Energy Consumption Prediction for Optimum Storage Utilization

Eric Boucher, Robin Schucker, Jose Ignacio del Villar

December 12, 2015

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.

Solving this issue involves providing businesses that use solar energy and storage with a system that would allow them to optimize in real time the choice between using the electricity they have produced and buying from their utilities provider. This is exactly what the startup eLum does. To make this happen, there is a need to have accurate predictions of the electricity consumption of each site at one or two days horizon. Using data provided to us by eLum, we have tried to solve this problem, and helped sites optimize the resources they spend on energy.

There have been many attempts to solve this issue in the past using extremely varied methods. Nonetheless, the literature points to the linear regression method (as in [1] and [2]), the KNN method (as in [3] and [4]), and specially neural networks ([5], [6], and [7]).

Main Objective

Much research has been realized in the field of Short Term Load Forecasting, The main objective of this project is to accurately predict the "next day" energy consumption needs for 100 businesses in the USA. The input data is consumption over a certain period (up to year) at 5 min intervals and we want to predict the consumption of the day after that period. Admittedly, this goal is complex, as we needed to predict 288 energy consumption needs to complete a whole day of prediction (with a prediction every 5 minutes). Nonetheless, we considered this goal to be challenging and engaging enough for us to try to tackle it. To find what we would consider a "good" method, we wanted to make sure that it worked well on all the different sites.

Methods and Algorithms used

Error Used

$$Error = \frac{1}{N_{Sites}} \sum_{site} \frac{(Y_{pred} - Y)^T (Y_{pred} - Y)}{Y^T Y}$$

This error, while not completely perfect, was made so that we do not favor sites that consume more energy on average (greater Y). Nonetheless, this gives us a good estimate for the overall error. Note that Y and Y_{pred} represent the true consumption and the estimated consumption respectively.

Linear Regression

Our first approach to the problem was to implement a simple linear regression model. We randomly separated our data points into 70% training set and 30% testing set. As features, we used what was available to us: the date. Thus, we trained a linear regression model using the weekday and the hour of the day:

$$Consumption_{est} = \sum_{i,j} \theta_{i,j} X_{i,j}$$

Where i=1..7 is the weekday (i.e. Monday, Tuesday...) and j=0..24 is the hour of the day. For example, $X_{2,8} = 1$ if we want to predict a Monday between 8:00 am and 8:59 am, and $X_{2,8} = 0$ any other time or day of the week combination. This very simple model gave us an average test error of 11% across all sites. It is promising since this algorithm models the consumption over the whole year and thus any day could be foretasted using only what weekday it is as information. This model corresponds to our baseline case, and any other more sophisticated time series model we need to beat this error to have potential. The difference between test and train error is very small (10.96% vs 10.87%) and we only use $24 * 7 = 168$ features to predict over 100,000 points. Thus, we believe that our error is mostly from high bias rather than high variance. As a result we need to use more features in order to reduce the error.

More Features

We have found site specific 30 min interval weather data (from NREL NSRDB), including solar irradiance,

temperature, wind speeds, relative humidity, and pressure for 2012. Intuitively, weather data, especially temperature and solar irradiance (if the site has solar panels) would play a large role in energy consumption. Adding only linear terms in weather data did not seem to help much, as the test error drop only to 10.7%. However, adding polynomial terms (especially $temp^2$, $wind^2$, $wind^2 temp$ ...) helped a lot and dropped the test error down to 7.0%. Again, the test and train errors were very similar so we are not over fitting. The distribution of errors can be found in Figure 2. We also tried adding holiday data, but this resulted in a over fit for those days as they are very few of them and we only have data for one year. Looking at school consumption in particular, the academic calendar has a huge impact as during the long summer break, the electric consumption is far lower than any other months. However, adding this feature did not change the error significantly.

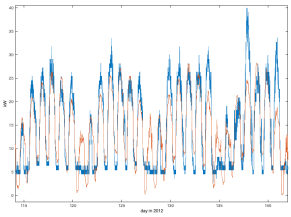

Figure 1
Linear regression model of a site that has an average error (blue = true consumption, red = modeled consumption)

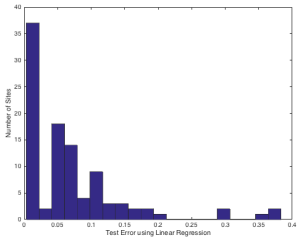

Figure 2
Error histogram for all sites using Linear Regression

K Nearest Neighbors

As mentioned in [3], a K nearest neighbor (KNN) algorithm seems to be promising for this application. We have implemented a KNN in order to predict an entire day (selected randomly) of our data set. The KNN algorithm works in the following way: For each facility:

- Looking at past and future 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 3)
- 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)

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

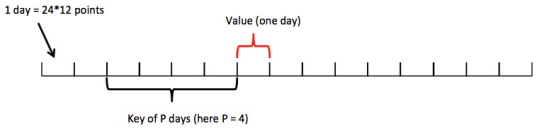

Figure 3
KNN key-value pair

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 in days) and the number of neighbors we include in the prediction (K). As a test set, we tried to predict 20 random days, that were never included in our training data. The lowest error we obtained is 13.5% for P = 1 and K = 5. Playing around with these values we see that this model becomes worse as P increases. The K dependence is not so strong and any value around 5 produces similar errors. This method does not seem to work well on our dataset as the best error is higher than using linear regression without weather data.

