CE 675 Civil Engineering Project: Machine Learning Air Quality

Lige Han

Nov. 30, 2020

Abstract

This project explored areas of improvement in a machine learning air quality prediction model. Implemented improvement includes: Quick data preview based on a Jupyter Notebook extension; visualization of the model data structures and information flows; enhanced variable selection approaches during the data pre-processing stage; and other tools that can be used working with recursive neural network models and data analytics in general.

Introduction

Process-based numeric atmospheric models follow explicit algorithms among spatial grids and temporal steps. They are more interpretable and widely accepted, while at the cost of more stringent standards for data acquisition and higher computational cost. Increasingly intensive availability of meteorological and emission inventory data made it possible to explore the capability of data-driven machine learning models in air quality prediction. For decades, machine learning has been enriched by novel training patterns and corresponding data normalization techniques. LSTM (Long-Short Term Memory) is one of the most popular recursive deep learning algorithms. Apart from feed-forward neural networks or convolutional deep neural networks specializing in regression prediction and computer vision, LSTM is extremely powerful in handling time series data structures. In cases where decision-making focuses on more on quick and precise prediction while disregarding interpretabilities, LSTM machine learning models will become a competitive option.

This project is a continuation of previous contributions from Dr. Fernando Garcia Menendez's Lab. The model is already capable of predicting ozone concentration in the upcoming 24 hours using 30 hours' data prior as input. There is still room for improvement in terms of precision. Another unique feature of the model is it considers relative locations of target points of interest from monitoring stations as its default explanatory variables.

Objectives

1). Easy ways of data visualization

Currently, data visualization is static. Only data at a certain time step can be ploted at a time, which makes intuitive comparison difficult. One of the objectives is to explore a better approach to derive and visualize data dynamically. Once achieved, similar patterns can be utilized under many different scenarios, such as data preview or result comparison.

2). Model structure visualization

As a subclass of recurrent neural networks, LSTM models are widely used in NLP (Natural Language Processing). Visualization of LSTM models taking one-dimensional input data requires abstraction of iterations only over a one-dimension sequence. In contrast, atmospheric observational data always has multiple dimensions, including but not limited to: species, time, longitude and latitude, etc. Additional dimensions add complexity to data pre-processing and model structure visualization as well. A graphic representation of the data structures and in-model information flows will help in understanding the model structure and provide more transparency.

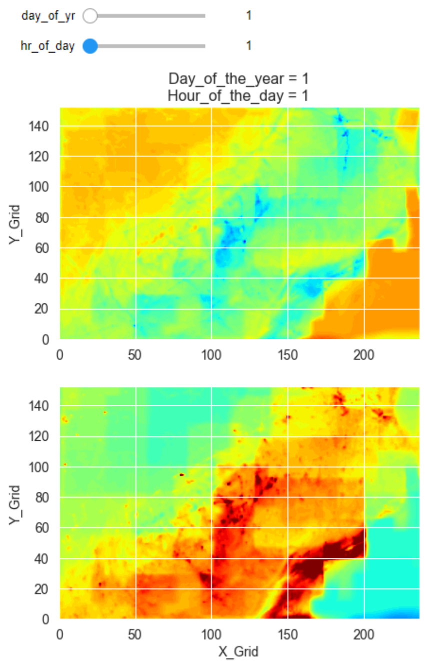
3). Feature importance identification
Current version of the model takes in features (explanatory variables) that are more logically relevant to ozone generation and removal. Even if in most cases, presence of ozone precursors and favorable meteorological conditions are dominant features affecting ozone concentrations. Chances are that, under certain circumstances, the learning and forgetting mechanism of the LSTM model will fail to capture the system dynamics due to stable emission rates or weather conditions among selected features. Additionally, as there are 67 emission variables and 41 meteorological variables to select from, it is also computationally infeasible to test all possible combinations to reach the optimal. To eliminate risks of missing out important features, and also provide justification for the feature selection, a quick cherry-picking process will be added to the model prior to entering the main LSTM forecasting model.

