Time Series Forecasting

Sai Prajwal Kotamraju[#1], Sai Pratyusha Gutti[#] , Rakshith Subramanyam[#1]

EEE, Arizona State University

[1]skotamra@asu.edu, [1]sgutti@asu.edu,[1]rsubra17@asu.edu

***Abstract*— In this project, a given time series data has been forecasted by using a multilayer perceptron. Feature formation was used to increase the number of input features and analyze the sequence in the series. A comprehensive study on the performance of feature formation to increase the dimensions of the data showed improvements on the performance of the network. Along with exploiting the dimensionality, a closed loop network was also experimented on for the prediction of the series. The results showed that implementing the autoregressive model gave a better prediction on the validation data when compared to the closed loop neural network model. The training error was about 519 (MSE) and validation data error was 287 (MSE). The results have been elucidated and corresponding graphs have been included.**

***Keywords*— Time series prediction, multi-layer perceptron, autoregressive model, dimensionality, recurrent neural network**

I. Introduction

Time series data is a type of temporal data which is a collection of ordered observations made in a chronological manner [1]. Data is collected from an experiment at regular intervals to form the time series which is used to predict future observations. There are usually two types of data in the prediction scenario - strictly stationary data and data with dependencies. Time series forecasting finds applications in many fields including business, meteorological studies, stock market, electricity demands, and in applications with seasonal changes in time, etc. [2]. From [3], we see that linear models of time series can be predicted by using Autoregressive Integrated Moving Average (ARIMA). Whereas Artificial Neural Network (ANN) has characteristics which can predict even nonlinear time series data. For this project, these two methods have been integrated to predict the given time series. For the purpose of prediction, the nature of the time series should be understood, its evolution should be modelled, its future values should be forecasted by understanding the factors impacting it.

275 data points of the time series data were given in a chronological order in a yearly manner and the next 30 points were asked to be predicted. The given data points are as shown below in Fig. 1. For training the network, about 95% of the data has been used as training data and the remaining 5% of the data was used for validation of the network created. The performance was tested using mean square error loss function.

The organisation of this report is as follows. Section II describes the approach used for the time series prediction which includes exploitation of properties for dependencies in the time series like auto-correlation and another approach of implementation of closed loop neural network. Section III covers the simulation procedure and the results obtained for each method. Section IV concludes the project report.

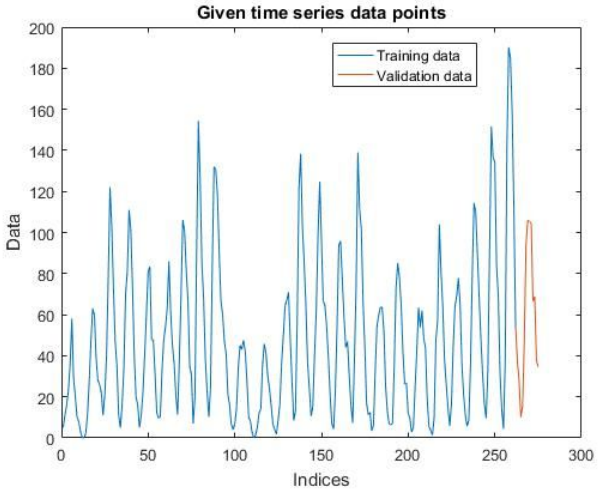

Fig. 1. Time series data points

II. Approach

To tackle the time series problem, described in the previous section, three models have been tested and analysed individually. They are as follows.

- Single Input Feed Forward Network using LM.
- Feed Forward Net using LM with increased input dimensionality.
- Closed Loop Narx Network with eleven delay taps.

In this section, the network architectures as well as the idea behind considering these models for time series prediction is addressed. Note that the shortcomings of individual approach and other important analysis is done in Section III of this paper.

A. Single Input Feed Forward Network using Levenberg-Marquardt (LM)

This model utilizes a Feed Forward Network architecture with Levenberg-Marquardt algorithm with one input neuron, one output neuron, two hidden layers with 16 neurons each. In this case, the indices of the data supplied are taken in vector form and supplied as input. The network architecture of this model is shown in Fig. 2.

