Artificial Neural Networks & Deep Learning
Challenge 2

Denaldo Lapi and Fabio Raffaeli

Team: fabio_dena

June 22, 2022

1 Introduction

In the second challenge of the course we were required to design a forecasting model for a multivariate time series.

In this document we report the development of our models by dividing the explanation in the main approaches we used: hierarchical LSTMs, sequence-to-sequence models, Seq2Seq models with attention and transformers.

2 Data preparation

The data we were given was a multivariate time series composed of 7 features. The first operation we did on it was the sequential split between training and validation. Additionally, we applied Min-Max normalization, by using the statistics computed over the training set also on the validation set. We have to remark that as a consequence of this, our model produces normalized predictions, which have to be de-normalized during the test phase. We also tried to train some models without normalizing the data, but we obtained very bad results. We even tried standardization, but it didn't give any improvement.

In order to solve a time series prediction task, we had to build samples made of a pair input-target. In this context, the input is a certain sequence of `w` observations and the target is the sequence of the following `t` observations, where `w` and `t` are arbitrary number (window and telescope respectively). By sliding the window over the whole time series by a certain stride we obtained multiple samples, represented by sequences. The steps of this procedure are clearly described in the notebook. The choice of the values of window size, telescope size and stride was not fixed, we tried various combinations depending on the structure of the model currently tried.

3 Some details on training and models evaluation

An important point we would like to focus on is the choice of the loss function. In this case we have a regression task, therefore our first choice was the MSE, which is very sensitive to outliers since it is based on the square of the errors. We also tried the MAE, which just computes the absolute difference between the target and the prediction and therefore it is less sensible to outliers, but we did't obtain any improvement (the same applies for MAPE, which is similar to the MAE, but is based on a percentage). We tried also the RMSE (in a custom version written by us, since it is not present in keras), as it is the metric used to asses the performance of models in the competition, but it gave results very similar to the mean squared error and sometimes worse, thus we abandoned it.

A solution which resulted to be good (better than MSE in some cases, including our final best model) was the Huber loss, which behaves differently depending on how large is the difference between target and prediction: for values under a threshold δ it is quadratic, while for values larger it is linear. Thanks to some hyperparameter tuning we found that the best value for δ in our models was the standard value (1.0).

We report also the two 2 callbacks we used for training: one being the early stopping, useful to prevent overfitting, and the other for reducing the learning rate when reaching a plateau in the validation loss.

To assess the performance of our models during our experiments using various metrics such as the mean squared error, the mean average error, the mean absolute percentage error and the root mean squared error.

4 Hierarchical LSTM

In our first attempts we used a model whose core consisted of two bidirectional LSTMs. The loss function was the MSE and the optimizer was Adam's. To build the samples we used a window of 200 and a stride of 10 (it was the same model seen at the exercise session). We tried both the extreme approaches: predicting all the 864 future samples at once; predicting only one sample at a time and iterate the procedure by autoregression. We observed a better result with the first approach, which obtained a score of 3.9819 of RMSE on the test set. We tried also some middle ground solutions like autoregression with 108 predicted samples at a time, but they were not as good as the first approach.

Starting from the model with the telescope of 864, we then tried to modify the size of the window and the stride. Proceeding in this way, we obtained an improvement on performance by using a window of 200 and a stride of 5, combined with a dropout of 0.4.

We did other attempts by modifying the structure of the neural network, specifically we tried to change the number of units of the two LSTMs, the number of neurons of the convolutional layers and their activation function (we tried the leaky ReLU). We also tried to use the Huber loss instead of the MSE. None of these worked to improve performance thus we moved on to an approach based on a sequence-to-sequence model.

5 Seq2Seq models

Our initial sequence-to-sequence model followed the classic encoder-decoder architecture. We decided to start with an encoder and a decoder both made of a bidirectional LSTM. The encoder encodes the input sequence into a compact representation, which is then fed to the decoder that produces the predictions. This model led to results slightly worse than those of the previous approach.

As a consequence we tried different structures for the encoder and the decoder, by stacking multiple recurrent layers instead of using only one, and by trying different layers other than the bidirectional LSTM, like the plain LSTM layer, the simple RNN layer and the GRU layer. We also made several experiments by changing the telescope, the window and the stride. An aspect worth of mention is that in this case we obtained very bad results when using a telescope of 864,

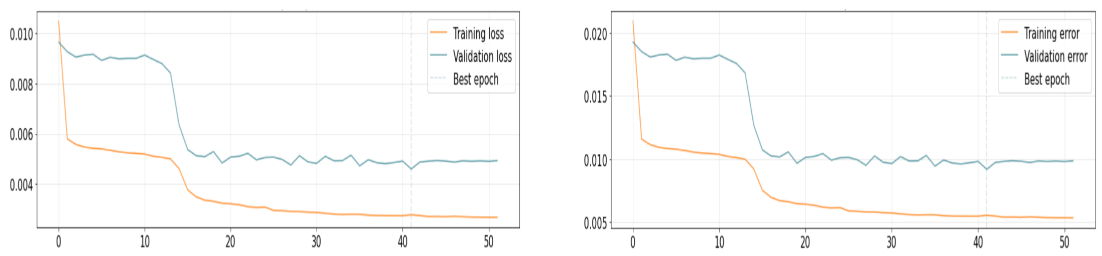

Figure 1: Huber loss (left) and MSE (right) in train (orange) and validation (blue) of the Seq2Seq model with Attention

contrarily to the approach of hierarchical LSTMs. Again, we tried not only the mean squared error as loss function but also the Huber loss.

Our best model based on the sequence-to-sequence architecture had an encoder and a decoder both made of a stack of 2 bidirectional LSTM layers of 128 units each, with a dropout of 0.2, and was trained with the Huber loss. The sequences in this case were built using a window of 450, a stride of 7, and a telescope of 216. This model achieved a score of 3.763 of RMSE on the test set.

6 Seq2Seq models with Attention

After struggling to improve our Seq2Seq models we decided to add the attention mechanism, as typically implemented in the Seq2Seq architecture, i.e. a content based attention built upon a memory represented by the encoder states. The details of this implementation are described in the notebook.

In this approach, as in the previous, we performed a lot of hyperparameter tuning both on the parameters related to sequence building (window, telescope and stride) and on the number and type of recurrent layers.

The best model we designed in this way, which was also the best overall, was composed of an encoder and a decoder both made of a bidirectional LSTM layer of 128 units, with a dropout of 0.2. It was trained with the Huber loss function and it worked with a window of 450, a stride of 10 and a telescope of 216. It obtained a score of 3.527 of RMSE on the test set. The Huber loss and the MSE for this model can be seen in Figure 1.

7 Transformer

After many fruitless attempts at improving the mentioned sequence-to-sequence model with attention we wanted to experiment with a transformer model. The transformer is the current state of the art in text processing and we thought that it would have been interesting to try to apply it to a time series task.

The architecture we tried was based on the multi-head attention mechanism. It was composed by a stack of encoder blocks, each one made of a Normalization layer, followed by a MultiHeadAttention layer, and then by a feed forward part. After the stacking of these multiple MultiHeadAttention layers we added a GAP layer followed by dense layers. The choice of window, stride and telescope was again of 450, 10 and 216, respectively. We did only few experiments with this model because of lack of time and they did not improve the results of our best model, but probably there would be space for further investigation in this field.