Polire - A toolkit for spatial interpolation and sensor placement

S. Deepak Narayanan*, Zeel Patel*, Apoorv Agnihotri, Nipun Batra
IIT Gandhinagar
{deepak.narayanan,patel_zeel,apoorv.agnihotri,nipun.batra}@iitgn.ac.in

ABSTRACT

Sensing is central to the SenSys and related communities. However, despite recent advancements, fine-grained spatial sensing remains a challenge, owing to cost, maintenance, among other factors. Thus, estimating the sensed phenomenon at unmonitored locations and strategically installing sensors is of prime importance. In this work, we introduce Polire - an open-source tool to lower the entry barrier for sensor deployments. Polire provides a suite of algorithms for spatial interpolation and sensor placement. We replicate several existing papers on these two tasks to show the efficacy of Polire. We believe that Polire is an essential step towards lowering entry barriers towards sensing and scientific reproducibility.

CCS CONCEPTS

• **Computing methodologies → Modeling methodologies**.

ACM Reference Format:
S. Deepak Narayanan*, Zeel Patel*, Apoorv Agnihotri, Nipun Batra. 2020. Polire - A toolkit for spatial interpolation and sensor placement. In *Conference Name.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Sensing is central to the SenSys and related communities to enable a host of applications. Prior work has sensed various environmental phenomena, including, but not limited to water distribution systems [14, 27, 43], atmospheric phenomena [33, 35] and soil [13, 24]. Sensing these phenomena has few fundamental challenges, including, but not limited to, missing or faulty measurements [48, 52], huge deployment cost [51], energy overhead [25] and optimizing the quality of sensor deployment [27].
The installation and maintenance of sensors entail a huge cost. Thus, it is common to *estimate* environmental variables at unsensed locations using spatial interpolation techniques. Spatial interpolation methods have different underlying assumptions that affect the estimated values. Thus, researchers use multiple spatial interpolation methods [63] while comparing estimates and choose the one having the maximum predictive accuracy [30]. Comparison among various interpolation methods [10, 36, 42] can help benchmark them on widely used datasets and can be used for the baselines for future work. A major hindrance with tackling the problems in spatial interpolation is lack of good quality open-source implementations of existing spatial interpolation methods. Widely used proprietary tools like ESRI's ArcGIS are expensive and non-trivial to integrate into data collection, processing and prediction pipelines.
Interpolation loses its efficacy when the sensor deployment is sparse. "Intelligent" sensor placement is central to maximize reconstruction accuracy and to account for the non-trivial sensor deployment and maintenance cost [27]. A set of work identifies flaws in uniformly placing sensors [23, 67]. While prior work has looked at various sensor placement/deployment algorithms [27, 29], there is a lack of open-source, generalized implementations.
To lower the entry barrier in this field of research, and to enable reproducibility; we propose an open-source spatial interpolation and sensor placement toolkit: Polire, implemented in Python. Polire is inspired by the impact of similar toolkits in other domains [4, 18]. We implement widely used spatial interpolation algorithms and a few state-of-the-art sensor placement algorithms in Polire. Polire has been i) made available open-source (with inbuilt documentation using Python docstrings) to enable collaborations among users within, and across communities; and ii) designed with an abstract though transparent API influenced by scikit-learn [45], a well-documented and consistent Python library.
We replicate two highly influential papers to demonstrate the efficacy of Polire. For interpolation, we replicate a paper by Wong et al. [64], which shows a comparison of several interpolation methods for air quality data estimation of Ozone and PM_{10}. Polire can replicate most results from this paper with just a few lines of code. For sensor placement, we replicate a state-of-the-art sensor placement paper [21]. We use the temperature and precipitation datasets used by Guestrin et al. [21]. We outperform the baselines and replicate the majority of the results successfully. Our code repository is hosted on GitHub below:
`https://github.com/sustainability-lab/polire`.
Our paper is organized as follows: We define our problem statement and convention in Section 3. In Section 4, we elaborate on the currently implemented interpolation schemes and sensor placement algorithms. We also demonstrate a simple use of Polire with a few code snippets. We explain the experiments and results in Section 5. We discuss the limitations of current work and directions to future work in Section 6. Finally, we conclude the paper in Section 7.

2 RELATED WORK

We broadly categorize our related work into four categories: spatial interpolation, sensor placement, toolkits, and reproducibility challenges. We describe each of them below.

2.1 Spatial Interpolation

Spatial interpolation techniques have been widely used by SenSys and related communities to get estimated values at unsensed locations. As the name suggests, these methods model spatial phenomena using spatial or Spatio-temporal data. Kriging [31] is a spatial

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Under Review, Dates, Location
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

interpolation technique used extensively in geostatistics [41]. Kriging assumes the observed values to be realizations of a random process. The estimate at an unsensed location is a linear combination of the values at the sensed locations giving the best linear unbiased estimate at unobserved locations. Inverse Distance Weighted (IDW) [60] interpolation is another widely used spatial interpolation technique. In IDW, the estimates at unobserved locations are the weighted average of the observed data locations. The weights are inversely proportional to the distance between the observed and unobserved locations. KNN (K-Nearest Neighbour) is another method used for predicting measurements at unsensed locations. In weighted KNN, observations are assigned weights inversely proportional to their distance from a test data point. Weighted KNN and IDW are the same when K is equal to the number of observed locations, and only the spatial features are considered. Wong et al. [63] use all of IDW, Kriging, Spatial Averaging and KNN in their work to interpolate PM_{10} values [63]. Recent work in SenSys demonstrates a hybrid air pollution reconstruction by adaptive interpolation techniques [65]. The authors use the Random forest algorithm to choose between Kriging and IDW adaptively.

2.2 Sensor Placement

Sensor deployment has been a well-studied problem. Krause et al. [28] propose an algorithm to simultaneously optimize the placement and the scheduling of sensors with constraints on the amount of the power used. Krause et al. [27] propose a sensor deployment model for detecting flaws in water distribution networks. The common aspect between [28] and [27] is that they deploy sensors from scratch, without any initial deployment. Hseih et al. [23], propose an incremental station deployment strategy for sensor deployment, assuming an initial deployment of sensors. They propose a semi-supervised approach to infer air quality and then subsequently recommend a fixed number of locations for installing sensors. The scheme proposes installing all the recommended sensors at once.

2.3 Toolkits

Various toolkits have been proposed in SenSys and related communities for various applications, including, but not limited to: i) energy analysis for smartphones [6]; ii) household energy disaggregation [26]; iii) thermal comfort sensing [56]; iv) solar modeling and forecasting [3]; v) rapid deployment of computing environments [20] and vi) intelligibility in context-aware applications [32]. An attempt was made for open-source spatial interpolation toolkit in 1997 by Robert et al. [49] for old decAlphas and SGI machines. Esri's ArcGIS is widely used proprietary toolkit nowadays for spatial interpolation. While various algorithms for placement and interpolation have been proposed, empirically comparing them remains a challenge.

2.4 Reproducibility challenges

Reproducibility is highly encouraged in many ACM conferences [1]. Call for papers for many conferences explicitly ask for reproducibility information. Moreover, various review forms for conferences now require the reviewer to fill the reproducibility score of submitted work. Prior studies [1] have discussed the various social and technical reasons impeding reproducibility. We believe our work can help lower the time and effort required towards replicating earlier studies, and also provide an added incentive to authors to make their work reproducible.

3 PROBLEM STATEMENT AND CONVENTION

We now formally define the problem statements for spatial interpolation and sensor placement.

Spatial Interpolation:

Given a set of sensors, along with information about their locations ($\mathbf{x}$) and the quantity they measure (y), the goal is to estimate the values for the measured quantity at unsensed locations. More formally, given sensory data $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$, where each $\mathbf{x}_i \in \mathbb{R}^2$, and $y_i \in \mathbb{R}, \forall i = 1, 2, 3, \ldots N$, and each y_i corresponds to the sensor measurement at location $\mathbf{x}_i$, we would like to estimate the value at locations $\mathbf{x}^*_j$, $\forall j = 1, 2, 3, \ldots k$. We denote the corresponding estimated values by $\hat{y_j}, j \in 1, 2 \ldots, k$. The two dimensions are coordinates in space and sensor readings/observations/measurements for $\mathbf{x}$ and y respectevely.

Sensor Placement: Given a set of sensors, along with information about their locations and the sensed values, our goal is to choose a subset of locations for placing new sensors from a set of candidate locations, such that a specific placement criterion (such as MI or entropy) is maximized. More formally, given sensory data $\mathcal{D} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$, where each $\mathbf{x}_i \in \mathbb{R}^2$, and $y_i \in \mathbb{R}, \forall i = 1, 2, 3, \ldots N$, and each y_i corresponds to the sensor measurement at location $\mathbf{x}_i$, we would like to choose a subset of locations $\mathcal{A}$ from a set of candidate of locations $\mathcal{V}$ so that a specific criteria is maximized.

4 TOOLKIT NAME

In the subsequent subsections, we present and describe Polire, our proposed toolkit for spatial interpolation and sensor placement. An important aim of the toolkit is to provide users with a consistent and easy-to-learn API. Our API is inspired by a popular machine learning package scikit-learn's API [7, 44]. We implement our toolkit in Python, owing to its easy to learn nature, and the enormous number of libraries that can be used for scientific computing [58, 59].

This section is further divided as follows: We describe the spatial interpolation and sensor placement algorithms implemented in Polire. Then, we show usage of our API and conclude this section with a kernel that we implement for non-stationary Gaussian processes.

4.1 Spatial Interpolation Algorithms

Kriging Interpolation

Kriging is a popular algorithm used for spatial interpolation [12, 57]. Kriging provides the best linear unbiased predictions [2]. Kriging interpolation assumes that a random process governs the underlying phenomena. An important step in Kriging is to compute an empirical variogram and fit a parametric variogram it. The variogram acts as a function of the dissimilarity among input data points. There are several variants of Kriging such as a) Ordinary Kriging; b) Simple Kriging; and c) Universal Kriging. They differ in how they model the expectation of the random process [2]. Ordinary Kriging assumes a constant unknown mean for the random process. Simple Kriging assumes a constant mean over the entire space of interpolation. Universal Kriging assumes that the random process can be decomposed into a non-random trend function and a random process. We build our Kriging interpolation implementation on the top of a popular Python package PyKrige [37]. Polire implements Ordinary Kriging and Universal Kriging. We do not implement Simple Kriging, which is limited and least general [39].

