Homework 2

Filippo Lazzarin, Davide Rizzotti, Simone Tognocchi, Enrico Marinelli

December 26, 2023

1 Introduction

The problem of interest is time series forecasting, which means to predict the future samples of a time series given the past samples. Our dataset is composed by timeseries of different length, and our goal is to design and implement a forecasting model which is able to learn how to exploit past observations in the input sequences to correctly predict the future. In our framework,in the first phase of the challenge it is required to predict the future 9 samples based on the past 200, while in the second the future 18 samples.

1.1 Approach used by the team

As for the first homework, we spent some time together to visualize, analyze and process the data. This phase was longer than the previous homework, since the preprocessing of time series requires a lot of steps. The meticulous execution of these steps is crucial to achieve favorable outcomes. After that, we started working in parallel so that each of us could experiment different models. Throughout this individual exploration, we maintained an ongoing feedback loop, sharing results and discoveries. This approach facilitated extensive experimentation, enabling us to derive optimal insights from our collective efforts. We found this way of collaboration to be the best, since from each result we outlined the response to different initializations, hyperparameters and architectures, in order to gain experience to build the most performing choices.

2 Dataset Preprocessing

2.1 Dataset Overview

The dataset is composed of 48000 univariate time series of different length, containing a minimum of 12 samples to a maximum of 2776 samples. Time series are divided into 6 different categories:

- A, with 5728 timeseries
- B, with 10987 timeseries

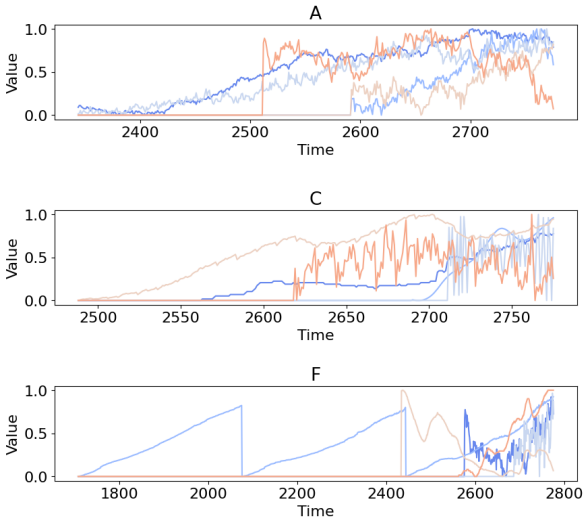


Figure 1: Visualization of time series of category A, C, F

- C, with 10017 timeseries
- D, with 10016 timeseries
- E, with 10975 timeseries
- F, with 277 timeseries

2.2 Data Visualization

To directly understand what kind of time series we had in our hands, we plotted a few time series for each category, to see if we could spot evident difference between different categories, but we couldn't find any. In each category we could find periodic time series with evident seasonality, positive or negative trends, very noisy time series or time series without any particular pattern (see Figure 1).

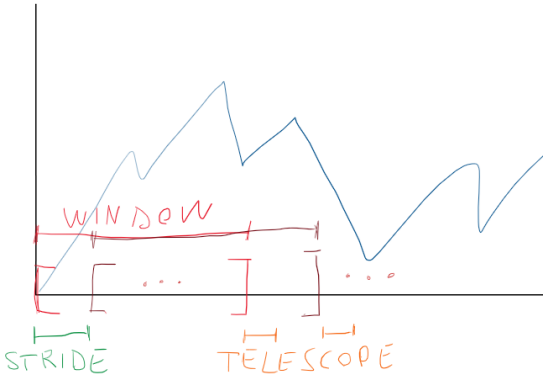
2.3 Train, Validation and Test sets

The dataset was partitioned as follows: we used 80% our our time series for the training of the models, and 20% for the testing phase; 10% of the training is used to validate the model. To create more balanced training and test sets, we did a

80/20 split for each category, and then we put together these divided sets into two distinct training and test sets.

2.4 Creation the training and testing data

Since our goal is to predict the future 9 samples based on the 200 before, we designed our architecture to take as input a vector of 200 samples and as output a vector of the 9 predicted samples. We used, then, a window based approach, where input sequences were generated by framing each time series with windows 200 samples in length. Labels were derived by considering the subsequent 9 samples relative to each window. The gap, in terms of number of samples, between one window and the consecutive one is called stride, and it's an hyperparameter for our problem. In order to perfectly fit the windows into our timeseries, a zero padding was added at the start of the time series. In the code this process is shown in details.



Our training dataset will consist of n sequences, each comprising 200 samples, accompanied by n corresponding target vectors containing 9 or 18 samples.

