Short-term Electrical Load Forecasting

Dr. Devender Singh, Ayush Kumar Goyal (15084005), Boragapu Sunil Kumar (15084006),
Srimukha Paturi (15084013) and Rishabh Agrahari (15084015)
Department of Electrical Engineering, IIT(BHU), Varanasi

Abstract— **Electricity demand forecasting is a central and integral process for planning periodical operations and facility expansion in the electricity sector. Demand pattern is almost very complex due to the deregulation of energy markets. Therefore, finding an appropriate forecasting model for a specific electricity network is not an easy task. Although many forecasting methods were developed, none can be generalized for all demand patterns. This review article aims to explain the complexity of available solutions, their strengths and weaknesses, and the opportunities and threats that the forecasting tools offer or that may be encountered.**

I. INTRODUCTION

Electricity as a product has very different characteristics compared to a material product. For instance, electricity energy cannot be stored as it should be generated as soon as it is demanded. Any commercial electric power company has several strategic objectives. One of these objectives is to provide end users (market demands) with safe and stable electricity. Therefore, Electric Load Forecasting is a vital process in the planning of electricity industry and the operation of electric power systems. Accurate forecasts lead to substantial savings in operating and maintenance costs, increased reliability of power supply and delivery system, and correct decisions for future development. Electricity demand is assessed by accumulating the consumption periodically; it is almost considered for hourly, daily, weekly, monthly, and yearly periods.

The Electric Load Forecasting is classified in terms of the planning horizons duration: up to 1 day/week ahead for short-term, 1 day/week to 1 year ahead for medium-term, and more than 1 year ahead for long-term. Short-term forecasts are used to schedule the generation and transmission of electricity. Medium-term forecasts are used to schedule the fuel purchases. Long-term forecasts are used to develop the power supply and delivery system (generation units, transmission system, and distribution system).

The electricity demand pattern is necessarily affected by several factors including time, social, economical, and environmental factors by which the pattern will form various complex variations. Social (such as behavior) and environmental factors are big sources of randomness (noise) found on the load pattern. Diversity and complexity in demand pattern have been leading to developing complicated Electric Load Forecasting methods. The literature is enriched with Electric Load Forecasting methods having many attempts to find the best estimation of load forecasting. The major methods include time series such as exponential smoothing, ARMA, BoxJenkins ARIMA, neural networks; Fuzzy logic; and support vector machine. Recently, sequential models such as RNN, LSTM and GRUs have come into picture for forecasting Electric Load.

The ARIMA models and their versions have achieved a considerable success for Electric Load Forecasting. In general, ARIMA models can be used when the time series is stationary without missing data. They can be further hybridized with artificial intelligence techniques. However, the complexity of demand pattern depends on its base period; it changes from fairly smooth curve (annually based) to most noisy and cyclic complex curve (hourly based) since the effect of environmental factors increases.

In this era the electric power consumption is growing fast and may be more randomly because of the increasing effect of environmental and human behavior. Therefore, the electricity demand pattern becomes more complex and unrecognized. For instance, the people all over the world are using increased number and variety of electric appliances most of them are environmentally related, that increases the cyclic variation and noise on the demand pattern. Though there are many forecasting methods, no single one can be generalized to perform enough for all cases, especially when many factors are considered. Thus, to get a proper forecast, it is not just adopting a famous method. In other words, an ideal method for a case may perform poorly for another one. Therefore, the research must be directed to specially assigned methods. In other words, each electric power plant in any country needs to follow its own forecasting method. For that purpose, the general methods can be also adopted but with efficient and effective modifications that suit the case; otherwise the results will be misleading.

