Estimating Fine Particulate Matter in the U.S. Using IDW-based Spatial Interpolation Extension

James Kyle Harrison

Georgia Southern University

Statesboro, GA, U.S.A.

Jh51381@georgiasouthern.edu

*Abstract* — This paper investigates using IDW-based spatiotemporal interpolation methods for the application of air pollution assessment, with the primary interest in this paper being fine particulate matter PM2.5 (air pollution). The choice of the time scale is of high concern when applying the spatiotemporal interpolation methods, and as a result this is further investigated beyond the scope of the implementation created. This document will also detail and explore validation through the leave-one-out-cross validation, as well as the formulation of error measurements in this approach such that we can estimate the population exposure to air pollution, and query the information based off of the interpolation results. Though we were able to create a solution and query results the design does have flaws, thus this document will then explore the findings of this implementation and what steps could be taken to make a more optimized approach to create finer results with less memory block usage and faster run time.

# Introduction

Spatiotemporal interpolation is a technique that plays a critical role in estimating values (of variable interest) at a given time and location in geographic information systems. The general process of spatiotemporal interpolation revolves around creating a model that can create relationships between the variables of interest and other observations that can be found within the data sets. It is important because of its predictive modeling about the state of measures (particulate matter PM2.5 in this case) that we do not actually have the data for. It is a vital tool for geographic information systems to find map data that we could calculate by other means. The reasoning for using this sort of technique can be seen in something as simple as in weather and rainfall estimation [1], and with researchers believing that climate change is causing more climactic events [2], the importance of using these techniques is becoming more apparent.

Although spatial interpolation is proven to work, there are not only various methods to achieve interpolation but also several ways to implement it as well. There are also various forms and methodologies to performing validation as well, and as a result there are quite a number of ways that a model can be designed. To help demonstrate the importance of spatiotemporal interpolation and the philosophies that are taken into consideration when creating a solution, we shall create our own application that allows one to import data, perform one type of interpolation, perform a type of validation, compute the error metrics for the data, and finally query the data. This document describes the creation of a model using an IDW-based extension interpolation method and leave-one-out cross-validation for particulate matter files. This document will also further investigate the reasonings for the methodologies used as well as explaining what could be done better to further improve the model and what one should take into consideration when creating their own model.

# Modeling

2.1. Graphical User Interface and Data Importation

To begin with developing our model we must start with designating the environment. To start, we will be using Python and tkinter for the development of the GUI for the model. Figure 1 shows the user interface in its entirety. The user interface consists of 4 primary features that will be explained one at a time, as the vertical order they are shown in is the order that one should use them. Starting off we must import data for Interpolation. To do so one must click ‘Import Data’, where a prompt showing only .txt files will allow you to select a data set to import. If the file matches the specified PM2.5 file format of id, time, x, y, and measurement then one is able to import their file using pandas. The user will then be prompted to select the Time Domain that they will be using from between Day, Month, and Year. The selection of Time Domain and the design of how one handles the time domain is of critical importance, and although we will discuss the implementation here the importance will be described later. The Graphical user interface, application, Word

Description automatically generatedcurrent selection of how the time domain is handled in this implementation is mostly done through pandas turning the data sets ([year, month, day], [month, year], [year]) into an additional column in the imported file labeled as time. This allows us to take in year, month, and day regardless of what data columns are actually there we can set the missing one’s equal to 1 and pass it along. With the data successfully imported, we are then able to look at the locations through our data visualization, which will populate a map of the entire U.S. using matplotlib. Although the implementation created imports and uses FuncAnimation in particular, there were difficulties creating animations in this model, and thus the points are static in display. The resulting map from the import of pm25\_2009\_measured.txt can be seen in Figure 2.

Figure 1

Diagram

Description automatically generated

Figure 2

2.2. IDW-based Spatial Interpolation

The next step in this model is where the model truly begins to work. As previously described, interpolation is of upmost importance when attempting to create measurements of unmeasured areas using other data, but how does one decide what type of interpolation to use? Amongst all of the interpolation techniques out there, the main methods of importance here are shape function-based reduction, shape function-based, IDW-based reduction, and IDW-based extension methods. To break it down, let’s first by explaining the difference between Reduction and Extension. Reduction is an approach to interpolation where one reduces the interpolation in a regular spatial interpolation case by turning the data into a set of one-dimensional data and then iterating over it and substituting the desired time instance into a similar interpolation to get results, whereas Extension is practically the opposite, where it deals with time as another dimension and extends the problem by adding an additional dimension [3]. Extension (as described) uses much more complex calculations than reduction in order to reach its goal, although it is difficult to tell if this means it will require more time to process. To have a better insight into interpolation as a whole, our approach will move forward with an Extension interpolation. With that settled, we then move on to the other part of selecting our interpolation method: IDW-based or shape function-based. IDW-based interpolation is a distance-based method that estimates values at unsampled locations based on the distance-weighted average of observation points specifically based off of the assumption that observation points closer to the point of prediction are more similar than those away from it [4], whereas shape function-based interpolation interpolates data based on the shape of the structure of the data [5]. The strengths of the two can get meddled when it comes to the data sets that are used in them, but a simple generalization of the implementations is that an IDW approach is a less computationally expensive model whereas the shape-based approach will provide a more optimal estimation for figures. As IDW-based interpolation seems to be a great selection when it comes to data that has a correlation between the value and location [6] and is a more straightforward method in implementation, this will be the approach we move forward with. Thus, we will move forward with IDW-based Extension spatial interpolation.

