On the "Time series forecasting using a hybrid ARIMA and neural network model"

by

Ramin Ziaei

Discussion 1

Under the Supervision of Dr. Cheng

Department of Mathematics

University of Nebraska - Omaha

Omaha, Nebraska

October 22^{nd}, 2020

Contents

List of Figures

List of Tables

1 Introduction

Time series forecasting is a very important type of predictions that is applicable to a variety of fields from finance to politics. In this type of forecasting, the past observations of the desired variable are collected and analyzed to construct a model that explains the trend of the variable over time. Then, it is used to predict the future's behavior of the variable. ARIMA (autoregressive integrated moving average) model is probably the most popular time series forecasting model that has been around for several decades. One of the drawbacks that this model has is its linearity which makes it less effective when nonlinear components are quite significant in a dataset. Artificial neural network (ANN) has become the most popular tool among all machine learning tools due to its flexibility and accuracy. ANN can be applied to a variety of problems including classification, regression, image detection and time series forecasting, to name but a few. One of the advantages that ANN has over the famous ARIMA is its capability for including nonlinearities. However, it needs a lot of data points to be trained and can easily overfit a dataset. Considering advantages and disadvantages of both methods, there is a new model called a Hybrid model that combines AAN and ARIMA to boost the current time series forecasting performance [1].

2 Datasets

2.1 Sunspot

First, one of the datasets in the paper is modeled to make sure that the developed model works well. The data set is a well-know dataset named "sunspot" that contains the annual number of sunspots from 1700 to 1987 with 288 observations. Figure 1 shows this dataset. There are 2 sets of predictions for this dataset in the paper, 67 and 35 points ahead. Only the results of the 67 points ahead will be shown in this discussion. We, first, start with the ARIMA model. Paper mentions that a subset autoregressive model of order 9 was found to be the most parsimonious among all ARIMA models. The neural network they used is a $4 \times 4 \times 1$ network. Figure 2 shows the graphs that is shown in the paper, and Figures 3 to 5 are the ones that was generated in this discussion. They show good match.

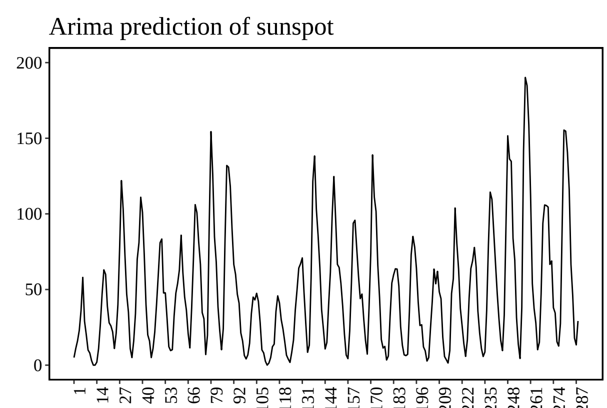

Figure 1: Sunspot series (1700-1987)