Inverse Distance Weighted (IDW) Interpolation

In IDW, the interpolated value at an unknown location is a weighted sum of all observations [53]. The weights are inversely proportional to the distance between the sensed locations and the unsensed location. As p increases, the influence of close-by data points is more profound. The prediction $\hat{y}$ at a location $\mathbf{x}_j^*$ in IDW is,

$$\hat{y}_j = \frac{\sum_{i=1}^{N} w_i(\mathbf{x}_j^*) y_i}{\sum_{i=1}^{N} w_i(\mathbf{x}_j^*)}$$

where the weights $w_i(\mathbf{x}^*{}_j) := \frac{1}{d(\mathbf{x}_i, \mathbf{x}^*{}_j)^p}$. Here $d(\mathbf{x}_i, \mathbf{x}^*{}_j)$ is the distance computed between the locations $\mathbf{x}_i$ and $\mathbf{x}^*{}_j$ and p is the exponent of this distance used for computing the weights. In our implementation, we allow the users to choose different weights based on the p^{th} power of the distance.

Spatial Averaging Interpolation

In Spatial Averaging, interpolated values of unsensed locations are evaluated as the average of the sensed locations within a circular region centred to the unsensed location. This method assumes that the interpolated values are dependent on the sensed values in a circular region and each sensor contributes equally irrespective of the relative distance within this circular region.

Trend Interpolation

In Trend Interpolation, the underlying phenomena is modelled as a polynomial that fits the dataset. In this interpolation, the order of the polynomial captures the extent of interplay between the coordinates of the input space. As an example, in a second order polynomial, when we fit an input space over latitudes and longitudes, there is an interaction term, which is the product of latitude and longitude. In our implementation, we support Trend Interpolation up to degree 2. For a degree 2 trend interpolator, the equation is as : $\hat{y_j} = a + bx_1 + cx_2 + dx_1{}^2 + ex_2{}^2 + fx_1x_2$, where x_1 and x_2 are the two coordinates of a test point $\mathbf{x}^*{}_j$).

Natural Neighbor Interpolation

Natural Neighbor interpolation is based on a Voronoi tessellation of the original input space and subsequently computing a weighted sum of a subset of the sensed locations. The Voronoi tessellation is made within the range of the sensed locations. Once we have a query point, the weights are assigned on the basis of the overlap area with all the nearby tessellations with which the data point shares a boundary. This method cannot typically extrapolate beyond range of the input space. We build our implementation on top of the Shapely package [17] in Python, which is used for manipulation, as well as analysis of planar geometric objects.

Spline Interpolation

Spline interpolation fits piecewise polynomials called as splines. In our implementation, we build a wrapper to use the bivariate B-spline interpolator from SciPy [59].

Random Interpolation

Random Interpolation interpolates an unsensed location using a value chosen uniformly at random from the range of the sensed measurements.

Learning Based Interpolation

In Polire, we do not just limit the users to the classical spatial interpolation methods discussed above. To make Polire more accessible and widely used, we allow any machine learning algorithm from the scikit-learn library [46] to be used as an interpolator. This allows an easy comparison against state-of-the-art machine learning algorithms. Sometimes, it may be desirable to exploit features beyond space. As an example, for air quality, we may want to use road networks or points of interests as potential features. This can help benchmark spatial interpolation algorithms while looking at the importance of features that go beyond space. Previous work [66, 68, 69] are examples of such work that exploit features beyond spatial locations.

A note on Gaussian Process Regression

We now discuss Gaussian Processes (GPs), which is a generalization of Kriging interpolation when used for interpolation tasks. More formally, Gaussian Process Regression is a non-parametric Bayesian machine learning algorithm. The GP can be viewed as a generalization of the multivariate Gaussian distribution having infinitely many variables such that any finite subset among them is a Gaussian distribution. In our case, we can model the sensed phenomena in 2D space as a GP. In many cases, a GP is used to model an environmental phenomena. Some examples of such phenomena include wind direction, air quality, temperature and precipitation [14, 22, 29]. In any GP model, we have a prior mean function $\mu : \mathbb{R}^d \to \mathbb{R}$ and a prior covariance function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. The covariance functions quantify the similarity among different data points. They are also well-known as kernels. A kernel function is used to compute the covariance matrix for a GP. The covariance matrix Σ has entries $\Sigma_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, where k is a kernel function and $\mathbf{x}_i$ and $\mathbf{x}_j$ are two data points in $\mathbb{R}^d$. We use the following notation in this section: Let $X \in \mathbb{R}^{n \times d}$ denote all the training data points and $y \in \mathbb{R}^n$ be the observations. Σ_{XX} refers to a covariance matrix. $\hat{\Sigma}_{XX} = \Sigma_{XX} + \sigma_n^2 I_n$ is a covariance matrix added with zero-mean Gaussian noise of variance σ_n^2, I_n is the identity matrix of order n. $\Sigma_{X\mathbf{x}^*}$ refers to a vector formed by calculating the covariance function between any test point $\mathbf{x}^*$ and all the train points. Once the model is trained on the data, we obtain the predictive posterior distribution. For a test point $\mathbf{x}^*$ and its corresponding predictive distribution $y*$, we have

$$\mathbb{E}[y^*|X, \mathbf{y}] = \mu(\mathbf{x}^*) + \Sigma_{X\mathbf{x}^*}^T \hat{\Sigma}_{XX}^{-1} \mathbf{y}$$

$$\mathrm{Var}[y^*|X, \mathbf{y}] = k(\mathbf{x}^*, \mathbf{x}^*) - \Sigma_{X\mathbf{x}^*}^T \hat{\Sigma}_{XX}^{-1} \Sigma_{X\mathbf{x}^*}$$

where $\mathbb{E}[y^*|X, \mathbf{y}]$ and $\mathrm{Var}[y^*|X, \mathbf{y}]$ are expectation and variance of a posterior distribution respectively [50]. In our case, to obtain the predictive mean and variance at a test point $\mathbf{x}_j^*$, where $\mathbf{x}_j^* \in \mathbb{R}^2$, we replace $\mathbf{x}^*$ by $\mathbf{x}_j^*$.

The effectiveness of modelling of underlying phenomena is significantly affected by the kernels. Widely used kernels such as Matern or RBF assume a few strong conditions. One such condition is stationarity, which means that the covariance between any two data points is a function of only the distance between the data points. This assumption may not always hold in practical scenarios. To demonstrate the same, we plot correlations between all the pairs of locations for temperature dataset described in section 5.2.1. Figure 1 demonstrates the inherent non-stationarity in the temperature dataset since there are several data points with different correlations having the same distance among themselves. Therefore, in addition to spatial interpolation algorithms implemented above, we also implement a non-stationary kernel proposed by Nott and Dunsmuir [38] and henceforth called NS kernel in this paper. The NS kernel came as a natural choice as it does not assume any of the conditions assumed by stationary kernels thus can be used to model a wide variety of phenomena. We believe that our implementation is the first open-source implementation of the NS kernel. Earlier work has shown that using this particular kernel over stationary kernels can provide significant gains in the predictive performance [21]. Guestrin et al. [21] used this kernel for sensor placement. The NS kernel learns spatial characteristics of the region locally and aggregates knowledge from all local regions into a kernel function. It requires Empirical Covariance Matrix[1] (ECM) as an input. The ECM is an estimate of the actual covariance matrix from a sample of a multivariate Gaussian distribution. Let us consider a set of sites $S = \{s_1, s_2, ..., s_n\}$ and a corresponding empirical covariance matrix Γ. We can learn stationary kernels at each of the sites locally by considering a region of closest N sites. If we denote stationary kernel function at site s_i as $K(h, \theta_i)$ and a set of closest sites as $\delta(s_i)$, where h is the distance between two sites and $theta_i$ are the kernel parameters. We want to minimize the difference between $K(h, \theta_i)$ and empirical covariance values by optimizing θ_i at all sites with Equation 1,

$$min\Sigma_{s_j,s_k \in \delta(s_i)} \left\{ \frac{\Gamma_{jk} - K(s_j - s_k, \theta_i)}{K(s_j - s_k, \theta_i)} \right\}^2 \tag{1}$$

Once we optimize for θ_i for $\forall i \in \{1, 2, ..., n\}$ using a least-square optimization technique, we can define the non-stationary kernel function $R(s_x, s_y)$ between any two sites s_x and s_y by Equation 5,

$$R_{\delta i}(s_x, s_y) = K_i(s_x - s_y) - c_i(s_x)^T C_i^{-1} c_i(s_y) \tag{2}$$

$$R_1 = \sum_{i=1}^{n} \sum_{j=1}^{n} v_i(s_x) v_j(s_y) c_i(s_x)^T C_i^{-1} \Gamma C_j^{-1} c_j(s_y) \tag{3}$$

$$R_2 = \sum_{i=1}^{n} v_i(s_x)^{\frac{1}{2}} v_i(s_y)^{\frac{1}{2}} R_{\delta i}(s_x, s_y) \tag{4}$$

$$R(s_x, s_y) = R_1 + R_2 \tag{5}$$

where, K_i is Kernel function at site s_i, $c_i(s)$ is a $n \times 1$ covariance vector $[K_i(s - s_j)]_{j=1}^n$, C_i is a $n \times n$ covariance matrix $[K_i(s_j - s_k)]_{j,k=1}^n$, $v_i(s)$ is a weight function. For any s, $\sum_{i=1}^n v_i(s) = 1$. Any kernel function $f(h)$ can be used in $v(s)$. One such function is $exp\left(\frac{||h||^2}{\eta}\right)$. $v_i(s)$ can be calculated as Equation 6,

$$v_i(s) = \frac{f(s - s_i)}{\sum\limits_{j=0}^{n} f(s - s_j)} \tag{6}$$

Distance Metrics

In Polire, we do not limit ourselves to just Euclidean distances between the data points in distance-based algorithms. Euclidean distance does not account for the curvature of the earth surface when interpolating over hundreds of kilometres in terms of distance.

