Machine Learning in Computational Engineering

ME 343

Forecast the geomagnetic storms

Author:
Thibaut Badoual

Winter 2021

1 Introduction: Background and previous investigations

1.0.1 Forecasting the disturbance storm time index

The earth is a gigantic magnetic dipole. In its outer core, the motion of convection currents of a mixture of molten iron and nickel generates incredible electric currents, themselves originally generated by Maxwell's laws of a magnetic field. Much more than guiding a compass or helping migrating birds move, the Earth's magnetic field acts as a vital shield against the high-energy current of particles released from the corona of the Sun, commonly known as the solar wind.

However, during solar flares, this fragile equilibrium is experiencing major disturbances due to an exceptional release of plasma from the sun. A very efficient exchange of energy from the solar wind into the space environment surrounding Earth occurs creating a major disturbance of Earth's magnetosphere: a geomagnetic storm.

During these storms, the currents in the ionosphere, as well as the energetic particles that precipitate into the ionosphere add energy in the form of heat that can increase the density and distribution of density in the upper atmosphere, causing extra drag on satellites in low-earth orbit. The local heating also creates strong horizontal variations in the in the ionospheric density that can modify the path of radio signals and create errors in the positioning information provided by GPS. While the storms create beautiful aurora, they also can disrupt navigation systems such as the Global Navigation Satellite System (GNSS) and create harmful geomagnetic induced currents (GICs) in the power grid and pipelines.

One index exists to measure these disturbances: the Disturbance Storm-Time Index, or Dst. In this project, the task will be to **forecast Dst in real-time to help satellite operators, power grid operators, and users of magnetic navigation systems prepare for magnetic disturbances**.

1.1 A brief literature review

It has been known since the work of Burton et al. [1975] that the Dst index can be well modeled using the solar wind as input. The Burton method specifies the change in Dst due to a driver term, function of the interplanetary Ey electric field, and a decay term, which gives a constant decay rate of about 7 hours. This model has proven to be very influential particularly due to its simplicity. Another important empirical model used to predict Dst is the Nonlinear Autoregressive Moving Average with eXogenous inputs (NARMAX) methodology firstly developed by Billings et al. [1989]. This methodology builds models by constructing polynomial expansions of inputs and determines the best combinations of monomials to include in the refined model by using a criterion called the error reduction ratio (ERR). However, these models are only able to predict the Dst for a very near future, of the order of the hour.

Thus, over the past three decades, several models were proposed for solar wind forecasting of Dst on a larger time scale, including empirical, physics-based, and machine learning approaches. Probabilistic methods have also been tried. For example, M. Chandorkar et all [2017] proposed a technique for probabilistic forecasting of Dst, using Gaussian Process Regression methodology to construct autoregressive models for Dst.

Recently, J. A. Lazzús and all [2017] developed a hybrid technique that combines an artificial neural network with a particle swarm optimization (ANN+PSO) to predict the Dst. The results reveal that the hybrid algorithm is a powerful technique for forecasting the Dst index a short time in advance like t + 1 to t + 3 (R from 0.98 to 0.90). However, t + 4 to t + 6 predictions become slightly more uncertain (R from 0.86 to 0.79).

While the ML models generally perform better than models based on the other approaches, there is still room to improve, especially when predicting extreme events.

1.2 Dataset used

The data is composed of solar wind measurements collected from two satellites: NASA's Advanced Composition Explorer (ACE) and NOAA's Deep Space Climate Observatory (DSCOVR). Live time data are available on the nasa website https://www.swpc.noaa.gov/products/real-time-solar-wind. However, for this project, we will only study a database already created regrouping the nasa data of 2019.

The objective of this project not being to prepare data, we have used here a code developed on the site of Kaggle, easily found in the following link https://www.kaggle.com/arashnic/soalr-wind. In the code provided in appendix, the whole first part is made of this code, the contribution of this project lies in the integration and optimization of a DNN algorithm to predict at time t and t+1 the Dst.

Our input data base will thus consist of 17 columns gathering the value of the DST index at time t and t+1 as well as the mean and standard deviation per hour of the solar wind data. The description of each feature is detailed in the appendix.

2 Modeling and results

2.1 Algorithm selected

For our study, we used an artificial recurrent neural network architecture, a **Long Short-Term Memory Model (LSTM)**. Unlike standard feedforward neural networks, LSTM has feedback connections and can not only process single data points but also entire sequences of data. For example, these models works really well with a video database or a timeseries. Indeed, LSTM models are capable of storing information over a period of time. In other words, they have a memory capacity, as the meaning of the abbreviation LSTM, Long Short-Term Memory Model, implies.