[#1] All the authors did equal amount of work

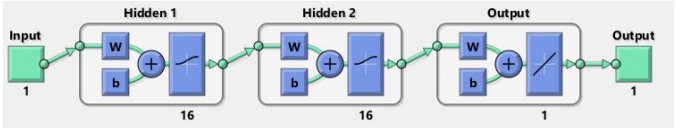

Fig. 2. Network Architecture for Single Input Feedforward Net with two hidden layers with 16 units in each layer.

In this model, the indices of the train data supplied are used without any modified representation. Thus, the input in this case is a row vector with the length of train set size. Output is also a row vector and has the same size as of input. Both the hidden layers use Logarithmic Sigmoid Activation function while the output layer uses a linear activation.

B. Feed Forward Net with increased input dimensionality

This is an improvised version of the previous model. The idea behind this architecture is mainly based on the autoregressive nature of the data. From Fig. 3., it can be noted that the autocorrelation of training data resulted in local peaks spaced out on an average of 11 indices. This proves that the data is highly correlated for every 11 samples.

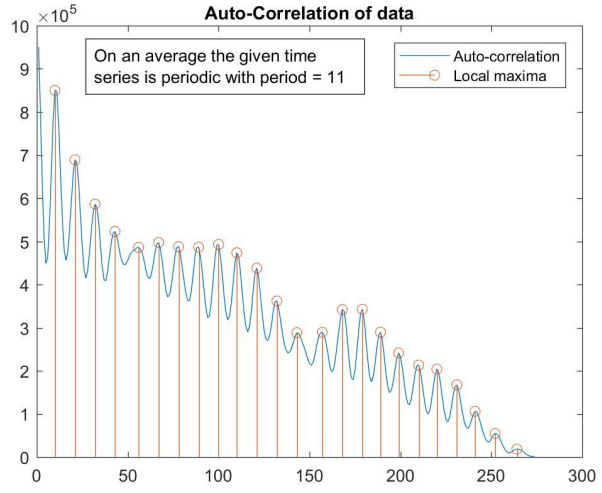

Fig. 3. Autocorrelation of Training Target Vector with the vertical orange lines (depicting local maxima) are spaced out at 11 samples.

To exploit this observation, the input dimensionality is increased such that the first row of the new input vector, represents the sequence number while the second row represents the sample's position in that sequence. Fig.4. represents the change in data representation in a pictorial way.

Given the index value n and the length of input N, the input for this model, *new input*, can be obtained as follows.

$$new\ input = \begin{bmatrix} input1 \\ input2 \end{bmatrix} \tag{1}$$

$$input1(n) = \frac{n}{11} + 1 \ ; \ n \in N \tag{2}$$

$$input2(n) = \begin{cases} rem(n,11) & ; if\ rem(n,11) \neq 0 \\ 11 & ; if\ rem(n,11) = 0 \end{cases} \tag{3}$$

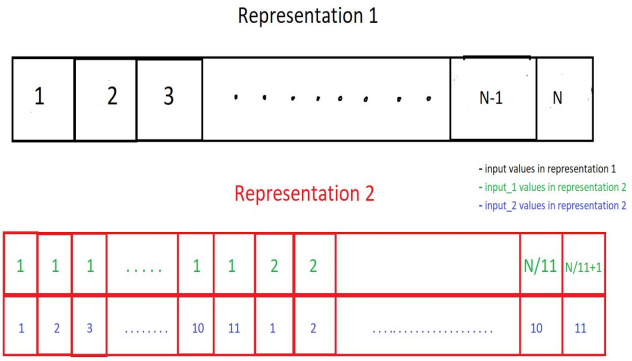

Fig. 4. In black color - Single input representation for model in Section II A. In red- Dual Input representation for Model in Section II B. Note that N needs to be a multiple of 11 to make the last value of input2 to be 11.

Except for the change in representation and the dimensions of the input, everything else was kept constant from the architecture specified in Section II A. i.e. the model uses two input neurons, two hidden layers with 16 neurons each with a logarithmic sigmoid activation function and one output neuron with linear activation function. Architecture for this model is shown in Fig. 5.

