Magnet Solution Details

ammarali32

February 2021

1 Introduction

MagNet challenge goal is to design a model for predicting the Disturbance Storm-Time Index for the current hour and the next hour. My solution has the same structure of the baseline model with some optimizations which i will go through in this report.

2 Model

My first enhancement step is to make the a model has the best performance on the validation without any changes to other factors. The proposed model structure consists of a Bidirectional LSTM connected to a Bidirectional GRU. (BI-LSTM-GRU) is a well known combination for time-series and text recognition problems. followed by a 3 dense layers connected to the GRU through a flatten layer. Then another dense output layer. number of neurons for each layer will be explained later.

3 Data

Second enhancement was related to the data. As it was shown in the baseline the some feature are highly correlated. Even though i used all features except (satellite_positions data). and left the decision of what features to use and what to ignore to the Neural Network and that really enhanced my CV. The imputation of missing values is a mystery.I tried different methods The one minimized my loss was a simple imputer with most frequent strategy. most frequent in the nature phenomena some how reasonable. But i thought the using iterative imputation with Bayesian Ridge estimation and most frequent as initial strategy should works better. Unfortunately that was not correct on my loss but i have no idea how it will perform on the testing data. i didn't send and submission with iterative imputation. Scaling is an important factor as well anyway standard scaler was the best on the validation set. a combination of standard scaling and power transformer was better on the validation but worse on the leader board. I also tried to use a normalizer because it is logical since

the activation layer of the LSTM is "tanh" but it also made the validation loss bigger. One of the most steps is to take the testing and validation sets from the head of the data and not from the tail as it was on the baseline. Because this data are not connected in time to the training data. But taking it from the tail makes it connected in time so the evaluation will not give appropriate results. i used 6000 for testing and the same data for validation + 4000. To summarize: i used standard scaler, simple imputer most frequent strategy and 29 features with std and mean. 6000 samples for testing and 10000 samples for validation (including testingset).

4 Model parameters

As i mentioned, I've used 29 features the nearest 2 power is 32. multiplied by 2 makes it 64 and i used 128 timestamps. The summation gives 192 which is the number of neurons used in my LSTM layer as beginning.For GRU i used 192*2 at first then changed to 192 * 3. Because it showed better performance on the validation loss. For dense layers number of neurons (192/2 = 96,128,64) in order. Then the output dense layer of 2 neurons. The last model was just a bigger model so i multiplied number of neurons for recurrent layers by two. Making LSTM neurons equals to 2 * 192 and GRU neurons equals to 192 * 6.

4.1 Hyper Parameters

Starting Learning rate equals to 0.0001 using ReduceLROnPlateau schedule with factor equal to 0.3 and patience 4 steps monitoring the validation loss. The loss was MSE loss. Batch size equals to 128. Almost all my hyper parameters was as standard and it is a good field to study because it may gives better performance.

4.2 Initialized weights

One of the most important points is to know how to initialize the weights for the training. There are multiple ways to do so. by running the training with different seed values or by redefining the model multiple times and run the training. or using well-known initializers on Tensorflow. What i have done is to define a dummy model before my model. The dummy model is bigger in size and actually will not run on time limit specified by the competition. The idea of the model is just to initialize my model with different weights values. anyway we can say the if we run the training maybe 1000 times maybe less or more it will get these initialized weights but that will take a lot of time. I guess the last idea was the reason of winning the competition. It reduced the loss from 147.9 to 146.9.

5 What about better performance

Of course i have multiple ideas to enhance the solution. i will mention some of them

1. Design a GANs model to generate new data (it will be better to generate data that is similar to the data which has huge dst absolute values because it was considered as outliers). Then train the model on the generated data. After that use the weights as a starting point for the training (transfer learning).

2. Data augmentation Also an important factor to enhance the model and minimize the loss see[1]

3. Use of satellite_positions data not as features but for another model for imputation or to build the GANs model or as a multi head model.

4. If it showed that some how Gaussian distribution is related to the problem then power transformer should be used (i have a strong feeling that it should be used).

6 Conclusion

Finally i would like to thank you for the interesting competition i really enjoyed to be part of it. And hope to see more competitions coming soon)).

References

[1] https://github.com/uchidalab/time_series_augmentation/blob/master/example.ipynb