Energy Consumption Prediction for Optimum Storage Utilization

Eric Boucher, Robin Schucker, Jose Ignacio del Villar
Stanford University
eboucher@stanford.edu, schucker@stanford.edu, jidvom@stanford.edu

Introduction

Continuous access to energy for commercial and industrial sites across the United States to cover their needs is essential for them to keep producing goods and providing services. Nonetheless, the price of electricity provided by utilities is variable throughout the day. In addition, Commercial and Industrial customers have to pay demand charges that are proportional to the maximum power drawn from the grid during each month. This rate structure makes these sites very sensible to how they consume power on a minute to minute basis. To solve this issue, eLum provides businesses that use solar energy and storage with a system that allows them to optimize in real time the choice between using the electricity they have produced and buying from their utilities provider. But to make this happen, accurate predictions of the electricity consumption of each site at one or two days horizon must be made. That way, eLum will allow these sites to optimize the resources they spend on energy.

Main Objective

The main objective of this project is to accurately predict the ”next day” energy consumption needs every 5 minutes for 100 businesses in the USA.

Methods and Algorithms used

Fourier and STD

In the case of time series such as energy consumption, a standard approach is to use an algorithm which finds periodicity patterns in the data and use theses patterns to predict the future. A standard algorithm in this case is the Fourier analysis which approximates the data by a sum of trigonometric functions. A more elaborated version is the STD - Standard Trend Decomposition, which approximates the function by a sum of periodic functions with additional seasonal and trend functions with lower or no periodicity. This enables more flexibility in our case as it allows the algorithm to take into account a rising demand or significant changes in equipment.

We have implemented both algorithm and unsurprisingly, STD performs systematically better than simple Fourier analysis. However, we found out that our error on the training set varied significantly with the number of days taken into account during training; and more data points is not always better. We found that on average, 3-weeks of data yields the best results (see Figure 1, ie the lowest test error. Our first hypothesis is that future points will be more similar to data points that happened a few days ago than what happened a long time ago.

Figure 1
STD Error vs Training Size

K Nearest Neighbors

We have implemented a K nearest neighbor (KNN) algorithm in order to predict the last day of our data set. The KNN algorithm works in the following way: For each facility:

- Looking at past data, stores the electricity consumption of P days (P*24*12 points) as *keys* and the electricity consumption of the day right after the P days as *value*
- Using all the data in the train set (353 days) we then have 353 - P (*key*, *value*) pairs stored in a table (see Figure 2)
- In order to predict the day in the test set, we look at the last P days of the train set (= *query_key*) and compare that *query_key* to the *keys* stored in our table. For each *key* in the table, the distance to our *query_key* is the norm of (*query_key* - *key*). Then we select the K *keys* which have the lowest distance.
- Our predicted value (= a day) is then: (i = 1 key that is closest to *query_key*, i = 2 second closest etc)

$$predicted\_value = \frac{\sum_{i=1}^{K} \frac{value(i)}{distance(i)}}{\sum_{i=1}^{K} \frac{1}{distance(i)}}$$

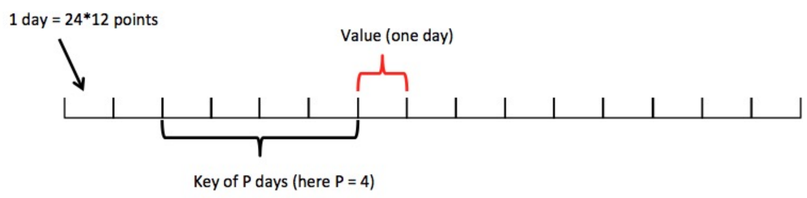


Figure 2
KNN key-value pair

The variables that we can tune on this model, is the number of days we look into the past to predict the next day (P = size of key vector) and the number of neighbors we include in the prediction (K). Looking at the error we get on the test set, the lowest error is 0.14 for P = 6 and K = 10. Playing around with these values we see that this model is very bad (error >1) for if P = 1 or P >7. The K dependence varies for different P values. (see Figure 3)

