POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Second project report - ANN2DL

Group: broccAlly

Members: Niccolò Bindi, Matteo Forlivesi, Lorenzo Franzè, Alessandro Lupatini

1. Introduction

This report summarizes the steps that we took for building a model for the univariate prediction of time-series belonging to different categories. After a section that goes into the details of the data exploration and preprocessing phases, we talk about the different models that we tried to build and their evaluation.

In particular, we built an ensemble model where we trained different models for each of the different classes in the dataset and two general models trained on the whole dataset, one that used bidirectional LSTM and a seq2seq model that also included an incorporated attention mechanism.

The report concludes with a summary of the results obtained by the evaluation of the different methods and the teachings we got from them.

2. Data exploration and preprocessing

Upon data loading, a thorough visual exploration using a custom function provided valuable insights into the time series dataset's inherent characteristics. Notable observations include discernible trends in category A, random patterns in category B, lack of correlation in category C, varied lengths in category D, strong correlations with seasonality in category E, and stationary noise atop a slow exponential trend in category F. (more details left in the notebooks).

In the post-loading phase, various experiments were conducted to optimize data preprocessing. Robust scaling was tested but did not outperform the existing min-max scaling in the dataset.

Detrending experiments involved applying first and second-degree polynomials to individual data samples. However, the diverse and varied nature of the data led to a counterproductive impact when using the same polynomial degree uniformly across samples. Consequently, this detrending approach was abandoned. The decision to forego additional detrending was grounded in the expectation that the models could effectively capture the inherent nonstationarities of the data during training.

Intra-category variance was identified during exploration, signaling that visualized trends might not fully represent the classes. An attempt to visualize the covariance function revealed nonstationarity, leading to the adoption of a compromise stride of 10(for most classes).

To achieve class balance, a custom sample-generating function was developed, incorporating class-specific strides and an upper limit on samples per class. Time series were transformed into windows of 200 data points with a prediction length (telescope) of 18, organized into a category-specific dictionary for efficient retrieval.

This meticulous approach ensures a representative and stratified dataset. The commitment to equal class distribution, facilitated by upper limits on sample counts, mitigates potential biases. The dataset was divided into training, testing, and validation sets using a 60-20-20 percent split.(In some notebooks like "Forecasting_using_seq2seq_models " this division was performed at the beginning but since the sample generation function discriminates by category the stratification of the sample was maintained.)

Two strategies were explored for padding time series to extract samples.

Initially, zero-padding was employed to standardize length. Subsequently, a refinement involved discarding samples with excessive padding at the beginning, enhancing dataset quality. The strategy reflects a commitment to optimizing both structural integrity and information content in the dataset, contributing to the effectiveness of subsequent modeling endeavors.

In a second moment we thought that the dataset given was very heterogeneous in terms of timeseries' length, in fact only the 45% circa of the dataset is at least a 200 length timeseries. So, to improve the performances of our model, we thought about a different type of padding, since using only 0 could possibly "deactivate" the information given in the first timestep of the timeseries. The padding used was to repeat the last part of the timeseries without mirroring it, unfortunately we notice in the development phase a drop of the performances (the model used was at 0.0055 with zero padding and after 0.0075), so this path was discarded.

3. Models

3.1. Models based on category splitting

For this approach after the splitting of the dataset into training and test sets, a model is trained every time only on a particular class of the data (ex. A) and saved. A different model is later trained on the remaining classes obtaining 6 different weights,a specific model structure could be used for each class. At prediction time the correct model is asked to predict according to the "categories" parameter received in input.

Even if at the beginning this model seemed to be the best choice, it has some negative sides:

- It is difficult to optimize and more time consuming since both the parameters of each model must be changed but also a different training for each model must be performed
- It doesn't perform well (for this reason we have developed the second approach later)
- It doesn't generalize enough the task since each model sees only a part of the data corresponding to their category, in case the category isn't kwon a priori and passed as parameter, or the time series to predict is a combination of more shapes, this approach isn't a good idea.

3.2. Models trained on the whole dataset

Since we didn't know what were the categories about and in the task description there was no mention about correlation between time series in the same category ("The time series of each domain are not to be understood as closely related to each other, but only as collected from similar data sources"), so it was obvious to try a different approach with ignoring categories and mixing all the time series together. In this case we tried two different routes.

3.2.1 Pure biLSTM approach

Starting from the model developed by the professor during the computer lab (so a bidirectional LSTM with 2 convolutional layers and a final cropping layer) we've managed to get a decent result (around 0.0075). After we tried to downgrade the first layer with only the same size LSTM and, as expected, the results were worse.

Subsequently we tried to raise the complexity using in cascade two bidirectional layers (the second with smaller complexity), with a convolution in the middle. The results were actually way worse than expected, even in the training part (a trend to overfit very fast, even with large batches).

