Report on forecasting systems using Deep Learning and Recurrent Neural Network techniques

By Ginevra Iorio

Abstract

The goal of this report is to give an introduction on Deep Learning and an overview on its main methods, such as Recurrent Neural Networks (RNNs), that can be used to build Time Series Forecasting systems.

Introduction

Deep Learning (DL) is the Artificial Intelligence's discipline in which machines, powered by AI software and hardware, try to learn from experience. These machines are essentially Neural Networks that are able to create multiple layers of abstraction to represent the data they process. Accordingly, compared to traditional Machine Learning algorithms, many of which have a limited ability to learn regardless of the amount of data they collect, the more the data to be accessed, the more DL systems are able to improve their performance. There exist different Deep Learning methods that are very useful to automatically learn the temporal dependencies structures which are used in time series forecasting problems.

Time Series Forecasting

Applying Deep Learning for Time Series Forecasting involves making scientific predictions based on historical time stamped data and entails creating models through historical studies, using them to draw conclusions and guide strategic decision-making in the future.

When approaching a time series forecasting problem, it is necessary to consider that not everything can be forecast, the predictability of an event depends on several factors:

- how well we understand the factors that contribute to it;
- how much data is available;
- whether the forecasts can affect the thing we are trying to forecast.

It is also required to understand what to forecast, the forecasting horizon and how frequently the forecasts are required in order to find and collect the appropriate data on which the forecasts will be based [1]. A time series is anything that will be observed sequentially, and it can be either at regular time intervals or irregularly spaced.

Dealing with sometimes unpredictable or unknown data, time series forecasting isn't infallible and can't be used in all situations. What is important in order to deal with good forecasting is to have a clean and time stamped set of data and to be able to identify trends and patterns in it. This means that more data is at hand, the better the data scientist is able to understand it [2].

Deep Learning Architectures for Time Series

Dealing with time series, entities representing a logical grouping of temporal information are used. There are different Deep Learning methods that, with the aid of Neural Networks, are used to predict future values for time series forecasting. Being the heart of Deep Learning algorithms, Neural Networks are used to enable computer programs to identify patterns and resolve common issues by mimicking the behavior of the human brain. In fact, their name and structure are inspired by it. Training data is essential for Neural Networks to develop and enhance their accuracy over time [3].

Artificial Neural Networks (ANNs) are the simpler existing, being comprised of a node layer, containing an input layer, one or more hidden layers, and an output layer. Moreover, each node, or artificial neuron, is connected to the others and has a weight and threshold that, if exceeded by any node's output, is activated, and starts providing data to the network's uppermost layer [3].

Distinct types of neural networks, each employed for a different purpose and needed for Deep Learning Techniques, can be categorized:

- Feedforward Neural Networks

  Also called Multi-Layer Perceptrons (MLPs), they approximate a mapping function from input variables to output variables. A Multilayer Perceptron has input and output layers, and one or more hidden layers with many neurons stacked together. The name of "Feedforward" Neural Networks comes from the fact that inputs are combined with the initial weights in a weighted sum and subjected to an activation function, just like in a Perceptron [4].

- Convolutional Neural Networks

This type of networks, which are traditionally designed for image datasets, are needed to extract local relationships that don't vary across spatial dimensions. They have proven to be effective in providing a component in hybrid models for problems such as object localization and image captioning. They accomplish this by operating on raw data, such as raw pixel values, rather than features that are domain-specific or manually created from the raw data. The model then, on the *representation learning* stage, learns how to extract useful features from the raw data. With respect to time series forecasting problems, a sequence of observations can be treated like a one-dimensional image that a CNN can read and process [5].

- Recurrent Neural Networks

RNNs are deep learning models that are typically used to solve problems with sequential input data. They are a type of neural network that can learn from earlier iterations during training because it keeps track of the information it has already processed, usually keeping a record of the information it processed in its most recent previous stages [6]. In RNN, a hidden state h_t can be computed for a given an input sequence $x=(x_1, x_2, \dots, x_t)$, as:

$$h_t = \begin{cases} 0 & t = 0 \\ \varphi(W_{x_t}, x_t) & \text{otherwise,} \end{cases}$$

where ϕ is a non-linear function. The recurrent hidden state is updated as follows:

$$h_t = g\left(W_{x_t} + uh_{t-1}\right),$$

where g is a hyperbolic tangent function (tanh) [7].

Recurrent Neural Network Variants

Since the late 1990s, several variants of RNNs algorithms have emerged:

- Long Short-Term Memory (LSTM)

This algorithm processes input data by looping over time steps and updating the network state, which then stores all the information retrieved from all prior steps. In order to train a LSTM network for time series forecasting, it is necessary to train a regression LSTM network with sequence output in which the responses are the training sequences with values shifted by one time step. Forecasting can be done with two different methods: *open loop*, that predicts the next time step in a sequence using only the input data, and *closed loop*, which predicts the following time steps in a sequence using the previous predictions as inputs [8].

In the LSTM, a cell is referred to as "gated." Through gates, information is added or withdrawn in a selective manner. How much incoming information is captured and how much of it is preserved is determined by the cell, which functions as a sieve. The model has the option of opening an input gate to store data, rejecting it and erasing it from long-term memory (*forget gate*), or passing the data to the subsequent layer (*output gate*). These decisions are made based on the importance weights that are assigned to the information when moving through the temporal loops and attempting to minimize the error. The LSTM adjusts the opening and closing of its gates by giving the information values higher or lower weights between 0 and 1 and, over time, it learns which information pieces are efficient in lowering the prediction error [6].