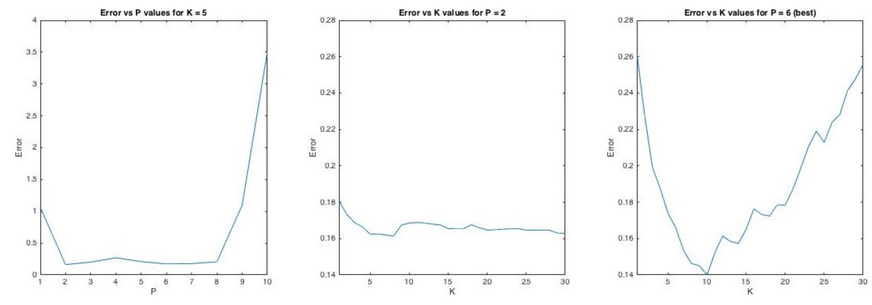
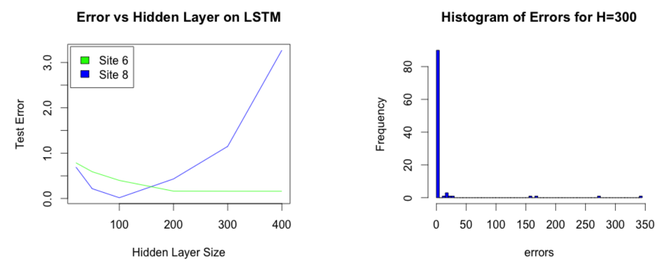


Figure 3
Error of KNN depending on model parameters

Neural Nets LSTM

Building on literature on the usage of neural nets for time series prediction, we decided to implement an LSTM. And started with a one hidden layer LSTM model with a time step of 7. The model takes into account a sequence of 7 days step-wise and outputs the prediction for the following one. The variables that we can tune on this model are: the number of days we look into the past to predict the next day; the number of hidden layers; and the number of training epochs. We chose T=7 as it allows us to have a complete view of a week and e=300 with dropouts to avoid over-fitting. We then tried to select the best possible parameter for H, the number of hidden layers. Unfortunately, different time series of different sites behaved very differently. H = 300 gave good results overall and even outperformed STL on some sites, but simultaneously gave extremely bad results for certain sites.



(a) Error vs Hidden Layer on LSTM **(b)** Histogram of Errors for H = 300

Figure 4: NN results

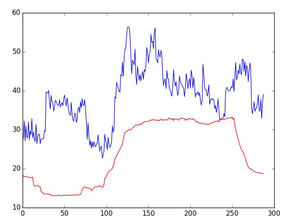
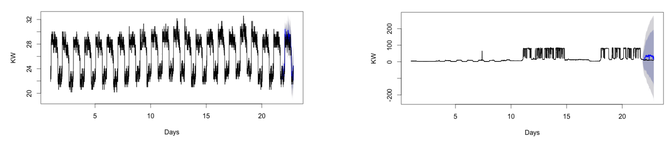


Figure 5
Prediction of consumption of site 6 with LSTM

Limitations of Models

STL and KNN models have similar errors (although KNN is slightly better with 0.14 error vs 0.16 for STL) and are good models for the facilities that are very cyclical. Those that have similar day to day consumption are very easy to predict whereas facilities which have strong day to day differences (for eg schools during holidays) have high errors (see figures 6a and 6b). This makes sense for STL because we implemented a very naive model that assume that there is one trend for the whole training set when, depending on the nature of the facility there is obviously a difference between a normal day and a holiday. The same goes the KNN where a holiday will be hard to predict. We have to keep in mind that we are trying to predict the 30th of December which is an unusual day (Christmas holidays) and which is why we could see such high error. Furthermore, STL seems to be consistently underestimating (but with the right shape) which could be why the KNN estimator is slightly better. This under estimations could be due to colder temperatures compared to the train days where heating loads are higher. Between KNN and STL, we will choose STL as the model to put in our pipeline because it has the key advantage of also modeling the training set (whereas KNN does not and just stores a table) so it will be easier to work with when we will add more features to our model. As we mentioned before, some limitations exist on the neural network that we implemented, as is shown in the bad performance on some sites, even though the performance was good on other sites.



(a) Best STD Forecast - Err = 0.9*10e-3 **(b)** Worst STD Forecast - Err = 4.9

Figure 6: STD Forecast

Conclusions

- Neural nets LSTM, while promising, gives us good results on 84 sites, but on the remaining 16 we get extremely bad results (error¿1). KNN actually performs better than neural nets LSTM except in 5 sites.
- After removing sites that had the same output we have that the error of the KNN is of 0.18, while the error of the neural net is 0.64.