POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Artificial Neural Networks and Deep Learning

Homework 2: Time Series Forecasting

Authors:
- Domenico Cacace
- Miguel Gonzalez
- Usevalad Milasheuski

Team name: DMU

---

1. Introduction

Our strategy for this homework is to start with simple Recurrent Neural Networks, evaluating pros and cons of different architectures. Afterwards we implement a seq2seq model, on which we can then test *novel* methodologies, such as the Luong attention mechanism.

2. Dataset

The dataset used in this project consists of 7 time series, each containing 68529 samples. As for preprocessing, we normalized the data (MinMax normalization) and extracted the sequences to be fed to the network with a sliding window approach; for our models, we found that a window size of 200 and a stride of 5 yield the best results.
The models submitted to the competition are trained on the whole dataset, with a 80/20 split between training and validation; to determine the performance of a model (based on how well they predict the future) and decide what models to submit we reserved ∼15% of the original dataset for testing.
We evaluated both oneshot and autoregressive predictions on most of our model, and found that the latter produced better predictions in almost all the cases.

3. Recurrent neural networks

As a first step, we evaluated the different recurrent layers that Keras offers. For the sake of consistency, we used the same model structure, consisting of:

- Recurrent layer(s)
- `Dense` layer
- `Reshape` layer
- `Conv1D` layer

For the recurrent layers we used `LSTM` and `GRU`; we did not consider at all the `SimpleRNN` layer, due to its vanishing gradient issue. We trained our models using the Mean Absolute Error and the Mean Squared Error as accuracy and loss functions respectively (we also tried RMSE as loss, and noticed no significant change). In addition, we employed early stopping (max 200 epochs) and learning rate reduction.
Tests with *unidirectional* layers yielded similar results (RMSE around 9.5) for both LSTM and GRU layers; changing the number of units per layer (50, 100, 256) and the number of layers (2, 3, 4) did not bring a significant improvements. During this phase we observed that, in some cases, there was a jump discontinuity between the last input and the first predicted value: to solve this issue we employed a dropout layer after every recurrent one, and observed that the gap disappeared.

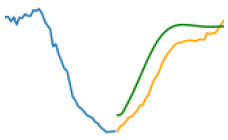

Figure 1: no dropout

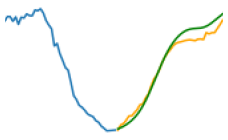

Figure 2: dropout (0.2)

3.1. Bidirectional layers

With bidirectional layers we observed significant differences among the models tested; as above, we changed the number of units per layer and the number of recurrent layers. We observed that the Bi-GRU models had significantly worse performances than BiLSTM ones for every variation tested: we attributed these outcomes to the structure of the memory units, as GRUs have less parameters than LSTM units.
Regarding BiLSTM models, we obtained the best results with 3 recurrent layers (RMSE 4.0945), while increasing the number of layers yields slightly worse results (RMSE 5.1252).

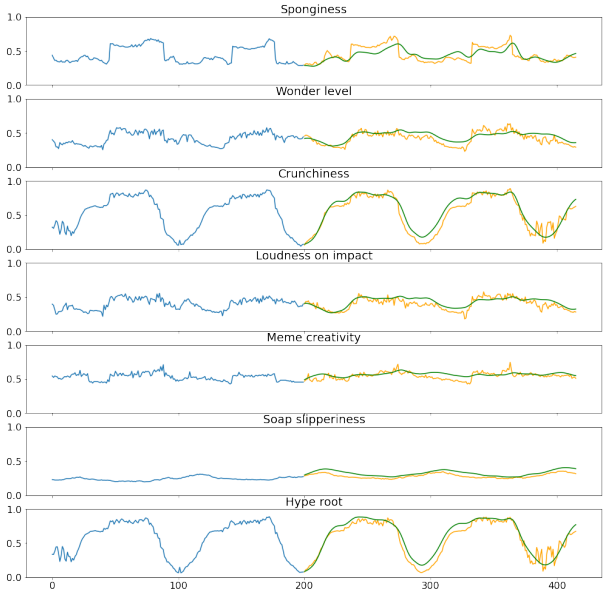

Figure 3: 3 BiLSTM layers (256 units), 0.2 Dropout

4. To seq2seq and beyond

Disclaimer: all the models presented in this section have been developed during the Codalab outage; the results we got via cross validation on our local test set differ significantly from the ones we got on the secret test set.

Our next step consists in building an encoder/decoder model. Instead of starting from scratch, we employed the results obtained in the previous experiments; following the classic encoder/decoder structure, we have:

- 3 BiLSTM layers, for the encoder
- 1 `RepeatVector`, to *connect the components*
- 2 BiLSTM layers, for the decoder
- 1 `TimeDistributed` Dense output layer

The results on the local test set suggested that this model performed similarly to the 3-layer BiLSTM; however, this model performs way worse (RMSE 9.7349): by taking a closer look at the results, we can see that the RMSE quickly diverges as we go *deeper* in the secret test set, going from 3.86 to 10.29 by moving from the first to the second quarter.

4.1. Luong attention

As a final step, we implemented the Luong attention mechanism; in this phase we had some trouble with the `Attention` layer provided by Keras, so we implemented it manually with dot products and Softmax (see the code and the model scheme in the appendix for more details).
By looking at the predictions on the local test set it looks like this model can outperform the 3-layer BiLSTM one, but the results on the secret test set say otherwise, showing a RMSE of 4.9470. Even though this isn't our best model, we were able to significantly improve the performance, keeping the RMSE lower among the whole secret test set (3.89, 4.69, 6.02 for the three quarters respectively).

