POLITECNICO
MILANO 1863

Artificial Neural Networks and Deep Learning

Homework 2

Pertusi Federica

Romano Giorgio

Zelioli Chiara

Introduction

This second homework aims at forecasting multiple univariate time series, involving the creation and implementation of appropriate Neural Networks architectures, and it is centered around developing a generalized model adaptable to a wide range of time series, with different characteristics.

Dataset Inspection and Preprocessing

The provided dataset consists of three separate datasets, one containing N = 48000 univariate time series belonging to different domains and grouped in six categories, which have been 0-padded in order to be stored in a Nx2776 array, one made of start and end indices for each time series, as each has potentially different length, and one with the respective category.
To split data into train set and test set, few expedients were thought. First, the split was carried out in order to maintain the original proportions of the categories, and, instead of picking the final part of each time series, 5% of the time series were entirely selected to be part of the test set, since an appropriate amount of training data was available. Furthermore, a fraction of 0.1 was used to split the data into training and test sets during the models' training.

Hyperparameters Tuning

Particular attention was devoted to adapting the function *build_sequences*, whose scope is preparing 200-timesteps sequences from the time series to train prediction models. It pads the time series with zeros to guarantee a compatible length with the window size, it discards those sequences with padding lengths exceeding the *max_padding* hyperparameter, to keep out sequences containing too many zeros that can negatively affect the models. It then constructs sequences according to the window, stride, and telescope.For the numerous cases where series are shorter than 200 timestamps, it was considered initially to replicate them until the window size was obtained, then to keep them unchanged (with zero padding to reach the size of 200) in order to avoid the introduction of possible biases or false patterns that don't reflect the actual nature of the series.

Concerning parameters setting, the *window* was fixed to 200 as the hidden test set on which prediction is performed is made of time series of length 200. Grid search was used to tune the *stride* value, among 10, 20, 25, 50. The selected value was 25, since 10 considerably slowed down model training, and 50 did not outperform 25, while 20 did not make a remarkable difference. Two were the possible types of forecasting: direct, with a *telescope* corresponding to the timestamps to be forecasted in the test set (9 in the first phase and 18 in the second), and autoregressive, with an *autoregressive_telescope* that was a submultiple of the *telescope.* Through grid search, we found that better results were obtained with direct forecasting and autoregressive forecasting using a value of 3 in the first phase and 9 in the second. As will be explained later, autocorrelation was used to choose a reasonable autoregressive telescope. However, the suggested values did not lead to better results.

Further Data Normalisation

Data was presented in the (0,1) range, presumably normalized using *MinMax* scaling, but, since this type of scaling may not be robust with respect to outliers, further scaling was performed. However, comparing models' performances when trained with re-scaled data and given data did not support the necessity of applying extra scaling. Before coming to this conclusion, it was thought to apply *RobustScaler*, as by removing the median and scaling data according to the interquartile range, it is robust with respect to outliers. Moreover, the spread data was re-scaled in the (0,1) range, because the training of the models gave much worse results with non (0,1)-ranged data.
Furthermore, the effectiveness of further scaling time series was tested carrying out post-processing, in order to de-normalise data and coherently make predictions.

Methods

Dealing with Categories

Aiming at building a general model able to predict time series independently on the domain of data, categories were not considered in any way except for data exploration and for fairly splitting train and test set, so that belonging to one of the six categories impacted model training the least possible.

Autocorrelation Analysis and Comparisons with Machine Learning Time Series Analysis

Autocorrelation measures the correlation between a time series and a lagged version of itself and its study is crucial in traditional time series analysis. Specifically, autocorrelation (ACF) and partial autocorrelation function (PACF) plots guide the type of model selection and its parameters estimation. For this homework's sake, previous knowledge was exploited to orient the type of forecasting, as a sharp cut-off was visible in the majority of PACF plots, one for each series, after 2 or 3 lags, suggesting the choice of an autoregressive prediction.
Dealing with time series usually employs stationarizing data, removing deterministic components, trend and seasonality, which may introduce misleading patterns. RNNs, LSTMs are good at correctly learning with non-stationary data, and stationarizing did not really improve models' performances.
Further study on autocorrelation dynamics as the window moved was explored. It could have been helpful in identifying time windows or adapting the prediction to the current autocorrelation structure because, instead of computing autocorrelation for the entire time series at fixed lags, *rolling autocorrelation* computes it for consecutive overlapping windows of the time series. However, it was not possible to transfer these thoughts to 48000 time series, inspecting its rolling autocorrelation plot.

Attention Mechanism

The underlying idea of attention mechanisms is to make the model focus on those parts of the input sequence that are most relevant for prediction. Instead of treating all input elements equally, attention assigns different importance to different elements, refining the Encoder-Decoder model.
There is not a unique way of implementing such a mechanism. In this research, three attention layer implementations were mainly exploited: the multi-head attention, the Bahdanau attention and a third custom attention layer. The latter, consisting of two sub-layers (two dense layers self.W and self.V) was the one achieving the best results.
The attention mechanism has proven to be a true game changer in this work, leading to models that outperformed their previous version, not exploiting any attention technique.