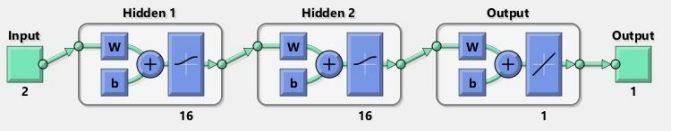

Fig. 5. Network Architecture of Feedforward Net with increased input dimensionality with two hidden layers with 16 units in each layer.

Because of increased input dimensionality, there would be more weights between input layer and hidden layer. This increases the degrees of freedom. Also, having a separate input for sequence number and the sample location helps the network understand the sequence in which peaks are changing for every cycle.

In Fig.3. it can be observed that the peaks at approximately every 100th sample shows an increase with respect to the previous peak which suggests that there might be another feature that correlates the data. Further increase in the input dimension to represent 3 features did not give much improvement in the performance and hence the results have been analysed using two features only.

C. Closed loop Feature augmented network

A closed loop network takes the prediction made in previous iterations and the past input that lead to the prediction as input features. This enables the network to predict future values based on past prediction. The network was given a delay of 11 values which means the network will observe t-11 values of the inputs and prediction to make the current prediction. The number 11 was chosen assuming the network to learn the correlation in the data the was observed

as explained in section II B. A two layer network with 11 neurons in both layers was structured, logarithmic sigmoid function was used as the activation function. The network followed Leven-Marquardt algorithm The network architecture is depicted in Fig. 6

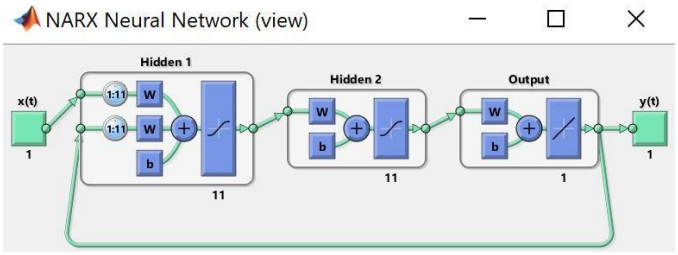

Fig. 6. Network architecture of Closed loop NARX network with 11 delay taps in the input.

III. Simulation and Results

Using MATLAB as the platform for analysis, the prediction of time series data was implemented using multilayer feedforward network and closed loop neural network. The results obtained have been compared in the following subsections.

A. Single input Feedforward Network vs Feedforward Net with increased input dimensionality

The network has been constructed based on the model described in Fig.2. The other hyperparameters, including learning rate, are set according to the MATLAB default values in the Neural Network Training.

The prediction based on the Levenberg-Marquardt algorithm with required number of hidden units and two layers gave a very good fit with less training error. But the validation error which represents the prediction, here, was very high. This network model has high complexity and implied overfitting. The mean square error (mse) for the training data is about 64 whereas the mse for the validation data is 5420. Fig. 7. shows the prediction obtained by the model representation in Section II. A. From the figure we can note that the output of the model for the last 11 data points (validation data) gives a bad prediction of the given time series. This model was also tested with multiple cases of weight initialization, number of active neurons, activation functions and feedforward network algorithms. Most of the cases had a similar result as shown in Fig. 7. Hence the features of the given series were analyzed and the observations are explained in the remaining part of this Section.

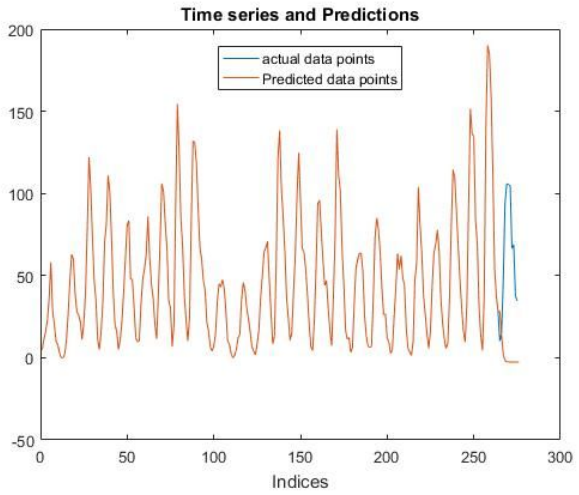

Fig. 7. Series Prediction by a complex Feedforward network

The next model was implemented based on the architecture given in Fig. 5. A random permutation was performed on the input matrix so that the robustness of the model is improved. The performance of the network trained with increased dimensionality was much better than the single input. The test data was divided into training and validation with a percentage of 95 and 5 respectively. The mean square error of the train set was 519.03 and on validation set it was 287.86. This is better than the single input network which has higher validation error. The model output for the training data and validation data is as shown in Fig. 8. Though the output for the training data is not perfectly aligned with the actual output, the validation data output is good implying good prediction of the series. This implies that there is no overfitting.

