KU Leuven

Smart Distribution Systems
B-KUL-H00P3A

Forecasting Competition: Report

Authors:
Alberte Bouso Garcia
Román Cantú Rodriguez

April 23, 2020

Contents

1 Introduction **2**

2 Methodology **2**

2.1 Data preparation 2

2.2 Training procedure 3

2.3 Network architecture 4

3 Results **5**

4 Conclusion **6**

1 Introduction

Day-ahead electricity prices are calculated in the electricity market depending on the bids of each generator and consumer, therefore, the prices are volatile and depend on many factors. Having knowledge on these electricity prices as soon as possible offers the players (e.g. consumers, retailers and supply companies) the capability of designing bidding strategies that would improve their benefits. Hence, accurate forecast methods for the day-ahead electricity prices are key for market players.

Historically, fundamental models are used for predicting this market data. For instance, by depicting demand and supply as accurately as possible. These models require a lot of information to be processed, consuming a lot of resources and time. In recent years, econometric time series models and machine learning methods have been developed. These approaches require less information and, additionally, this can be easily extracted from historical data series [1].

This assignment deals with the design, tune and performance evaluation of an Artificial Neural Network (ANN) model which aims to predict the hourly price of electricity based on historical data. ANNs is a powerful machine learning method which can learn from data, i.e. it can be trained to recognise patterns, classify data and forecast future events [2]. In this way, an ANN can detect and extract correlations in historical data-series autonomously.

The implementation of ANN for price forecast is not new in the literature, as several examples can be found. In [3] the author forecasts the price of the next hour or [4] where the study focused on predicting the average peak price of the next day. Also, a vector forecast of electricity prices from a specific period (in this case, the day ahead) is performed in [5].

2 Methodology

Modelling with ANNs requires an adequate selection and preparation of the data, i.e. which data is going to be used as an input to predict the hourly prices. Then, an appropriate architecture is defined and evaluated, with the aim of test how well-fitting price forecast is achieved.

2.1 Data preparation

The data used for **training** the ANN comes from three different sources: historical hourly price, historical wind and solar production and Day-ahead predictions. The last two data groups are presented on a 15-minute basis while spot prices are extracted from Belgium market exchange operator, 'Belpex', which actuates on an hourly basis. All of them include the data for two full years, 2016 and 2017. Note that wind and solar data include 'NotANumber' values, as well as prices, which will be taken into account.

By inspecting the data on Excel, one considers the following:

- *DayAheadForecast* is acceptable for both wind and solar as predictions of the real production on the same day, observed by normalising this value against the normalised value of *LoadFactor*.
- The test data (i.e. the 3 given random weeks) includes *DayAheadForecast* for the day after the week, which is considered as valuable information.

As a first step, all training data is imported in the script as a data frame, a data structure from the Python package `pandas`. This allows creating a unique variable with labelled rows and columns which is easy to handle. Hence, one imports 3 CSV, selecting only the columns *price* from Belpex and *DayAheadForecast* from wind and solar dataset.

To create a unique granularity, i.e.price data is resampled in a 15 minutes basis, taking the same price for each of the 4-time steps in the hour. Additionally, it was detected that prices can be missing or doubled due to the seasonal clock change. This is solved by assuming the price is the same as the previous hour. Pursuing error prevention, wind and energy forecast NaN are filed with previous valid step data. Finally, to prevent outliers, samples that are far from the median of the whole data, are replaced by

$$D_{new} = \mu \pm n_{treshold} \cdot \sigma \tag{1}$$

with $n_{treshold} = 5$.

After inspecting autocorrelation in the electricity price, one can find a weekday correlation: Mondays tend to look similar to other Mondays, Tuesdays to Tuesdays and so on. Hence, it was decided to reshape the training data such as one week of historical data targets "8th-day" prices, e.g. week from 1st of January to 7th of January will be used as input and the 8th of January prices used as an output, then the week from 2nd to 8th of January will target 9th of January and so on. In addition, as mentioned before, wind and solar energy *DayAheadForecast* are also considered in the model. This is done by appending to the input an array with wind and solar predictions for the "8th day".

In summary, input data will be composed of 723 rows with 864 columns each one. The last number is due to adding seven days hourly prices in a 15-minutes basis, 672 measures, with "8th-day" predictions of solar and wind, 96 measures each. 723 is the total number of samples one can take from two years since 730 days can only form 723 groups of 7 consecutive days. On the other hand, output data will be composed only by the daily prices of the "8th-day" in a 15-minutes basis, i.e. its shape will be (723, 96). At this point, one is ready to train its Neural Network.

2.2 Training procedure

When it comes to training/test division, one always have overfitting as the phenomena to avoid: the model would have a perfect score when repeating the samples used to train but failing to predict anything useful on unseen data. A common practice in supervised learning is the use of cross-validation. Its basic approach is the so-called k-fold cross-validation. The function `KFold` from the package `sklearn` implements this function, by allowing one to train and test the model k-times on different subsets of training data and build up an estimate of the performance on unseen data.