LSTM

Long Short-Term Memory networks are an evolution of RNNs that avoid the vanishing gradient issue employing a three gates mechanism: input, forget, and output gates. They are particularly effective in processing and predicting sequences by maintaining a memory of important past information for an extended period. The LSTM layer was present in almost every of the candidate models, because of its key characteristics and advantages. LSTMs also have some limitations, such as high computational complexity and vulnerability to overfitting; in fact, every attempt of models with multiple LSTM layers and a more complex architecture implied overfitting and complexity issues.

GRU

Gated Recurrent Unit NN works like LSTM but with two gates (the forget and input gates are combined into a single update gate) and instead of having a separate memory cell, it stores long and short-term information in a single hidden layer, resulting in an overall less complex model, with a remarkable gain in computational speed. GRUs can perform better in less complex tasks or scenarios due to their simpler architecture. This might explain why the GRU outperformed the LSTM in this case: aiming to develop a universal model for medium to small-sized univariate time series.

Models

Baseline

First of all, a straightforward model, referred to as baseline, was developed and used as a benchmark for our future models. Specifically, a model predicting a constant forecasting was chosen as baseline, equal to the last value of the input sequence. Such a model, evaluated on the test set, showed a mean squared error (MSE) of approximately 0.03.

Initial Model

The first and principal model, reaching low MSE values on the hidden test set, is a quite simple model. Its first portion consists of a Bidirectional LSTM layer with 128 units followed by a convolutional layer. The obtained embedded volume is then fed to the aforementioned custom attention layer. Lastly, a dense layer connects the network to the output shape. On the hidden test set, this model obtained a MSE value of 0.01038, substantially better than our baseline value.
Many attempts of improving the previous model were carried out, leading to many different and also more complex architectures. In what follows, some examples of such attempts are reported.

Time2Vec representation

In order to build a better model, the addiction of an embedding layer was tried, as it should be able to better represent the input time series. A Time2Vec layer was then implemented, with the aim of learning an effective representation of time, in order to fully exploit the temporal information.
This layer made the model deeper, showing better learning abilities and even outperforming the previous model on our test set. Unfortunately, this did not improve the performance on the hidden test set, which was approximately the same as before.

Final Model

After substituting the LSTM layer with a GRU one to simplify the model's complexity, a straightforward architecture with a GRU layer with 256 units was carried out, coupled with an Attention layer and a Dense layer. This minimalistic setup unexpectedly led to the best results, reaching an MSE of 0.0082 with autoregressive forecasting and 9 as *autoregressive_telescope.* Given the nature of the task, this simple model was the perfect trade-off between model performance and computational efficiency, in order to provide a forecasting as universal as possible.

ResNet

Additional attempts were made by exploring more complex models: a ResNet-inspired architecture was created with 3 residual blocks and different layer types. Employing Conv1D layers, we developed a convolutional model that emerged as a valid alternative, yielding an MSE value of 0.0083. Additionally, more experiments with LSTM and GRU layers were tried. However, integrating these recurrent layers escalated the model's overall complexity, resulting in overfitting issues.

Conclusion

The structure of the data presented several challenges, forcing the attention to the exploration and the preprocessing phases, and the meticulous construction of appropriate training and test sub-sequences. Furthermore, the model selection posed the challenge of detecting a suitable trade-off between computational complexity, adaptability and universality, in order to effectively handle the large volume of timeseries, the zero-padding, and the diverse categories of time series. Nevertheless, the goal of finding a model capable of predicting future timestamps while maintaining a certain level of generality has been achieved, while carefully monitoring the risk of overfitting.

Contributors

Every member addressed different sub-problems, read documentation and explored techniques while actively participating and discussing the overall group progress. Therefore, the following indications cannot utterly illustrate the single contributions, as we worked together to interpret each other's results and decide which direction to proceed in. In particular:

- Federica implemented the Final Model, the ResNet-like models and handled the building of the sequences;
- Giorgio implemented the Initial model, the Attention mechanism, the Time2Vec version and handled the building of the sequences too;
- Chiara took care of preliminary analysis, the analysis of the autocorrelation, the data scaling and the report.

Bibliography

Attention is all you need:
https://arxiv.org/abs/1706.03762

Time-series forecasting using Attention Mechanism (Initial Model code):
https://www.analyticsvidhya.com/blog/2023/06/time-series-forecasting-using-attention-mechanism/

A comprehensive guide to attention Mechanism in Deep Learning:
https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/

Time Series Forecasting with Deep Learning and Attention Mechanism:

https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc

Time2Vec: Learning a Vector Representation of Time:

https://arxiv.org/abs/1907.05321

Time2Vec for Time Series features encoding:

https://towardsdatascience.com/time2vec-for-time-series-features-encoding-a03a4f3f937e

Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline :
https://arxiv.org/pdf/1611.06455