Analysis of Recurrent Neural Networks for Short-Term Energy Load Forecasting

Luca Di Persio[a)] and Oleksandr Honchar[b)]

[a,b)] *Dept. of Computer Science, University of Verona and HPA-High Performance Analytics*

[a)]luca.dipersio@univr.it
[b)]oleksandr.honchar@univr.it

Abstract. Short-term forecasts have recently gained an increasing attention because of the rise of competitive electricity markets. In fact, short-terms forecast of possible future loads turn out to be fundamental to build efficient energy management strategies as well as to avoid energy wastage. Such type of challenges are difficult to tackle both from a theoretical and applied point of view. Latter tasks require sophisticated methods to manage multidimensional time series related to stochastic phenomena which are often highly interconnected. In the present work we first review novel approaches to energy load forecasting based on recurrent neural network, focusing our attention on long/short term memory architectures (LSTMs). Such type of artificial neural networks have been widely applied to problems dealing with sequential data such it happens, e.g., in socio-economics settings, for text recognition purposes, concerning video signals, etc., always showing their effectiveness to model complex temporal data. Moreover, we consider different novel variations of basic LSTMs, such as sequence-to-sequence approach and bidirectional LSTMs, aiming at providing effective models for energy load data. Last but not least, we test all the described algorithms on real energy load data showing not only that deep recurrent networks can be successfully applied to energy load forecasting, but also that this approach can be extended to other problems based on time series prediction.

INTRODUCTION

The general problem of how efficiently exploit time series to forecast trends of specific quantities is one of the key issue of the on going research in applied mathematics. In particular, the latter is the basis of convergent analysis performed using approaches that belong to different mathematical areas, spanning from statistics, to stochastic processes theory, to computer science, just to mention a few. The possible scenarios of applications are large as well. Precise forecasts are ever much more required in finance, to meteo previsions aims, concerning the future behavior of social networks users, the latter particularly from a commercial point of view, etc. From a classic point of view, the use of techniques such as, e.g., ARIMA, GARCH as long as the implementation of various type of smoothing filters, such as the Kalman-type, are all well known as rather powerful tools for forecasting aims. Nevertheless, such approaches are often unable to deal with non-linear and/or non-seasonal patterns, being also very sensitive to data outliers. This is one of the main reasons why, during last decades and also thanks to the increase of the computational powers of modern computers, machine learning algorithms have been started to be used more and more to obtain real value - regression, hence extending the traditional approach of polynomial regression to fit a curve given set of parameters. Such recent techniques, mostly differ between each other from the point of view of implemented parameters sturcture, e.g., the decision trees regressors, the cost function, etc. Between the latter, the artificial networks type techniques allow to efficiently solve regression problems, typically by mean of several hierarchical levels. However, related time series cannot be processed at every time step with simple neural networks, then saving some hidden state of the sequence. Therefore, a better choice to handle temporal data is represented by recurrent neural networks [5, 6]. They are still *connectionist type models*, but they pass input data inside the network across time steps, hence processing one element at a time and changing temporal state of representation after processing every next time step. In this work we show how popular lately LSTM can perform on energy load forecasting. We describe data preprocessing and a simple way to inject information about seasonality to the neural network as another dimension. We compare performance of multilayer LSTMs, bidirectional LSTMs and decoder-encoder architectures, also showing further ways to improve obtained results.

DATA PREPARATION

In figure 1, we have reported a time series sample from the dataset provided by a leading company acting in the Italian energy market scenario. Is it easy to recognize how such data exhibit periodic patterns and stationarity. Such features also suggest an easy way to analyze the data, aiming at developing related forecasts.