For this assignment, it was decided to use 10 folds (so-called 10-fold validation) so the model trains on 90% and test on 10% from the total training set (years 2016 and 2017). This allows one to compare the performance across folds, and evaluate the averages of the folds. Since the average error in the different models evaluated was similar for the 10 folds, it was concluded that the data was not sensitive regarding the chosen split. Hence, it is valid to finalize the model using the full data to train the model, as suggested in [6].

On the other hand, for the test, it is decided to use data from three random weeks of 2019. Belpex price is obtained from [7] whereas wind and solar forecasts from [8]. This allows to test the performance of the previously trained model on unseen information and evaluate a good generalization and, consequently, low overfitting. A summary of the training process can be visualized in the workflow presented in Figure 1.

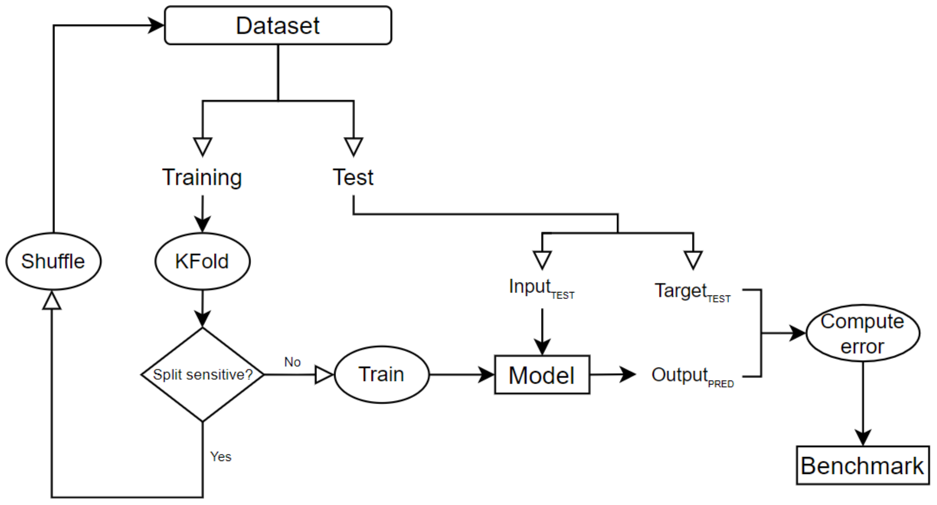


Figure 1: Workflow of the algorithm

2.3 Network architecture

The ANN model is built using Python package `Keras`, a high-level neural networks API capable of running on top of `TensorFlow`. Sequential API is used, as it allows an easy implementation of stacked layers one after the other and widely used.

As some authors suggest ([9], [10]) one hidden layer is sufficient for the large majority of problems as well as it keeps the training fast. Nevertheless, since the train of the ANN is considerably fast, two hidden layers are used as a configuration. Results show better performance with the use of two hidden layers, which lead to opt for this structure. Hence, the model will have four layers: one input, one output and two hidden.

Input and output layer sizes will be automatically defined by training data and its targets, i.e. 864 and 96, respectively. For the last one's activation function, linear was chosen. In this way, the values are not bounded.

Regarding hidden layers, an initial guess about the number of neurons was given by one of the rules of thumb presented in [11]: use a size between the size of the input and output layers. After several trials, a configuration of 168 neurons for the first and 96 for the second brought the best results in terms of the benchmark detailed in the previous section. Additionally, activation function `ReLu` was the election of choice, as it is the most popular selection worldwide according to [12].

`RMSprop` was the selected optimizer for the model. RMSprop is a fast and very popular optimizer, widely used in Deep Learning as shown in [13]. When it comes to solving each unit, this solver keeps its mean gradient feedback through the learning and for each step given, the gradient value is normalized by this mean, i.e. it stabilizes the learning. All parameters were set to default, including `lr` (learning rate).

Initially, 500 epochs will be used for the K-Fold validation. It allows a fast computation of the problem and it is not computational expensive to run several configurations with this amount of iterations. Nevertheless, for the final architecture i.e. the one presented in the document, a run of 250 epochs was enough to achieve an accurate model, after observing overfitting regarding the test set with a high number of iterations.

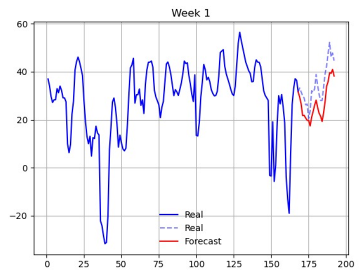
3 Results

The output from the ANN model has a shape (96, 1). Nevertheless, Belgium market operator does not work in quarter-hourly-basis, which means a resample is needed. This is done by averaging the 4 values of each hour. Note that for most of the architectures tested, a quasi-horizontal line was observed for each hour, implying good learning from the model.

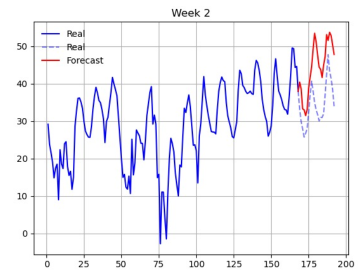
As mentioned earlier, the model is tested by running it over three random weeks of 2019. It is worth mention that these three weeks are not the same as the given ones in the assignment, which aim to evaluate the performance of the forecast. The result of the 2019 weeks is shown in Figure 2.