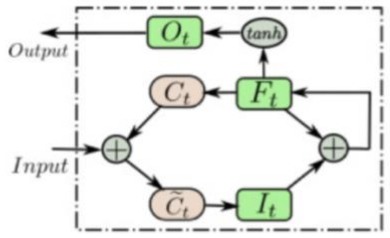

Figure 1: basic structure of LSTM. I_t, F_t, *and* O_t *represent the three LSTM gates (input, forget and output gates respectively),* C *and* $C^{\sim}$ *represent the candidate memory cells and memory cell content.*

The 3 gates of LSTM network are represented with the following equations:

$$J_t = sigmoid\left(w_J\left[h_{t-1}, k_t\right] + b_J\right)$$

$$G_t = sigmoid\left(w_G\left[h_{t-1}, k_t\right] + b_G\right)$$

$$P_t = sigmoid\left(w_P\left[h_{t-1}, k_t\right] + b_P\right)$$

Where: J_t is the function of input gate, G_t the function of the forget gate, P_t the function of the output gate, W_x the coefficients of the neurons at gate (x), H_{t-1} is the result from the previous time step, k_t the input to the current function at time-step t and b_x the bias of neurons at gate (x).

The input gate in the first equation gives the information that needs to be stored in the cell state, the second equation throws the information based on the forget gate activation output, the third one combines the information from the cell state and the output of forget gate for generating the output [9].

- Gated Recurrent Units (GRU)

They are a variation of the LTSM algorithm, being designed very similarly. They solve the vanishing gradient problem of standard RNN, which is caused by the multiplicative nature of the backpropagation algorithm, since gradients calculated at a deep stage of the RNN have too small of an impact [10]. In order to solve it, GRU uses the so-called *update gate and reset gate*. These are two vectors that decide what information should be passed to the output and they can be trained to keep information from long ago, without washing it through time [11].

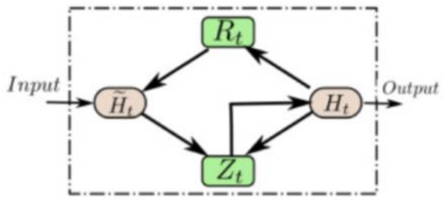

Figure 2: basic structure of GRU. R_t *and* Z_t *are reset gate and update gates respectively,* H_t *and H˜t are the candidate hidden state and hidden state respectively.*

- Vanilla RNNs

This method uses the backpropagation algorithm, and its name refers to the fact that it doesn’t contain the latest contain the latest and more advanced mathematical ingredients to retain a memory of long-term patterns. For this reason, it struggles to learn more long-term dependencies and can encounter the vanishing gradient problem [6].

Conclusions

Mimicking the function of the human brain, Deep Learning and its algorithms have many applications nowadays and in numerous different fields. Using brain simulations, they make learning algorithms much better and easier than they have ever been. There are many different techniques to be applied when it comes to using Deep Learning for training Neural Network models and, when it comes to building a forecasting system, these are highly beneficial. After having seen all their characteristics, it is possible to conclude that LSTM Recurrent Neural Networks are the most valuable option for time series. As a matter of fact, this type of model can store information for a certain period of time, which can subsequently be used to predict future outcomes more precisely.

Bibliography

[1] Deepak Yadav (2020). Approach Any Time Series Forecasting Problem. *The Startup*, accessed on September 25, 2022, https://medium.com/swlh/approach-any-time-series-forecasting-problem-8f87bd07e2a9

[2] Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, *379*(2194), 20200209, https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2020.0209

[3] IBM Cloud Education (2020). Neural Networks. *IBM*, accessed on September 26, 2022, https://www.ibm.com/cloud/learn/neural-networks

[4] Carolina Bento (2021). Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis. *Towards Data Science*, accessed on September 26, 2022, https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

[5] Jason Brownlee (2018). Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python. *Machine Learning Mastery*. https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/

[6] Heiko Onnen (2021). Temporal Loops: Intro to Recurrent Neural Networks for Time Series Forecasting in Phyton. *Towards Data Science*, accessed on September 26, 2022, https://towardsdatascience.com/temporal-loops-intro-to-recurrent-neural-networks-for-time-series-forecasting-in-python-b0398963dc1f

[7] Abdelhafid Zeroual, Fouzi Harrou, Abdelkader Dairi, Ying Sun (2020). Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study, Chaos, Solitons & Fractals, Volume 140, 110121, ISSN 0960-0779. https://doi.org/10.1016/j.chaos.2020.110121

[8] Time Series Forecasting Using Deep Learning. *MathWorks*, accessed on September 27, 2022, https://www.mathworks.com/help/deeplearning/ug/time-series-forecasting-using-deep-learning.html

[9] Vinay Kumar Reddy Chimmula, Lei Zhang (2020). Time series forecasting of COVID-19 transmission in Canada using LSTM networks. *Chaos, Solitons & Fractals*, Volume 135, 109864, ISSN 0960-0779, https://doi.org/10.1016/j.chaos.2020.109864

[10] Nic McCullum (2022). The Vanishing Gradient Problem in Recurrent Neural Networks. Accessed on September 27, 2022, https://www.nickmccullum.com/python-deep-learning/vanishing-gradient-problem/

[11] Simeon Kostadinov (2017). Understanding GRU Networks. *Towards Data Science*, accessed on September 27, 2022, https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be