Once one presses Perform Interpolation (assuming they have imported their data) interpolation may begin by two prompts showing up asking for the number of neighbors and the exponent one would like to select. If the user chooses 0 on either of these selections, they will be given the default testing variables for this model (neighbors: 3, 4, 5, 6, and 7 exponents: 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5). From here the console will say “Performing Interpolation…” and the user must then wait for a length of time. What’s truly going on while one waits is our IDW-based extension algorithm. Our algorithm first starts by using Scipy’ cdist, that allows us to calculate the distance matrix between the data points and locations. From here, we are then able to use the input neighbor count to locate that many nearest neighbors in our distance matrix by using pandas’ argpartition. We are then able to calculate weights for each of these neighbors, and finally the interpolated values for each location are calculated and returns the results in a way that allows one to save the results as a file. The results of this interpolation can be shown in our county\_id\_t\_w.txt file.

2.3. Validation

To ensure that the results are accurate and reliable we must incorporate some sort of validation. The choice between a 10-fold cross-validation and leave-one-out-validation is one of severe importance to the data given. A k-fold crossover is less computationally expensive yet utilizes a less trained data set, whereas the leave-one-out cross-validation is incredibly expensive yet gives better data [7], meaning that it can take many hours to get an output file. The use case for on deciding to use a k-fold cross-validation is understandable; if one is using a large enough data set or are just trying to get results as fast as possible the choice of implementing it is clear, however as we are running this program on a device with 16GBs of RAM and an Nvidia RTX2060, we elected to go for the leave-one-out crossover as we believed that even with the additional time it would take to process, we could complete our task still within our personal time parameters. The problem with this, however, is what could be seen with the creation of file loocv\_idw.txt. The data in the file did not follow the proper format of the columns by not creating the original values column. Even the current version of this file still does not have this, as attempts to add said column failed in the resulting few file generations to try and make this with rather empty columns or incorrect data storage. Although this is the case, the file in its current form is still usable as long as we keep in mind that we are going to have to pull the original numbers for comparison separately.

2.4. Error Metrics

With interpolation and validation completed, we also need to confirm the data via error metrics. Due to not having the original values column in the loocv\_idw.txt, there was a bit of difficult creating this function as we needed to import them separately and fix them to the value format expected, so we had to implement it in such a way that we can do so and also fit these original values to the same shape file size and loocv\_idw.txt. Regardless, we were still able to create a file that allowed us to see the error values. The shape of the Error Metrics file is quite a bit busy to read (error\_statistics\_idw.txt) and large, thus an additional csv (errorStatsReadable.csv) has been added to properly view the results. Looking at the results of our data though, we can find that running this application with 7 neighbors and an exponent of 5 gave the lowest mean absolute error at 5.355295. It seems that overall, the mean absolute error continued to lower as the complexity of the case continued. This can also be seen in fact with all of our error metrics, and thus it is safe to say that in this case 7 neighbors and 5 exponents was the best performing overall. Some of the examples of this pattern can be seen in Table 1 below, where those with the lowest exponent always have the highest MAE.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Stat | N3e1.5 | N3e3 | N4e1 | N5e2.5 | N6e3 | N7e1 | N7e5 |
| MAE | 5.627746383 | 5.573559 | 5.652893 | 5.443735 | 5.403477 | 5.703161 | **5.355295** |
| MSE | 56.42727327 | 56.77956 | 56.30631 | 53.16313 | **52.76574** | 56.22326 | 53.0653 |
| RMSE | 7.511808921 | 7.535221 | 7.503753 | 7.291305 | 7.264003 | 7.498217 | **7.284593** |
| MARE | 1.043786151 | 1.016651 | 1.064534 | 0.987459 | 0.976768 | 1.087963 | **0.963657** |

Table 1

2.5. Querying

Finally, we’ve come to the final part of our implementation, SQL-like querying. With the creation of such expansive data, the use of querying is practically a necessity to filter through the data. The query that we elected to implement was a general query of selecting the id, location, time, and measurement from a specific time (in YYYY-MM-DD format) and between a minimum and maximum measured value. This querying also has its own import when the button is pressed, so that you can select what file to do this querying for. An example of querying the results of county\_id\_t\_w.txt can be seen below in Figure 3.

