Time Series Forecasting

Cecchetti Francesco, Concina Lorenzo

December 2022

1 Introduction

Time series forecasting is a very important research field, whose applications are multiple such as weather forecasts, forecasts of stock values or numerical series predictions for industrial purposes. In the last two years the approach to this problem has changed radically [1], passing from probabilistic and statistical models, to techniques based on Machine Learning and in particular Deep Learning [2], which have considerably improved the results. Of particular importance in this field are the challenges called Makridakis M-competitions [3], among which the last two (M4 2018, M5 2020), which have the objective of evaluating the accuracy of existing or new forecasting methods. In this report we summarize our approach to this challenge, presenting the models and techniques used to try to predict the considered multivarite timeseries, reducing the RMSE as much as possible.

2 Direct approach

Predicting the values of a time series is a complex task, which requires particular neural networks, capable of memorizing the evolution of the sequence, in order to be able to predict new values. To this end, it is possible to use recurrent networks (RNN), which unlike feed-forwards, have feedback in their architecture that implement the storage mechanism. However, vanilla RNN are not adequate for this challenge, as they cannot memorize sequences longer than a dozen samples due to the vanishing gradient problem during the training. We therefore decided to start experimenting with the use of LSTM (Long Short Term Memory), which on the contrary, are able to learn much longer dependencies. We therefore created a model with two LSTM layers of 64, 128 neurons each, followed by a single Dense layer, using a window of 900 samples and a stride of 90. This first submission gave us a poor result, 5.378 of RMSE. The idea of stacking two LSTM was gave by the fact that, additional hidden layers are understood to recombine the learned representation from prior layers and create new representations at high levels of abstraction, in particular, given that LSTM operate on sequence data, it means that the addition of layers adds levels of abstraction of input observations over time. Gave this observation, we tried to add to the previous model another third LSTM layer with 256 neurons with a dropout of 0.4 to prevent overfitting (since in this way the number of parameters increased considerably), but the result on the submission was worse (5.67). With subsequent experiments, we realized that the initial intuition of placing such a large window was wrong and that by decreasing its size the results improved considerably. We therefore decided to test a model made up of only two LSTM layers of 64 and 128, with a Dense layer to follow and a window and a stride of 500 and 5 samples respectively. This gave us our best result with a 3.621 of RMSE *(/Notebooks/Direct_LSTM.ipynb)*.

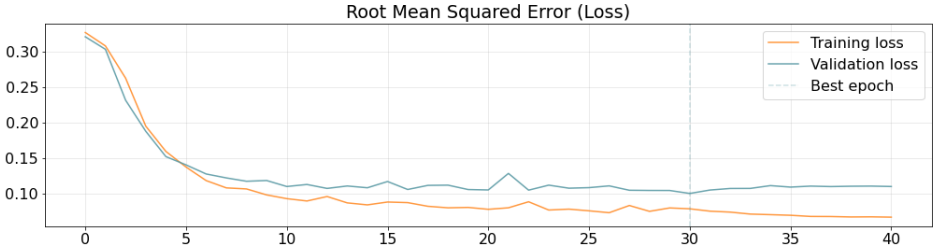

Figure 1: Plot of the RMSE

Having obtained this promising result, we then tried to fine-tune the hyperparameters, to further improve, such as varying the size of the window or the stride, the number of neurons of the LSTM layers (from 64

[1] https://towardsdatascience.com/an-overview-of-time-series-forecasting-models-a2fa7a358fcb

[2] https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0

[3] https://www.researchgate.net/publication/344487258_The_M5_Accuracy_competition_Results_findings_and_conclusions

and 128 to 128 and 256) or the amount of dropouts, however, without being able to further improve the submission result. For example, we also tried to increase the number of LSTM layers to three (64, 128, 256 neurons) but also in this case the submission result worsened to 4,322, probably because the number of trainable parameters was too large (just over two million, which is more than double that of the two-layer model).

Failing to further improve the results with this model, we decided to try to use the GRU layers instead of the LSTMs, experimenting with the same architectures as before, so once again two recurrent layers (GRU 64 and 128) followed by a Dense layer. After several attempts (playing also with the hyperparameters and the number of layers), the best result on the submission obtained with this type of recurring networks is 4,135.

