CS230

LOAD AND SOLAR POWER FORECASTING IN A RESIDENTIAL DISTRIBUTED ENERGY SYSTEM

Wael Abid*
Department of Computer Science
Stanford University
waelabid@stanford.edu

Abstract

Lately, it has been possible to acquire data about electricity consumption in a household precisely and efficiently thanks to smart meters. We are therefore able to observe this data and analyze it then use it to forecast power demand and solar power generation in a household. In this paper, we describe out work in forecasting these quantities using state of the art models to achieve performance that is comparable or an improvement of what is achieved in the literature.

1 Introduction

The infrastructure that defines the U.S. electric grid is based largely on pre-digital technologies developed in the first part of the 20th century. In subsequent decades, grid development has evolved through emphasis on safety, accessibility, and reliability to security and resiliency. Throughout this evolution, the grid mainly relied on centralized power plants and developed protocols to provide system reliability based on that model. However, the increasing use of renewable generation and distributed energy resources (DER), such as residential solar and home energy storage, along with customers' changing energy usage patterns are leading to greater uncertainty and variability in the electric grid. New tools are required to create a flexible and modern electric grid that can meet this increase in renewable generation and DERs, while providing the quality of service, resiliency, and reliability that customers expect. If successful, we will have contributed significantly to resolving the challenges the US faces regarding energy consumption, management and grid performance maintenance. This could shrink the need for massive energy storage facilities, standby natural gas-fired generators and long-distance transmission lines. Electrical load helps the electrical generating and distribution companies plan their capacity and operations in order to reliably supply all consumers with the required energy so that there are no under generation or over generation. It also helps in deciding and planning for maintenance of the power systems with least impact on users. The input is a time series consisting of a week worth of power demand data in a household that the model uses to predict the next day in that time series as a sequence of 48 timesteps, which represent 30-minute steps in that day. We use models that are traditionally used for time series predictions such as the SARIMA (Seasonal Autoregressive integrated moving average) as well as more novel models such as LSTM and CNN, and other models that are a fusion of these models like the CNN-LSTM Encoder-Decoder, and ConvLSTM or Convolutional LSTM.

CS230: Deep Learning, Fall 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

In his paper Short-Term Forecasting: A Review of Procedures in the Electricity Supply Industry Derek W. Bunn details many forecasting procedures which have been developed in the electricity supply industry to serve the short-term needs of load dispatching. Most importantly, he talks about ARIMA, autoregressive models and General Exponential Smoothing. Another paper Electricity load forecasting for residential customers: Exploiting aggregation and correlation between households by Samuel Humeau et al. discusses which of s support vector machine for regression (SVR) and multilayer perceptron (MLP) is better for the forecasting tasks depending on the scale of the dataset (dataset that comprises information about a single house vs. a dataset that comprises information about many houses). In another paper, Deep Learning for Household Load Forecasting—A Novel Pooling Deep RNN, Heng Shi et al. discusses A novel pooling-based deep recurrent neural network is proposed in this paper which batches a group of customers' load profiles into a pool of inputs. Essentially the model could address the over-fitting issue by increasing data diversity and volume. The 3 papers discussed above present a good time-line of what is being discussed in the literature from the 1980s until 2018. To the best of our knowledge, some of the models we discuss, haven't been used to forecast household power consumption.

3 Dataset and Features

We have data for the month of July 2015 sampled at a 1-minute rate over 95 nodes (group of houses). We down sampled it so that it is sampled at a rate of 30 minutes because sampling it at a rate of 1 minute has a very high variance and the meaning of the oscillations of the data between a minute and the next one doesn't have a meaningful trend.