This capacity is extremely useful when we deal with time series or sequential data. With an LSTM model, we are free to decide on which time range the algorithm will use to predict time t+1 or how much information the algorithm should store at each time.

The first layers of our final model will then be LSTM layers completed by "Dense" type layers. The ratio and the numbers of these two "layer type" will be discussed later. The optimization method chosen is the Adam one coupled with a L2 regularization penalty. A "relu" activation function is used for every layers except for the last one where a "linear" activation function is preferred. All these choices are summarized below.

Table 1: Main characteristics of the algorithm selected

optimizer method	adams
activation function	relu
output activation function	linear
regularization	L2 regularization penalty

Many other parameters had to be studied for the implementation of the algorithm described above. This will be the subject of our next parts.

2.2 Key parameters for this time series database

2.2.1 LSTM and Dense Layers

The usual way to implement a LSTM model is to create one input LSTM layer followed by one or more Dense type layer. In this section, we are interested in the number of typical LSTM layers to implement to optimize the performance of our model. The total number of layers is set here to 10, we will see later why.

Table 2: RMSE results of different model with the same depth (10 layers) but a number of LSTM layers different

Number of LSTM Layers	Loss	Validation Loss
1	271	290
2	278	289
5	338	339
8	331	343

As we can see on the table above, increasing the number of LSTM type layers doesn't significantly decrease the loss, or even increases it. For this reason, we set the first two layers as LSTM type ones, the other will be Dense type.

2.2.2 Timesteps

As mentioned before, one of the strength of the LSTM model is to give the opportunity to determine the number of timesteps in the past used to predict each step at t0 and t1. Our data is aggregated by time, so timesteps is equal to the number of hours we want to use for each prediction.

Let's try to follow a physical reasoning in order to optimize the choice of the timesteps. As we can learn from the literature, geomagnetic storm often occurs after solar flares. Or, the solar wind takes 4 days to reach the earth. However, with such a time step, the results not only make no sense but also diverge. Indeed, take a too wide window to predict the time t+1 leads to hide the interesting patterns of our date. Moreover, what makes sens physically is to choose the solar flare duration and not the time it takes to the wind to reach the Earth.

Guided by our physical sense, we thus decided to choose a timesteps equal to 12 hours, the maximum solar flare duration. Immediately, the improvement of our performance is unquestionable, our error is almost divided by two as exposed below.

Table 3: RMSE values for different timesteps

Timesteps (hour)	Loss	Validation Loss
32	355	387
100	inf	inf
12	173	250

2.2.3 Batch size

Again, one key advantage of the LSTM models is to be able to determine the number of samples to process before the parameters of a model are updated. As before, we realized a quick study to figure out the best batch size to set. Our preliminary study concluded that with a too low batch size (10 and below), the risk of overfitting is too important and the best balance found between accuracy and rapidity was around a batch size equal to 500.

The possibility to add a batch normalization in each layer has also been studied, to fight against internal covariate shift and make our DNN faster and more stable (through normalization of the layers by re-centering and re-scaling). However, no improvement has been noticed despite a higher calculation time.

2.3 Tuning of the hyper-parameters

2.3.1 Number of epochs

The number of epochs determines the number of complete passes our model takes through the training data. We used here for our final training 50 epochs, the results converging afterwards. For more precision, this number could have been increased but this is of little interest in the context of our study. For the tuning of the other hyperparameters, we only used 5 epochs, for the sake of efficiency and simplicity.

2.3.2 Number of neurons and deep vs wide

A neural network could either be wide or deep i.e., with a large number of neurons or a large number of layers. In our case, we choose to implement a deep DNN. Indeed, deeper networks are essentially a stack of many important features. To predict the Dst index, we do not need many different possibilities stored (in the different neurons) but a time hierarchy provided by the layers.