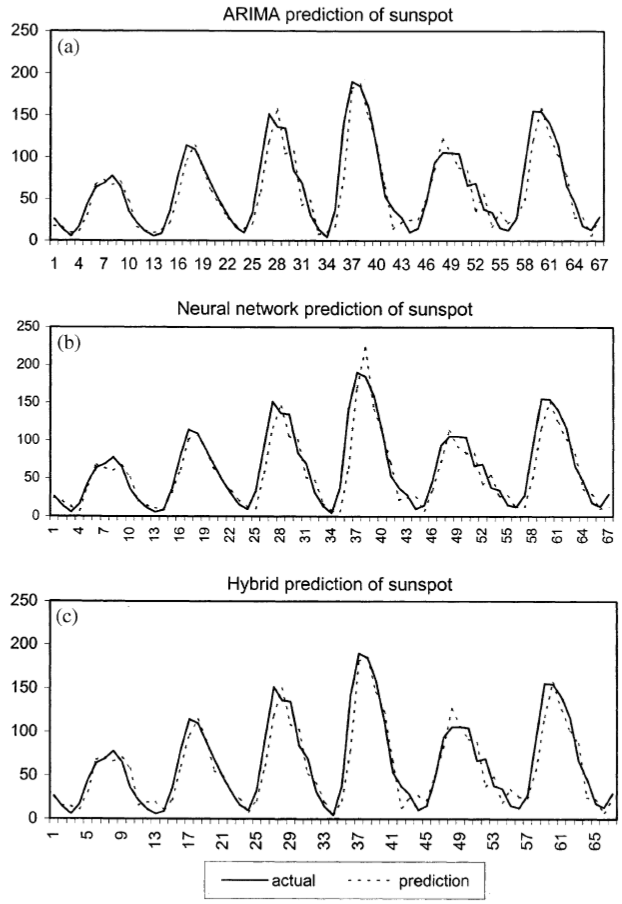

Figure 2: (a) ARIMA prediction of sunspot (b) Neural network prediction of sunspot and (c) Hybrid prediction of sunspot.

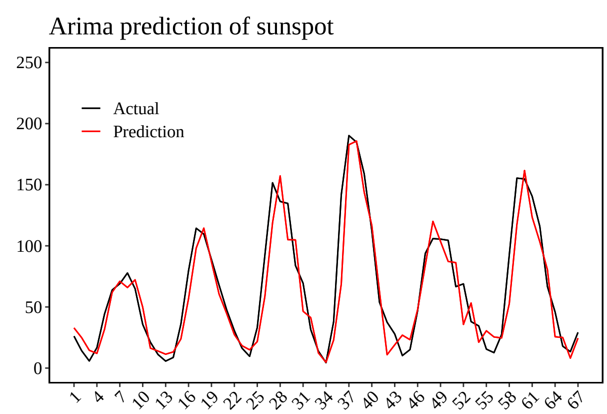

Figure 3: Simulated ARIMA prediction of sunspot

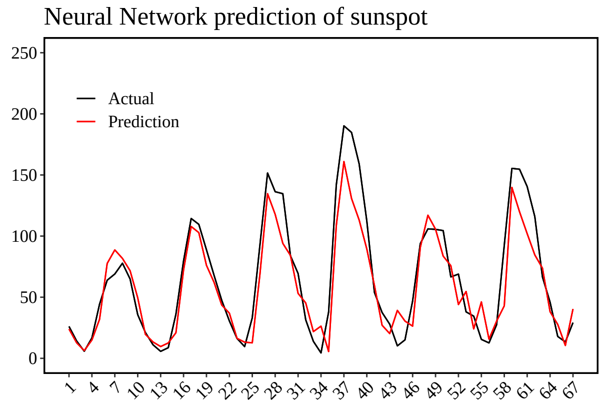

Figure 4: Simulated ANN prediction of sunspot

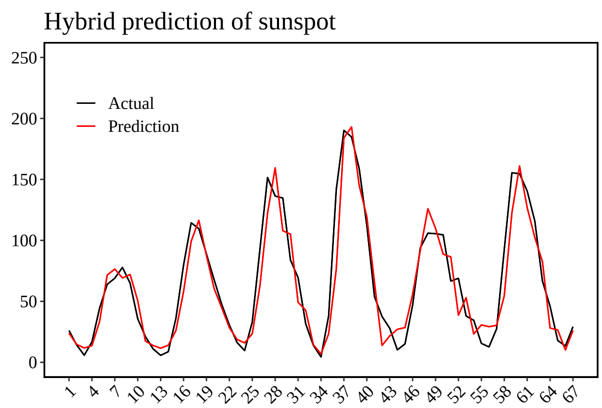

Figure 5: Simulated Hybrid prediction of sunspot

2.2 Delhi Daily Temperature

My final project is a classification problem with a column of date that shows the date at which the last status of the booked room was changed, and it does not mean and show any trend over time. So, I used another dataset from the Kaggle website which contains some information about temperature, humidity, wind speed and air pressure in india from 2013 to 2017 [2]. For this discussion, I only use the temperature section of the dataset. Data from 2013 to 2016 (1095 points) are used to train the model and data from 2016 to 2017 (367 points) is used to validate the accuracy of it. Figure 6 shows the time history of the temperature change over 4 years in India.