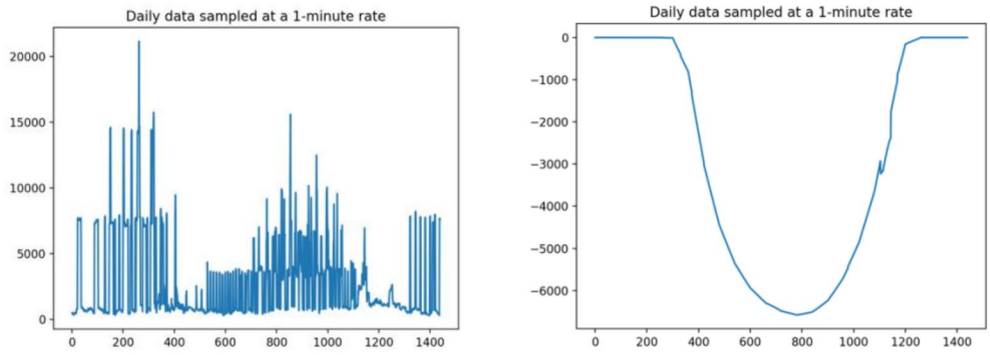

Figure 1: Daily data sampled at a 1-minute rate

I tried different sampling frequencies (5, 10, 15, 20, 30 and 60 minutes) and 30 minutes is judged to be the sampling rate at which the patterns of power usage seen are variable and conforming with the general consumption in a household.

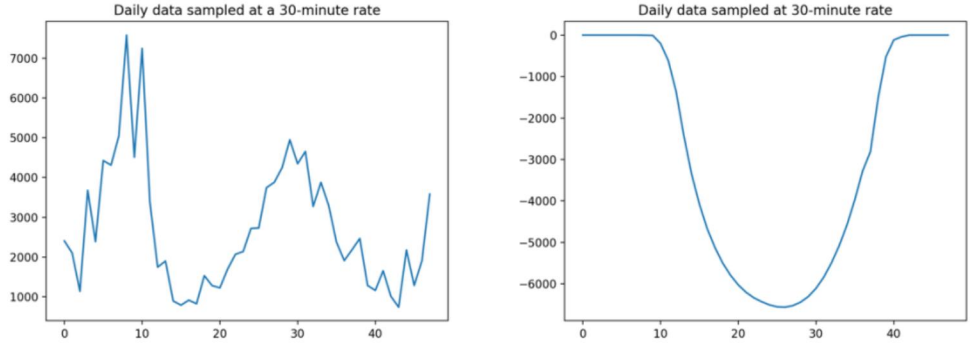

Figure 2: Daily data sampled at a 30-minute rate

I analyzed the data by separating it into weekdays/weekends to see if there are any different trends between the data in those 2 categories but haven't found a major difference apart from that the peak of power consumption during weekdays is bigger than that of the weekends. Dividing the dataset into the different days of the week yields that the trend during the days is similar across all days. The variance across the 95 nodes isn't different so I opted out of an architecture that outputs predictions for each of the 95 houses and chose to consider the 95 nodes as training examples of the time series of 31 days sampled at a rate of 30 minutes. After down sampling, we obtain a dataset that has 31 days, 48 time steps each over 95 homes. The figure below describes it best.

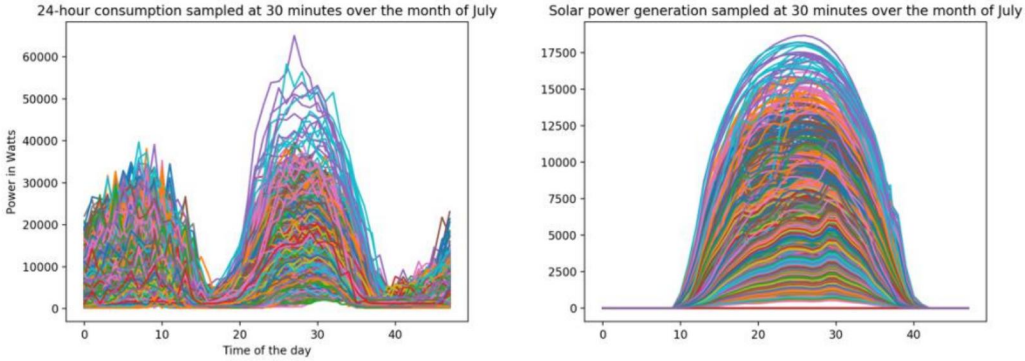

Figure 3: All power data over the month of July sampled at a 30-minute rate

The reasoning behind the following data split is that we wanted to train, develop, test our model on both weekdays and weekends. Therefore, training is first 21 days, development and testing are the next 10 days split equally, resulting in roughly a 70-15-15 split.