4). Educational objectives
This project involves comprehensive implementation of environmental modeling and computational tools across many platforms. As part of the project, educational objectives are listed below:

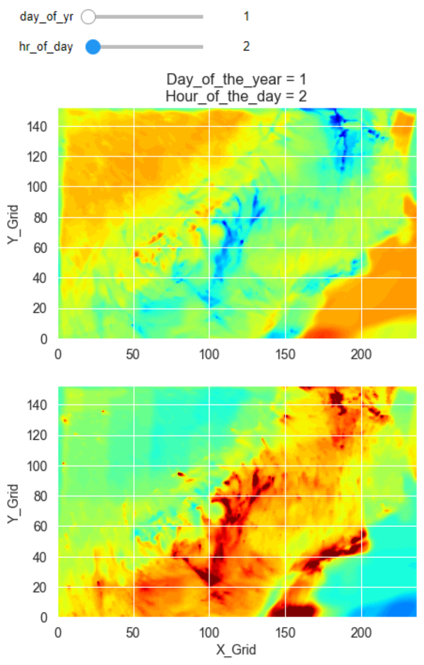
1. Learn basic python syntax and semantics.
2. Learn basic Linux commands and shell scripting.
3. Learn how to configure python IDEs both on local machines and remote servers.
4. Learn how to achieve multitasking on computing clusters.
5. Learn the structures of multidimensional meteorological and emission datasets
6. Explore interactive visualization of observation and prediction data using Jupyter Notebook (a web-based python IDE).
7. Explore visualization of the model structure and its data structure.
8. Optimize feature selections to improve model performance and mitigate computational intensity.
9. Learn academic writing and graphing

Results

1). Interactive visualization using Notebook widgets:
Jupyter Notebook is an open-source web IDE equipped with python kernels and customize extensions. The library 'ipywidgets' is a JavaScript extension, which derives data either from the memory or the disk and displays changes on the web page synchronously when called by the user. In its JavaScript source code, it follows an object-oriented design pattern: 'observer pattern'. This pattern redo the user-identified processes every time a change occurs in the GUI of the web page. As is shown in figure 1, the user can specify which day of the year and which hour of the day's data he/she wants to see by drawing the sliders above. Similar patterns can be adopted under other scenarios. For instance, in data analytics, the user can dynamically pull out data as specified and make changes to it, and eventually display the results using sliders or drop-down menus, etc.



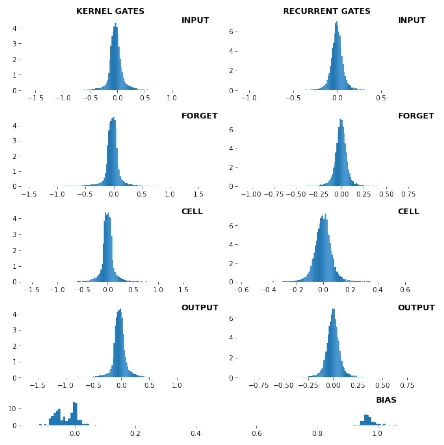
(a) 12AM on Jan. 1st



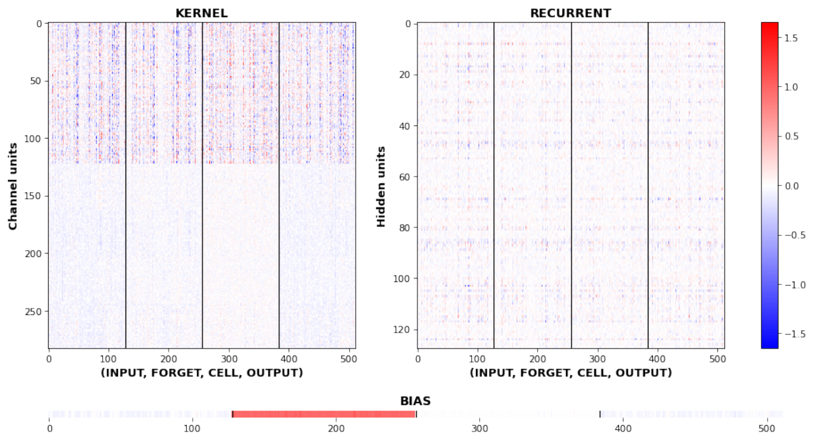
(b) 1AM on Jan. 1st

Figure 1: Example of Notebook interactive widgets

2). Data structures and visualization of the LSTM model:
Keras is a popular open-source deep learning library under the Tensorflow framework with many contributors perfecting its capabilities. Other than its built-in methods that show basic model attributes, third party sub-libraries help in visualizing these attributes. Figure 2 represents the weight and bias distribution of all components among both forward kernel and backward recurrent training units. For the current version of the model, weights are all normally distributed around the mean value of 0. More efforts can be invested in this part.