The aim of this paper is to demonstrate a pragmatic forecasting methodology for analyzing the electric load pattern and predicting the future load demand for short, medium, and/or long terms. This methodology can integrate different forecasting models. The rest of the paper is organized as follows. Section 2 is about the dataset used for the proposed methodologies which are presented in Section 3. Section 4 applies the methodology to a typical power load pattern. Concluding remarks are contained in Section 5. Note: All the source code developed during the course of the project is open source and is available on Github[1]

II. DATASET

We are using load data from State Load Dispatch Center (SLDC), Delhi. Load data is available at a time step of 5

[1]`https://github.com/pyaf/load_forecasting`

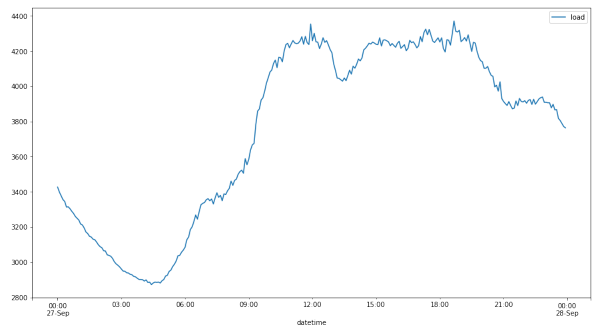


Fig. 1: Day wise plot

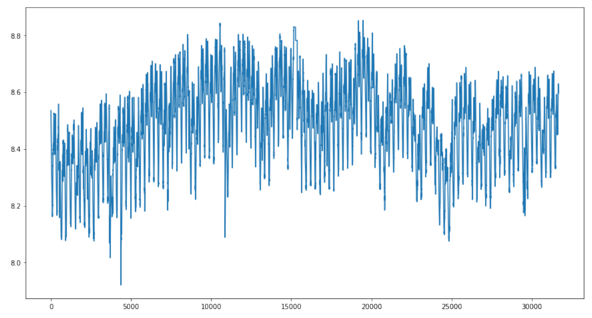


Fig. 2: Plot for a month

minutes, totally 288 values for each day. A general trend of the load data can be seen in the day wise plot and month plot in Fig(1) and Fig(2) respectively.

This data is being extracted from the SLDC website of Delhi [2] by an automated script. The mentioned website updates the data for every 5 minutes. Thus we have data in the time steps of 5 minutes. We acquired the data of the past one year till date, from the SLDC delhi website which made the information public. However, there are certain days when the SLDC website was not functional due to maintenance or any other factors, which implies the data collected might have some missing values.

III. METHODS

In this paper several methods, from moving averages to non-traditional algorithms such as RNNs have been implemented on the data set described in the above section. The working of all these methods are discussed thoroughly below

A. Simple Moving Average (SMA)

When demand for a product is neither growing nor declining rapidly, and if it does not have seasonal characteristics, a moving average can be useful can be useful in removing the random fluctuations for forecasting. Although moving averages are frequently centered, it is more convenient to use past data to predict the following period directly.

Although it is important to select the best period for the moving average, there are several conflicting effects of different period lengths. The longer the moving average period, the more the random elements are smoothed (which may be desirable in many cases). But if there is a trend in the data-either increasing or decreasing-the moving average has the adverse characteristic of lagging the trend. Therefore, while a shorter time span produces more oscillation, there is a closer following of the trend. Conversely, a longer time span gives a smoother response but lags the trend. The formula for a simple moving average is

$$F_t = \frac{\sum_{i=1}^{n} A_{t-i}}{n}$$

Where F_t = Forecast for the coming period, n = Forecast for the coming period, and $A_{t-1}, A_{t-2}, A_{t-3}$ and so on are the actual occurrences in the in the past period, two periods ago, three periods ago and so on respectively.

B. Weighted Moving Average (WMA)

Whereas the simple moving average gives equal weight to each component of the moving average database, a weighted moving average allows any weights to be placed on each element, providing, of course, that the sum of all weights equals 1. The formula for the weighted moving average is

$$F_t = \sum_{i=1}^{n} w_i A_{t-i}$$

$$\sum_{i=1}^{n} w_i = 1$$

Where F_t = Forecast for the coming period, n = the total number of periods in the forecast, w_i = the weight to be given to the actual occurrence for the period t-i, A_i = the actual occurrence for the period t-i.