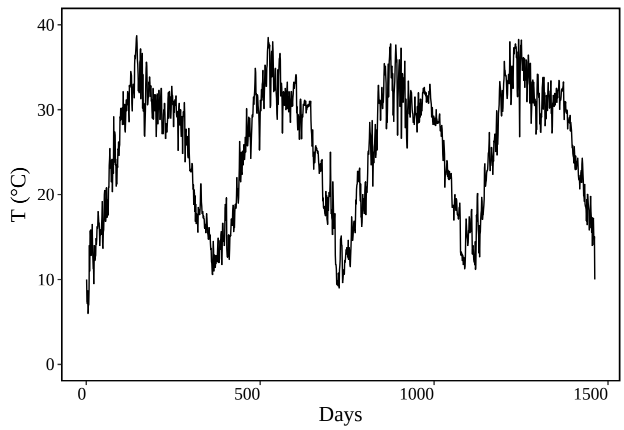

Figure 6: Delhi daily temperature change

3 ARIMA

I use an auto arima model to find the best fit to the dataset. The auto.arima() function in R uses a variation of the Hyndman-Khandakar algorithm (Hyndman & Khandakar, 2008) [3], which combines unit root tests, minimization of the AICc and MLE to obtain an ARIMA model. The best fit is an Arima(1, 0, 3)(0, 1, 0)[365], and it was used to fit the model and predict the future data. Like the original data, figure 7 shows that the prediction is very noisy and has sharp spikes and changes, but overall, the trend looks similar to the original data and seems reasonable. It is also capable of capturing the highest and lowest temperatures between 2016 and 2017 which is an advantage of this model. The mean squared error (MSE) and mean absolute deviation (MAD) are selected to be the forecasting accuracy measures which are shown in table 1.

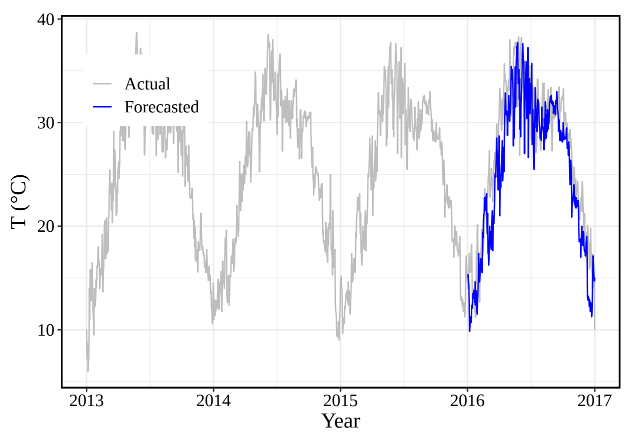

Figure 7: ARIMA model prediction

Table 1: ARIMA forecasting accuracy measures

MSE	MAD
13.13531	2.877289

4 Artificial Neural Network

After, the neural network was fit to the data to predict the temperature from 2016 to 2017. After some trial and error, NNAR(4, 1, 3)[365] was found to be working the best for this dataset. It needs to be mentioned that the prediction did not seem to be super sensitive to the parameters of NNAR, and the prediction would slightly change if parameters were changed. The prediction from the ANN (figure 8) looks much smoother compared to both the original data and the prediction from the ARIMA model. However, it cannot quite catch the lowest and highest temperatures recorded during the forecasted year (especially the high temperatures). Table 2 shows MAD and MSE for the ANN approach.