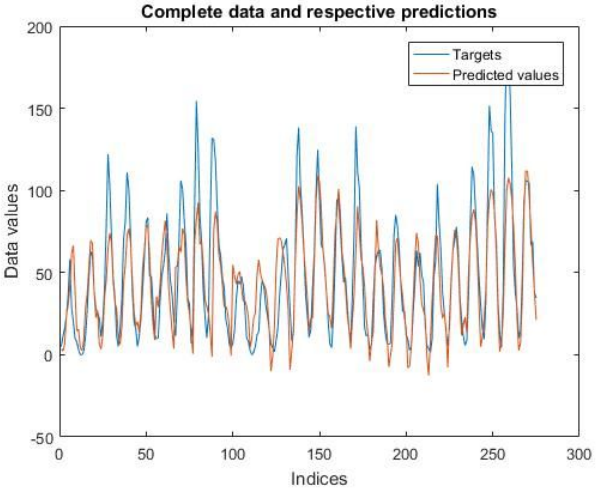

Fig. 8. Output of the Feed forward network trained on augmented input.

Fig. 9. shows the prediction of the next 30 data points (in red) along with time series (in blue). From the pattern of the complete series (given series and predicted series), we might conclude that the built network did a good prediction.

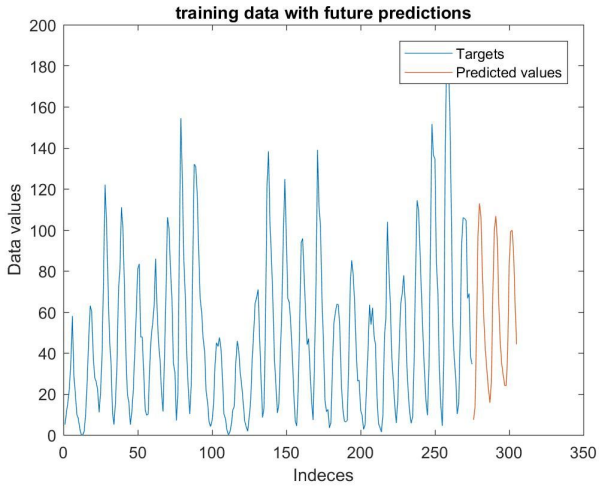

Fig. 9. Prediction made by Feed forward network trained on augmented input.

As discussed in Section II parts A and B, the only difference between those two architectures is the input representation. Fig. 10. shows the performance plot of Single Input Feedforward Network while Fig. 11. shows the performance plot of Feedforward Net with increased input dimensionality. It can be noted that Feedforward Network with modified input representation does good job when compared to a single input Feedforward Network in terms of minimum validation error for the same set of parameters as specified in Section II.

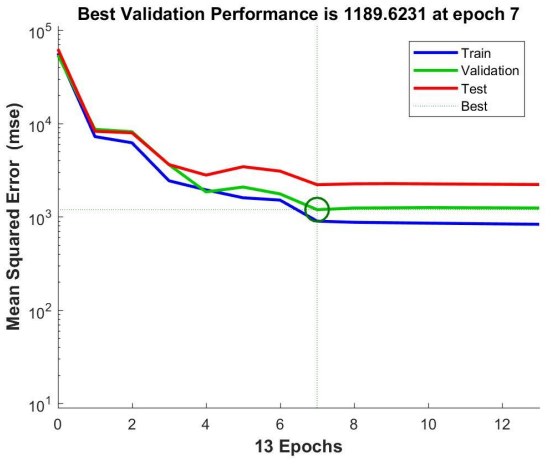

Fig. 10. Performance plot of Single Input Feedforward Net.