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 test set (i.e. the day to be predicted)

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, 2-weeks of data yields the best results (see Figure 5, 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 as in general weather patterns are usually on longer time scales. 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 as in general weather patterns are usually on longer time scales.

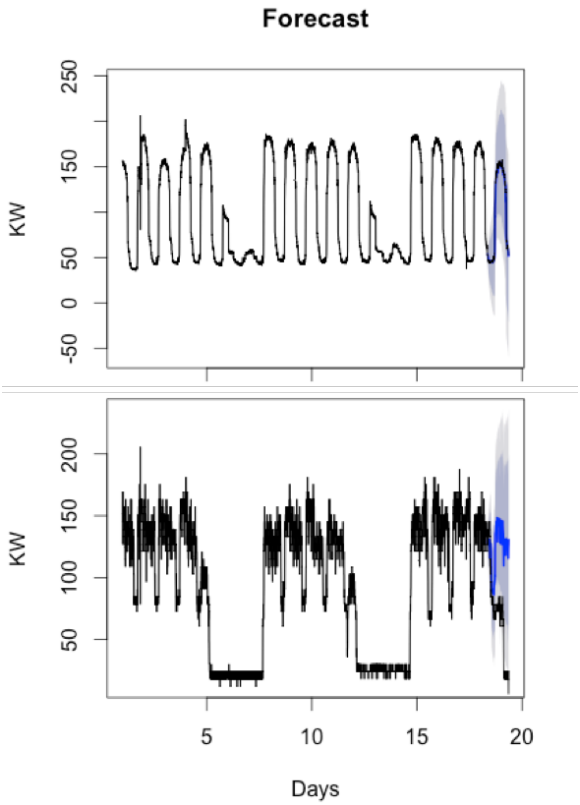

Figure 4
Best and Worst STD Forecasts
E = 0.12*10e-3 vs E = 0.89

Figure 5

STD Error vs Training Size

Neural Nets - LSTM

Building on literature on the usage of neural nets for time series prediction, we decided to implement an LSTM. LSTM stands for Long-Short-Term-Memory, a kind of Recurrent Neural Network algorithm that can "learn from experience" thanks to memory gates (see figure 6).

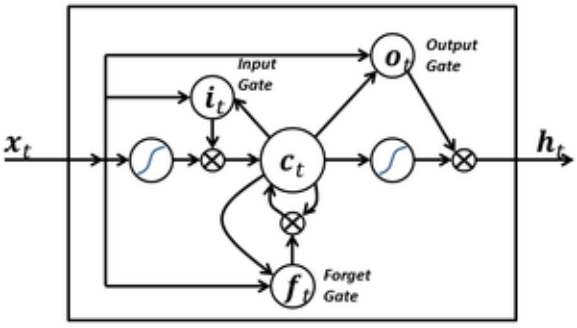

Figure 6
Long-Short-Term-Memory

Using the python package Keras [8] we first implemented an LSTM with a *tanh* activation before a linear activation for the output. 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 stepwise 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 STD on some sites, but simultaneously gave extremely bad results for others. Feature selection was a therefore a tough process and our hope of finding an universal algorithm did not seem very realistic in the case of neural nets. Adding more weather data did not improve our results significantly. Although our results on linear regression suggest that including higher order terms or having a deeper neural net might help.

We then went on with different models, looking not at a vector but at each five minute value individually in the LSTM and a size-step of 500. Despite being a more classical approach, it return non significant results. We suspect that we did not have enough data to really grasp the trends and admittedly were asking a lot of our model. Indeed, there is tremendous variability at the 5-minute levels even on two days that look extremely similar from a distance.

Lastly, we tried a simple feed forward neural network on a 5-minute basis but the results were inconclusive, likely due to a not so surprising error propagation.

