Machine Learning Air Quality Future Scope

Lige Han

Apr. 20, 2021

The primary goal of this project: optimization of new monitoring station locations

The naive solution is to iterate through all possible new locations and pick the ones that bring the greatest improvement to the error metrics. The beta is more of a framework, and is tested over only 2 variables (1 emis and 1 met) and a short duration of time. It is still slow even to generate the results for 40 locations (around 6 mins for each location, as shown in the csv recorder). It can be concluded that it is computationally infeasible to use brutal force to reach the optimal with the current build.

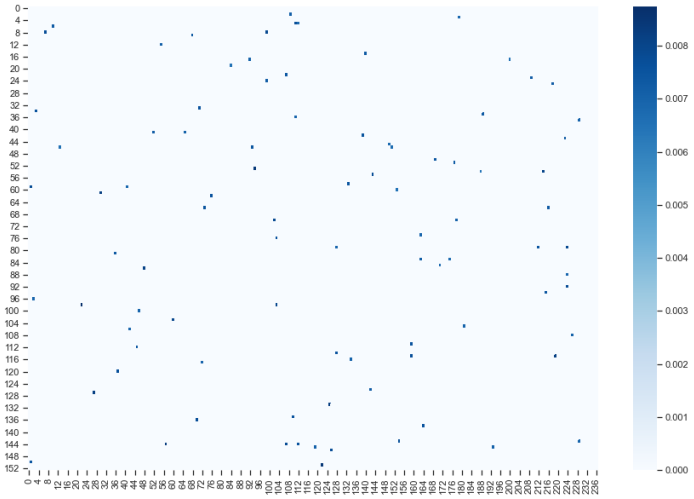

Figure 1: Heatmap of random location performances (RSME)

Potential areas of improvement:

1). Location screening
Project the netCDF data on a basemap on ArcGIS or other GIS platforms, manually exclude locations that are not qualified to be chosen as monitoring stations (lakes, rivers, ocean, natural reserves, etc.). Find the relative coordinates of such locations and exclude them from the optimization iteration. This will reduce the running time proportionally.

2). Pipeline rework
The beta calculates each location's performance from start to finish of the pipeline, which may include repetition of certain steps that are time consuming, such as data extraction and splitting and model training. When a model is trained, it is possible to save and reuse it with the keras '.to jason' function. Cautions need to be taken when reusing the model, because the input data size and format have to be consistent with those of the original training data. Exceptions might be thrown otherwise.Therefore, it is worth further

study on how to save and reuse the LSTM model for each input data configuration to reduce the running time.

3). Test on the PC, do the hard part on the server
Even after the location screening and the pipeline rework. A PC might still not be fast enough to produce desired results under realistic conditions, nor does it have the disk memory space to contain the raw data. As is mentioned in the project final report, some IDEs have extensions that allow direct connections to remote servers through ssh that can save the redundancy of transporting data. Testing of a more variant datasets is possible once such connections are achieved. However, if the model is to be trained over a longer time span or a large number of variables, it is inevitable that the main work needs to be done on a computing server. It will be necessary to configure the python environment on the server, and the testing code will also need to be revised into working code.

Use ArcGIS to provide better illustrations of netCDF geospatial data extraction

ArcGIS has many built-in models and tools that can easily handle geospatial data including netCDF without breaking pipelines into explicitly written code pieces. ArcGIS uses a python kernel to do the calculations in the background under a module named 'Arcpy'. ArcGIS models also feature extraction of workflows into python code to enable advanced users to revise the process in a finer scope. Unfortunately, the 'Arcpy' module is not open-source and, therefore, unavailable through 'pip', 'conda', or other python package management software. It can only be imported and run within the ArcGIS python kernel, making it hard to be incorporated into our main python pipeline.

However, there are still ways to implement ArcGIS features in this project. In the 'dist predictor' function definition, we calculate the centroid of neighboring monitoring station locations, and include the distances and angles from the centroid to the 3 vertices as relative location predictors. The idea of including distance predictors is referenced from this paper (U-Air: When Urban Air Quality Inference Meets Big Data). We may also want to explain why and how we take these factors into consideration. AcrGIS has a tool 'create TIN' that does the same thing and creates an 'elevation' (pollutant concentration in our case) gradient over the whole 'TIN'. Also it is possible to use other ArcGIS default functions like 'Geometry Calculation' based on 'TIN' points to present actual locations of the predictors above the TIN layer. Adding a basemap and other elements like north arrow and legend will make better-looking maps.

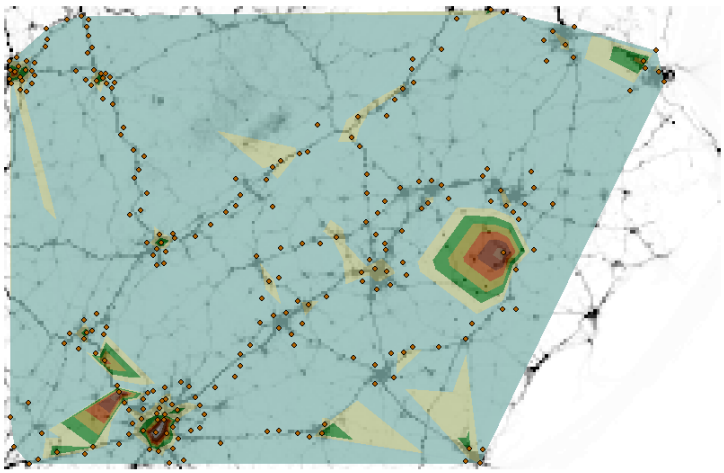

Figure 2: Example of Triangulated Irregular Network for Pollutant Concentration

Note that the use of ArcGIS should not make major changes to the main python pipeline without careful deliberation due to its closed-source nature. Auxiliary illustrations of the workflow is the main goal.

Other thoughts

1). Model structure and parameters
LSTM air quality models take data extracted from 4-dimensional (2D spatial, temporal, and species) netCDF files, and reuniformed into unidimensional time-series. It is essential to showcase the data structure alternation for easier comprehension . Some initial work visualizing the information flow from raw data to input data and from input data to the output has been done in a 3D view. The parameters of the model are yet to be explored to achieve better model performance.

2). Workflow archive and collaboration
Instead of passing python scripts around, using git to archive the workflow will be a better option. Add participants as collaborators in the common repository to work on the project simultaneously. One person should be responsible for the admission of pull requests to avoid conflicts among commits. The current repository could be a place to start from. GitHub