B. Closed loop Narx Network with eleven delay taps

The training result of the closed loop network was very satisfying but the validation performance was not as satisfying. The validation error was not near the training error.. Multiple weight initialization with the use of regularizers were tried with not much improvement in the result. The Fig.12 depicts the performance

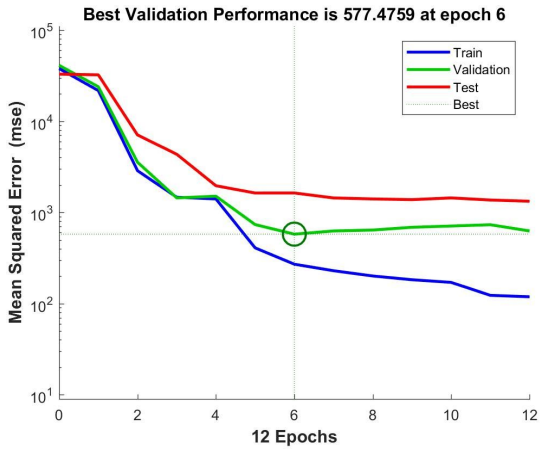

Fig. 11. Performance plot of Feedforward Net with increased input dimensionality.

measure of the network. This may be due to the reason that the data that is available is very less and the network do not have enough timesteps to go back and predict and also that the network was overfitting of the training data. The feedforward closed loop network performed better in both training and validation-test.

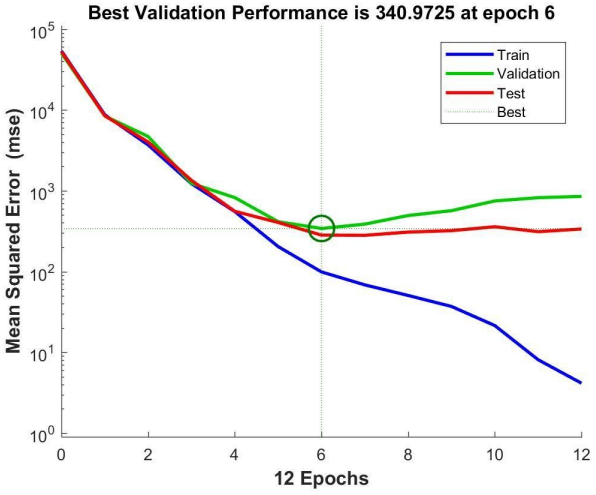

Fig. 12. Performance plot of the Closed loop Network with increased input dimensionality.

IV. Conclusion

The time series data prediction was carried out using a feedforward single input and augmented multiple input network and also a closed loop narx network. The feed forward network trained on input augmented data was used to predict the target value for the next 30 inputs by analysing the 275 input values of the training data. The predicted values are represented in the Appendix section. The mean square error

for the training data modelled with Feedforward network with increased dimensionality was around 520 and the validation error was around 280. It is concluded that the feed forward network with increased input dimension performs better than a closed loop network or a non augmented input feed forward network for less training examples.

References

[1] Soheila Mehrmolaei, Mohammad Reza Keyvanpour, "Time series forecasting using improved ARMA," IRANOPEN, Iran, Aug. 2016

[2] G. Mahalakshmi et al., "A survey on forecasting of time series data ", ICCTIDE, India, Oct. 2016.

[3] Aarshay Jain (2016, February 6). A comprehensive Beginner's Guide to create a Time Series Forecast (with odes in Python) [Online]. Available: https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/

[4] MathWorks, MATLAB Documentation [Online]. Available: https://www.mathworks.com/help/matlab/