4 Methods

The model is given as input a week worth of data and is expected to return the next day as a 48-step sequence. The metric used is NRMSE (Normalized Root Mean Squared Error). In the literature, the two well-known methods for evaluating forecasted values are the mean of average percentage error (MAPE) and the root mean square error (RMSE). In this paper, we sometimes make forecasting at the household level. For many households, there are periods when the consumption is 0, or extremely low. In these cases, the MAPE is infinite, making its evaluation inconvenient. RMSE does not have this problem. We observe that the RMSE is strongly influenced by the order of magnitude of the data. Our data set contains numerous households whose hourly consumption ranges between 0.05 kWh and 60 kWh. In order to reduce their influence, we use the following normalization for the RMSE:

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}}$$

- SARIMA:
    - SARIMA stands for Seasonal Autoregressive Integrated Moving Average. It is an extension of the ARIMA model that handles seasonality better. ARIMA models are, in theory, the most general class of models for forecasting a time series which can be made to be stationary.
- LSTM:
    - Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.1 They work tremendously well on a large variety of problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn. Our model consists of 2 layers of LSTM, each layer

having 48 cells, then a 48 Dense layer to interpret the output of the LSTMs and then an output Dense layer.

- Encoder-Decoder models:

    - An encoder-decoder LSTM is a model comprised of two sub-models: one called the encoder that reads the input sequences and compresses it to a fixed-length internal representation, and an output model called the decoder that interprets the internal representation and uses it to predict the output sequence.
        - LSTM-LSTM: our model consists of an LSTM encoding layer of 48 cells, a repeat vector layer for the model to output the sequence of the right length, and a decoding LSTM layer of 48 cells with a 48 Time Distributed Dense layer and an output layer to interpret the outputs of the LSTM layer.

        - CNN-LSTM: our model consists of an CNN encoding layer of 48 filters and a kernel size of 2, a repeat vector layer for the model to output the sequence of the right length, and a decoding LSTM layer of 336 cells with a 336 Time Distributed Dense layer and an output layer to interpret the outputs of the LSTM layer.

- ConvLSTM:

    - A variation on the CNN LSTM architecture is the ConvLSTM that uses the convolutional reading of input subsequences directly within an LSTM's units. This approach has proven very effective for time series forecasting for its ability of interpreting spatiotemporal data as observed in the Convolutional LSTM Network: *A Machine Learning Approach for Precipitation Nowcasting* paper of Xingjian Shi et al.

5 Experiments/Results/Discussion

The following table represents the results we have obtained with each model we tried.

Model	***NRMSE***	***RMSE***
SARIMA	33.45%	1170kWh
CNN	13.62%	440kWh
LSTM	11.64%	376kWh
ENCODER-DECODER CNN-LSTM	15.09%	487kWh
ENCODER-DECODER LSTM-LSTM	11.36%	366kWh
ConvLSTM	**11.29%**	**364kWh**

The following six figures represent in order: Load forecast for the models: SARIMA, CNN, LSTM, CNN-LSTM Encoder-Decoder, LSTM-LSTM Encoder-Decoder, ConvLSTM.
In the seventh figure, we use the ConvLSTM model, which has the best performance to predict the solar power.