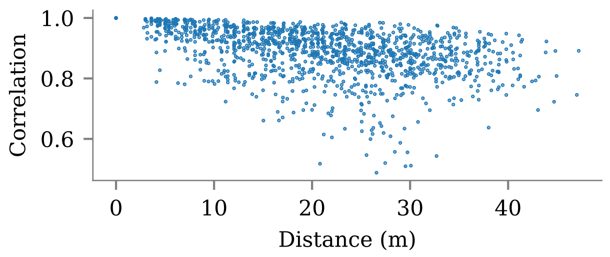


Figure 1: Pairwise correlation among sensor enabled locations v/s distance for temperature dataset. Pairs having same distance between themselves show different correlations due to non-stationarity in the data.

To this end, we also incorporate the great circle distance, using the Haversine formula. We chose to use the Haversine formula due to its numerical stability. [55]. Incorporating this distance into the interpolation algorithms naturally accounts for the curvature of the earth and can thus interpolate over regions with significant changes in latitudes and longitudes. This is essential since the Euclidean distance assumes that points lie on a plane and change along each direction for a specific distance is the same which is not correct when we are dealing with interpolations over large areas of the earth, given that unit distances along latitude and longitude are unequal.

Gridded Interpolation

Apart from providing point-wise estimates, Polire also provides estimates over a grid. Gridded Interpolation is preferred for visualization and interpretation in several spatial interpolation tasks.

Comparison with ArcGIS Interpolation Toolset

In Table 1, we compare the spatial interpolation algorithms that we implement in Polire with the algorithms present in ArcGIS's Interpolation Toolset [2]. We not only implement almost all the algorithms present in ArcGIS (which are not open-source) but also implement additional algorithms like Spatial Averaging and also Learning-Based Interpolation that can make use of already popular machine learning packages such as Scikit-Learn. We do not implement Topo to Raster since it is a highly specialized algorithm designed for making hydrological modelling [15]. We do not implement Spline with Barriers again for similar reasons.

4.2 Sensor Placement Algorithms

Many criteria have been proposed to optimize the quality of sensor placements for Gaussian Process models, including placing sensors at highest entropy locations, maximizing MI between sensed and unsensed locations, A- D- or E-optimal design [8, 11, 54, 70, 71]. We will first discuss entropy in detail.

4.2.1 Entropy as a criteria. Gaussian Process models enable us to calculate posterior mean and variance of unsensed locations using Bayes rule. We want to place sensors at locations with the highest entropy conditioned on observations. Entropy (H) of a Gaussian random variable $\mathcal{X}$ conditioned on a set of variables $\mathcal{Y}$, is

[1]http://math.mit.edu/~liewang/ECM.pdf

[2]https://bit.ly/2YSsHEY

Algorithm	ArcGIS Toolset	Polire
IDW	✓	✓
Kriging	✓	✓
Spatial Averaging	✗	✓
Trend	✓	✓
Natural Neighbor	✓	✓
Spline	✓	✓
Spline with Barriers	✓	✗
Topo to Raster	✓	✗
Learning Based Interpolation	✗	✓

Table 1: Comparison of interpolation algorithms in ArcGIS and Polire. We implement almost all the algorithms present in ArcGIS's toolkit, along with some algorithms that are not present in ArcGIS's toolkit.

a monotonic function of its variance. Variance is also a measure of uncertainty. Naturally, uncertainty is higher at points far from the observed points. Thus, GP with entropy criterion will place sensors at points having highest posterior variance [54]. Entropy $H(\mathcal{X}|\mathcal{Y})$ of a Gaussian random variable $\mathcal{X}$ conditioned on another variable $\mathcal{Y}$ is $\frac{1}{2}log(2\pi e\sigma^2_{\mathcal{X}|\mathcal{Y}})$, where $\sigma^2_{\mathcal{X}|\mathcal{Y}}$ is posterior variance.

4.2.2 Mutual Information (MI) as a criteria. Caselton and Zidek [8] proposed a criterion which places sensors by reducing uncertainty about all unsensed locations. We want to place a set of sensors $\mathcal{A}^*$ on a set of candidate locations $\mathcal{V}$ which gives accurate predictions at unsensed locations as shown in Equation 7.

$$\mathcal{A}^* = argmax_{|\mathcal{A}|=k} H(\mathcal{V}\backslash\mathcal{A}) - H((\mathcal{V}\backslash\mathcal{A})|\mathcal{A}) \qquad (7)$$

Maximizing Equation 7 is similar to maximizing MI $I(\mathcal{A};\mathcal{V}\backslash\mathcal{A})$.

4.2.3 MI v/s Entropy: A comparison. Entropy as a criterion suffers from a problem of placing sensors near boundaries as entropy tends to higher at the boundary [47]. Unlike entropy, MI takes unsensed locations into account (Equation 8). Therefore, MI tends to place sensors central to the unsensed locations and avoids wasted information unlike entropy by placing sensors at the boundaries. Thus, MI is proposed as a better design criteria by Guestrin et al. [21].

4.2.4 Near optimal sensor placement algorithm. Guestrin et al. [21] prove that maximizing MI is an NP-complete problem. Thus, they propose a greedy approximation algorithm (Algorithm 1) for sensor placement having polynomial time-complexity. Any submodular function such as entropy or MI can be used as a criterion in this method to greedily select one sensor at a time and still achieve near optimal placement. Guestrin et al. [21] also give an approximation bound $(1 - 1/e)$ for the greedy algorithm which is approximately 63%, which implies that MI calculated with Algorithm 1 will be at least 63% of the maximum MI. If the sensed and unsensed locations are denoted as $\mathcal{A}$ and $\bar{\mathcal{A}}$ respectively, gain in MI after placing a new sensor at location y can be calculated as Equation 8.

$$MI_{Gain} = MI(\mathcal{A} \cup y) - MI(\mathcal{A}) = H(y|\mathcal{A}) - H(y|\bar{\mathcal{A}}) \qquad (8)$$

We implement the approximation Algorithm 1 proposed by Guestrin et al. [21] in our toolkit, which greedily places the sensors. We use 'Greedy algorithm' as an alternative name to Algorithm 1 in subsequent sections.

Algorithm 1: Approximation algorithm for sensor placement using MI [21]

```
Input  : $\Sigma_{\mathcal{V}\mathcal{V}}$ (Covariance matrix), $k$ (Number of
         placements), $\mathcal{V}$ (set of candidate locations)
Output : $\mathcal{A} \subset \mathcal{V}$ (set of newly placed locations)
$\mathcal{A} \leftarrow \phi$;
for $i \leftarrow 1$ to $k$ do
    $\delta_{old} \leftarrow -\infty$;
    $selected \leftarrow \phi$;
    for $y \in \mathcal{V}\backslash\mathcal{A}$ do
        $\delta \leftarrow \frac{\sigma^2_y - \Sigma_{y\mathcal{A}}\Sigma_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}y}}{\sigma^2_y - \Sigma_{y\bar{\mathcal{A}}}\Sigma_{\bar{\mathcal{A}}\bar{\mathcal{A}}}\Sigma_{\bar{\mathcal{A}}y}}$;
        if $\delta \geq \delta_{old}$ then
            $\delta_{old} \leftarrow \delta$;
            $selected \leftarrow y$;
        end
    end
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{selected\}$;
end
```

4.2.5 Optimal placement v/s Greedy placement. To compare the performance of the Greedy algorithm with the optimal solution, we do a brute force search to select the best combination of k locations maximizing the MI. An important caveat is that this brute force algorithm is computationally efficient only for small values of k.

4.2.6 Random Placements. We randomly place k sensors, $k \in \{1, 2, ..., N\}$ M times with M distinct random seeds. We use this as a baseline to compare against the solution obtained via the greedy algorithm.

4.3 API

In this section, we demonstrate a simple use of the interpolation and sensor placement methods.

4.3.1 Data Formats. consists of a two-dimensional NumPy [40, 58] array for the input locations, `X` that typically have information about the spatial locations and similarly a vector of NumPy array for the sensed values or targets, `y`. For the special case of the Learning-Based Interpolation, the input information can go beyond just the spatial locations and could involve other factors affecting the underlying phenomena like time, etc.

4.3.2 Spatial interpolation API. In Polire, we follow the Scikit-Learn API [7] for fitting the models and then generating predictions using the fitted models. Our choice of following this particular API is motivated by its simplicity and ease of use. We would also like to emphasize that this consistent interface is already well tested and thus easy for algorithm developers to contribute. To demonstrate how to use our API, we show a minimal working example in Listing 1.

```
1 from Polire.interpolate import Kriging
2 intpr = Kriging()
3 intpr.fit(X, y)
4 prediction = intpr.predict(X_unsensed)
5 grid_prediction = intpr.predict_grid()
6 # Access parameters of algorithm
7 intpr.variogram_model
```

Listing 1: A simple usage of Polire.interpolate

Firstly, the user needs to import the class of the interpolator (L1) to be used. Then, the user initializes the class (L2) with any pertinent parameters, followed by the `fit()` (L3) method which is then subsequently followed by the `predict()` (L4) method. To predict over a default grid spanning the range of the input data points, the user needs to call the `predict_grid()` (L5) method. The users can also provide custom ranges for gridded interpolation by passing the range as an argument. To access various parameters of the interpolation algorithm, the users need to use the '.' (dot) notation. As seen in the above listing, to access the variogram model, `intpr.variogram_model` is used.