Although many periods may be ignored (that is, their weights are zero) and the weighting scheme may be in any order (for example, more distant data may have greater weights than more recent data), the sum of all the weights must equal 1. Experience and trial and error are the simplest ways to choose weights. As a general rule, the most recent past is the most important indicator of what to expect in the future, and, therefore, it should get higher weighting.

C. Simple Exponential Smoothing (SES)

In the previous methods of forecasting (simple and weighted moving average), the major drawback is the need to continually carry a large amount of historical data. (This is also true for regression analysis techniques, which we soon will cover) As each new piece of data is added in these methods, the oldest observation is dropped, and the new forecast is calculated. In many applications (perhaps in most), the most recent occurrences are more indicative of the future than those in the more distant past. If this premise is valid that the importance of data diminishes as the past becomes more distant - then exponential smoothing may be the most logical and easiest method to use.

$$F_t = \alpha A_{t-1} + (1-\alpha) F_{t-i}$$

where F_t = exponentially smoothed forecast for period t,

[2]https://www.delhisldc.org/

D. AutoRegressive Integrated Moving Average (ARIMA)

ARIMA models provide another approach to time series forecasting. Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting, and provide complementary approaches to the problem. While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

An ARIMA model can be understood by outlining each of its components as follows:

- **Autoregression (AR)** refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
- **Integrated (I)** represents the differencing of raw observations to allow for the time series to become stationary, i.e., data values are replaced by the difference between the data values and the previous values.
- **Moving average (MA)** incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each component functions as a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with p, d, and q, where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as:

- **p**: the number of lag observations in the model; also known as the lag order.
- **d**: the number of times that the raw observations are differenced; also known as the degree of differencing.
- **q**: the size of the moving average window; also known as the order of the moving average.

Data Preprocessing: We used last one months data at a frequency of 30 minutes instead of 5 minutes due to computational overload. Best hyper-parameters were searched using grid search method and used for model training. As the data is seasonal with seasonality and their of varying nature of trends in data given the season of year the optimized values of p, d, q were obtained accordingly.

Model Architecture: Once the optimized hyper parameters were available after the grid search finished, standard ARIMA model was used for training and prediction using Python's *statsmodels* library.

E. Recurrent Neural Networks (RNN)

Recurrent Neural Networks are the state of the art algorithm for sequential data and among others used by Apple's Siri and Google's Voice Search. This is because it is the first algorithm that remembers its input, due to an internal memory, which makes it perfectly suited for Machine Learning problems that involve sequential data. It is one of the algorithms behind the scenes of the amazing achievements of Deep Learning in the past few years.

In a RNN, the information cycles through a loop. When it makes a decision, it takes into consideration the current input and also what it has learned from the inputs it received previously.

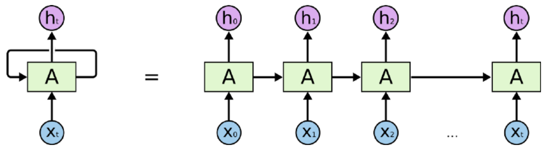


Fig. 3: RNN architecture

Data Preprocessing: For RNN, LSTM and GRU models Data was made stationary by detrending using one lag differencing and was re-scaled to [-1, 1] scale. This helps in faster convergence and prevents features with relatively large magnitudes like previous year load demand to carry a larger weight during training. The model was trained on last 60 days of data, with each training data vector containing load of a specific time for last 10 days and the label was the load of the $11th$ day at the same corresponding time.

Model Architecture: The model consisted of two layers one with two simple RNN cells with hyperbolic tangent function (*tanh*) as activation function and followed by a dense layer without any activation. Metric used for loss was mean square error optimized with Adam optimizer with a learning rate of 0.001. The model was trained for 15 epochs. The code was written in Python using Keras library.

