EPiC Series in Computing
Volume 69, 2020, Pages 161–169
Proceedings of 35th International Conference on Computers and Their Applications

Reducing error propagation for long term energy forecasting using multivariate prediction

Maher Selim[1], Ryan Zhou[1], Wenying Feng[1,2], and Omar Alam[2]

[1] Department of Mathematics
[2] Department of Computer Science
Trent University
Peterborough, Ontario, Canada, K9L 0G2
{maherselim, ryanzhou, wfeng, omaralam}@trentu.ca

Abstract

Many statistical and machine learning models for prediction make use of historical data as an input and produce single or small numbers of output values. To forecast over many timesteps, it is necessary to run the program recursively. This leads to a compounding of errors, which has adverse effects on accuracy for long forecast periods. In this paper, we show this can be mitigated through the addition of generating features which can have an "anchoring" effect on recurrent forecasts, limiting the amount of compounded error in the long term. This is studied experimentally on a benchmark energy dataset using two machine learning models LSTM and XGBoost. Prediction accuracy over differing forecast lengths is compared using the forecasting MAPE. It is found that for LSTM model the accuracy of short term energy forecasting by using a past energy consumption value as a feature is higher than the accuracy when not using past values as a feature. The opposite behavior takes place for the long term energy forecasting. For the XGBoost model, the accuracy for both short and long term energy forecasting is higher when not using past values as a feature.

1 Introduction

Time series forecasting is a well-studied field with many critical applications. For example, machine learning models have become widely used in the energy industry for forecasting future energy prices and demands [1, 15]. Advances in sensor and smart meter technology have made large quantities of energy data available [10]. This, combined with increasingly accurate predictions produced by machine learning has made it possible for technologies such as smart grid to flourish.

However, most machine learning models such as LSTM make use of historical values of the load as an input feature. This works well for single timestep predictions, but when a forecast further into the future is required, it becomes necessary to feed predictions back in recursively. In addition, if the model makes use of external features such as weather, forecasts of these features must be generated as well. All of these predictions introduce error, which is compounded when fed back into the model as inputs. Without any external input, models generally become inaccurate or even unstable over multiple timesteps, making multiple-time-step forecasting challenging even for models with high single-time-step accuracy.

In this paper, we propose introducing generated features, features which can be calculated from known variables with perfect accuracy even far into the future. These features limit the effect of the accumulated error, as the model is trained on both these features and recursive inputs. We demonstrate this effect using a benchmark energy dataset. Two machine learning models are trained to perform single-timestep predictions: a long short-term memory (LSTM) neural network and a gradient boosted tree model. Predictions are then made over a period of one month by recursively feeding the model outputs from earlier timesteps in as inputs for later timesteps. We show that without any generated features, error accumulates rapidly over time while inclusion of the generated features leads to smaller accumulated errors over time. We also demonstrate the accuracy of predictions made entirely using generated features, with no recursive term. This version of the model is of interest as it allows forecasts for arbitrary times in the future, without having need of predicting all values in between.

The remainder of this paper is organized as follows. Section 2 provides some background in time series forecasting using univariate and multivariate prediction. Section 3 describes the computational models used for the study. Experimental set-up and results are discussed in Section 4. Lastly, Section 5 concludes the paper.

2 Time Series Prediction by LSTM and XGBoost

We first briefly discuss the mathematical background for time series prediction, as well as specific machine learning algorithms. This forms the basis for our model development and implementation as explained in Section 3.

2.1 Time series prediction

Time series prediction is a problem which aims to predict future values using past values. These are generally past values of the target variable, but this is not necessarily the case. Forecasting models can be broadly classified into univariate and multivariate models based on the number of features used. When forecasting multiple timesteps into the future, models can also be classified into direct, recursive and MIMO approaches [14].