4.3.3 Sensor placement API.

As explained in the earlier section, we make our sensor placement API simple as well. A minimal example for the sensor placement is given in Listing 2.

```
1 from Polire.placement import NottDuns
2 model = NottDuns(N=10)
3 model.fit(X, y)
4 index, loc = model.place(X_unplaced, N=1)
5 # Access parameters of placement algorithm
6 param_dict = model.get_all_params()
7 #or
8 #param = model.get_param(parameter_name)
```

Listing 2: A simple usage of Polire.placement

For sensor placement, user needs to import a class from Polire.placement. Valid class names are `NottDuns (L2)` (for the NS kernel) and `Stationary` (for a set of stationary kernels) as of now. In above example we pass `N` to mention number of nearest points to learn kernel locally. Once we call `model.fit (L3)` method, model is ready to use for sensor placement. We can use `model.place (L4)` method to place sensors at `X_unplaced` locations and pass in parameter `N` to specify number of placements in the above example. Model parameters can be accessed using `.get_all_params (L6)` or `.get_param (L8)` method as shown in Listing 2.

5 EXPERIMENTS AND EVALUATION

We explain our experiments and results for two paper replications in the subsequent sections.

5.1 Replication of Spatial Interpolation Paper

To demonstrate the efficacy of Polire for spatial interpolation, we firstly replicate the paper by Wong et al. [64]. We chose to replicate this paper due to the following reasons: 1) This work was highly impactful and was published at a reputed journal. (2) It is relevant to the SenSys community [9, 34, 61] (3) This work is intended towards assessing the role of exposure to ambient air pollution as a risk factor for respiratory diseases in children. This is extremely important since poor air quality has been shown to cause chronic illnesses including asthma and cancer and (4) The authors use the United States Environmental Pollution Agency's (EPA) AIRS monitoring data to estimate O_3 and PM_{10} levels at a block group level[3] in the USA. This air quality dataset is public, contains data spanning decades, and is easily accessible.

We would like to note here that while the dataset for air quality is publicly available, the dataset of the locations for which the authors perform spatial interpolation is not wholly available publicly. This was due to the authors using protected data obtained from National Health and Nutrition Examination Survey - III (NHANES-III), which provides very detailed health information on a statistically significant proportion of the United States population. The authors interpolate PM10 and O_3 over 82 different counties in the United States where NHANES-III was conducted. The publicly released data only contains counties with a population of over 500,000, due to confidentiality and privacy reasons. This data corresponds to a total of 35 of the total 82 counties Further, the authors interpolated at a block group level. In our current replication of the work, we interpolate over all the block groups over the United States. This is because of the following reasons: (1) We do not have access to the exact block groups the authors used; and (2) The NHANES-III survey is a statistically representative sample of the US population; If we only interpolate over the counties in the publicly released data, our results may not be representative of the US population. By choosing to interpolate over all the counties in the US, we solve this issue of having a statistically significant representation of the US population. In many cases, the authors consider specific regions for comparing different interpolation methods. In such cases, we appropriately only consider those regions in our experiments. We also note here that we use the block groups data for the US from

5.1.1 Datasets, Statistics and Quality Assurance.

Ozone (O_3)

Datasets: The authors use the dataset from AIRS database of O_3 values sampled hourly from all the stations in the USA for the year 1990. The unit of measurement is parts per billion (ppb). This dataset can be retrieved from EPA website.

Statistics: The authors propose using 'Summer Daytime Ozone' as their statistic for measuring exposure. This metric computes the average ozone exposure during the daytime in the summer months from May to September.

Quality Assurance: While computing the above statistic, the authors enforce the following conditions: (1) For computing daily average; there must be at least 8 of 9 hourly measurements for any station during daytime (10:00 AM to 6:00 PM). (2) For computing the exposure over summer, each station requires at least valid readings from 115 out of the 153 days between May and September.

PM_{10}

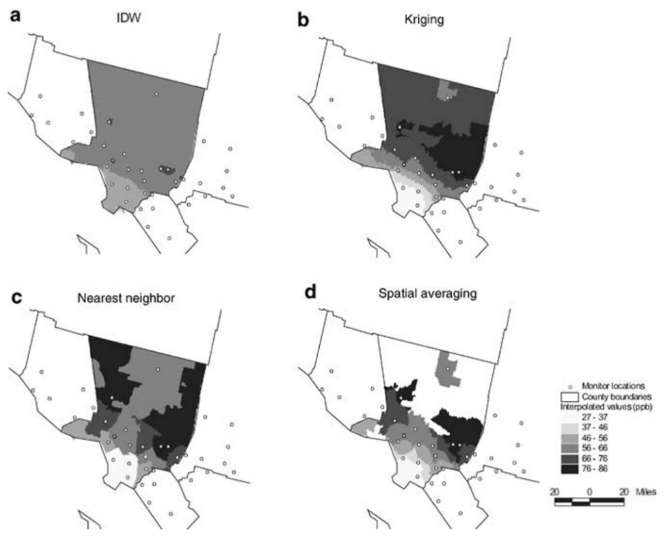
Datasets: The authors use the dataset from AIRS database of PM_{10} values reported every sixth day for a 24 hour sampling period for 1990. The unit of measurement is $\mu gm/m^3$. This dataset can be retrieved from EPA website.

Statistics: The authors use two statistics for measuring exposure

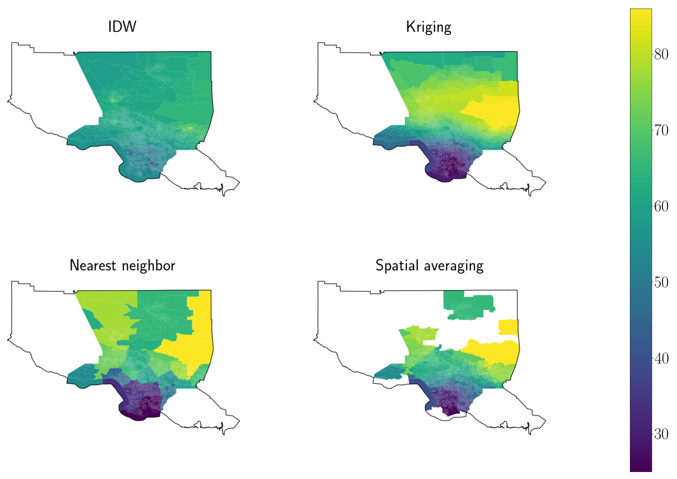
[3] We refer to census block groups as block groups in this paper

Pollutant	Statistics	25th (Theirs)	25th (Ours)	50th (Theirs)	50th (Ours)	Mean (Theirs)	Mean (Ours)	75th (Theirs)	75th (Ours)
Smr. Day. O_3	Monitor	40.0	40.6	45.2	46.3	45.9	46.8	51.2	51.9
	Kriging	36.9	42.3	47.5	52.6	49.4	51.2	61.2	60.1
	IDW	39.9	42.2	43.0	45.5	45.0	45.7	49.8	49.0
	NN	35.9	39.0	40.9	45.2	42.8	45.2	47.5	50.3
	Sp Avg	36.7	39.9	41.4	44.3	43.4	45.2	48.0	49.3
Ann. Avg PM10	Monitor	23.0	22.4	27.7	27.3	29	28.9	33.6	33.6
	Kriging	27.1	26.4	29.9	29.5	33.2	30.0	37.0	31.5
	IDW	27.8	26.9	30.5	29.4	33.5	30.8	37.5	32.4
	NN	26.0	24.5	30.4	30.2	33.7	31.9	39.0	34.7
	Sp Avg	27.7	25.8	32.4	30.2	34.8	31.9	39.4	34.7
Max Qtly PM10	Monitor	29.0	25.6	35.6	33.2	37.8	33.6	43.7	40.4
	Kriging	34.8	30.2	38.2	34.7	42.6	35.5	46.9	40.8
	IDW	34.1	31.2	38.9	35.6	42.9	36.0	51.4	40.4
	NN	33.5	28.9	38.1	34.6	43.3	35.6	52.0	40.7
	Sp Avg	34.5	30.1	39.8	30.6	44.3	36.3	52.5	41.0

Table 2: Summary Statistics. In the above table, we show the summary statistics of the interpolated values as reported by the authors [64], as well as the values we obtain. In the Statistics Column, Monitor refers to the air quality sensor and its statistics. All the other values except monitor are estimates and the statistics concerning them. NN and Sp Avg refer to Nearest Neighbor and Spatial Averaging respectively. Smr. Day. O_3 refers to Summer Daytime Ozone, Ann. Avg PM10 refers to Annual average PM_{10} and Max Qtly PM10 refers to Maximum Quarterly PM10. 25th **refers to the 25th percentile of value,** 50th **to the median and** 75th **refers to the 75th percentile. We observe that most of our statistics are consistent and very close to that observed by the authors. We note that there is a minor difference in the statistics for Max Qtly PM10 between ours and the authors'.**



(a) Author's Mapping Estimates for LA County [64]



(b) Estimates of four interpolation algorithms for Summer Daytime Ozone for Los Angeles.

Figure 2: Mapping estimates of Ozone in Los Angeles. As can be seen, the interpolated estimates and the trends are consistent with [64].

- Annual Average; and the maximum quarterly average of PM_{10} values.

Quality Assurance: For every PM_{10} monitor AIRS computes the percentage values recorded out of the total number of values. The authors require that, for any station to be included into this computation of quarterly average, at least 75% of these values must be present in each quarter for the monitor.

5.1.2 *Interpolation Algorithms and Settings.*

(1) **Kriging Interpolation**: In their work, The authors use Ordinary Kriging algorithm with a Spherical variogram as their parametric variogram. They firstly evaluate the feasibility of performing Kriging using the datasets. They do this by checking the fit of the parametric variogram to the semivariance plot. After checking for feasibility for (1) PM_{10}: They divide the United States into five aerosol regions and conduct Kriging on each of them separately and, (2) O_3, they perform Kriging only in Southern California.

(2) **Inverse Distance Weighted Interpolation (IDW)**: For IDW, the authors consider a radius of 250km around an unsensed location and interpolate using all the monitors within this radius. They choose this particular radius to ensure that an IDW estimate is available at all block groups. They choose the exponent of the distance to be 1 in their paper. For both O_3 and PM_{10}, they use the same conditions.