F. Long Short-term Memory (LSTM)

A usual RNN has a short-term memory. There are two major obstacles RNNs have or had to deal with, exploding gradients and vanishing gradients. Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks, which basically extends their memory. Therefore it is well suited to learn from important experiences that have very long time lags in between.

LSTMs enable RNNs to remember their inputs over a long period of time. This is because LSTMs contain their information in a memory, that is much like the memory of a computer because the LSTM can read, write and delete information from its memory. This memory can be seen as a gated cell, where gated means that the cell decides whether or not to store or delete information (e.g if it opens the gates or not), based on the importance it assigns to the information. The assigning of importance happens through weights, which are also learned by the algorithm. This simply means that it learns over time which information is important and which not.

In an LSTM you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isnt important (forget gate) or to let it impact the output at the current time step (output gate).

Data Preprocessing: Data preprocessing used in LSTM is similar to the one used in RNN model.

Model Architecture: The model consisted of two layers one with one LSTM cell with hyperbolic tangent function (*tanh*) as activation function followed by a dense layer without any activation. Metric used for loss was mean square error optimized with Adam optimizer with a learning rate of

0.001. The model was trained for 15 epochs. The code was written in Python using *Keras* library.

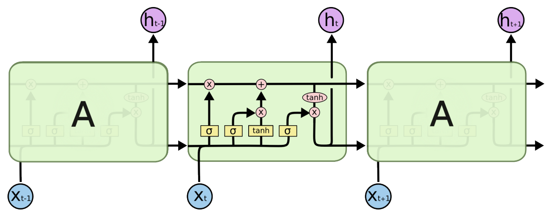


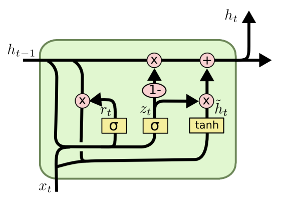
Fig. 4: A LSTM cell

G. *Gated Recurrent Unit (GRU)*

A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU. It combines the forget and input gates into a single update gate. It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.

Data Preprocessing: Data pre-processing used in LSTM is similar to the one used in RNN model.

Model Architecture: The model consisted of two layers one with one GRU cell with hyperbolic tangent function (*tanh*) as activation function followed by a dense layer without any activation. Metric used for loss was mean square error optimized with Adam optimizer with a learning rate of 0.001. The model was trained for 15 epochs. The code was written in Python using *Keras* library.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Fig. 5: A GRU cell

IV. RESULTS

This section shows the results of the above mentioned methodologies on the load data for date: 27/11/2018. All the plot below show the real load data obtained from the SLDC Delhi and the load curve forecasted by the above mentioned methods. The real load curve clearly shows typical day wise variations.

