UTRECHT UNIVERSITY

Faculty of Science

Department of Mathematics

Multi-fidelity spatial regression for air temperature predictions using first, second and third party data

1st July 2022

Author:
Daniëlle van Beekvelt
5953669

Supervisors:
Prof. Dr. Ir. Jason Frank
Utrecht University

Dr. Ir. Jouke de Baar
Dr. Irene Garcia-Martí
KNMI

Koninklijk Nederlands
Meteorologisch Instituut
Ministerie van Infrastructuur en Waterstaat

Abstract

There has been a growing interest in using second and third party data in addition to first party data to make weather predictions. This development is necessary, because it can provide predictions for a higher spatial resolution than the predictions made using only first party data. Consequently, weather phenomena with a high spatial variability such as rain and wind could be predicted more accurately. However, it has not been shown before that using WOW-data as third party data leads to accurate predictions. In this thesis a modified version of the interpolation method of Kriging will be used to demonstrate the value of WOW-data in weather predictions. The Kriging procedure is modified such that it will be able to work with noisy data and such that it can differentiate between the systematic and random errors in first, second and third party data. The robustness of the method is examined by using synthetic data such that the model can be tested for weather of various spatial variabilities. These tests show that up to a certain spatial variability first, first and second, and first, second and third party data perform equally well. However, after a certain threshold is reached ,first and second, and first, second and third party data perform better than just first party data.

Keywords: Kriging, Gaussian Processes, third-party data

Contents

Chapter 1

Introduction

The Royal Netherlands Meteorological Institute (KNMI) is the national information and research facility for meteorology, climate and seismology. Two of the KNMI's core tasks are weather forecasting and climate monitoring. In order to perform these tasks, observations are needed. Previously, the KNMI almost exclusively relied on observations from their own observational networks, such that the quality of observations would be up to the standards of the World Meteorological Organisation (WMO). The data gathered by the KNMI will be referred to as 'first party data' or '1PD'. However, the spatial sparsity of the official network implies that large parts of the country remain unobserved which might be detrimental to monitor local weather conditions [1].

In this thesis, we intend to use data from two different sources in addition to the KNMI observations to make higher resolution weather forecasts with a nation-wide reduction in prediction uncertainty. The first additional source is data from Rijkswaterstaat, a part of the Dutch Ministry of Infrastructure and Water Management, which will be referred to as 'second party data' or '2PD'. Rijkswaterstaat uses weather stations next to roads to, for example, keep track of possible formation of black ice. We assume that Rijkswaterstaat's data is not up to the WMO's standards, but we consider it trustworthy enough to use [1]. The second additional source will be called 'third party data' or '3PD' and consists of a network of personal weather stations (PWS) that are part of the Weather Observations Website (WOW) initiative, commenced by the UK Met Office in 2011 and joined by the KNMI in 2015. PWS are stations installed by citizens with an interest in monitoring weather in their private spaces, such as their home or at school. PWS monitor weather phenomena such as temperature, rain, pressure and wind. These devices are provided by commercial manufacturers, hence the price range and the quality of the sensors used in the stations are highly variable. In addition, PWS are often installed in urban and peri-urban areas, locations that seldom meet the WMO's standards for station siting [1].

Using second and especially third party data offers a huge potential increase in the spatial density of the weather station network. Being able to operate at a higher spatial resolution opens the door for a wide range of new applications and operational services at KNMI. Two of these activities that 3PD can contribute to are related to advancing towards a high-resolution weather forecast (e.g. local rainfall and wind gusts) and the increased capacity of issuing weather warnings for severe weather conditions (e.g. hailstorms). This thesis illustrates how the combination of 1PD with 2PD and 3PD data might help creating such high-resolution services in the future. In addition, the work carried out in this research is the first demonstration of value for WOW data, which might encourage and propel new research lines in this 3PD direction. Nevertheless, the combination of data with different quality levels entangles a number of analytical challenges. In this work, these challenges are modelled with multi-fidelity spatial interpolation techniques, which are properly adapted to incorporate data of variable quality. Therefore, this thesis focuses on the application of a multi-fidelity spatial interpolation method to air temperature observations coming from three quality-variable monitoring networks. The objective of this analysis is demonstrating the potential of alternative weather observations to devise high-resolution interpolations that can expand the existing services and motivate new ones.