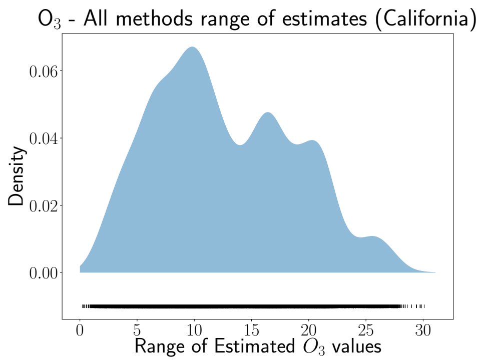
(3) **Spatial Averaging**: In their work, the authors consider spatial averaging with a search radius of 10 miles. For both O_3 and PM_{10}, they use the same conditions. We do not go into the reasons for this choice of radius as it is beyond the scope of our work.

(4) **Nearest Neighbor**: In their work, the authors also propose to use the nearest monitor value for interpolation at the unsensed locations, irrespective of the distance.

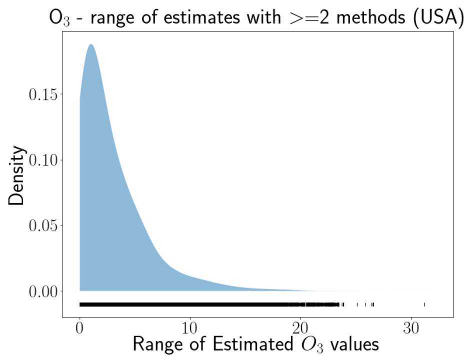
5.1.3 *Replication of Experiments and Results in [64].*

(1) **Summary Statistics**: The authors compute summary statistics in which they compare the statistics of the interpolated values and the dataset for different interpolation algorithms and pollutants. For any interpolation method, this summary was computed over all the counties that had estimates for that particular method. We report both the authors' summary statistics and ours in Table 2. As can be seen, our values for Summer Daytime O_3 and Annual Average PM_{10} are very similar. Our Maximum Quarterly PM_{10} values are relatively lower than what the authors report. We believe the reason could be that the differences in the datasets due to lack of information about all the counties. This is evident in Table 2, where the Monitor value for our dataset is lower than the authors'.

(2) **Summer Daytime Ozone - Interpolated values in California**: As mentioned in the subsection 5.1.2, the authors could perform Kriging interpolation only in Southern California. To compare the different algorithms, the authors employ all the interpolation algorithms in Los Angeles (LA) County in Southern California. They observe that in LA County (1) IDW predicted fairly uniform concentration across the county, with a small increase from coast to inland; (2) Kriging values increased more strongly from coast to inland; (3) Nearest-Neighbor interpolations were a patchwork of areas of constant values and (4) Spatial averaging was notable in the number of locations in the county for which interpolated values were not computed. They map these interpolations of O_3, and this can be seen in Figure 2a. The corresponding map that we generate can be seen in Figure 2b. Our results are consistent with the authors' observations and trends. The authors also report the mean interpolated summer daytime ozone values for California. We note here that the authors do not provide specific regions in California that they interpolated over, while they do mention that Kriging could only be performed in Southern California. Therefore, we compute interpolated values for the entire state of California as well as for Southern California. We report their values and our values in Table 3. Our interpolated values are similar to what the authors report, albeit with some minor difference. We believe that this difference is due to us not having access to the actual block groups used by the authors.



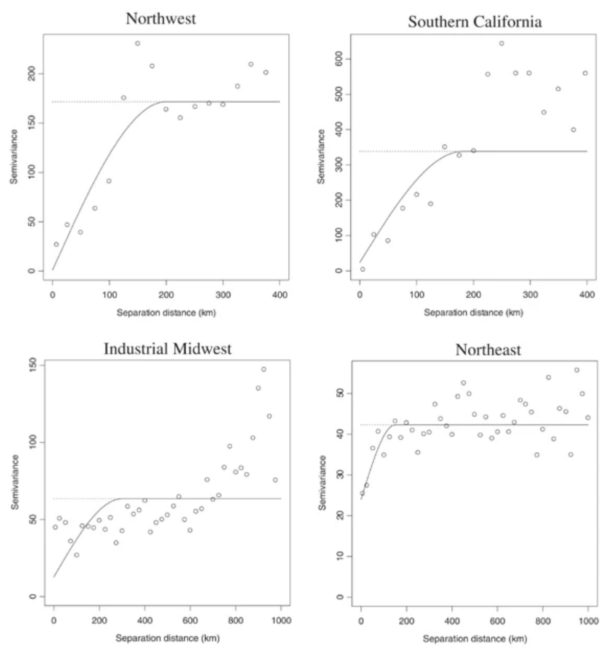
(a) Range of Estimates of Summer Daytime Ozone when all the estimates are available in California.



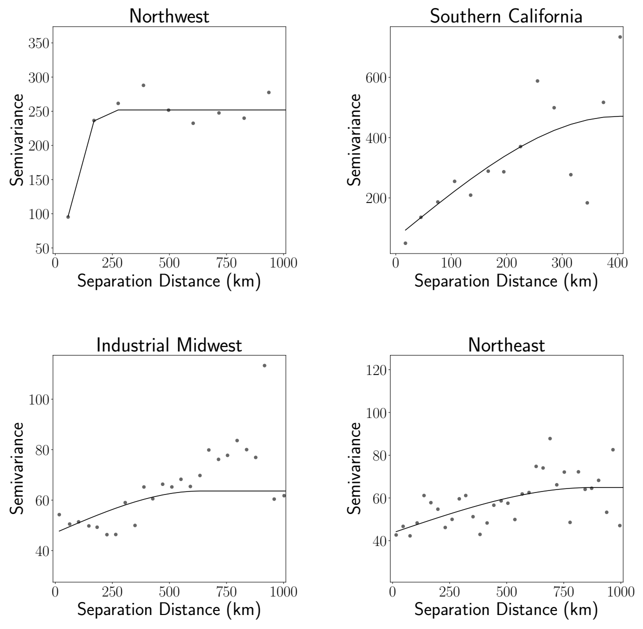
(b) Range of Estimates of Summer Daytime Ozone when at least 2 estimates are available at each block group.

Figure 3: The top figure (a) above shows the ranges of estimates being normally distributed at a block group level with all estimates. When this condition is relaxed, we get a positively skewed distribution, with most block groups having low range of estimates. This is consistent with the authors' observations.

(3) **Comparison of variogram fit for different aerosol regions for PM_{10}**: The authors perform Kriging by dividing the United States into five aerosol regions. They show plots of the variograms fit the data along with the empirical values. They observe that the variogram fit is not satisfactory in 2 aerosol regions, Northeast and Industrial Midwest, while they are better in the cases of the remaining three regions - Northwest, Southern California, and Southeast. Their figures, along with ours, can be seen in Figure 4. For Industrial Midwest, they report that semivariances computed remained less than the total sample variance till separation distances of 700 km. We observe a very similar trend in our replication. We



(a) **Empirical Variograms for five modified aerosol regions. This figure is taken from [64]. [64]**



(b) **Empirical Variograms for five modified aerosol regions. This was obtained using Polire.**

Figure 4: Comparing variograms across the author's and our implementation. As can be seen, the variograms we generated via Polire are consistent with that of the authors.

Implementation	IDW	Kriging	Spt. Avg.	Near. Nbr.
Wong et al [64]	53.3	49.4	49.5	49.2
Ours (Cal.)	51.1	51.2	48.8	48.5
Ours (S.Cal)	58.4	55.2	55.6	55.1

Table 3: Comparison of spatial interpolation algorithms' mean estimate in California for O_3. Cal. refers to California and S.Cal refers to Southern California. Our interpolation for the whole of California roughly corresponds to the values obtained by the authors. Near. Nbr. is Nearest neighbor and Spt. Avg. is Spatial Averaging. All values are in ppb.

also note that the semivariances values that we obtain are slightly different from those reported by the authors. We believe that this is due to us having to make approximations while dividing the US into aerosol regions, and because we were interpolating over many more block groups than the author. We note here that our patterns for Southeast were similar to that of the authors. However, we do not show the variogram for Southeast due to space constraints.
(4) **Comparison of interpolation algorithms**:
a) *Correlation Coefficients*: The authors compute the pairwise correlation amongst the different interpolation algorithms. They observe that the correlation values between the pairs of interpolation algorithms is generally high (0.80 - 0.97). They obtain the lowest correlation while comparing O_3 estimates between IDW and Kriging and between IDW and nearest neighbor interpolation. For most methods, they find that correlation values exceed 0.9. We observe similar trends in our implementation, though our lowest correlation is 0.86. Consistent with the observations of authors, our lowest correlations for O_3 are obtained with: IDW and N. Neighbor (0.86), IDW and Spatial Averaging (0.89) and IDW and Kriging (0.9).
b) *Range of Estimates*: The authors compare the range of estimates across all the interpolation algorithms within each block group. They do this by computing the difference in the maximum and minimum estimated values. They found O_3 values in California to be normally distributed, implying that the differences in interpolated values were moderately higher for a substantial percentage of the block groups. They find that the distribution is positively skewed for PM_{10}. As already mentioned in section 5.1.2, interpolation of O_3 by all the methods is limited to California. Therefore, to obtain the range estimates at other locations, the authors relax this condition of having all the algorithms' estimates to instead allow for at least two interpolation algorithm estimates to be present for the computation of range. They report that this resulted in a much more positively skewed distribution and also that more than 50% of block groups have a range < 4 ppb. We verify their observations by plotting the density estimates in Figure 3 for O_3. Our trends are consistent with what the authors report since initially, it is closer to a normal distribution in California but once the conditions are relaxed and the region of interpolation expanded, we get a positively skewed distribution.
c) *Leave-one-out cross validation*: The authors perform leave-one-out cross validation using the interpolation algorithms Kriging, Nearest

	Kriging (Theirs)	Kriging(Ours)	N. Neighbor (Theirs)	N. Neighbor (Ours)	Spt. Averaging (Theirs)	Spt. Averaging (Ours)
R^2	0.72	0.78	0.80	0.68	0.88	0.78
Coefficient	1.06	1.07	0.75	0.81	0.83	0.96
n	55	73	55	73	33	52

Table 4: Results from the Cross-Validation on O_3 data. Above n corresponds to number of samples and coefficient is the linear regression coefficient obtained by fitting the predictions and the ground truth. We can observe that our results are similar to the others for regression coefficients, but the R^2 values vary significantly.