After some tests, the number of layers was fixed at 10. A higher number creates an overfitting. The number of neurons in each layer was also the object of a study, the most important factor being the number of neurons in the LSTM layers. The final model is presented below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_82 (LSTM)               (None, 12, 30)            5520
_________________________________________________________________
lstm_83 (LSTM)               (None, 20)                4080
_________________________________________________________________
dense_537 (Dense)            (None, 20)                420
_________________________________________________________________
dense_538 (Dense)            (None, 20)                420
_________________________________________________________________
dense_539 (Dense)            (None, 20)                420
_________________________________________________________________
dense_540 (Dense)            (None, 20)                420
_________________________________________________________________
dense_541 (Dense)            (None, 20)                420
_________________________________________________________________
dense_542 (Dense)            (None, 20)                420
_________________________________________________________________
dense_543 (Dense)            (None, 20)                420
_________________________________________________________________
dense_544 (Dense)            (None, 40)                840
_________________________________________________________________
dense_545 (Dense)            (None, 2)                 82
=================================================================
Total params: 13,462
Trainable params: 13,462
Non-trainable params: 0
```

Figure 1: Summary of the model

2.3.3 Dropout and stateful

We mention here for purely informational purposes two parameters of the model which are intended to be optimized but which have not been carefully studied in this project:

- **dropout**: regularization by randomly "ignoring" a dropout fraction of a layer's neurons during each pass through the network during training, so that no particular neuron overfits its input.The value chosen in our model is 0.4.

- **stateful**: determines whether the model keeps track of the historical data that its seen within each batch. Since each sample within a batch encodes the entire sequence we care about, we can set this to False.

2.3.4 Learning rate and L2 regularization factors

This section aims to realize a tuning of two hyper parameters: the learning rate and the L2 regularization factors. We ran our model over two ranges of these two parameters in order to figure out the best couple. A first tuning was realized over a large range before a second one was ran over a finer interval. The results obtained are as follows:

- learning rate = 0.0005
- reg = 0.005

2.4 Code validation and benchmarks

To validate our code, we used as metric the root mean squared error. Before any training, we split our data into two datasets: a train/validation dataset of 6000 rows and a test dataset of 3000 rows. This allowed to compute accurately the RSME of our algorithm on a database that the latter did not know.

2.5 Results

.

Finally, our final run gives the following results:

- model loss = 143.168
- validation loss = 247.881
- test loss = 226.2922

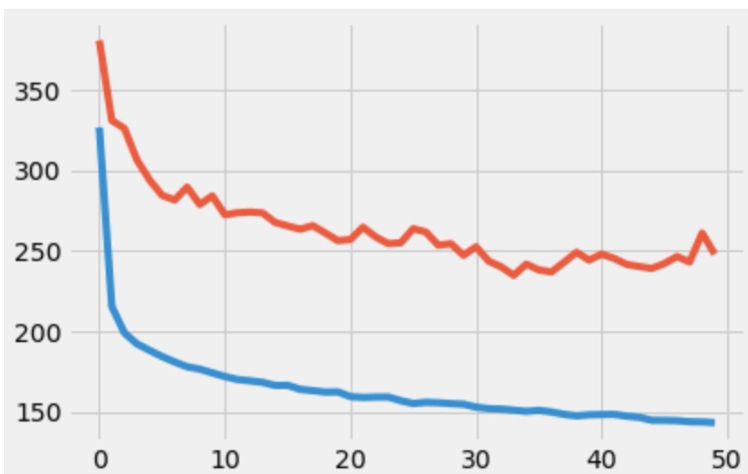

Figure 2: Results - the validation loss is the red curve - the training loss is the blue curve

2.6 Limitations and future perspectives

First, our dataset doesn't take into account the satellite positions even though all the data is available online. Or, the earth magnetic field varies in the space, not taking into account this variation can lead to an intrinsic error independent of our model.