An improvement was made with substituting the cropping layer (our thought was that was a waste of information going from 200 to 18 discarding the data) with only one dense layer (to adapt the new architecture was used a flatten layer, the dense one and a reshape one) and the results were really good (going to 0.0060).

Our best model using this approach was at the end obtained enlarging the bidirectional LSTM layer to 100 neurons (we've seen that larger numbers used to overfit the data) and, necessarily, adapting the convolutional layer after. Also, a dropout layer (at 0.5) was used before the dense layer, along with an increased batch. The results in the development phase were around 0.0055, we didn't succeed to go below that.

During the last days of the development, we tried other configurations (but obtaining no improvement): with similar results compared to our best model was adding another dense layer (size 256), but without dropout (we've seen that at a certain point the validation loss started to suddenly increase without any explanation).

3.2.2 Seq2seq approach

A second route undertaken was a lightweight yet performing seq2seq architecture with an incorporated attention mechanism. The model's simplicity and impressive performance were noteworthy.

The architecture comprised an encoder LSTM, passing its final state to a decoder LSTM as repeated input. Attention was calculated using the dot product between the decoder's output and the sequences returned by the encoder LSTM. The resulting attention mechanism was then concatenated to the decoder output. A time-distributed layer in the output concluded the model.

Despite not being the most computationally intensive, this model delivered outstanding results. Its swift training speed and generalization capabilities were particularly remarkable, especially given the initial conservative setting of the number of LSTM cells. An enhancement was introduced through the incorporation of a batch normalization layer, effectively addressing overfitting concerns when scaling up the number of LSTM cells. This judicious addition further optimized the model's performance without compromising its efficiency.

4. Evaluations on test set

The assessment of model performance on the test dataset involved two distinct approaches. Firstly, standard Mean Squared Error (MSE) and Mean Absolute Error (MAE) calculations were employed to quantify accuracy. Subsequently, random predictions were visually compared against the true values of the future to gain insights into the models' strengths and weaknesses.

Notably, the experimentation with LSTM nodes revealed an interesting trade-off: augmenting nodes enhanced the models' capacity for oscillation in predictions. However, this often came at the expense of losing sensitivity to the overall trend exhibited by the data. This observation suggests that, under appropriate preprocessing conditions such as scaling and detrending, further increasing LSTM nodes might be a viable strategy.

On the other hand, the 1D convolutions demonstrated proficiency in capturing long-range effects such as trend and seasonality. However, optimal results were achieved when these convolutions were positioned before the LSTM layers. When used afterward, a significant degradation in performance was observed. This finding underscores the importance of layer ordering in the model architecture and the nuanced interplay between different architectural components. Overall, these insights provide valuable guidance for refining and optimizing the models based on their specific capabilities and limitations.

The ensemble model performs poorly on some classes, while others get better results, the total performance is given by the average on all the 6 classes. A side effect is represented by the fact that when the model is evaluated on the hidden dataset we don't know which classes perform worst and so which classes we should optimize.

The generalized model with the biLSTM approach has achieved the best performance and the slowest training among all the models, both the double dense one and the dense-dropout model. In our opinion further implementation would consist of using more convolutional layers (reducing bottlenecks in our proposed model) and expanding the final dense net. Also concatenating multiple nets (to follow seasonality) could improve the performances.

While the seq2seq model may not have achieved its utmost potential due to limited experimentation and fine-tuning, its notable strengths in terms of speed and impressive results suggest that it was a promising direction. The model's efficiency and excellent outcomes obtained during initial trials hint at its potential as a preferred choice. Further exploration and refinement could likely unlock its full capabilities, making it a viable candidate for the task at hand. The balance of speed and performance achieved with the current configuration underscores the model's promising nature, making it a compelling avenue for future optimization and experimentation.

5. Conclusions

The model that showed the best performance was the one with bidirectional LSTM trained on the whole dataset, but the results obtained did not show such a big improvement to clearly state that this solution is superior in general. Further development of all the models presented could lead to a breakthrough in performance that can ascertain the superiority of a certain approach over the other.

6. Contributions

Forlivesi Matteo : worked on data preparation, seq2seq models
Franzè Lorenzo : worked on ensemble method
Lupatini Alessandro : worked on data preparation, generalized model
Bindi Niccolò : worked on generalized model

6.1. Notebooks

Forecasting_using_seq2seq_models : Forlivesi Matteo (various versions exist corresponding to variations of the model core but were not submitted)
Pure_biLSTM_model : Lupatini Alessandro
Forecasting_notebook_lorenzo: Franzè Lorenzo
Forecasting_niccolo: Bindi Niccolò