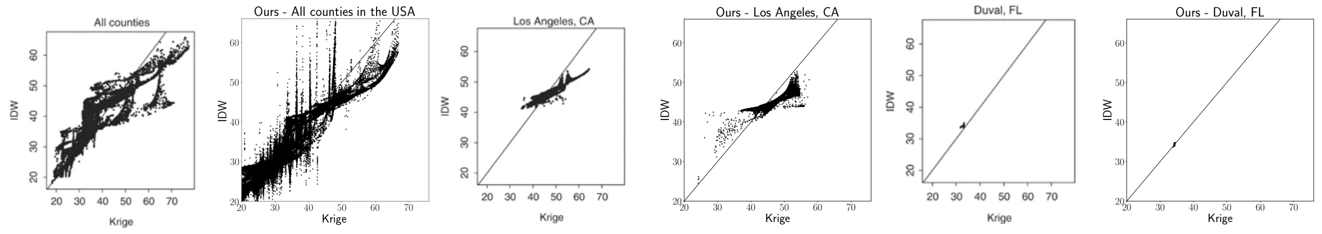


Figure 5: Scatter plots comparison between Kriged and IDW estimates for All counties, Los Angeles, CA and Duval, FL. From left, the 1st, 3rd and the 5th plots were generated by the authors. Corresponding to those plots are our plots, which are the 2nd, 4th and 6th plots from the left. We can see from our plots the while the overall correlation is high, there are discernible patterns that we can observe in different counties in our plots, confirming the observations of the authors.

Neighbor and Spatial Averaging. They regress interpolated values with true values and examine the R^2 value as well the coefficients obtained for regression. They report these results for O_3 in South California. They report that all three methods are minimally biased, though Kriging produces slightly less biased results than the other two methods. We report their results and our results in Table 4. We observe that while our regression coefficients are similar in the ordering (highest to lowest), our R^2 values vary significantly. We believe that this is due to the region we chose for Southern California. To the best of our knowledge, the authors do not report the exact counties for South California in their paper.

d) *Range of estimates on a small fraction of block groups with an NHANES subject*: In NHANES-III, the subjects are mapped to the nearest block group. In this particular experiment, the authors consider the 82 counties where NHANES-III data was available and each subject mapped to a block group was actually present in the block group. The authors study the range of the interpolation estimates within these block groups. Since we do not have access to this exact data, we do not conduct this experiment.

e) *Comparison of pairs of methods via scatter plots*: To better understand the relationships across different interpolation algorithms, the authors also compare estimated values of different algorithms via scatter plots. The authors observe that while there is a high correlation in the estimated values, there are internal structures that are discernible within the scatter plots. In their paper, the authors show the scatter plots for Kriging and IDW for PM_{10} (a) All counties (b) Duval and (c) Los Angeles. They choose these two counties to show examples of two distinct types of relationships that they observe. The patterns and scatter plots that they show, as well as what we observe can be seen in Figure 5. We obtain a similar overall trend that the authors obtain based on the interpolated values and our internal structures are consistent with what the authors report.

5.2 Replication of Sensor Placement Paper

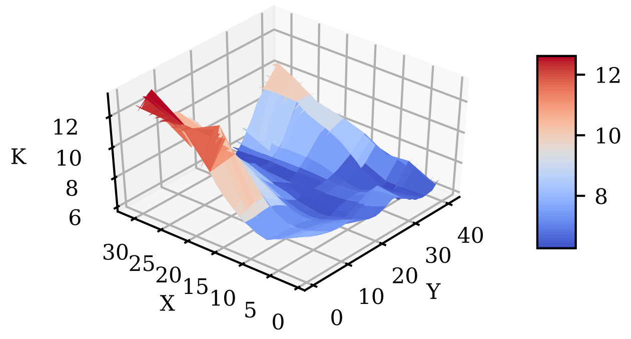
We now discuss our replication of sensor placement by Guestrin et al. [21].

5.2.1 Datasets. (1) **Berkeley Intel lab temperature dataset**: The dataset was collected from 54 sensors between 28th February and 5th April, 2004. in Berkeley Intel lab. The dataset contains humidity, temperature, and light sampled every 31 seconds. Starting from 28th February, we train on 3 days of data following the authors and test on the next day, 2nd*March*, 2004 of data. The dataset can be downloaded `here`.

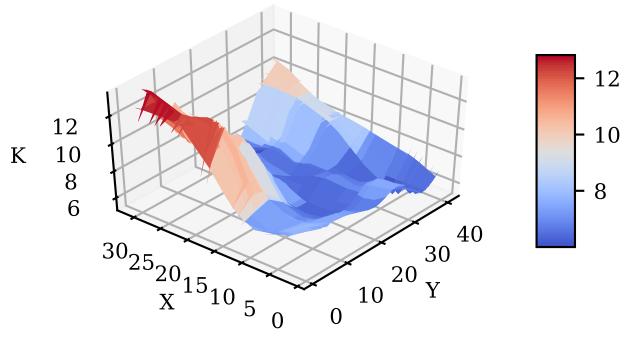
(2) **Precipitation dataset**: Our second dataset is precipitation data collected over 46 years (1949 – 94) in Washington and Oregon, USA by the Joint Institute for the Study of the Atmosphere (JISAO) and Ocean [62]. Daily precipitation was reported in 190 stations $\approx$ 50 km apart from each other. We use the first 15 years data for training and the subsequent five years data for testing. We do not go beyond 20 years due to anomalous and significant amounts of missing data. The dataset can be downloaded `here`.

5.2.2 Validation of NS kernel. We demonstrate the validity of the NS kernel using the temperature dataset. The authors Guestrin et al. [21] use the following procedure to test the efficacy of the kernel. They first compute the empirical covariance matrix Γ and learn the NS kernel using it. They then compare the 41st sensor's covariance entries of the NS kernel $K(s_{41}, *)$ with the interpolated empirical covariance vector $\Gamma(s_{41}, *)$ where s_{41} is location of the 41st sensor. We follow the same steps and use Kriging with spherical variogram as our interpolator. Our results in Figure 6 shows how the learnt NS kernel K has comparable characteristics with the interpolated empirical covariance matrix Γ. This implies that we have successfully modelled the non-stationary phenomena using

NS kernel. Note that stationary kernels would not be able to model these underlying phenomena easily.



(a) Interpolated emprirical covariance matrix for sensor 41.



(b) NS kernel K(x, *) where x is sensor 41.

Figure 6: Comparison of Empirical covariance matrix v/s non-stationary (NS) kernel. We observe that the interpolated empirical covariance matrix and the covariance matrix learnt by the NS kernel are very similar, demonstrating the efficacy of the NS kernel in modeling the underlying nonstationarity.

5.2.3 Experiments and Experimental setup. In the first experiment, we compare MI for sensor placement using Algorithm 1 versus random placements. For this purpose, we do 100 random placements of 30 sensors with different random states and compare the mean and maximum of MI with MI obtained using Algorithm 1. In the second experiment, we would like to strategically place sensors and obtain minimum interpolation error at unsensed locations. We would like to note that there are multiple instances of missing data in both datasets. For the temperature dataset, the sampling strategy to mitigate this problem was unclear to us from the paper[4]. After an exploratory literature review, prior work [16] showed that sampling rate of 22 minutes is optimal for Spatio-temporal modelling preserving characteristics of the original data, and we resample our data accordingly. For the precipitation dataset, we downsample the data to 60 days for similar reasons as the temperature dataset. We did not consider the data from 39 stations for not having a sufficient number of samples. Of the 190 stations, we finally have the data from about from 151 stations. Prior literature often leverages a pilot deployment to model the environment [23]. Our experiment has three phases: i) Training from a pilot deployment; ii) Placement of k sensors; iii) Testing (Interpolating) at remaining locations. Training phase includes learning kernel parameters of a Gaussian Process. We use both stationary and non-stationary kernels in our experiments. We use a Matern32 kernel for the stationary GP, after empirically observing that it yields better results during prediction when compared with other stationary kernels. We learn our hyperparameters using GPy [19]. We use the non-stationary kernel (NS kernel) that we implemented and learn the kernel parameters as described in Section 4.1.

In the placement phase, we place k sensors, $k \in \{1, 2, ..., 30\}$. We use entropy and MI as two different criteria to optimize the sensor placement. The candidate set of locations is where ground truth data is available for testing. We greedily place k sensors at the remaining locations using Algorithm 1. Our placement strategy is the same for stationary and non-stationary kernels as it is independent on the nature of the kernel. In the third phase, we interpolate at remaining locations to compute the RMSE (root mean squared error) in the predictions. For each value of k, $k \in \{1, 2, ..., 30\}$, we interpolate the values at remaining locations. We use the same kernel for interpolation and sensor placement to maintain consistency.

In the third experiment, we compare the geometry of placements using entropy v/s MI as a criterion for sensor placement.

5.2.4 Results and Analysis. We show the results and analysis of our experiments on the temperature and precipitation datasets in this section. Extensive results (which we could not mention here due to space constraints) are attached in appendix of our repository.

Temperature dataset

To verify that placements by the Greedy algorithm maximize MI, when compared with the random baseline, we show the comparison of MI at $\forall k \in \{1, 2, ..., 30\}$ in Figure 7 for Matern32 kernel. As explained in the earlier sections, higher MI implies a better quality of sensor placements. We can see in Figure 7b that Greedy algorithm yields higher MI than mean and even maximum MI yielded by random baseline algorithm at all values of k. We observe the same phenomena in case of NS kernel in Figure 7a.