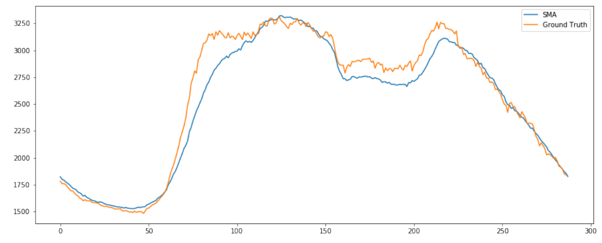


Fig. 6: SMA

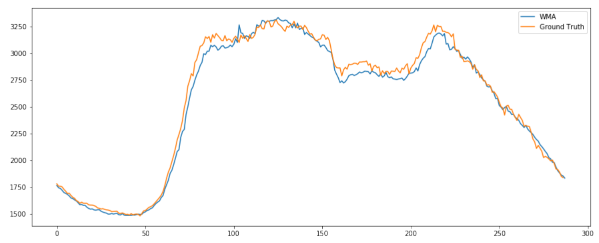


Fig. 7: WMA

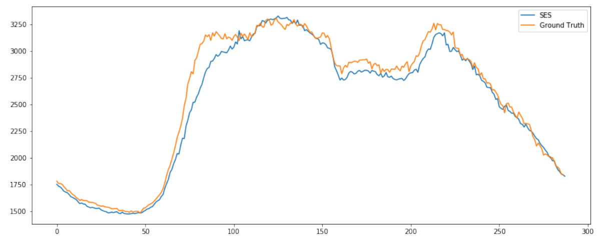


Fig. 8: SES

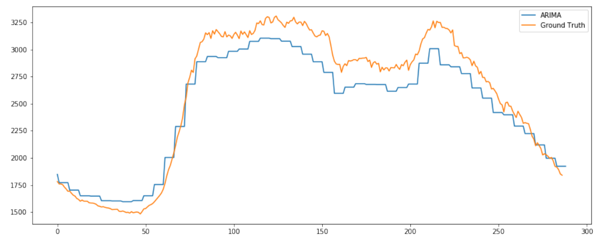


Fig. 9: ARIMA

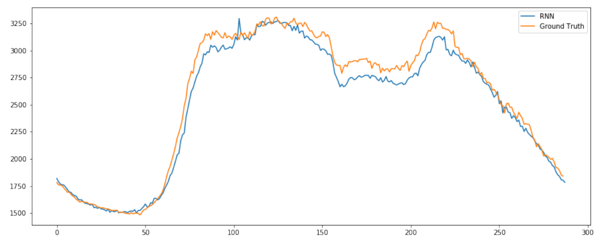


Fig. 10: RNN

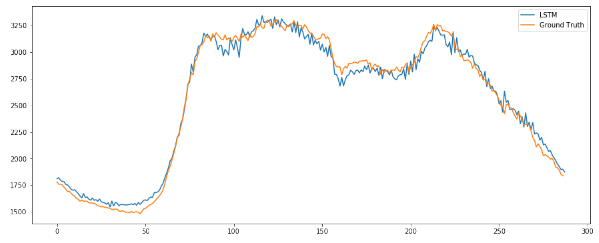


Fig. 11: LSTM

Forecasted results by all 7 algorithms:

For the sake of comparing these seven methods performance wise, the error metrics RMSE[3] and MAPE[4] are taken

[3]Root Mean Square Error
[4]Mean Absolute Percentage Error

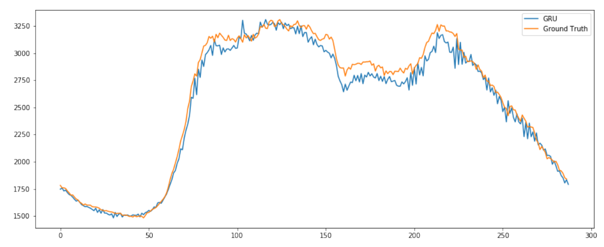


Fig. 12: GRU

into account. The below table shows the error metrics for the date 27/11/2018.

Serial No.	Method	RMSE	MAPE
1	SMA	126.17	3.16
2	WMA	62.86	1.85
3	SES	100.01	2.73
4	ARIMA	178.66	5.82
5	RNN	96.86	2.8
6	LSTM	73.56	2.52
7	GRU	88.35	2.48

V. CONCLUSION

Six algorithms were implemented in the project namely SMA, WMA, SES, ARIMA, RNN, LSTM and GRU. While the performances vary every day because all the models are trained everyday incorporating the last day's data in training set, WMA, LSTM or GRU are among the best performing models. The performance of the LSTM and GRU models can be further improved by incorporating weather parameters and other features like day, weekend, month into the training dataset.

ACKNOWLEDGMENT

We would like to extend our sincere thanks to our mentor and guide Prof. D. Singh for his constant guidance and supervision which has helped us in realizing this project to its completion.

REFERENCES

[1] Long Term Load Forecasting with Hourly Predictions based on Long-Short-Term-Memory Networks, 2018 IEEE Texas Power and Energy Conference (TPEC) Feb 2018
[2] http://colah.github.io/posts/2015-08-Understanding-LSTMs/
[3] https://machinelearningmastery.com/
[4] https://www.delhisldc.org/