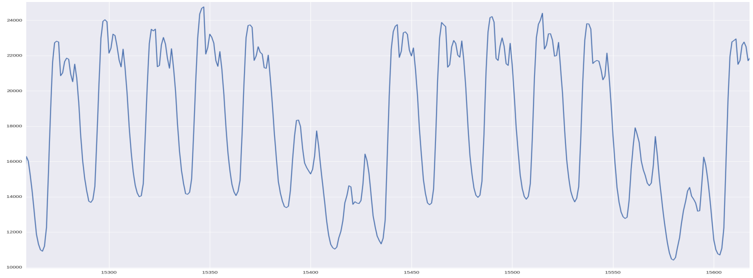

FIGURE 1. Sample data points from energy load dataset

A typical check for the latter hypotheses consists in calculating the related Hurst exponent H, namely a scalar value that helps to identify, within the limits of statistical estimation, if a given time series has the mean reverting poperty, if it behaves as a *random walk* and if it is characterized by a trend component. In particular, we have: if $H < 0.5$, then the time series is mean reverting ; if $H = 0.5$, then the time series represents a discretizaion of *geometric Brownian motion*, if $H > 0.5$, then the time series has a trend component. Analyzing our dataset, we derive a Hurt exponent equal to 0.13. Latter value indicates that we are dealing with a stationary time series. Therefore we can normalize data using statistical parameters from dataset, namely mean, standard deviation, resp. minimal or maximal values, in order to increase the performance of machine learning models we want to implement. In order to normalize our data, we chose a $z - score$-type normalization, hence we subtract the mean of test or train dataset, and then we divide by its standard deviation. Then, we split our dataset in *training part*, resp. in *test part*, taking 90% of first historical values as train set. Therefore, we check the reached performance on the last 10%. In particular we have decided to avoid to have mixing between values belonging to the past, resp. to the future. Every training example consists of a pair (X_i, y_i), where X_i is a time window of length 48 (for two days of measurements), while y_i is a value in the next hour. Our training set consists of 35456 samples, while the test set has 12291 samples. To take into account the seasonality feature of data, we first encode them exploiting concrete values, namely day of the year, day of the week and hour of a day, and then we use such values as a spatial dimension on every time step. In order to normalize each dimension, we subtract the relative maximum value, namely 365 days for the whole year, 7 days for the weeks, 24 hours concerning the *daily dimension*. As an example of the normalization performed, we can consider a vector of every time step, such as $(-1.34, 0.08, 0.14, 0.5)$, where first dimension is the normalized load, the second is decoded as the 30th of January, the third is Monday and the last one is 12 AM. Following such an approach, we have that every input to the neural network is composed by 48 historical vectors, constituting a tensor of dimensions (48, 4, 1).

LSTM NEURAL NETWORK

The training phase of a recurrent neural network is based on pairs (x_t, y_t), where x_t is an input sequence, while y_t could be a single output, or even a sequence. Sequence modeling is obtained considering, at every time step, some hidden state. The latter allows the RNN to *remember* the current state of a sequence, or its context and processes it forward to future values with previous time steps. Moreover, corresponding to every new input x_t, a new hidden state, let us indicate it with h_t, is added according to h_{t-1}. To every couple of inputs x_t and h_{t-1}, have been associated three weight matrices, namely W_{hx} indicating the weights from input to hidden layer, W_{hh} from hidden to hidden, and W_{yh} for the output's weights. The resulting basic equations for RNN read as follows:

$$s_t = tanh(W_{hx}x_t + W_{hh}s_{t-1} + b_t) \quad , \qquad o_t = softmax(W_{yh}s_t) \; .$$

Basic RNNs perform good on modeling short sequences, up to 10 time steps, while longer ones cause problems of vanishing or exploding gradients. In order to overcome such type of issues dealing with long sequences, an interesting approach for long-short term memory has been developed by Schmidhuber [1]. Comparing to RNNs, LSTMs single time step cell has a more developed hidden structure. Inside these cells, often called memory blocks, instead of single input and a single hidden state, there are three adaptive and multiplicative gating units, which are called the *input gate*, the *forget gate* and the *output gate*, on the basis of the information they have to learn about current state of a given sequence. The new development, forget gate, plays the role of determining, or to keep, or to forget the information about previous time steps. This latter feature allows to catch more complex temporal patterns, not only between time steps t and $t - 1$, but also between t and $t - n$, where n can be on a long distance from t. Accordingly, the forward propagation equations, characterizing the LSTM gates, reads as follow: $i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$, $c\_in_t = tanh(W_c x_t + U_c h_{t-1} + b_{c\_in})$, $f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$, $o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$, and for the forget state update, we have: $c_t = f_t \cdot c_{t-1} + i_t \cdot c\_in_t$, and $h_t = o_t \cdot tanh(c_t)$, where x_t represents the input to the memory cell, while $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ are the weight matrices, and b_i, b_f, b_c, b_o are biases.

ARCHITECTURES COMPARISON

In this section we compare different architectures of deep recurrent neural networks and their special characteristics.

Multilayer LSTM

Multilayer recurrent neural networks are built in the same way as general feed-forward multilayer networks. In particular, after processing a sequence, every memory block has its own output y_t. The latter gives us a new sequence constituting the first-level representation of the original one. The, we pass this sequence to the next LSTM layer that is trained the same way. Last recurrent layer has return which is not constitute by the whole sequence. In fact, only last hidden state is passed to affine layers of neural network, linear classifier or other classification algorithm.