For a more mathematical description of the problem, some notation must first be introduced. Random variables will be denoted with capital letters such as X, Y, fixed or observed quantities by lower-case letters such as x, y. Maximum likelihood estimators will be denoted by letters with a circumflex like $\hat{x}, \hat{y}$. Matrices and vectors will be denoted using bold letters like $\mathbf{A}, \mathbf{B}$ and $\mathbf{v}, \mathbf{w}$, respectively. In addition, $p()$ is used for probability density functions and $p(a|b)$ for the probability density function of a conditional upon b.

Now the problem can be described as the estimation of a function

$$g(\mathbf{x}) : \mathcal{D} \to \mathbb{R} \tag{1.1}$$

representing, for example, the temperature in a geographical region $\mathcal{D} \in \mathbb{R}^2$, for instance the Netherlands. The estimation

$$f(\mathbf{x}) : \mathcal{D} \to \mathbb{R} \tag{1.2}$$

is made based on given discrete data obtained from measurements, where the quality of measurements is uncertain. Let n be the number of weather stations and $i \in \{1, \cdots, n\}$ be the index of the i-th station. We distinguish between three classes of stations which we refer to as 1PD, 2PD and 3PD,

$$S_k = \{i \mid i \text{ is a station providing } k\text{th party data}\}\ ,\ k = 1, 2, 3. \tag{1.3}$$

The given discrete data consists out of a set of pairs $\{(x_i, y_i)\}$, $i = 1, ..., n$, where $\mathbf{x}_i \in \mathcal{D}$ is the coordinate of station i expressed in longitude and latitude. The measured temperature data $\mathbf{y}_i \in \mathbb{R}$ is given by $\mathbf{y}_i = g(\mathbf{x}_i) + \epsilon_i$, where ϵ_i is a measurement error that follows a normal distribution. This measurement error consists of a systematic error b, called the bias and a random error σ_n, called the noise. We assume that b and σ_n are 0 for $i \in S_1$. The challenge is making the right estimations for b and σ_n for 2PD and 3PD. The given data combined with the estimates for b and σ_n will be used to make the prediction $f(\mathbf{x})$ of $g(\mathbf{x})$ whose prediction uncertainty needs to be small and uniformly bounded.

In the next chapter, the data used for the implementation of the interpolation method will be discussed in more detail. This is followed by a chapter where the choice of method will be motivated and where the mathematical background of the prediction method is explained in detail. After this has been discussed, there will be an explanation of the implementation in Python. Finally, the results of the synthetic and real data experiments will be shown and discussed.

Chapter 2

Data

In the implementation of the multi-fidelity spatial interpolation method, temperature observations are used. In the following section, the geographical locations of the stations that measure the temperature will be discussed. We have chosen to use air temperature data, because there is an experimental quality control available for air temperature observations for 3PD. The quality control will be explained in section 2.2. In addition, since a new method is being implemented it is easier to test it with a phenomenon such as temperature that is quite homogeneous in space and time. The robustness of the method will be tested on synthetic data that will be introduced in the last section of this chapter.

2.1 Description of geographic area

The KNMI operates a network of 35 lands stations that are relatively uniformly distributed over the country see Figure 2.1a, which ensures monitoring capabilities on the national scale. However, the visual inspection of the image shows that large regions between stations have a sub-optimal coverage, thus local weather remains unseen by the official network. Rijkswaterstaat operates sensor networks that are installed along the road network, see Figure 2.1b. The WOW network is located throughout the country, but tends to be clustered around urban and peri-urban environments as shown in Figure 2.1c.

In this thesis, we selected the date of 25th of January of 2019 to conduct the analysis. On that day, there were 35 KNMI stations, 319 Rijkswaterstaat stations, and 409 WOW stations yielding observations. Note that the 3PD stations have undergone a quality control procedure that will be further explained in section 2.2, thus the stations present on the selected date have the highest quality possible. To use these stations to make a prediction for the Netherlands, the country was divided in a grid of a 100 by 100 points, with a mesh size of approximately 4 by 4 kilometres, see Figure 2.1d. This might be still considered a rather large spatial resolution, but it is necessary due to computational constraints.

