POLITECNICO
MILANO 1863

ARTIFICIAL NEURAL NETWORK AND DEEP LEARNING

Challenge 2: Signal Forecasting

Authors:
Naclerio Andrea - 10934883
Pivetta Federico - 10932991
Spolaor Alessandro - 10717380

Professor:
Prof. Boracchi Giacomo

Thursday 21st December, 2023

1 Introduction

The aim of this challenge was to develop a deep model able to forecast signals. In particular, we were provided by 3 vectors: training-data, categories and valid-periods. Training-data was a vector containing N=48000 rows, one for each signal recorded, and 2776 columns. Signals were zero-padded in order to obtain the same length for all the signals. Categories was a vector containing N elements, each i-th element represents the class of i-th signal. Finally, valid-signal was a vector containing, for each i-th row, the point where the signal starts and the padding ends. To assess performances, Codalab platform was used. In particular, in the first phase model was tested on 9-points forecasting while in the second phase, the model was evaluated on 18-points forecasting.

2 Exploratory data analysis

Before approaching preprocessing and model definition, we performed an exploratory data analysis to understand the characteristics of the signals and classes. First of all, we extracted valid signals by removing zero padding. Then we analysed the signals' length distribution. The resulting histogram of the valid lengths was mostly concentrated below 500 samples. It is characterized by a huge peak of around 50 and its mean is 198 samples. This information was used to select the size of the window in the Long-Short-Term-Memory, LSTM, models. 50 was considered a reasonable value because it allows to detect important features of the train while not including a lot of zero-padding. Then we approached signals clustered by class, in order to identify class-related features. We analysed valid length distribution, mean and standard deviation of the signals and mean and standard deviation of the dynamic range. There were no evident differences among classes. Therefore we directly looked at some of the signals and no common pattern were identified for each class. Therefore, we decided to drop class information and create only one model that aimed to generalize all the signals. Finally, we analyzed the average distribution of the signals on the y-axis. Data have been already normalized between 0 and 1. Indeed the mean is 0.434 ± 0.130 with averaged standard deviation equal to 0.234. Dynamic range is fully exploited by most of the signals, in fact, the mean dynamic range is 0.93 ± 0.12 and only 613 exploited half of the available range. This analysis was performed considering the whole valid signal, moving to the windows we noticed a different behaviour: a more consistent number of windowed-signals were distributed in the lower part of the y-axis. It was confirmed by the mean dynamic range of the windowed-signals that is equal to 0.40 ± 0.26. As a consequence, models were not able to perform near 0. We implemented different preprocessing methodologies to overcome this feature that will be discussed in 3.3.

3 Data preparation

3.1 Data stratification

The time series provided cover six different domains: Demography, Finance, Industry, Macro-economy, Micro-economy and other; in fact the 48.000 signals are splitted in 6 different categories, from A to F. The number of signals in the different classes was quite unbalance, moving from more than 10.000 samples in 4 classes (B,C,D,E) to 277 in the F one.

Despite no characteristic features have been identified between the different classes, we decided to create our training, validation and test set preserving the same stratification.

3.2 Padding

The inputs of our models are windows of a fixed length. In order to generate them, only the valid part of each signal has been taken into account and used in the windowing procedure. As already mentioned in the Exploratory Data Analysis section, we decided to use a window of 50 samples in order to reduce the number of input signals with mostly zero padding. Despite this procedure, there were still some windows with a relevant padding's occurrence. As first, since padding does not provide any relevant information, we decided to discard those signals in which the padding length (number of consecutive samples equal to 0 starting from the left part of the signal) was greater than the 40% of the valid signal length.

Looking at the remaining signals, we noticed that there were unusual peaks in the connection between the last padding sample and the first valid one since the signal started from 0 and went instantaneously to an higher values. To avoid this behaviour, we tried different approaches: the first was to substitute the padding with a noise with the mean and standard deviation of the valid signal component in each window. Since the performance was worst, we decided to replace the padding with the signal flipped version. We replicated the samples in a symmetric way, keeping fixed the 1st valid samples and flipping the signal, cropping only the portion corresponding to the padding length. Since the performance was improved using the last preprocessing, we decided to adopt this preprocessing (still combined with the discarding phase of those windows with a padding part higher than the 40% respect the valid signal length.