Figure 7
Error vs Hidden Layer on LSTM

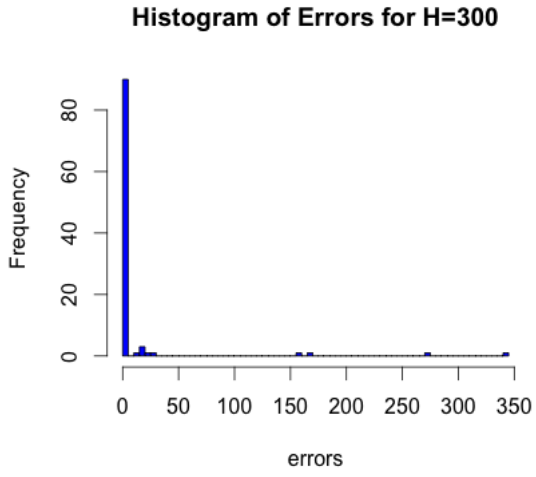

Figure 8
Histogram of Errors for H = 300

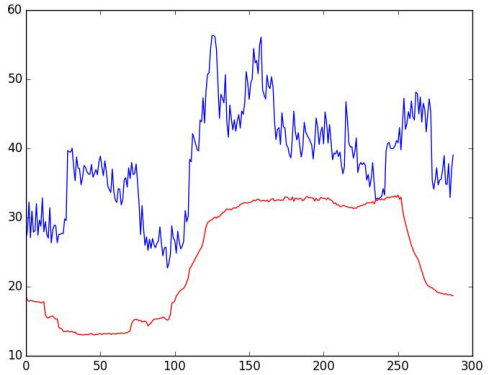

Figure 9
Prediction of consumption of site 6 with LSTM

Limitations of Models and Next Steps

STD is the model that works best (4.7% on average across sites) even though it does not use any weather data.(see figures **??** and **??**). This makes sense because STD does not try to model the whole year and only models the last two weeks and predicts the next day from that. Furthermore, STD seems to be consistently underestimating (but with the right shape) which could be why this error is still very high. We think that this underestimation could be alleviated by augmenting our STD with weather data and this would be our main goal for future work. Interestingly, it is hard to tweak the model to become a "one-fit-all" algorithm as very regularized sites are having variance issues while less regularized ones are having bias issues.

Another next step would be to improve our linear regression model to only model a few weeks before our query day rather than modeling the whole data that we have in order to reduce the bias of the model. This would be a fast implementation but we decided to explore other methods rather than the classic linear regression.

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. We believe that the key reason for that is the lack of data (only one year) relative to the daunting task at hand, predicting a vector of 288 values.

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). Linear regression actually performs better than neural nets LSTM overall except in 5 sites.

- Adding weather data is crucial to get an error lower than 10% and currently our best algorithm for prediction the consumption is linear regression. Adding weather features to a linear regression model is straight forward and we were able to integrate all the weather data we had in our model. In contrast, as STD or KNN are time series prediction model, augmenting them with weather features was more challenging and we were not able to implement a solution that leveraged all of the weather information on our hands.

References

[1] Amral, N.; Ozveren, C.S.; King, D., *Short Term Load Forecasting using Multiple Linear Regression*, Universities Power Engineering Conference, 2007. UPEC 2007. 42nd International , vol., no., pp.1192-1198, 4-6 Sept. 2007

[2] Papalexopoulos, A.D.; Hesterberg, T.C., *A regression-based approach to short-term system load forecasting*, Power Systems, IEEE Transactions on , vol.5, no.4, pp.1535-1547, Nov 1990

[3] Al-Qahtani, F.H.; Crone, S.F., *Multivariate k-nearest neighbour regression for time series data A novel algorithm for forecasting UK electricity demand*, Neural Networks (IJCNN), The 2013 International Joint Conference on , vol., no., pp.1-8, 4-9 Aug. 2013

[4] Troncoso Lora, A; Riquelme Santos, J.M.; Riquelme, J.C.; Gmez Expsito, A.; Martnez Ramos, J.L., *Time-Series Prediction: Application to the Short-Term Electric Energy Demand*, Current Topics in Artificial Intelligence, Springer Berlin Heidelberg, vol. 3040, 2004

[5] Hippert, H.S.; Pedreira, C.E.; Souza, R.C., *Neural networks for short-term load forecasting: a review and evaluation*, Power Systems, IEEE Transactions on , vol.16, no.1, pp.44-55, Feb 2001

[6] Lee, K.Y.; Cha, Y.T.; Park, J.H., *Short-term load forecasting using an artificial neural network*, Power Systems, IEEE Transactions on , vol.7, no.1, pp.124-132, Feb 1992

[7] Bakirtzis, A.G.; Petridis, V.; Kiartzis, S.J.; Alexiadis, M.C., *A neural network short term load forecasting model for the Greek power system*, Power Systems, IEEE Transactions on , vol.11, no.2, pp.858-863, May 1996

[8] keras.io, *Keras: Deep Learning library for Theano and TensorFlow*, Last accessed: December 10th 2015