A recursive approach trains a single model to predict a single step in the future, known as a one-step ahead forecast:

$$\hat{x}_t = F(x_{t-1}, x_{t-2}, \ldots)$$

where $x_{(i)}$ represents the value of the variable at timestamp i. This forecasted value is then fed back in as an input and the next timestep is forecasted using the same model:

$$\hat{x}_{t+1} = F(x_t, x_{t-1}, \ldots)$$

This process is repeated until the desired time horizon has been reached. This approach is sensitive to accumulated errors, as any error present in the initial prediction will subsequently be carried forward to later predictions when the predicted value is used as input. However, as only one model is used for all predictions, this allows more resources to be invested in the single model. In addition, this approach is flexible in that it allows forecasting for any time horizon, whether or not the model has been trained on that time horizon.

A direct approach aims to avoid error accumulation by creating a separate model for each potential time horizon. Thus, a collection of models is trained:

$$\begin{aligned}\hat{x}_t &= F(x_{t-1}, x_{t-2}, \ldots)\\ \hat{x}_{t+1} &= G(x_{t-1}, x_{t-2}, \ldots)\\ \hat{x}_{t+2} &= H(x_{t-1}, x_{t-2}, \ldots)\\ \ldots &= \ldots\end{aligned}$$

This avoids propagated errors as no predicted values are used as input. However, as each model is trained independently, the models may not learn complex dependencies between the values $\hat{x}_t$, $\hat{x}_{t+1}$, $\hat{x}_{t+2}\ldots$. This approach is also computationally much more expensive as multiple models must be trained and stored.

The multi-input multi-output (MIMO) strategy attempts to combine the advantages of these approaches by training a single model with multiple outputs to predict all timesteps up to the time horizon simultaneously:

$$[\hat{x}_{t+H}, \hat{x}_{t+H-1}, \ldots, \hat{x}_t] = F(x_{t-1}, x_{t-2}, \ldots)$$

This avoids accumulated error by performing all predictions in one step, as well as modeling any interdependencies between future timesteps. However, this comes at the cost of less flexibility, as all horizons are forecasted using the same model and possible time horizons are limited to those built into the model.

Based on the input features, time series prediction models can be categorized as univariate or multivariate. Univariate models use a single feature, generally the target variable, to predict a future value:

$$\hat{x}_t = F(x_{t-1}, x_{t-2}, \ldots).$$

This has the advantage of allowing smaller and computationally lighter models. Univariate models do not require extra external data and require no feature engineering. However, as they are tied to a single variable, they exhibit more sensitivity to noise and reduced stability for recursive models.

Multivariate time series models use observations of multiple variables or features, often taken simultaneously, and attempt to also describe the interrelationships among the features [3]:

$$\hat{x}_t = F(x_{t-1}, x_{t-2}, \ldots, a^{(1)}_{t-1}, a^{(1)}_{t-2}, \ldots, a^{(2)}_{t-1}, a^{(2)}_{t-2}\ldots)$$

where each $a^{(i)}$ represents the time series of an external feature. This has the obvious advantage of modeling relationships between the target and external variables, but at the cost of a bulkier model and higher computational costs. Building such a model generally also requires obtaining measurements of external features; the difficulty of this is highly dependent on data availability.

It is also possible for a multivariate model to employ no past information about the target variable:

$$\hat{x}_t = F(a^{(1)}_{t-1}, a^{(1)}_{t-2}, a^{(1)}_{t-3}, \ldots, a^{(2)}_{t-1}, a^{(2)}_{t-2}, a^{(2)}_{t-3}, \ldots).$$

In this case, predictions must be made solely based on the relationships of external features to the target variable. Such a model is rarely used in practice as training the model in the first place requires knowledge of past values of the target variable, but may see use if obtaining a full time series of the target value is difficult due to missing or unusable values. In addition, as the output of the model is never used as an input, error accumulation is limited. If future values for the external features can be obtained, this approach allows prediction based on those values without first predicting earlier time horizons.