We show the results of interpolation after k placements for the Matern32 and NS kernel in Figures 8a and Figure 8b. We were able to achieve similar RMSE values as the authors. The authors also report that MI yields better predictions (and therefore less RMSE) when compared with entropy in most cases for the NS kernel. For NS kernel, MI is outperforming entropy 60% of the time yielding a nearly equal range of RMSE. We believe that small variations in our results are due to the differences in experimental settings (based on our understanding of the original paper).

We can see from Figure 9 that entropy is placing sensors near to the boundary of the room whereas MI is placing sensors central to the unsensed locations.

Precipitation dataset

We compare MI for Greedy algorithm and random for precipitation dataset. We observed similar results as in the earlier case with

[4]We contacted the authors to understand the setting better. However, we did not receive a response.

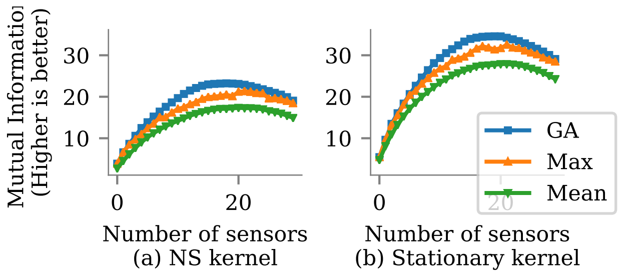


Figure 7: Mutual Information: Random baseline v/s Algorithm 1 for the temperature dataset. Greedy algorithm (GA) always achieves better MI comparing either with the maximum (Max) or mean (Mean) of random placements.

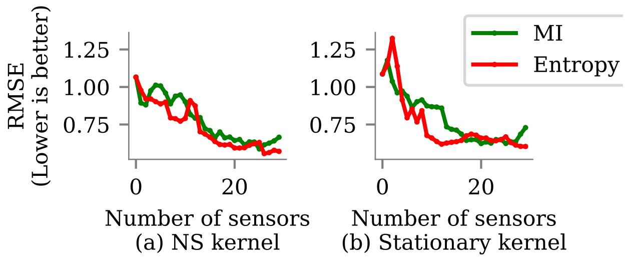


Figure 8: RMSE for interpolation using Gaussian process regression at k placements, $k \in \{1, 2, 3, ..., 30\}$(Temperature dataset). Errors are comparable with similar results obtained by Guestrin et al. [21].

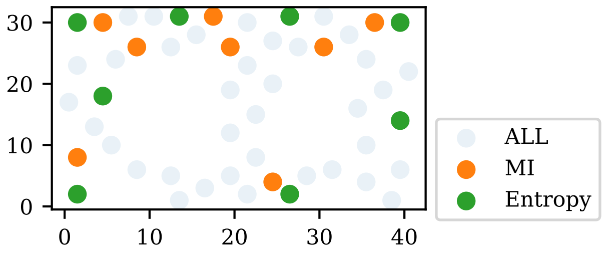


Figure 9: 8 sensors placed with Greedy algorithm using MI and entropy with temperature dataset. Entropy has placed sensors near to the boundary. MI has placed sensors centrally.

Greedy algorithm yielding higher MI than mean and maximum of the random placement for both Matern32 and NS kernel. We note here that the authors do not report results for this experiment, and therefore owing to this and space constraints, we do not show these plots in the paper. For the interpolation experiment, RMSE for Stationary and NS kernel is shown in Figure 10. NS kernel is noisy in the beginning but gets accurate with increment in the number of sensors.

6 LIMITATIONS AND FUTURE WORK

Polire is currently not optimized for performance. While we vectorize our implementations, we currently do not take advantage

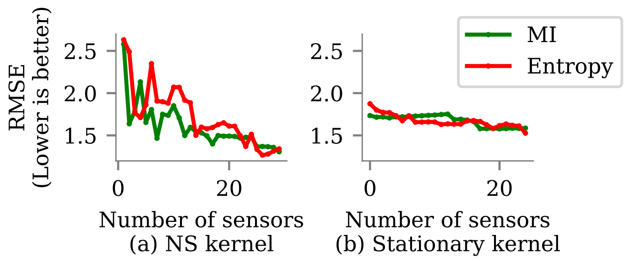


Figure 10: RMSE for interpolation using Gaussian process regression at k placements, $k \in \{1, 2, 3, ..., 30\}$ (Precipitation dataset). MI outperforms Entropy for NS kernel. Errors variate most by 1 mm compared to similar results achieved by Guestrin et al. [21].

of hardware accelerators like GPU. As an example, there are faster implementations of algorithms like Natural Neighbor [5] that we do not implement currently.
As part of future work, we plan to do an exploratory user study to understand the efficacy of our package. We will try to address the difficulties or problems that arise while using our toolkit. We plan to do this by conducting a set of well-designed experiments. We would also like to create a reproducibility challenge, similar to those conducted in the machine learning conferences such as ICLR and NeurIPS. We believe reproducibility challenges are a great way to promote more openness and clarity in research.

7 CONCLUSION

In this paper, we proposed Polire: An open-source spatial interpolation and sensor placement toolkit, implemented in Python. As part of our toolkit, we implement several widely used interpolation algorithms as well as state-of-the-art sensor placement algorithms. By doing so, we address the limitations of existing commercial tools by creating an open-sourced implementation of several algorithms and help reduce the barrier of entry into this field. Besides, we also implement a generalized non-stationary kernel to model environmental phenomena. We demonstrate the efficacy of our toolkit by replicating results successfully from influential research papers. We hope that Polire and similar tools will become an important step towards reproducible research and help researchers focus on science than individually implementing baselines.

REFERENCES

[1] 2017. *Reproducibility '17: Proceedings of the Reproducibility Workshop*. Association for Computing Machinery, New York, NY, USA.

[2] Andreas Lichtenstern. 2013. Kriging methods in spatial statistics. *Bachelors' Thesis* (2013).

[3] Noman Bashir, Dong Chen, David Irwin, and Prashant Shenoy. 2019. Solar-TK: A Solar Modeling and Forecasting Toolkit. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '19)*. Association for Computing Machinery, New York, NY, USA, 372. https://doi.org/10.1145/3360322.3361006

[4] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. 2014. NILMTK: An Open Source Toolkit for Non-intrusive Load Monitoring. In *Fifth International Conference on Future Energy Systems*. Cambridge, UK. https://doi.org/10.1145/2602044.2602051 arXiv:1404.3878

[5] Alex Beutel, Thomas Mølhave, and Pankaj K. Agarwal. 2010. Natural Neighbor Interpolation Based Grid DEM Construction Using a GPU. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '10)*. Association for Computing Machinery, New York, NY, USA, 172–181. https://doi.org/10.1145/1869790.1869817

[6] Niels Brouwers, Marco Zuniga, and Koen Langendoen. 2014. NEAT: A Novel Energy Analysis Toolkit for Free-Roaming Smartphones. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems (SenSys '14)*. Association for Computing Machinery, New York, NY, USA, 16–30. https://doi.org/10.1145/2668332.2668337

[7] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. 2013. API design for machine learning software: experiences from the scikit-learn project. (2013).

[8] William F Caselton and James V Zidek. 1984. Optimal monitoring network designs. *Statistics & Probability Letters* 2, 4 (1984), 223–227.

[9] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, Yilong Li, Ji Jia, and Xiaofan Jiang. 2014. AirCloud: a cloud-based air-quality monitoring system for everyone. In *SenSys '14*.

[10] Thomas M Cover, Peter E Hart, et al. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory* 13, 1 (1967), 21–27.

[11] N Cresssie. 1991. Statistics for spatial data.

[12] J. P. Delhomme. 1978. Kriging in the hydrosciences. *Advances in Water Resources* 1, 5 (Jan. 1978), 251–266. https://doi.org/10.1016/0309-1708(78)90039-8

[13] Jian Ding and Ranveer Chandra. 2019. Towards Low Cost Soil Sensing Using Wi-Fi. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 39, 16 pages. https://doi.org/10.1145/3300061.3345440

[14] Wan Du, Zikun Xing, Mo Li, Bingsheng He, Lloyd Hock Chye Chua, and Haiyan Miao. 2014. Optimal Sensor Placement and Measurement of Wind for Water Quality Studies in Urban Reservoirs. In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks (IPSN '14)*. IEEE Press, 167–178.

[15] ESRI. 2020. How Topo to Raster Works, ESRI ArcGIS. https://pro.arcgis.com/en/pro-app/tool-reference/3d-analyst/how-topo-to-raster-works.htm#GUID-11C3CA25-3558-415D-90B4-84BE448E5400

[16] Sahil Garg, Amarjeet Singh, and Fabio Ramos. 2012. Learning non-stationary space-time models for environmental monitoring. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[17] Sean Gillies et al. 2007–. Shapely: manipulation and analysis of geometric objects. https://github.com/Toblerity/Shapely

[18] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation* 101, 23 (2000), e215–e220. https://doi.org/10.1161/01.cir.101.23.e215

[19] GPy. since 2012. GPy: A Gaussian process framework in python. http://github.com/SheffieldML/GPy.

[20] Chris Greenhalgh, Shahram Izadi, James Mathrick, Jan Humble, and Ian Taylor. 2004. ECT: A Toolkit to Support Rapid Construction of Ubicomp Environments System Support for Ubiquitous Computing Workshop. *University of Illinois at Urbana Champaign, Nottingham, UK* (2004).

[21] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. 2005. Near-Optimal Sensor Placements in Gaussian Processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*. Association for Computing Machinery, New York, NY, USA, 265–272. https://doi.org/10.1145/1102351.1102385

[22] Vitor Guizilini and Fabio Ramos. 2015. A Nonparametric Online Model for Air Quality Prediction. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 651–657.