Bidirectional LSTM

Even if standard LSTM are able to handle long-term dependencies, they cannot use future input information coming from the current state. On the contrary, Biderectional recurrent neural networks (BRRNs), see, e.g., [2], do not require their input data to be fixed, moreover future input information is reachable from the current state. The basic idea behind BRNNs type architectures is to connect two hidden layers of opposite directions to the same output. Exploiting this structure, the output layer can get information from both the past and the future states. It is worth to mention that BRNN are especially useful when the context of the input is needed, while handling seasonal time series it is important to observe the full cycle, providing advantages over standard LSTM.

Encoder-decoder LSTM

The idea of encoder-decoder RNNs, also called *sequence-to-sequence*, see, e.g., [3], is to use one LSTM, or even several layers, to read the input sequence, as well as to obtain fixed-dimensional vector representation, in order to later use another LSTM, or a set of layers, to extract the output sequence from that vector. This approach directly exploits the idea of distributed representation learning, assuming that it is possible to learn a representation of a time window with one architecture and then forecast with a second one, possibly totally different, resulting in a so-called *feed-forward* or *convolutional network*.

EXPERIMENTAL RESULTS

In what follows we provide the results of computation performed with networks that have been all trained during 100 epochs with batch size equal to 128. We have also ussed a rather standard optimization algorithm, namely the *Adam* one, see [**?**], with scheduled learning rate, starting from 0.02 and decresing by factor 0.9 after every 10 epochs, but without decreasing the loss function (MSE). In Figure 2, we have reported results obtained exploiting the MAE and MAPE approaches on de-normalized data. As we can see, the architecture characerized by bidirectional structure performs better than the regular one. Moreover idea with encoding and decoding works well. Nevertheless, we obtain lower performances if compared with bidirectional LSTM. In Figure 3, we have shown a sample prediction visualization from test dataset. We would like to underline how it looks very accurate as well, being the black line the one determined by data points, while the blue one constitutes the neural network prediction.

	Mean average error	Mean average percentage error (%)
LSTM	324.88	1.88
Bidirectional LSTM	302.48	1.76
Encoder-decoder LSTM	336.16	1.97

FIGURE 2. MAE and MAPE after training different architectures

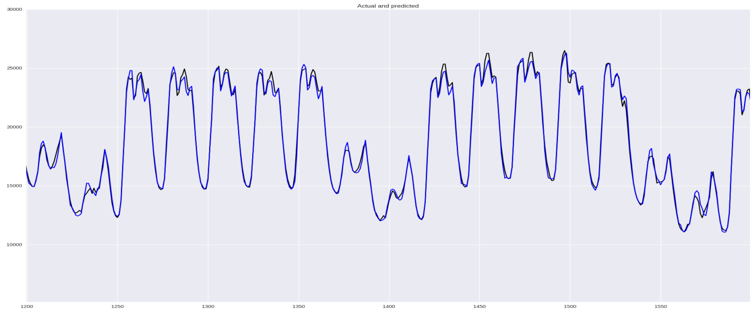

FIGURE 3. Results of bidirectional LSTM forecasts on test data

CONCLUSIONS

In this work we have considered recurrent neural networks, focusing our attention on the implementation of different architectures of long-short term memory networks to energy load forecasting. We have shown how to pre-process data in order to take into consideration the *seasonal information* as well as suitably normalize data. Moreover we have recalled the basic principles of deep LSTMs, bidirectional LSTMs and encoder-decoder framework. Our analysis indicates that bidirectional LSTMs provide the best performance on particular type of time series. We would also like to underline that obtained results can be improved by increasing the depth of the architecture considered, namely adding more recurrent or affine layers, as well as adding smoothing procedures to avoid possible overfitting.

REFERENCES

[1] S.Hochreiter, J.Schmidhuber, *Bidirectional recurrent neural networks*, Signal Processing, IEEE Transactions,1997

[2] Schuster, Mike, and Kuldip K. Paliwal, *Bidirectional recurrent neural networks*, Signal Processing, IEEE Transactions, 1997.

[3] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, *Sequence to Sequence Learning with Neural Networks*, NIPS, 2014.

[4] D. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*, *arXiv:1412.6980.*

[5] L. Di Persio, O. Honchar, *Recurrent neural networks approach to the financial forecast of Google assets*, International journal of Mathematics and Computers in simulation, Vol. 11, 2017.

[6] L. Di Persio, O. Honchar, *Artificial neural networks architectures for stock price prediction: Comparisons and applications*, International Journal of Circuits, Systems and Signal Processing, 10, pp. 403-413, 2016.