To demonstrate our approach, we will apply recursive univariate, multivariate and the modified multivariate techniques to the two machine learning algorithms described as follows.

2.2 Long Short-Term Memory (LSTM) neural networks

Long short-term memory is a type of recurrent neural network architecture designed to extract long-term dependencies out of sequential data and avoid the vanishing gradient problem present in ordinary recurrent networks [9, 13]. These properties make it the method of choice for longer time series and sequence prediction problems [8, 16]. Several variations of the LSTM unit have been successfully applied to energy forecasting and other areas [2, 12]. The standard LSTM architecture [9] described below is applied in our study.

Each LSTM cell contains a cell state (h_{t-1}), the long-term memory, and a recurrent input (y_{t-1}) - the short-term memory. It also contains three "gates": neurons which output values between 0 and 1 and are multiplied with the information flowing into and out of the cell. The forget gate σ_f controls the amount of information discarded from the previous cell state. The input gate σ_u operates on the previous state $h[t-1]$, after having been modified by the forget gate, and decides how much of a new candidate state $\tilde{h}[t]$ to add to the cell state $h[t]$. The output $y[t]$ is produced by squashing the cell state with a nonlinear function $g_2(\cdot)$, usually tanh. Then, the output gate σ_o selects the overall fraction of the state to be returned as output.

2.3 XGBoost regression

Gradient boosting is an ensemble technique which creates a prediction model by aggregating the predictions of weak prediction models, typically decision trees. With boosting methods, weak predictors are added to the collection sequentially with each one attempting to improve upon the entire ensemble's performance.

In the XGBoost implementation [5], given a dataset with n training examples consisting of an input $\mathbf{x_i}$ and expected output y_i, a tree ensemble model $\phi(\mathbf{x_i})$ is defined as the sum of K regression trees $f_k(\mathbf{x_i})$:

$$\hat{y_i} = \phi(\mathbf{x_i}) = \sum_{k=1}^{K} f_k(\mathbf{x_i}). \tag{1}$$

To evaluate the performance of a given model, we choose a loss function $l(\hat{y_i}, y_i)$ to measure the error between the predicted value and the target value, and optionally add a regularization term $\Omega(f_k)$ to penalize overly complex trees:

$$L(\phi) = \sum_{i}^{n} l(\hat{y_i}, y_i) + \sum_{k}^{K} (\Omega(f_k)). \tag{2}$$

The algorithm minimizes $L(\phi)$ by iteratively introducing each f_k. Assume that the ensemble currently contains K trees. We add a new tree f_{K+1} that minimizes

$$\sum_{i}^{n} l(\hat{y_i}, y_i + f_{K+1}(\mathbf{x_i})) + \Omega(f_k). \tag{3}$$

In other words, the tree that most improves the current model as determined by L are greedily added. We train the new tree using the objective function (2); this is done in practice by approximating the objective function using the first and second order gradients of the loss function $l(\hat{y_i}, y_i)$ [7].

3 Models

We built the LSTM model using Keras 2.2.5 on TensorFlow 2 backend running on aconda distribution Python 3.7. The model consists of three layers: the input layer, a hidden layer of 50 LSTM neurons, and the 1 neuron output layer. The model's internal parameters are optimized for MAE loss function through the Adam optimizer. In the training stage of the model, the data was fed in batch size 72, and the training was done for 300 epochs.

The second model was built using optimized gradient boosting XGBoost library in python with 100 numbers of estimators, 3 max depth and 0.1 learning rate.

In order to ensure the replicability of the experiment, the 2001 EUNITE competition dataset [6] is used in this paper. This benchmark dataset is well-studied in Energy forecasting research [4, 11].

The EUNITE dataset spans over two years from January 1997 until January 1999. It contains the following fields: the half-hourly electricity load, the daily average temperature, and a flag signifying whether the day is a holiday. In the statistical analysis of the dataset [4, 11], it was found that the electricity load generally decreases during holidays and weekends. This phenomenon depends on the type of the holiday, e.g., Christmas or New Year.