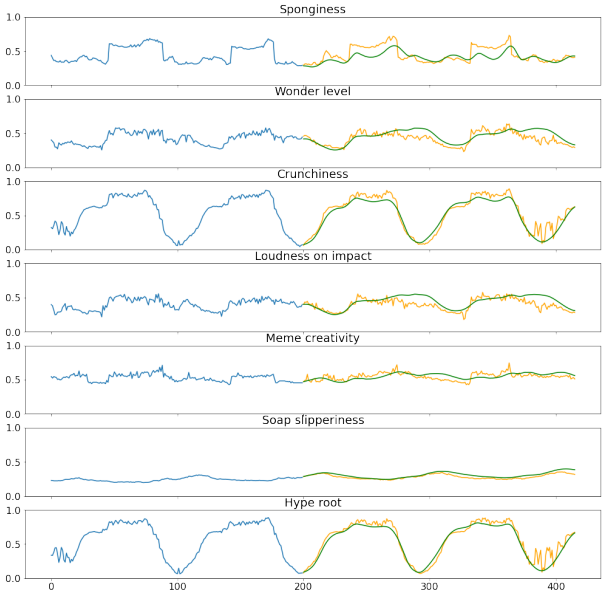

Figure 4: Encoder/Decoder with Luong attention

5. Conclusions

In this report we presented our small journey in the world of time series forecasting with neural network architectures. Starting from the basic memory blocks we built and evaluated several models, spacing between different architectures. We expected the encoder/decoder with attention to be our best performing model but, due to our implementation (we think we might have overcomplicated things a little bit), the simpler 3-layer BiLSTM model showed the best results; despite this, we had the chance to see first-hand how powerful the attention mechanism can be and why it is such a big deal.

Appendix A: *BiLSTM* model

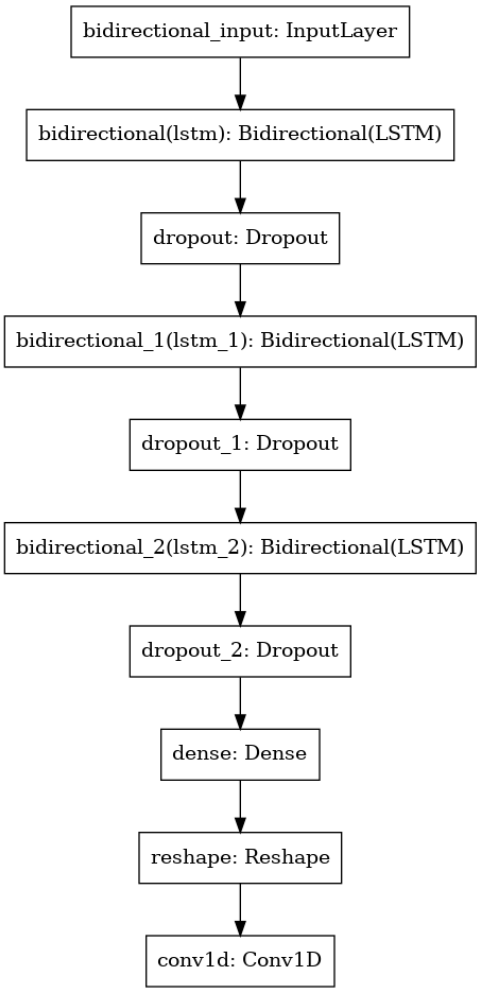

Figure 5: Encoder/Decoder with Luong attention

Appendix B: *attention* model

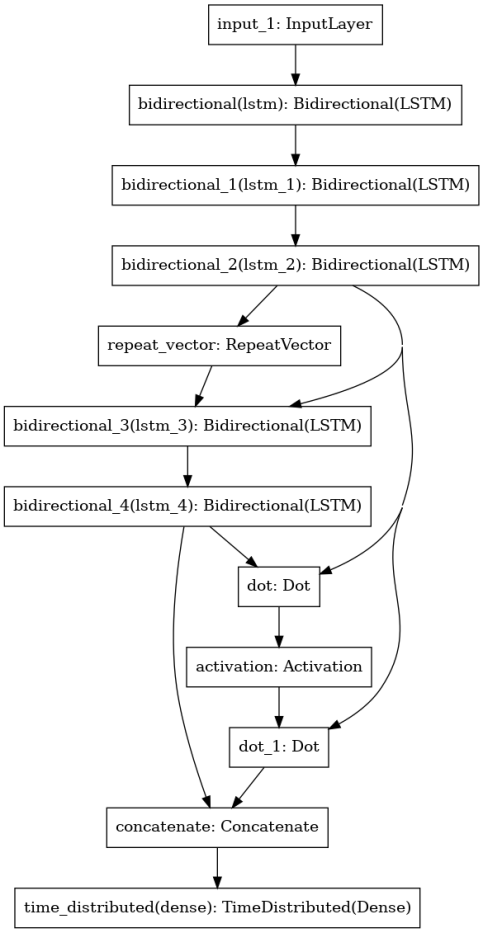

Figure 6: Encoder/Decoder with Luong attention

For the sake of clarity, we omitted the dropout layers; it is important to note that without dropout this model is way too overfit, scoring a RMSE of 32.68 on the secret test set.