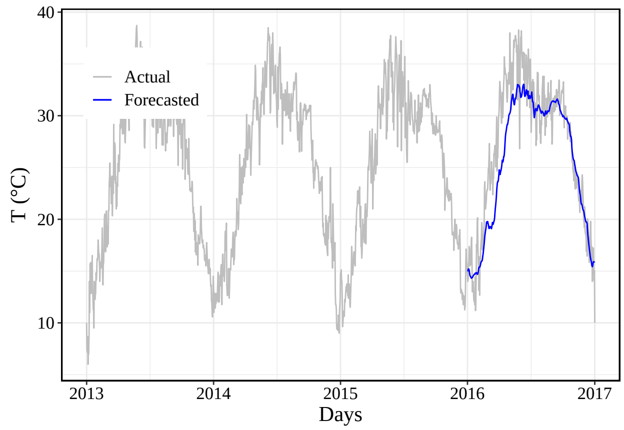

Figure 8: ANN model prediction

Table 2: ANN forecasting accuracy measures

MSE	MAD
10.01809	2.755489

5 Hybrid

Both ARIMA and ANN have been successful in their own linear and nonlinear domains. However, neither can be used universally for all time series problems since some might be mostly linear where ANN might give mixed results or mostly nonlinear where ARIMA shows weaknesses. In the hybrid approach, first, the ARIMA model is utilized to analyze the linear portion of the time series. Then, its residuals are fed into the ANN model to capture nonlinearities that were left out of the ARIMA model. Finally, the forecasts from both steps are linearly combined as the final prediction. As figure 9 displays, the hybrid model, it less noisy compared to the ARIMA model and can capture higher temperatures compared to the ANN model. Table 3 shows MAD and MSE for the Hybrid model.

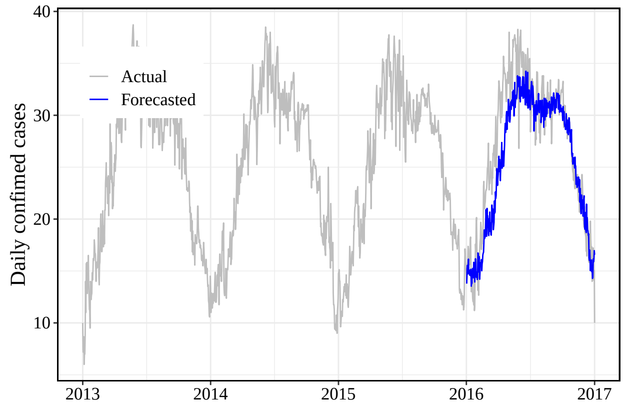

Figure 9: Hybrid model prediction

Table 3: Hybrid forecasting accuracy measures

MSE	MAD
10.5023	2.515971

6 Comparison

The comparison among all three models is brought in table 4. This table shows that the ANN model has lower MSE and MAD compared to the ARIMA model. The hybrid model has a lower MSE compared to the ARIMA model, but its MSE is higher than that of the ANN model. When it comes to MAD, the hybrid model is the best among all three.

Table 4: Models comparison

Model	MSE	MAD
ARIMA	13.13531	2.877289
ANN	10.01809	2.755489
Hybrid	10.50230	2.515971

7 Summary

When considering different fitted models to the same data, it is very important to recognize for what application they will be used. For instance, in the case of ARIMA and ANN, the later seems much less noisy and smoother, and more accurate if either MSE or MAD is considered. However, it is not capable of capturing minimum and maximum possible temperatures in the future. Due to the context of the research, these high values might be of great importance. So, if a model has lower MSE or MAD, it does not necessary mean it is superior to others. This is also the case for the hybrid model since it is not capturing the extreme values of the temperature. On the other hand, according to the literature, a hybrid model can increase the accuracy of the prediction as we saw in the sunspot dataset. Also, in the second example, the hybrid model gives the smallest MAD. All in all, the hybrid approach seems promising, but only based on this short discussion, I cannot make a general argument that it will always be better than either ARIMA or ANN.

References

1. Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159-175.

2. https://www.kaggle.com/sumanthvrao/daily-climate-time-series-data

3. Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. Journal of Statistical Software, 27(1), 1–22. https://doi.org/10.18637/jss.v027.i03