(a) Weights



(b) Bias

Figure 2: Model weights and bias

The pipelines from high-dimensional atmospheric data to input LSTM model batches can be difficult to interpret. 3-D models help visualize the data structure and the information flows from large high-dimensional tensor blocks and within hidden layers of the LSTM model. As shown in the figure 3, each tensor block (cube) represents one day's data of one species: horizontal axes represent the longitudinal and latitudinal coordinates, and the vertical axis represents the temporal dimension of 24 hours. Data extraction derives temporal arrays at the same horizontal coordinates over all tensor blocks. After extracting arrays denoted by red dots in Figure 3, sorted data was fed into the main LSTM model, as shown in Figure 4.

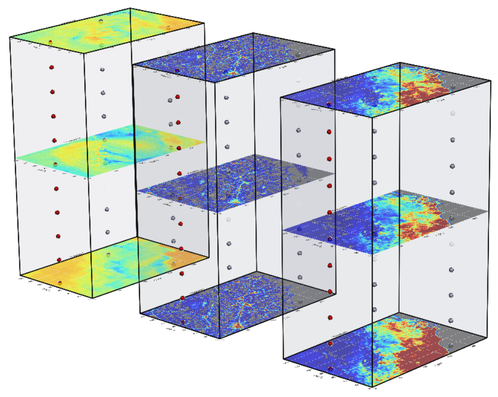


Figure 3: Data structure of explanatory variables

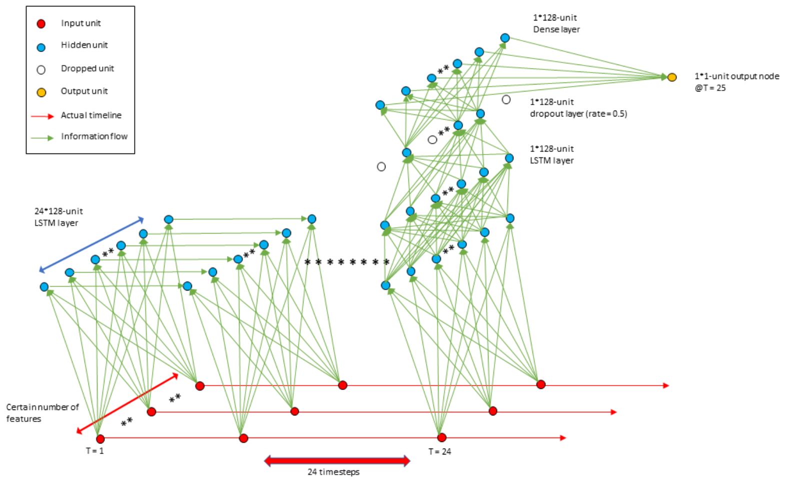
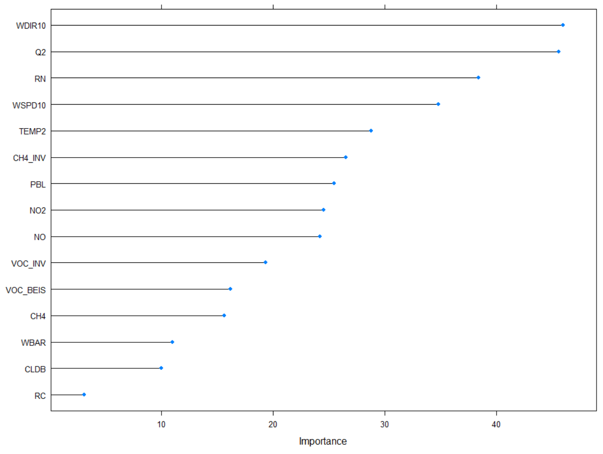