To test the effect of using only external features in reducing error propagation for long term forecasting, the two year data was divided into two datasets: the first was used for a multivariate time series model consisting the following features: the previous half-hourly electricity load, the daily temperature, time of day, month, day of the week, and whether the day is a holiday. The second was used for a multivariate time series with the same features but without considering the previous half-hourly electricity load as a feature.

Both datasets were converted into input-output pairs for supervised learning using the python package *scikit-learn*. The package was used to encode the categorical features to numerical values and to normalize all features so that their values lie in the interval $[0, 1]$. The two models were trained and tested on each dataset with an 80/20 training/testing split.

When the two models, LSTM and XGBoost, were trained on the first multivariate time series dataset, they were able to forecast only one step ahead, since the previous half-hourly electricity load is needed in every step. For predicting the first value of electricity consumption in January 1999, the last known half-hourly electricity load on December 31, 1998, was used as an input value for the *previous load reading* feature. Then the predicted consumption value was used as input value for that feature to predict the energy consumption in the next step. This process was repeated to the end of January 1999. As a result of using the predicted value as an input in every step, the forecasting error propagated and accumulated throughout the predicted interval.

To reduce error propagation and accumulation in long term forecasting, both models were trained on the second dataset which does not contain the *previous load reading* feature. The models were used for multi-step forecasting for the month of January 1999. The predication error was only due to the model error for every step and was not a result of error accumulation.

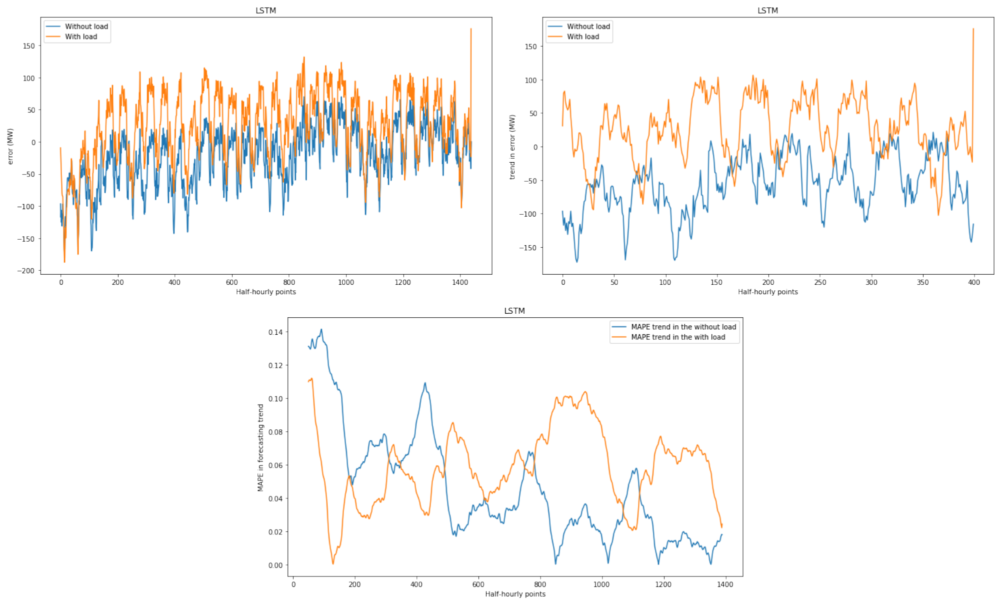

Figure 1: (a) January 1999 forecasting error for the LSTM model using the two datasets. The orange line depicts the first model that was recursively uses the past predicted values for forecasting the next steps of energy consumption, while the blue line depicts the second model which does not depend on past values of energy consumption in forecasting. (b) Forecasting error trend with averaging window of 100 points. (c) MAPE trend with averaging window of 100 points.