2.5 Scaling data

In order to deal with outliers and varying scales among the different features of our dataset we decided to use in combination a Robust Scaler followed by a MinMaxScaler.
The Robust Scaler removes the median and scales the data according to the interquantile range, also called IQR. This makes the scaling resistant to outliers. After applying the Robust Scaler, the central tendency and variability of data are better represented, making the data more uniformly distributed.
After the Robust Scaler, the MinMaxScaler is applied. This scaler transforms features by scaling each feature to the range between 0 and 1. The transformation is given by subtracting the minimum value of the feature and then dividing by the difference between the maximum and the minimum value of the feature.
The combination in sequence of these two scalers allowed us to improve significantly our results. The main reason for which we decided to apply them sequentially is because with Robust scaling we were able to give a smoother distribution of the features, while with MinMaxScaler we re-transformed our feature into the original domain, which had features between 0 and 1.
In the following chart we can see how the scalers changed our initial data, giving a more polished shape.

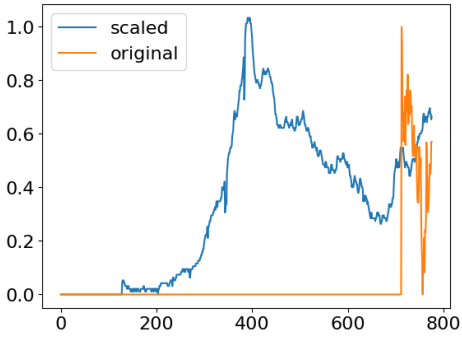


Figure 2: Scaled vs Original time series

3 Model development

3.1 Introduction

Initially we tried to tackle the problem with an ensemble approach, dedicating space of a model for each category, knowing that in the prediction phase was given the category of the single time-series, hence we could have chosen the model with a conditional statement on that information. The lack of expected performances, combined with the low time to test these models let us move towards a unique larger model, with many more neurons and layers with respect to the little models we were training in the first place.
Since the task to be solved is time series forecasting, we decided to use LSTM based architectures in order to keep memory of the past samples of each time series, adding Convolutional layers in order to extract features from these compact representations of the past. In this way, we are able to understand general patterns of the time series of our dataset, and predict the future with fairly high accuracy.

3.2 Training

The idea behind the network was, indeed to combine Convolutional and LSTM layers, as shown during laboratory classes. To enhance this combination and have a more complex unique network we thought to add more than one couple of the aforementioned layers. Hence we have experimented different trials their size, in terms of neurons, and the number of layers theirselves, starting to 1 up to 3 couples of layers. In addition to these simple layers we have added layers to process features in between, such as the Attention layer, used to let the network focus on specific parts of the sequence transfered from a convolutional layer. Moreover parallelizing the output of a "feed-forward sequence" Convolutional-Batch-Attention-LSTM with one or more extra Convolutional-Batch layers that process the same feed-forward output has been an idea that improved network performances, that we kept using after every feed-forward sequence. The addition of the two resulting outputs are then sent to the next layer. The best performing repetition of this scheme has proved to be 2. The network ends with another larger Convolutional layer, a GAP layer and a Dense layer to determine the future 9 or 18 time-steps of the given 200-steps sequence, the only difference in the model itself is the output size based on this choice.

3.3 Submission

Before submitting the model and make predictions about time series forecasting, we had to firstly scale and normalize the data. Therefore, since before training the model we used the Robust scaler and the MinMaxScaler, we had to apply the same procedure also to the unknown test data. What we did was to apply the formulas directly without the supply of any external library, using the statistic values obtained from training data. Thus, we firstly subtracted by the median of 0.0001693465979 and divided by the IQR of 0.0017968303597. Sequently, we applied the MinMaxScaler formula with the min value of -0.09270888444287544 and the max value of 538.3877997431756.

4 Conclusions

We have tried very different models, starting from an approach with many simple models, going through a unified model for each class. We kept increasing complexity of the model also thanks to the *optuna* library used to perform multiple tests in sequence, having a more clear view on the relationship complexity-performances. With this coding behaviour we have reached discrete results, that have been enhanced with a more systematic approach on data-preprocessing, by a normalization pipeline with median, IQR, minimum and maximum obtained from the full dataset. Once in prediction phase we post-processed outputs repeating that sequence backwards. The obtained results, with a stride of 10 for the sequence building, have been satisfying. We'we been able to achieve a result of 0.00484224 of MSE in development phase, and of 0.00892815 during the final phase.

5 Contributions

- Davide Rizzotti: Development of code to split the initial dataset into training and test, to create the sequences and to create the sets of data usable to train the models. Training of models with different parameters. Contribution to the report mainly in the Introduction and Data preprocessing parts.
- Simone Tognocchi: Main contributions to the model development and training procedures. Great effort in highlighting model differences and relationship with hyperparameters in the pre-processing phase. Contributed in writing and revising the report.
- Filippo Lazzarin: Preprocessing of the data, in particular with the implementation of the data normalization through the Robust scaler and MinMaxScaler both for training and submitting. Tried different model configuration with different hyperparameters. Contributed in writing and revising the report.
- Enrico Maria Marinelli: Data visualization for each single class contribution it. Main focus on tuning and selecting number of layers and neurons in the final network. Studying the connection between model complexity and sequence building. Contribution on report development.