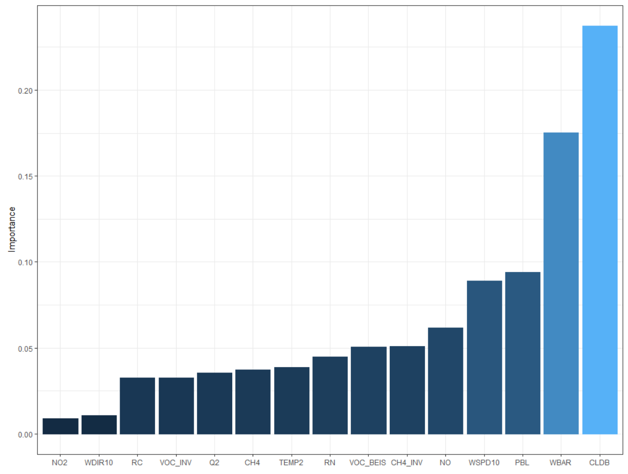
Figure 4: LSTM model information flows

At each time step, a batch of input nodes enter the first fully connected hidden layer via the input gate. An output gate with activation functions representing the 'short term' memory transports information from the current hidden node to the next input layer. Long term memory information flows from the first hidden layer forward is controlled by the forget gate. For simplicity, only the long term memory is shown in the figure as horizontal parallel lines. After such iteration reaches the final batch of input layer (T=24), the model activates traditional back-propagation dense and dropout layers to make the final prediction of the target response variable at the consequential timestep (T=25).

3). Feature importance identification:
Observation data netCDF files contain 67 emission variables and 41 for meteorological variables. Genetically, it is the best to include as many eligible variables (excluding those with unreasonable data) as possible. However, this approach will dramatically increase the computational cost while not improving the model performance to the same extent. Conducting sensitivity analysis and selecting features with the highest importance scores prior to starting the main LSTM model training stage is more cost-effective. To simplify the screening process, data was organized on an hourly basis with no connection among prior or latter timesteps as is in the main LSTM model, and only a gridded area near Raleigh, NC is used for training and testing (refer to code for details). Both a random forest and a neural network model were trained for evaluation. The random forest uses 'dropout' iterations to score variables with variation dealt to the model output when they are ruled out from the model. And the single layer feed-forward neural network employs Garson's algorithm to identify variable importance. A testing combination was used by both algorithms, and results are shown in Figure 5.



(a) Importance scores from Random Forest



(b) Importance scores from Neural Networks

Figure 5: Model structure

Random forest and neural network algorithms yielded inconsistent results. Possible reasons may include: small studied area (only near Raleigh); short tested temporal scale (one day); generic difference between two algorithms. More investigations can be put into this part.

Importance weights from the random forest model is tested in the main LSTM model. RMSE of the testing dataset was used as the error metric the in evaluation. Results of different variable combinations are shown in Figure 6. Detailed combinations are listed in Table 1. Note that combination A through F are considered as 'relatively' important variables as specified in the random forest selection model, and combination G and H are tested as 'less' important variables for comparison. Among 'relatively' important variables, once the most important variables are included, there will be no statistically significant improvement by adding other variables. It is noteworthy that including 'less' important or seemingly irrelevant variables in the combination undermines the predication accuracy of the model. These results provide justification and confidence in the feature selection process. Similar approaches can be applied to other multi-variable machine learning models to both reduce the computational cost and improve the model performance.

