Report -Time series

Gayathri Yenigalla
Harshith Reddy Suram

We set out to find the best effective models to unravel the mysterious patterns concealed in our datasets as we ventured through the maze of time series data analysis. Our journey took shape throughout the development and examination of unique models, each with unique potential and drawbacks.

The first model yielded a Mean Absolute Error (MAE) of 2.62 and served as a benchmark. Inspired by our results, we continued and created a simple machine learning model with a thick layer. Even with our best efforts, the model's MAE of 2.71 was somewhat higher. Even while the rise in mistake was not great, it made us reconsider our strategy and look for ways to make it better in the next version. Because the thick layer model flattened the temporal context of the time series data, its performance was insufficient. Furthermore, the poor performance of the convolutional models was caused by the way they treated every data segment in the same way, which upset the sequential order. These difficulties highlighted how crucial it is to preserve the temporal structure to make accurate predictions.

We concluded that processing time series data is better suited for Recurrent Neural Networks (RNNs). The capacity of RNNs to incorporate data from earlier time steps into their present decision-making process is one of its key features. With the help of this feature, the network can identify patterns and relationships in sequential data. As a kind of memory, the RNN's internal state holds onto data

from earlier inputs. Consequently, sequences of different lengths may be efficiently modeled by the network. Still, many people believe that the basic Simple RNN model is too simple to be truly useful. Interestingly, this model has a major flaw in that it constantly performs the worst out of all the models, as shown by graphical displays.

The infamous "vanishing gradient problem" causes Simple RNN to have practical difficulties, especially in deep networks, even if in theory it can store information from all prior time steps. Because of this problem, the network is essentially untrainable. More sophisticated RNN variations, such as the GRU and Long Short-Term Memory (LSTM), were created in response and included into systems like Keras. Our testing showed that out of all the models, the straightforward GRU model produced the best results. This advantage stems from its ability to capture long-range relationships in sequential data with computational efficiency—a skill that sets it apart from LSTMs.

The final effort was an attempt to combine an RNN and a 1D convolutional model. Regretfully, the hybrid technique produced a greater Mean Absolute Error (MAE) of 3.17, which can probably be attributable to the intrinsic limits of the convolution in terms of maintaining the information's sequential sequence. Considering our results, we recommend against using basic RNNs for time series analysis because of their vulnerability to the vanishing gradient problem and their poor ability to capture long-term relationships.

Alternatively, we suggest looking into more sophisticated RNN architectures made expressly to tackle these problems, including LSTM and GRU. Although LSTM is commonly used due to its efficiency in processing time series data, our research suggests that GRU might yield more beneficial outcomes. Fine-tuning hyperparameters including the number of units in stacked recurrent layers, recurrent dropout rates, and the use of bidirectional data presentation is crucial to improving the performance of GRU models. Prioritizing RNN architectures designed for sequential data processing is also a wise move. Our results imply that the 1D convolution and RNN combination did not yield the best outcomes. Convolutional methods are less useful for time series data analysis since they tend to mess with the information's sequential order. For time series modeling, it is therefore advised to concentrate on RNN architectures that are tuned for sequential data to get better performance.