APPENDIX

A. *Prediction of next 30 data points.*

7.38080606889777

13.3116166737061

50.0098091388285

97.4084328658552

112.861734438967

106.269228501675

77.8072592926804

55.6521575324633

40.8094885035930

31.5218919516399

22.6191447601389

15.6633030734115

25.5949826767590

69.4159022290015

99.8865704151883

106.719239769013

95.2151863071758

63.9961182109755

44.5976870450989

34.3709629338225

28.9475911995206

24.0634766226960

24.1407460948748

41.3765872673218

83.1692103211127

99.2922390966408

99.7007092251910

87.9316875273361

63.0964646157983

44.2010299553336

B. *Feed Forward network with input augmentation*

```
% EEE 511: Project 2
% Timeseries Prediction.
%   Author:   Sai   Prajwal   Kotamraju,   Rakshith
Subramanyam,  % Sai Pratyusha Gutti
% Date: 10/8/2017
clear;clc;

%% Data acquisition, shuffling and splitting into
training and validation
display = 0; % Set display = 1 for graphs of
results and to 0 for no display

raw_data    =    xlsread('3-project_time    series
data_students.xlsx'); % Importing the training
data
target = raw_data'; index = 1:length(target);
input1 = ceil((index)./11); %Sequence Number
input2 = rem(index,11);
input2(input2==0)=11;
X = [input1;input2];
X_total = [X;target];
perc_train = 95;%Percentage of data to be used for
training
n_train = ceil((perc_train/100)*length(target));
[rows, cols] = size(X_total);

%% Understanding the data (Why division with 11?)
%  Uncomment  the  section  below  to  see  the
correlation graphs
% corr = xcorr(target,target);
% corr = fftshift(corr);
% corr = corr(1:length(corr)/2);
%  figure;  plot(corr);  title('Auto-Correlation  of
data');
% hold on;
% [peak_val, peak_loc] = findpeaks(corr);
% period = floor(mean(diff(peak_loc)));
% stem(peak_loc,peak_val);
%       annotation('textbox',[.2       .6       .3
.3],'String',{['On  an  average  the  given  time'],[
'series    is    periodic    with    period    =
',num2str(period)]},'FitBoxToText','on')
% hold off;
% legend('Auto-correlation','Local maxima');

%% Training the network
seed1 = 92; seed2 = 3;

rng(seed1);
rand_idx = randperm(cols) ;
X_total_shuffled = X_total ;
X_total_shuffled(:,rand_idx) = X_total(:,:);
X_shuffled = X_total_shuffled([1, 2],:);
target_shuffled = X_total_shuffled(3,:);

% Shuffled Train Data
X_train_shuffled = X_shuffled(:,1:n_train);
target_train_shuffled                              =
target_shuffled(1:n_train);
```

```
%Shuffled Validation Data
X_val_shuffled =
X_shuffled(:,n_train+1:length(target));
target_val_shuffled =
target_shuffled(n_train+1:length(target));

rng(seed2);
net = feedforwardnet([16 16],'trainlm');
net.performParam.regularization = 0;
net.layers{3}.transferFcn = 'purelin';
net.layers{2}.transferFcn = 'logsig';
net.layers{1}.transferFcn = 'logsig';
[net,tr] =
train(net,X_train_shuffled,target_train_shuffled);
y_train = net(X_train_shuffled);
train_idx =
11*(X_train_shuffled(1,:)-1)+X_train_shuffled(2,:)
;
if display == 1
    plotperform(tr);
end
train_mse       =       (1/n_train)*(y_train       -
target_train_shuffled)*(y_train                    -
target_train_shuffled)';
%fprintf('MSE for Train set- %f ',train_mse);
%% Validation data accuracy check
y_val = net(X_val_shuffled);
valid_mse  =  (1/(length(target)-n_train))*(y_val  -
target_val_shuffled)*(y_val                        -
target_val_shuffled)';
%fprintf('MSE for Validation set- %f ',valid_mse);
test_idx =
11*(X_val_shuffled(1,:)-1)+X_val_shuffled(2,:);
%% Complete regression plot
out = net(X);
if display ==1
    figure;
    plot(index,target);
    hold on;
    plot(index, out);
    hold off;
          title('Complete   data   and   respective
predictions');
    xlabel('Indeces'); ylabel('Data values');
    legend('Targets','Predicted values');
end
%% Prediction Test Data
test_indeces = 276:305;
test_input1  =  ceil((test_indeces)./11);  %Sequence
Number
test_input2 = rem(test_indeces,11);
test_input2(test_input2==0)=11;
X_test = [test_input1;test_input2];
y_test = net(X_test)
if display ==1
    figure;
    plot(index,target);
    hold on;
    plot(test_indeces, y_test);
    hold off;
           title('training   data   with   future
predictions');
    xlabel('Indices'); ylabel('Data values');
    legend('Targets','Predicted values');
end
```