Moreover, one of the major limitations of our model is simply the fact that we are trying to predict an exact value even though we are not interested in the latter. What we want to know above all is whether a magnetic storm is approaching the earth. The solution that could be developed in further studies could be to implement a classification model (with a binary cross entropy loss function for example), using the classification developed by Tatiana Podladchikova et al. (2018). They found that we can define thresholds for moderate, intense and extreme geomagnetic storms around –80 nT, –150 nT and –330 nT. The only thing to code is then a new feature defining the "class" of a geomagnetic events.

3 Conclusion

We succeed in this project to implement a LSTM algorithm to forecast the Dst index. The motivations behind the prediction of such an index are multiple. In addition to predicting more and more accurately the Dst, which is an important space weather parameter since it is the main indication of magnetic storms, this study can improve our knowledge on which features of the interaction between the solar wind and the magnetosphere are most important in producing magnetospheric activity. Moreover, forecasting a magnetosphere index allows to learn how predictable the magnetosphere is, or, in other words, to what extent the magnetosphere is directly driven by the solar wind.

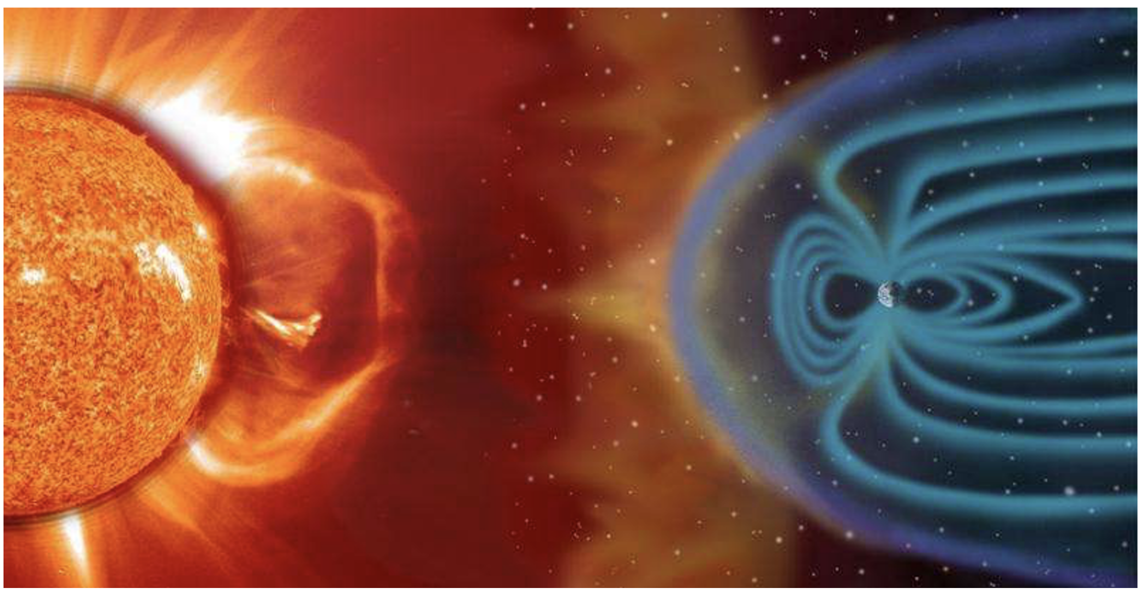

Figure 3: Visualization of the earth's magnetic field and its interaction with the sun, drawing not to scale (https://phys.org/news/2016-03-powerful-geomagnetic-storms-solar.html)

Features

t0	Dst index at the current timestep
t1	Dst index one hour ahead
bx_gse	Interplanetary-magnetic-field (IMF) X-component in geocentric solar ecliptic (GSE) coordinate (nanotesla (nT))
by_gse	Interplanetary-magnetic-field Y-component in GSE coordinate (nT)
bz_gse	Interplanetary-magnetic-field Z-component in GSE coordinate (nT)
theta_gse	Interplanetary-magnetic-field latitude in GSE coordinates (defined as the angle between the magnetic vector B and the ecliptic plane, being positive when B points North) (degrees)
phi_gse	Interplanetary-magnetic-field longitude in GSE coordinates (the angle between the projection of the IMF vector on the ecliptic and the Earth–Sun direction) (degrees)
bt	Interplanetary-magnetic-field component magnitude (nT)
density	Solar wind proton density (N/cm^3)
speed	Solar wind bulk speed (km/s)
temperature	Solar wind ion temperature (Kelvin)
smoothed_ssn	Number of spots on the solar disk

References

[1] R. L. McPherronm R. K. Burton et C. T. Russell : An empirical relationship between interplanetary conditions and dst. *Journal of geophysical research, Volume 80*, 1975.

[2] Consuelo Cid Elena Saiz Judith Palacios, Antonio Guerrero et Yolanda Cerrato : Defining scale thresholds for geomagnetic storms through statistics. *Space Weather Group, Departamento de Física y Matemáticas, Universidad de Alcalá, Alcalá de Henares, Spain.*, 2018.

[3] Balikhin M. A. Billings S. A. Boaghe, O. M. et H. Alleyne : Identification of nonlinear processes in the magnetospheric dynamics and forecasting of dst index. *J. Geophys. Res., 106(A12),30047–30066*, 2001.

[4] Nancy U. Crooker Nick Arge Robert L. McPherron, George Siscoe : Probabilistic forecasting of the dst index. *Institute of Geophysics and Planetary Physics, University of California Los Angeles, Los Angeles California*, 2005.

[5] E. Camporeale Chandorkar, M. et S. Wing : Probabilistic forecasting of the disturbance storm time index: An autoregressive gaussian process approach. *SpaceWeather, 15, 1004–1019, doi:10.1002/2017SW001627*, 2017.

[6] P. Vega P. Rojas andI. Salfate Lazzús, J. A. : Forecasting the dst index using a swarm-optimized neural network. *SpaceWeather, 15, 1068–1089,doi:10.1002/2017SW001608*, 2017.

[7] Anatoly Petrukovich Tatiana Podladchikova et Yuri Yermolaev : Geomagnetic storm forecasting service stormfocus: 5 years online. *J. Space Weather Space Clim. Volume 8*, 2018.