3.3 Transformations

Looking a the graphical predictions of our models, we saw that the worst results were present when window's amplitude was very small, under 0.1 while were more accurate with middle-range (0.3-0.7) amplitude. For this reason we choose several preprocessing transformations to solve this behaviour. As first, $log1P$ transformation has applied; this type of transformation reduces more the high values while the smaller one are slightly. Since our aim is to map the small values in the middle-range, we decided to compute the log1p(1-X); in this way the signal is reversed, so the small values are mapped in the higher one and then they are brought in the middle through log1p transformation. We need to underline that this type of transformation squeezes the interval; thus the final windows amplitude has a dynamic range smaller than the initial one (0-1).

A second transformation was tested; we still tried to increase the small values but, conversely the previous one, keeping the same dynamic range between 0 and 1. The chosen function was the gamma transformation $y = x^{\gamma}$ with $\gamma = 0.7$ In order to implement these transformation, it's necessary to apply them on the training and validation set but it's important to come back on the original domain.

We tried to adopt those transformations in out models but they did not show a consistent behaviour; the first transformation returned better results with some models while the second one with others. In order to choose which one to apply, we toke the one that return better results on our most performing model.

3.4 Fast Fourier Transform

By looking at the prediction of the models that will be discussed in 4.2, we noticed that the model predicted the average of the signal in that window and its ending trend. Therefore, it was not able to learn high-frequency patterns of the signals. In order to facilitate the learning process of frequency features, we implemented architectures with two inputs: signal in time and frequency. The spectrum of the signal was computed using the FFT, *Fast Fourier Transform*, function which implements the DFT, *Discrete Fourier Transform*. To avoid large values at frequency equal to 0, it was computed on the difference between the signal and its mean. Then the second half was dropped to not induce repetitive data due to the symmetry property of the Fourier transform. The results of the models that implement both time and frequency information will be discussed in 4.3.

4 Models

All the models were implemented to forecast 18 points in order to be consistent with the final test output shape, therefore we fixed a telescope of 18 samples. Despite that, we also tried different prediction lengths (3, 6 with auto-regressive model and 9) but the performances were lower. For the first CodaLab test phase, since the number of predicted samples was 9, we truncated our model predictions taking only the first 9. For what concerns the loss function, the mean squared error function was adopted.

4.1 Baseline

The evaluation metrics (mean squared error and mean absolute error) were not tangible, therefore we decided to identify a standard forecasting model which provided us with a baseline for assessing the effective performance of the model. The selected standard model predicted all 9 samples equal to the average of the signal in all the input points. In the first test, the resulting mean squared error was equal to 0.0997.

4.2 Basic models

We approached the problem using Recurrent Neural Networks, RNN, because they are able to implement a memory state which can store information about the relationship of consecutive input elements. This characteristic is fundamental in analysing time sequences because all the time points are correlated. Firstly we implemented a simple network composed of 2 hidden layers: one LSTM layer and one convolutional layer. The output layer shape was fixed to (18,1) using a dense layer and a cropping layer. These last two layers were included in all the models, consequently their discussion will not be provided anymore. The model was performing better than the baseline, suggesting that it was extracting information from signal trends. Then LSTM layer was substituted with a bidirectional-LSTM layer in the same architecture. This second model outperformed the first one, consequently, bidirectional-LSTM layer was selected as the recurrent part in the next models. Thirdly, a multitude of models composed of multiple concatenations of bidirectional-LSTMs and convolution layers were implemented. Moreover, we tested also blocks composed by convolutional layer and bidirectional-LSTM. The model named *model-basic* that outperformed the others on the mean squared error on the test set was composed of two blocks bidirectional-LSTM and conv1D layer with kernel=3. The last approach was to implement a ResNet-style architecture preceded by a bidirectional-LSTM layer. A ResNet-style block was defined by three consecutive groups of conv1D, BatchNormalization, ReLU activation and dropout layer. Each block ended with an add layer (receiving as input the first and the last layers) that allowed the propagation of the residual. Although multiple models were performed by varying the number of blocks included, the number of activations in the conv1D layers and their kernel-size, anyone outperformed *model-basic* on test.