Text

Description automatically generated

Figure 3

2.6. Complications

Although everything in this model works and is able to produce the required results, there were many difficulties faced along the way. The first major issue faced was the handling of the time dimension. Initially, we take in our import data and the user is asked what time domain to use. The design used fills the missing data fields (day, month, or both) as 1 for the merger into what would be the time dimension, however this resulted in more processing than necessary as this design would then need to be changed into different formats in various methods to properly work. An implementation that could have potentially bypassed this would be to have converted the information into the general YYYY/MM/DD format from the start to minimize the difficulties with handling time, however using the general encoding schema described in the project document should have been enough. Using 1-365 or .1-36.5 to represent every day of the year would make a difference immediately, but between the two described in the document we believe 1-365 would have worked better for our implementation due to its simplistic interpretability. Scaling the information down, however, could give some slight advantages in severely large data sets (requiring multiple years of data), but for the requirements of our implementation 1-365 would have been just fine.

Another complication that was faced is within the selection of using an extension method instead of a reduction method. The datasets used did not seem too large in size initially, but the problem first reared itself when attempting to use the ‘blkgrp\_xy.txt’ file for Interpolation. The resulting error of trying to run this stated that it could not run because it required 964GBs in order to process. Reduction would have allowed for not only faster creation of the solution altogether, but a better understanding of the results initially. Although the results were just fine as they were, additional time needed to be taken to reperform long tasks (such as interpolation, which for loocv\_idw.txt took 6-8 hours each run). Perhaps using a k-fold cross-validation would have also further minimized the time necessary to create results, as the leave-one-out cross-validation is quite computationally expensive.

There were also small issues in things such as animation (as previously described in 2.1) in design, but ultimately these complications are all hindsight observations.

# Conclusion

We have taken a look at spatial interpolation methods and created our own application to import data (of our file structure), select a time domain, perform an inverse distance weighing extension spatial interpolation, enable the use of leave-one-out cross-over to validate the interpolation, compute the error metrics, and finally query the created files to replicate SQL queries of selecting entries between minimum and maximum measured points. Although this implementation serves its function there are ultimately a few tweaks that could be made to allow for better performance all around. The choice of how time was handled was ultimately rather poor and effected performance by having to convert it so frequently, and mixing this issue with the use of the leave-one-out cross-over ultimately created a quite slow program that can still provide results from comparison.

# References

1. Q Fu et al. “Rainfall Spatial Estimations: A Review from Spatial Interpolation to Multi-Source Data Merging” (2019) [https://www.mdpi.com/2073-4441/11/3/579 Retrieved April 23](https://www.mdpi.com/2073-4441/11/3/579%20Retrieved%20April%2023), 2023
2. K. Fung et al. “Evaluation of spatial interpolation methods and spatiotemporal modeling of rainfall distribution in Peninsular Malaysia” Ain Shams Engineering Journal 13 (2022). <https://reader.elsevier.com/reader/sd/pii/S209044792100335X?token=9C9DD6A6D5F28B774523D956DEDF60B5B656537BB6FD575E0C127A37697E795E98D4EDA56A455ADFF0919C426967BC8D&originRegion=us-east-1&originCreation=20230424193112> Retrieved April 23, 2023
3. Revesz, P., & Li, L. (2002a). Constraint-based visualization of spatial interpolation data. In Proc. of the

Sixth International Conference on information visualization (pp.563–569).London, England: IEEE

Press. Retrieved April 24, 2023

1. P. Ohlert, M. Bach, and L Breuer. “Accuracy assessment of inverse distance weighting interpolation of groundwater nitrate concentrations in Bavaria (Germany)” Environmental Science and Pollution Research (2023) 30:9445–9455 https://link.springer.com/article/10.1007/s11356-022-22670-0 Retrieved 04/21/2023
2. K. Zheng, Y. Kangkang, G. Zhou. “Response simulating interpolation methods for expanding experimental data based on numerical shape functions” (2019) https://www.researchgate.net/profile/Kaikai-Zheng/publication/332405126\_Response\_simulating\_interpolation\_methods\_for\_expanding\_experimental\_data\_based\_on\_numerical\_shape\_functions/links/5cb59cea4585156cd79b1e10/Response-simulating-interpolation-methods-for-expanding-experimental-data-based-on-numerical-shape-functions.pdf
3. Esri. (n.d.). How Inverse Distance Weighted interpolation works. ArcGIS Pro Help. Retrieved from [https://pro.arcgis.com/en/pro-app/latest/help/analysis/geostatistical-analyst/how-inverse-distance-weighted-interpolation-works.htm Retrieved April 21](https://pro.arcgis.com/en/pro-app/latest/help/analysis/geostatistical-analyst/how-inverse-distance-weighted-interpolation-works.htm%20Retrieved%20April%2021), 2023
4. P. Antoniadis. Cross-Validation: K-Fold and Leave-One-Out (March 2023) https://www.baeldung.com/cs/cross-validation-k-fold-loo Retrieved 04/23/2023