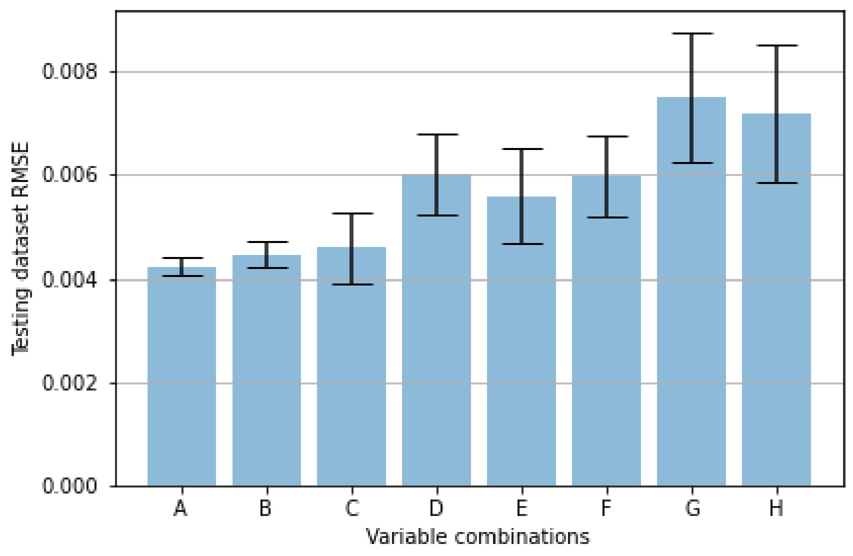


Figure 6: Model evaluations under different explanatory variable combinations

Variable combinations	Emission variables	Meteorological variables
A	NO	TEMP2
B	NO2	WSPD10
C	NO, NO2	WSPD10, TEMP2
D	NO2, VOC_INV	TEMP2, RN
E	NO, NO2	WSPD10, TEMP2, RN
F	NO, NO2, VOC_INV	PBL, Q2, TEMP2, WSPD10, WDIR10
G	ACET, ACROLEIN	PRSFC, USTAR
H	ACET, ACROLEIN	PRSFC, USTAR, WSTAR, ZRUF, HFX

Table 1: Variable combinations

***Emission variables:** NO: Nitric oxide; NO2: Nitrogen dioxide; VOC_INV: Total VOC excluding biogenics; ACET: Carboxylic acid derivatives; ACROLEIN: Acrolein

***Meteorological variables:** TEMP: Temperature at 2m; WSPD10: Wind speed at 10m; WDIR10: Wind direction at 10m; RN: Nonconvective precipitation; PBL: Height of planetary boundary layer; Q2: Mixing ratio at 2m; PRSFC: Surface pressure; USTAR: Cell averaged friction velocity; WSTAR: Convective velocity scale; ZRUF: surface roughness length; HFX: Sensible heat flux

Future Scopes

Jupyter Notebook is powerful in handling python scripts. It will be promising to discover more extensions both on the Notebook and other IDEs. VS code is another lightweight IDE integrated with many powerful extensions, such as remote SSH that can connect to remote servers. Once local python kernel can derive data directly from the remote machine through SSH, error-prone operations and the redundancy of switching between platforms can be eliminated.

LSTM air quality models have complicated structures. It will be worthwhile adjusting hidden layer arrangements to further improve model performance. Efforts can also be put into better understanding of model attributes and analyzing their impact to the output.

The current variable selection process adopts random forest algorithms, which neglect temporal correlations and treat input data points as discrete instances. Future efforts can be put into developing simple recursive models to align with the main LSTM model structure. Also under circumstances where atmospheric model input has unreasonably high number of variable combinations to be tested thoroughly, the above variable selection process can be adopted.

Github repository

Machine Learning Air Quality

Acknowledgement

I would like to thank Dr. Fernando Garcia Menendez for offering this opportunity for me to apply and expand knowledge in environmental modelling under real-world scenarios, and for coordinating resources during the project. I would also like to send my appreciation to Mr. Muhundhan Mohan and Dr. Satish Vutukuru for their strong and patient assistance along the way.

References

[1] Freeman, B. S., Taylor, G., Gharabaghi, B., & Thé, J. (2018). Forecasting air quality time series using deep learning. Journal of the Air amp; Waste Management Association, 68(8), 866-886. doi:10.1080/10962247.2018.1459956

[2] Zheng, Y., Liu, F., & Hsieh, H. (2013). U-Air. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '13. doi:10.1145/2487575.2488188

[3] Chang, Y., Chiao, H., Abimannan, S., Huang, Y., Tsai, Y., & Lin, K. (2020). An LSTM-based aggregated model for air pollution forecasting. Atmospheric Pollution Research, 11(8), 1451-1463. doi:10.1016/j.apr.2020.05.015

[4] Beck, M. W. (2018). NeuralNetTools: Visualization and Analysis Tools for Neural Networks. Journal of Statistical Software, 85(11). doi:10.18637/jss.v085.i11

[5] Roesch, I., & Günther, T. (2018). Visualization of Neural Network Predictions for Weather Forecasting. Computer Graphics Forum, 38(1), 209-220. doi:10.1111/cgf.13453