In addition to calculating the errors for both models, we also calculated the mean absolute percentage error (MAPE) which is defined as the follows:

$$\text{Error} = (y_i - \hat{y}_i), \tag{4}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100, \tag{5}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and N is the number of fitted points.

4 Results

Figure 1 (a) shows January 1999 forecasting error for the LSTM model with the two datasets. The orange line shows the error for the first dataset that recursively used the past predicted values for forecasting the next steps of energy consumption, while the blue line used the second dataset which did not use the past values of energy consumption. Figure 1 (b) shows the forecasting error trend for an averaging window of 100 points. Figure 1 (c) shows the MAPE plot for an averaging window of 100 points. It is clear from the figure that using first datatset which includes recursively predicted values (orange) yields higher accuracy, i.e., lower MAPE. However, around a time horizon of 500 timesteps this trend reverses with the blue line showing higher accuracy. This could be because of irregularity in the data.

LSTM is useful for short term time series forecasting, and the model has the ability to pickup the short term trends from the historical data [8, 16]. This is reflected in high accuracy in forecasting short term after the training period. However, using the predicted consumption value as the input of the previous half-hourly electricity load feature in every next step leads to the propagation and the accumulation of forecasting error. Therefore, the error for long term forecasting is higher than short term forecasting. Using the second data set for training LSTM ensures removing the propagation and the accumulation of forecasting error. Also, it seems that the high correlation between the daily peak load and the daily average temperature for the two-year historical data [4, 11] could explain that even with the absence of the previous half-hourly electricity load feature in the second data set, the accuracy is not decreasing so far than the first dataset since the daily average temperature feature still used in the second dataset.

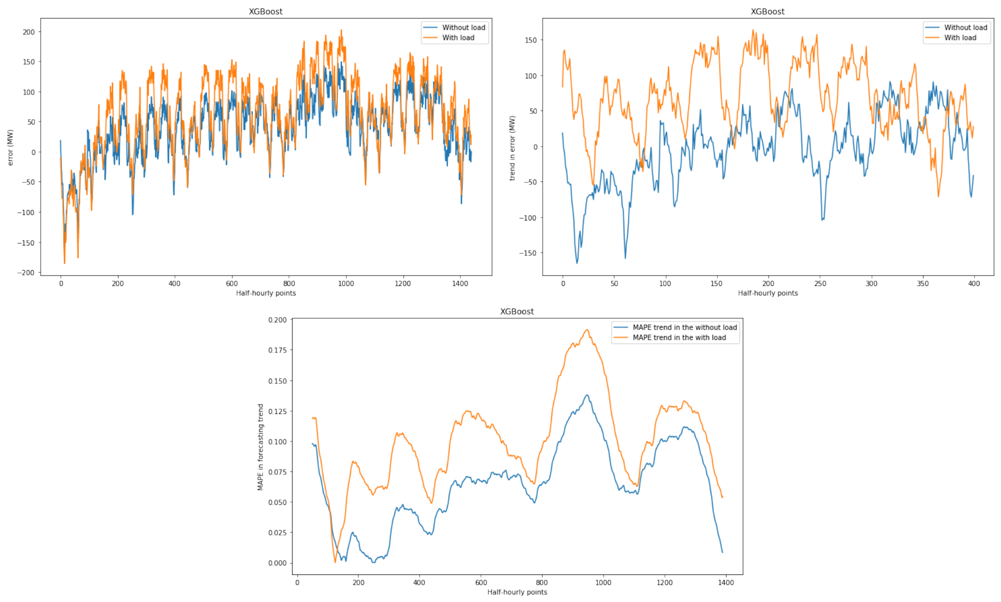

Figure 2: (a) January 1999 forecasting error for the XGboost model using the two datasets. The orange line depicts the first dataset that was recursively fed by past predicted values for forecasting the next steps of energy consumption, while the blue series uses the second dataset which does not depend on past values of energy consumption in forecasting. (b) Forecasting error trend with averaging window of 100 points. (c) MAPE trend in forecasting with averaging window of 100 points.