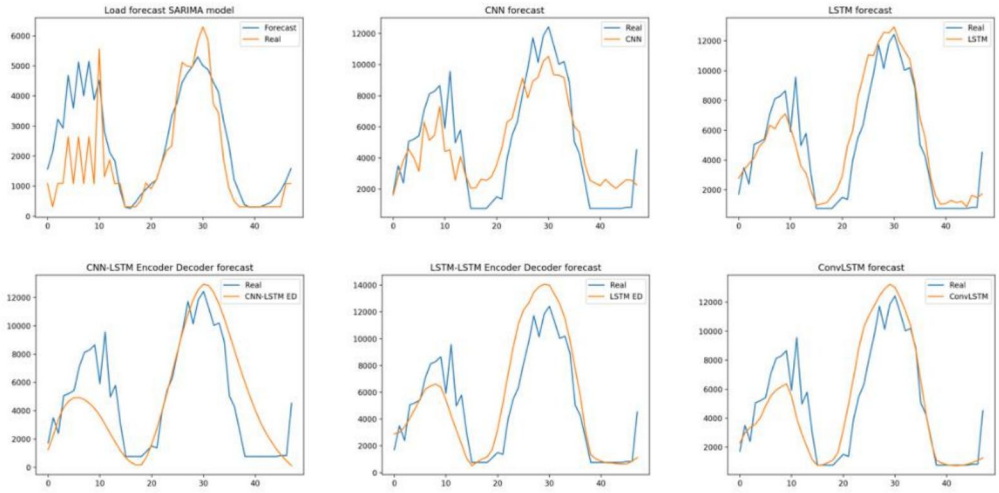

Figure 4: Load forecasting using all the models

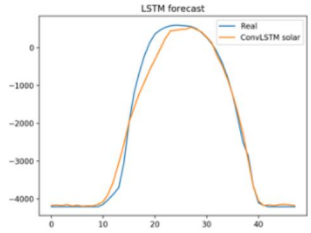

Figure 5: Solar forecasting using the ConvLSTM model

The performances we got are commensurate with what Heng Shi obtained in his paper [3] (view table 1: performance comparison) and in the literature in general, which are values that range from 9-15% for state of the art models. This is a good indicator and shows that the models work and are fitting the data well. The best performance as seen in the table in the one of the ConvLSTM because it is particularly good at capturing spatiotemporal aspects of the data. It is also the second fastest model to train after the CNN. It is fast because unlike a traditional LSTM model, it relies on parameter sharing.
The fastest model to train is the CNN model. However, it tends to overfit easily. Therefore, we used Dropout and early stopping to achieve better performance on the test set.
The LSTM-LSTM Encoder-Decoder model performs better than the CNN-LSTM Encoder-Decoder model because the encoding part of the first model is an LSTM which is particularly better at capturing and keeping time-series information even at further LSTM cells.
We initially had issues with the logic we're using to preprocess our data. But as soon as that was fixed as discussed in the data section above, we started choosing which models we're going to work with. We started with a simple one-layer LSTM and as we made the layer size bigger or added another layer, we found problems with exploding gradients as we train the model. It has turned out that changing the activation of the first LSTM layer from 'ReLu' to 'sigmoid' solved the problem. This was the problem with all the models that had LSTM as their first layer, even though we normalized the dataset. It turned out that data normalization wasn't enough for the exploding gradients problem and the reason we thought of using 'sigmoid' activation is that it limits the output of the first layer to values between 0 and 1.

6 Conclusion/Future Work

The performances we got are commensurate with what is found the literature, (9-15% for state of the art models). This is a good indicator and shows that the models work and are fitting the data well. The best performance as seen in the table in the one of the ConvLSTM. Future work is focused on integrating a battery (e.g. Tesla Powerwall) within the house and trying to minimize spending on electricity making use of the battery to charge from PV panels and to adapt to the fluctuations of the price of electricity every day and season and use solar.

7 Contributions

Wael Abid: worked on all aspects of the project solo. It was a challenging, at some times discouraging, at most times exciting, overall rewarding journey with the final project and the class in general. Thanks to the CS 230 teaching staff. Special mention to my TAs Steven, for making almost everything easy to understand, listening, helping and giving great tips, and to Cristian for being cheerful at every conversation, for helping me solve problems with the final project, within TA appointment time or outside of it.

Project code: https://github.com/abidwael2/CS-230-Final-Project

References

[1] Bunn, Derek W. “Short-Term Forecasting: A Review of Procedures in the Electricity Supply Industry.” The Journal of the Operational Research Society, vol. 33, no. 6, 1982, pp. 533–545. JSTOR, JSTOR, www.jstor.org/stable/2581037.

[2] S. Humeau, T. K. Wijaya, M. Vasirani and K. Aberer, "Electricity load forecasting for residential customers: Exploiting aggregation and correlation between households," 2013 Sustainable Internet and ICT for Sustainability (SustainIT), Palermo, 2013, pp. 1-6. doi: 10.1109/SustainIT.2013.6685208

[3] H. Shi, M. Xu and R. Li, "Deep Learning for Household Load Forecasting—A Novel Pooling Deep RNN," in IEEE Transactions on Smart Grid, vol. 9, no. 5, pp. 5271-5280, Sept. 2018. doi: 10.1109/TSG.2017.2686012

[4] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Comput. 9, 8 (November 1997), 1735-1780. DOI=http://dx.doi.org/10.1162/neco.1997.9.8.1735

[5] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[7] Chollet, François, et al. Keras. https://keras.io (2015).