The root mean squared error (rmse) for the test set, i.e. 3 random weeks from 2019, was 10.5. On the other hand, rmse over the training set, i.e. years 2016 and 2017, was 10.7. Under the experiments, it was possible to reduce the rmse over the training set to very small numbers, at the expense of a poorer performance over the test. Hence, it is decided to choose an ANN architecture and configuration such as both errors are similar, which ensures an acceptable learning and avoids overfitting. Table 1 summarizes all the tuning parameters.

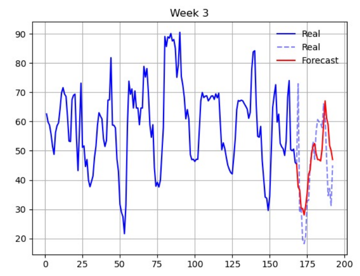
Code is available at the GitHub repository [14].



(a) First week



(b) Second week



(c) Third week

Figure 2: Performance under unknown data

Parameter	Value	Parameter	Value
Hidden layers	2	*Epochs*	250
Neurons hidden layer 1	168	*Activation function*	`ReLu`
Neurons hidden layer 2	92	*Activation function*	`ReLu`
Neurons input layer	216	*Activation function*	-
Neurons output layer	96	*Activation function*	`linear`
Optimizer	`RMSprop`	*Learning rate*	0.001

Table 1: Neural Network architecture and configuration

4 Conclusion

This document details the process of designing, tuning and testing an ANN which attempts to forecast daily prices in the Belgium spot market. The number of layers, neurons or epochs, are some of the parameters needed to be tuned during the process. Nevertheless, good and justified criteria for data treatment and preparation has equal relevance in the performance of the model.

Besides achieving acceptable results with the model in terms of accuracy, overfitting and generalization, there is still room for improvement. One possible upgrade while using `Sequence` package, is the implementation of a grid search methodology to find the accurate parameters for optimal tuning of the ANN. Another idea is to test a different solver, such as Adam, which has gained popularity in the last years among the machine learning community. On the other hand, the ANN can achieve better results if more data of interest is added, e.g. load consumption estimations.

Additionally, one could use advanced models as Recurrent Neural Networks (RNN), which have shown good results in time-series prediction. Specifically, Long-Short Term Memory is a kind of RNN which overcome problems related to memory management that vanilla RNN faces.

References

[1] D. Keles, J. Scelle, F. Paraschiv, and W. Fichtner, “Extended forecast methods for day-ahead electricity spot prices applying artificial neural networks,” *Applied Energy*, vol. 162, pp. 218 – 230, 2016.

[2] MATLAB, “What is a neural network?.” Available at https://www.mathworks.com/discovery/neural-network.html.

[3] A. M. Gonzalez, A. M. S. Roque, and J. Garcia-Gonzalez, “Modeling and forecasting electricity prices with input/output hidden markov models,” *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 13–24, 2005.

[4] H.-T. Pao, “A neural network approach to m-daily-ahead electricity price prediction,” in *Advances in Neural Networks - ISNN 2006* (J. Wang, Z. Yi, J. M. Zurada, B.-L. Lu, and H. Yin, eds.), (Berlin, Heidelberg), pp. 1284–1289, Springer Berlin Heidelberg, 2006.

[5] H. Yamin, S. Shahidehpour, and Z. Li, “Adaptive short-term electricity price forecasting using artificial neural networks in the restructured power markets,” *International Journal of Electrical Power & Energy Systems*, vol. 26, no. 8, pp. 571 – 581, 2004.

[6] Chris, “How to use k-fold cross validation with keras?,” Feb 2020. Available at https://www.machinecurve.com/index.php/2020/02/18/how-to-use-k-fold-cross-validation-with-keras/.

[7] “Spot belpex,” Feb 2020. Available at https://my.elexys.be/MarketInformation/SpotBelpex.aspx.

[8] “Grid data: Power generation,” Feb 2020. Available at https://www.elia.be/en/grid-data/power-generation.

[9] J. de Villiers and E. Barnard, “Backpropagation neural nets with one and two hidden layers,” *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 136–141, 1993.

[10] D. Stathakis, “How many hidden layers and nodes?,” *International Journal of Remote Sensing*, vol. 30, no. 8, pp. 2133–2147, 2009.

[11] J. Heaton, *Introduction to Neural Networks for Java, 2nd Edition*. Heaton Research, Inc., 2nd ed., 2008.

[12] S. Sharma, “Activation functions in neural networks,” Feb 2019. Available at https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[13] A. Karpathy, “A peek at trends in machine learning,” Apr 2017. Available at https://medium.com/@karpathy/a-peek-at-trends-in-machine-learning-ab8a1085a106.

[14] A. Bouso and R. Cantú, “Forecasting competition,” 2020. Available at https://github.com/albertebouso/forecasting_SDS.