Figures 2 (a) and (b) show January 1999 forecasting error and its forecasting error trend, respectively, using the XGBoost model. We used an averaging window of 100 points for the XGBoost model with the two datasets. The orange plot uses the first dataset, while the blue series uses the second dataset. Figure 2 (c) shows the MAPE trend for an averaging window of 100 points. It is clear from the figure that the MAPE of the orange plot which includes the predicted values remains slightly larger than the MAPE of the blue plot, which does not include predicted values for most of the time. For XGBoost model, the algorithm learning mechanism was able to pick up more long term patterns in the historical data than short term patterns. This is reflected in higher accuracy for using the second dataset.

5 Conclusion

As machine learning prediction using historical data has been widely applied to our daily life, reducing errors of the prediction results has become paramount for the design of the algorithms. In this paper, aiming to preserve the flexibility of recursive forecasting while mitigating error accumulation over long time horizons, we investigate the approach whereby external features are generated and models are trained on these generated features. The ides is to convert an univariate model into a multivariate model without the normal difficulty of obtaining a multivariate dataset. Results from the experiments show that the external features anchor predictions and limit the amount of accumulated error.

Using the LSTM model as an example, it is found the accuracy for short term energy forecasting with using a past energy consumption value as feature is higher than the accuracy when not using past values as a feature. the opposite behavior takes place for the long term energy forecasting. For XGBoost model the accuracy for both short and long term energy forecasting is higher when not using past values as a feature.

The idea could be applied to other fields where time series forecasting is used. As future work, other datasets such as stock market forecasting will be evaluated. We will also consider the effects of composition among the input variables of machine learning prediction.

6 Acknowledgment

Support from the Natural Sciences and Engineering Research Council of Canada (NSERC) is greatly acknowledged.

References

[1] Kadir Amasyali and Nora M El-Gohary. A review of data-driven building energy consumption prediction studies. *Renewable and Sustainable Energy Reviews*, 81:1192–1205, 2018.

[2] Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of recurrent neural networks for short term load forecasting. *arXiv preprint arXiv:1705.04378*, 2017.

[3] C. Chatfield. *Time-Series Forecasting.* CRC Press, 2000.

[4] Bo-Juen Chen, Ming-Wei Chang, and Chih-Jen lin. Load forecasting using support vector machines: A study on eunite competition 2001. *IEEE transactions on power systems*, 19(4):1821–1830, 2004.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[6] EUNITE. Eunite electricity load forecast 2001 competition. *Proceedings of EUNITE*, Dec. 2001 2001.

[7] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[8] John Cristian Borges Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[9] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[10] Katarina Grolinger, Alexandra L'Heureux, Miriam AM Capretz, and Luke Seewald. Energy forecasting for event venues: Big data and prediction accuracy. *Energy and Buildings*, 112:222–233, 2016.

[11] Jawad Nagi, Keem Siah Yap, Farrukh Nagi, Sieh Kiong Tiong, and Syed Khaleel Ahmed. A computational intelligence scheme for the prediction of the daily peak load. *Applied Soft Computing*, 11(8):4773–4788, 2011.

[12] Apurva Narayan and Keith W Hipel. Long short term memory networks for short-term electric load forecasting. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1050–1059, Banff Center, Banff, Canada, October 5-8 2017.

[13] Christopher Olah. Understanding lstm networks. *GITHUB blog, posted on August*, 27:2015, 2015.

[14] Souhaib Ben Taieb, Gianluca Bontempi, Amir Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition, 2011.

[15] Kaile Zhou, Chao Fu, and Shanlin Yang. Big data driven smart energy management: From big data to big insights. *Renewable and Sustainable Energy Reviews*, 56:215–225, 2016.

[16] Lingxue Zhu and Nikolay Laptev. Deep and confident prediction for time series at uber. *arXiv preprint arXiv:1709.01907*, 2017.