?searchHighlight=curve%20fitting%20tool&s_tid=doc_srchtitle

[5] Shumway, Robert H., and David S. Stoffer. "Time series regression and exploratory data analysis." Time series analysis and its applications. Springer New York, 2011. 47-82..

.

APPENDIX – I

Predicted n=30 values (i.e. 276 – 305 data points)

Time Units	Predicted Values
276	36.91304371
277	68.84890268
278	100.8600914
279	93.30344076
280	75.12912378
281	55.77992827
282	49.9600858
283	44.31314226
284	31.51873521
285	33.56467868
286	50.68482404
287	98.73592677
288	135.6553189
289	132.4787362
290	116.631021
291	59.05328674
292	38.81010123
293	27.98210444
294	27.4932551
295	23.08977685
296	36.1868329
297	92.68889145
298	123.1598375
299	140.2166739
300	115.1590711
301	81.11621195
302	43.88800728
303	35.54076265
304	19.32380402
305	12.72296585

APPENDIX – II

Matlab Code:

The below code is used for Model Training

```
%========For Model Training==========%
close all, clear all, clc;
data=xlsread('project2_time series data_students.xlsx'); % Given data
y_train = data(1:245)'; % y in rows for training
y_test = data(246:275); % y in column for testing
x_train  = 1:245; % x for training
x_test = (246:275); % x for testing


subplot(2,1,1)
plot(data,'b');
legend('Given data');
xlabel('Time');
ylabel('y values');
hold on;
subplot(2,1,2)
plot(y_train,'r');
legend('Training data');
xlabel('Time');
ylabel('y values');
% plot(y_test,'g');
% legend('Testing data');


y_train=con2seq(y_train);% y train to sequential (create arays)

%x=con2seq(y_train');
% NAR (USING NTS TOOLBOX)
net=narnet(1:25,4);% 25 delay and 4 Hidden Neurons
net.trainFcn='trainlm';% Training using Levenberg- Marquadt Algorithm
[Xs,Xi,Ai,Ts] = preparets(net,{},{},y_train);

net.divideParam.trainRatio = 75/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 10/100;
net.trainParam.max_fail = 100;
net.trainParam.min_grad=1e-5;
net.trainParam.show=5;
net.trainParam.lr=0.9;
net.trainParam.epochs=5000;
net.trainParam.goal=1e-8;

i=0;
for i = 1:8
    net = train(net,Xs,Ts,Xi,Ai);
end

view(net)

[netc,xic,aic] = closeloop(net,Xi,Ai);
[y_predict,xfc,afc] = netc(cell(0,30),xic,aic);% Predict 30 values
y_predict=cell2mat(y_predict);% change Cell Array to dataset
y_predict=y_predict';
```

```
predictdata=y_predict;
%newdata=append(data,predictdata);

ytest = y_test';
figure(2);
plot(x_test,y_predict,'b');
hold on;
plot(x_test,ytest,'r');
legend('prediction','actual output');
e = y_test - y_predict; % Errors
meanVal = mean((y_test - y_predict).^2);    % Mean Squared Error
RMSE = sqrt(meanVal);  % Root Mean Squared Error

figure(3)
subplot(2,1,1)
xlabel('Time')
ylabel('Y Value')
plot(246:275,y_predict,'g');
hold on;
plot(246:275,y_test,'b');
legend('Predict Data','Actual Data');

subplot(2,1,2)
xlabel('Time')
ylabel('Error')
plot(246:275,e,'r');
legend('Error');

y_train=cell2mat(y_train);
figure(4)
plot(y_train,'r');
hold on;
plot(246:275,y_predict,'g');
plot(246:275,ytest,'b');
legend('Training Data','Predicted Data','Actual Data');
xlabel('Time');
ylabel('Y Value');
%========For Model Training==========%
```

The below code is used to predict n=30 data points.

```
%========For Prediction==========%
close all, clear all, clc;
data=xlsread('project2_time series data_students.xlsx'); % Given data
y_train = data(1:275)'; % y in rows for training
x_train  = 1:275; % x for training

subplot(2,1,1)
plot(data,'b');
legend('Given data');
xlabel('Time');
ylabel('y values');
hold on;
subplot(2,1,2)
plot(y_train,'r');
legend('training data');
xlabel('Time');
ylabel('y values');
```

```
y_train=con2seq(y_train);% y train to sequential (create arays)

%x=con2seq(y_train');
% NAR (USING NTS TOOLBOX)
net=narnet(1:25,4);% 25 delay and 4 Hidden Neurons
net.trainFcn='trainlm';% Training using Levenberg- Marquadt Algorithm
[Xs,Xi,Ai,Ts] = preparets(net,{},{},y_train);

net.divideParam.trainRatio = 75/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 10/100;
net.trainParam.max_fail = 100;
net.trainParam.min_grad=1e-5;
net.trainParam.show=5;
net.trainParam.lr=0.9;
net.trainParam.epochs=5000;
net.trainParam.goal=1e-8;

i=0;                %(8 iterations)
for i= 1:8
    net = train(net,Xs,Ts,Xi,Ai);
end

view(net)

[netc,xic,aic] = closeloop(net,Xi,Ai);
[y2,xfc,afc] = netc(cell(0,30),xic,aic);% Predict 30 values
y2=cell2mat(y2);% change Cell Array to dataset
y2=y2';
predictdata=y2;
%newdata=append(data,predictdata);

figure(3)
% subplot(2,1,1);
xlabel('Timesteps in years');
ylabel('Value');
plot(276:305,y2,'g')
hold on;
plot(1:275,data,'r')
legend('Predicted data','Actual Data');
%========For Prediction==========%
```