[23] Hsun-Ping Hsieh, Shou-De Lin, and Yu Zheng. 2015. Inferring Air Quality for Station Location Recommendation Based on Urban Big Data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. Association for Computing Machinery, New York, NY, USA, 437–446. https://doi.org/10.1145/2783258.2783344

[24] Natsuki Ikeda, Ryo Shigeta, Junichiro Shiomi, and Yoshihiro Kawahara. 2020. Soil-Monitoring Sensor Powered by Temperature Difference between Air and Shallow Underground Soil. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1, Article 13 (March 2020), 22 pages. https://doi.org/10.1145/3380995

[25] Neal Jackson, Joshua Adkins, and Prabal Dutta. 2018. Reconsidering Batteries in Energy Harvesting Sensing. In *Proceedings of the 6th International Workshop on Energy Harvesting Energy-Neutral Sensing Systems (ENSsys '18)*. Association for Computing Machinery, New York, NY, USA, 14–18. https://doi.org/10.1145/3279755.3279757

[26] Jack Kelly, Nipun Batra, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. 2014. NILMTK v0.2: A Non-Intrusive Load Monitoring Toolkit for Large Scale Data Sets: Demo Abstract. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings (BuildSys '14)*. Association for Computing Machinery, New York, NY, USA, 182–183. https://doi.org/10.1145/2674061.2675024

[27] Andreas Krause, Jure Leskovec, Carlos Guestrin, Jeanne VanBriesen, and Christos Faloutsos. 2008. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management* 134, 6 (2008), 516–526.

[28] Andreas Krause, Ram Rajagopal, Anupam Gupta, and Carlos Guestrin. 2009. Simultaneous Placement and Scheduling of Sensors. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks (IPSN '09)*. IEEE Computer Society, USA, 181–192.

[29] Andreas Krause, Ajit Singh, and Carlos Guestrin. 2008. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research* 9, Feb (2008), 235–284.

[30] Nina Siu-Ngan Lam. 1983. Spatial interpolation methods: a review. *The American Cartographer* 10, 2 (1983), 129–150.

[31] Andreas Lichtenstern. 2013. Kriging methods in spatial statistics. (2013).

[32] Brian Y. Lim and Anind K. Dey. 2010. Toolkit to Support Intelligibility in Context-Aware Applications. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing (UbiComp '10)*. Association for Computing Machinery, New York, NY, USA, 13–22. https://doi.org/10.1145/1864349.1864353

[33] Ning Liu, Yue Wang, Jiayi Huang, Rui Ma, and Lin Zhang. 2019. Enhanced Air Quality Inference with Mobile Sensing Attention Mechanism: Poster Abstract. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems (SenSys '19)*. Association for Computing Machinery, New York, NY, USA, 418–419. https://doi.org/10.1145/3356250.3361938

[34] Ning Liu, Yue Wang, Jiayi Huang, Rui Ma, and Lin Zhang. 2019. Enhanced Air Quality Inference with Mobile Sensing Attention Mechanism: Poster Abstract. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems (SenSys '19)*. Association for Computing Machinery, New York, NY, USA, 418–419. https://doi.org/10.1145/3356250.3361938

[35] Xinyu Liu, Xiangxiang Xu, Xinlei Chen, Enhan Mai, Hae Young Noh, Pei Zhang, and Lin Zhang. 2017. Individualized Calibration of Industrial-Grade Gas Sensors in Air Quality Sensing System. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys '17)*. Association for Computing Machinery, New York, NY, USA, Article 74, 2 pages. https://doi.org/10.1145/3131672.3136998

[36] George Y Lu and David W Wong. 2008. An adaptive inverse-distance weighting spatial interpolation technique. *Computers & geosciences* 34, 9 (2008), 1044–1055.

[37] Benjamin Murphy, Sebastian Müller, and Roman Yurchak. 2020. *GeoStat-Framework/PyKrige: v1.5.0.* https://doi.org/10.5281/zenodo.3739879

[38] David J. Nott and William T. M. Dunsmuir. 2002. Estimation of nonstationary spatial covariance structure. *Biometrika* 89, 4 (12 2002), 819–829. https://doi.org/10.1093/biomet/89.4.819 arXiv:https://academic.oup.com/biomet/article-pdf/89/4/819/667332/890819.pdf

[39] R. A. Olea. 1999. *Geostatistics for engineers and earth scientists.* Boston (Mass.) : Kluwer academic. http://lib.ugent.be/catalog/rug01:000523946

[40] Travis E Oliphant. 2006. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA.

[41] MA Oliver and R Webster. 2014. A tutorial guide to geostatistics: Computing and modelling variograms and kriging. *Catena* 113 (2014), 56–69.

[42] Margaret A Oliver and Richard Webster. 1990. Kriging: a method of interpolation for geographical information systems. *International Journal of Geographical Information System* 4, 3 (1990), 313–332.

[43] Amitangshu Pal and Krishna Kant. 2019. Water Flow Driven Sensor Networks for Leakage and Contamination Monitoring in Distribution Pipelines. *ACM Trans. Sen. Netw.* 15, 4, Article 37 (Aug. 2019), 43 pages. https://doi.org/10.1145/3342513

[44] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[47] Naren Ramakrishnan, Chris Bailey-Kellogg, Satish Tadepalli, and Varun N Pandey. 2005. Gaussian processes for active data mining of spatial aggregates. In *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 427–438.

[48] Nithya Ramanathan, Thomas Schoellhammer, Eddie Kohler, Kamin Whitehouse, Thomas Harmon, and Deborah Estrin. 2009. Suelo: Human-Assisted Sensing for Exploratory Soil Monitoring Studies. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*. Association for Computing Machinery, New York, NY, USA, 197–210. https://doi.org/10.1145/1644038.1644058

[49] Robert G Raskin, Christopher C Funk, Scott R Webber, and Cort J Wilmott. 1997. Spherekit: The Spatial Interpolation Toolkit (97-4). (1997).

[50] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

[51] Slim Rekhis, Nourhene Ellouze, and Noureddine Boudriga. 2012. A Wireless Sensor Network Based Water Monitoring System. In *Proceedings of the 8h ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet '12)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/2387218.2387225

[52] Jia Shao, Wei Meng, and Guodong Sun. 2017. Evaluation of Missing Value Imputation Methods for Wireless Soil Datasets. *Personal Ubiquitous Comput.* 21, 1 (Feb. 2017), 113–123. https://doi.org/10.1007/s00779-016-0978-9

[53] Donald Shepard. 1968. A Two-Dimensional Interpolation Function for Irregularly-Spaced Data. In *Proceedings of the 1968 23rd ACM National Conference (ACM '68)*. Association for Computing Machinery, New York, NY, USA, 517–524. https://doi.org/10.1145/800186.810616

[54] Michael C Shewry and Henry P Wynn. 1987. Maximum entropy sampling. *Journal of applied statistics* 14, 2 (1987), 165–170.

[55] Roger W Sinnott. 1984. Virtues of the Haversine. *S&T* 68, 2 (1984), 158.

[56] Adam Tyler, Oliver Bates, Adrian Friday, and Mike Hazas. 2019. A Toolkit for Low-Cost Thermal Comfort Sensing. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '19)*. Association for Computing Machinery, New York, NY, USA, 348–349. https://doi.org/10.1145/3360322.3360994

[57] Muhammad Umer, Lars Kulik, and Egemen Tanin. 2008. Kriging for Localized Spatial Interpolation in Sensor Networks. In *Scientific and Statistical Database Management*, Bertram Ludäscher and Nikos Mamoulis (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 525–532.

[58] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* 13, 2 (2011), 22.

[59] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[60] S. Vogl, P. Laux, W. Qiu, G. Mao, and H. Kunstmann. 2012. Copula-based assimilation of radar and gauge information to derive bias-corrected precipitation fields. *Hydrology and Earth System Sciences* 16, 7 (2012), 2311–2328. https://doi.org/10.5194/hess-16-2311-2012

[61] Péter Völgyesi, András Nádas, Xenofon Koutsoukos, and Ákos Lédeczi. 2008. Air Quality Monitoring with SensorMap. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN '08)*. IEEE Computer Society, USA, 529–530. https://doi.org/10.1109/IPSN.2008.50

[62] Martin Widmann and Christopher S Bretherton. 2000. Validation of mesoscale precipitation in the NCEP reanalysis using a new gridcell dataset for the northwestern United States. *Journal of Climate* 13, 11 (2000), 1936–1950.

[63] David W Wong, Lester Yuan, and Susan A Perlin. 2004. Comparison of spatial interpolation methods for the estimation of air quality data. *Journal of Exposure Science and Environmental Epidemiology* 14, 5 (2004), 404.

[64] David W Wong, Lester Yuan, and Susan A Perlin. 2004. Comparison of spatial interpolation methods for the estimation of air quality data. *Journal of Exposure Science & Environmental Epidemiology* 14, 5 (Sept. 2004), 404–415. https://doi.org/10.1038/sj.jea.7500338

[65] Min Wu, Jiayi Huang, Ning Liu, Rui Ma, Yue Wang, and Lin Zhang. 2018. A Hybrid Air Pollution Reconstruction by Adaptive Interpolation Method. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys '18)*. Association for Computing Machinery, New York, NY, USA, 408–409. https://doi.org/10.1145/3274783.3275207

[66] Xiuwen Yi, Junbo Zhang, Zhaoyuan Wang, Tianrui Li, and Yu Zheng. 2018. Deep Distributed Fusion Network for Air Quality Prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining (KDD '18)*. Association for Computing Machinery, New York, NY, USA, 965–973. https://doi.org/10.1145/3219819.3219822

[67] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. 2013. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1436–1444.

[68] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. 2013. U-Air: When Urban Air Quality Inference Meets Big Data. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. Association for Computing Machinery, New York, NY, USA, 1436–1444. https://doi.org/10.1145/2487575.2488188

[69] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. 2015. Forecasting Fine-Grained Air Quality Based on Big Data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. Association for Computing Machinery, New York, NY, USA, 2267–2276. https://doi.org/10.1145/2783258.2788573

[70] Zhengyuan Zhu and Michael L Stein. 2006. Spatial sampling design for prediction with estimated parameters. *Journal of agricultural, biological, and environmental statistics* 11, 1 (2006), 24.

[71] Dale L Zimmerman. 2006. Optimal network design for spatial prediction, covariance parameter estimation, and empirical prediction. *Environmetrics: The official journal of the International Environmetrics Society* 17, 6 (2006), 635–652.