4.3 Time and frequency

In addition to the standard LSTM, we thought on combine the information regarding the time domain with the frequency, as mentioned in the section 3.4. In this way, the architecture manages 2 inputs. On one hand the signal window in the time domain is processed with *model-basic*. On the other, the window signal's FTT is the second input where the information are processed by sequence of 1D Convolutional layers. The layer combinations was chosen to adjust the output dimensions of each branch, such that they can be concatenated in a unique latent space representation. The resulting model was called *model-TF*. Transformations (sec. 3.3) were performed. *Model-TF-1P* has been obtained applying the transformation log1p(1-X). In order to process these information, several combination of dense layers were tested; the one that gave the best result was the one with a single dense layer with 128 neurons. This model was called *model-128D*.

Starting from this structure, we tried to add another window as 3rd input. Our idea was to have the larger window in order to allow the model to understand the general behavior and a smaller one that crop the last part since it contains the most relevant samples for the forecasting. The 3rd input was processed in the same way as the larger window (Bidirectional, MaxPolling and 1D convolutional layer) trying several length combinations. Despite this intuition, the performance of this model called *model-TTF*, was lower than the architecture with a single window.

We also tried an additional implementation on the base model, this time keeping the 2 inputs (time and frequency) but changing the frequency branch structure. The first approach was to replicate the same structure adopted in the time branch of *model-basic*, the aim was to incorporate a recursive structure also on the frequency branch. The second approach was to use a bidirectional-LSTM followed by a ResNet-style block composed of 3 Conv1D layers and one final adding layer. Both implementations resulted in good forecasting of new time points, in particular, the results of the second model, called *model-TF2* and its optimization *model-TF2-128D-1P* (transformation log1p(1-X) and 128 dense layer after concatenation) are reported in figure 1.

4.4 Multi-Head Attention

The last implementation approach to the problem was the use of transformers. In particular, we tried to implement a transformer with 4 multi-head attention blocks, following the architecture reported in [1]. We followed the backbone proposed during the last lecture and we adapted the hyperparameters to our problem (e.g. sequence-length=50). The performances of this model, called *model-MHA*, were poor compared to the others. Unfortunately, due to the lack of time, it was not possible to optimize it and fully exploit its potentiality.

5 Results

In figure 1 the results of some of the implemented models are reported. As previously stated, the first external test was a 9-point forecasting problem while the training, validation, internal test and second external test were performed on 18-points. Some results report an * which indicates that they are metrics obtained in the numerical space of the log1p(1-X) transformation, therefore they can't be comparable with the others.

MODEL	TRAINING MSE	VALIDATION MSE	TEST MSE	FINAL_TEST 1 MSE	FINAL_TEST 2 MSE
model-basic	0.0115	0.0116	0.0113	0.00594	0.01063
model-TF	0.0112	0.0113	0.0110	0.00584	0.01005
model-TF-1P	0.0048 *	0.0048 *	0.0162	0.00528	0.01034
model-TF-128D	0.0112	0.0114	0.0111	0.00565	0.01048
model-TF2	0.0095	0.0107	0.0105	0.0055	0.01078
model-TF2-128D-1P	0.0048 *	0.0051 *	0.0107	0.00531	0.01050
model-TTF	0.0110	0.0104	0.0161	0.00610	/
model-MHA	0.0639	0.0632	0.0643	0.1098	/

Figure 1: Results of the models

In order to choose our final model, we have considered the one with the best performance in 2nd external test. Results in test 1 were not taken into account because they were computed on 9-points forecasting, while the project request was 18 points. Internal test was not considered as an evaluation criteria because it was not independent. Therefore *Model-TF* resulted the best one.

6 Conclusion

In conclusion, implemented models resulted in acceptable in this forecasting problem. With respect to the baseline, all of them outperformed in the internal test and in external test 1. As suggested by the performances, model-TF reached promising results in 18-points forecasting. The integration of frequency-based information with time-based data led to consistent improvements. Indeed, oscillation information helped to model forecast high-frequency patterns of the signal. Despite promising intuition, no relevant improvements were obtained by data transformation. Other typologies of data processing may achieve better results. Further implementation to resolve this problem could be a deeper exploitation of transformer-based models.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: `1706.03762`. [Online]. Available: `http://arxiv.org/abs/1706.03762`.

7 Contribution

All the members worked together in all the phases of the project. In particular, we approached the task using a brainstorming approach. Then different ideas were followed by different members of the team. More in detail Federico Pivetta developed time and frequency models, Andrea Naclerio developed models with one additional time window and autoregressive models. Alessandro Spolaor focused on multi-head-attention models. The final report was written by all the members.