The next step was to experiment with bidirectional recurrent layers. These layers are very useful in cases where all the time stamps of the sequence are already available in the training phase, just like in the case of this challenge. They consists of duplicating the first recurrent layer in the network so that there are now two layers side-by-side, then providing the input sequence as-is as input to the first layer and providing a reversed copy of the input sequence to the second. As a first experiment we tried to use a single bi-directional layer of 128 neurons, followed by a Dense layer, obtaining 4.33 of RMSE in the submission. We therefore experimented with other more complex models, starting from the stack of two bi-directional both 128 neurons with a dropout of 0.5 that prevented overfitting, obtaining in submission 4.218, therefore a small improvement. Given this observation, we tried to add a third bidirectional layer (this time with 32, 64, 128 neurons each), with however an immense deterioration on the submission, obtaining 13.9 of RMSE.

The last direct approach we tackled was the combination of 1D convolution and LSTM...

3 Autoregressive approach

Another possible technique to perform a multivariate timeseries forecast is to make use of an autoregressive model in which we use previous observations to essentially predict a single sample, of all the variables, for the next time step.
This can be simply done by modifying previous models telescope variable, setting it to 1. Telescope is prediction length of our model, as it will now predict 1 sample from each of the sequences we input at training time.
Important to notice is also the modifications bring to the model.py script, in which we made the requested predictions shifting 1 by 1 on the input test set and concatenating each prediction with each other. We therefore begun our experiments with some of the models we already tried with a direct approach to compare models' accuracy.

Thus, we trained a model with 2 bidirectional LSTM layers of 128 neurons each, that achieved 5.81 of RMSE on test set, which although is not that good, it was a promising starting point for this new approach. In fact, already with the second attempt, adding a dense layer of 512 neurons and tanh as activation function, before the output to the previous model, we have been able to reach 4.11 as RMSE on test set *(/Notebooks/Autoregressive_BiLSTM.ipynb)*.
Little did we knew that we could have not reached this precision with an autoregressive approach on later attempts. Indeed, we proceeded trying to do variations on the model architecture, such as switching back to LSTMs, increasing and decreasing model complexity with dense layers, and tuning of hyperparameters, especially of the window and stride, but never got better results nor in validation or test.

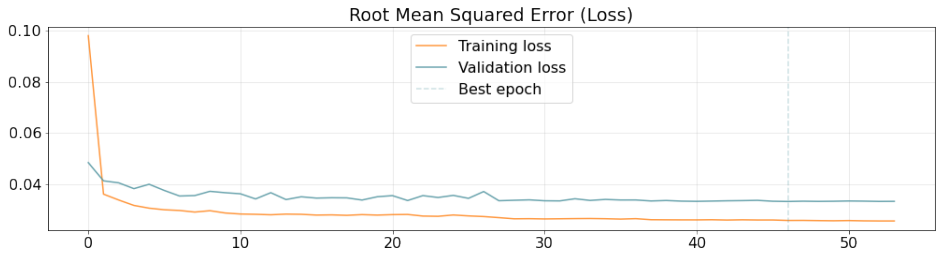

Figure 2: Plot of the RMSE

4 Other approaches

Another approach we explored was the combination of convolutional and recurring layers. The idea is to extract useful features that identify the time series through convolutional layers and to predict the next step in the series through recurring layers. A CNN does not support a sequence as input, but a 1D CNN is able to read the input sequence and learn its most important characteristics. We made many attempts of combinations with Conv1D layers searching the most balanced model complexity wise and performing fine tuning of model's hyperparameters. The best one consists in two Conv1D layers o 16 filters and 2 of kernel size, followed by a MaxPooling1D layer, a LSTM layer of 200 neurons and a Dropout layer as regularizer which resulted in a good 3.76 of RMSE in test phase *(/Notebooks/Direct_CNN_LSTM.ipynb)*.

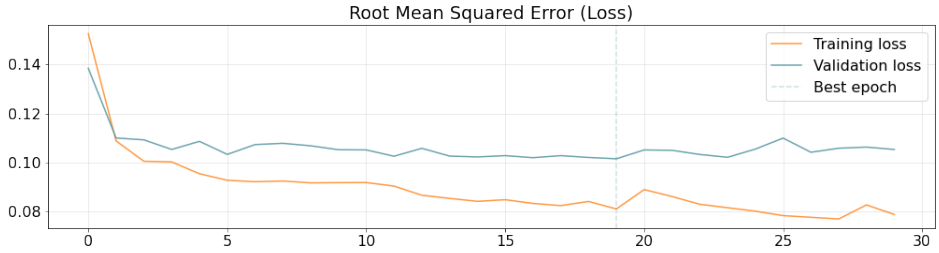

Figure 3: Plot of the RMSE