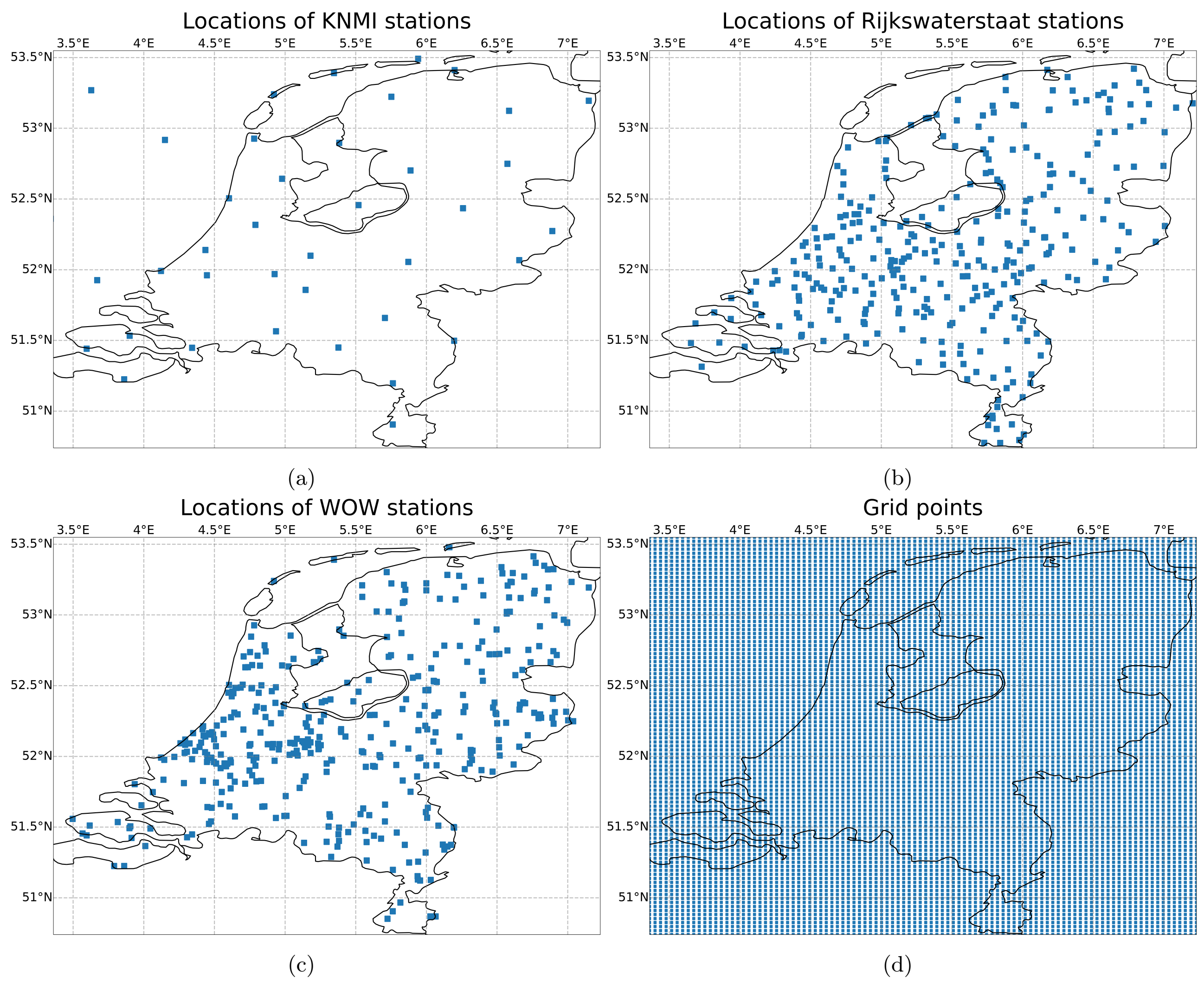

Locations of KNMI stations
Locations of Rijkswaterstaat stations
(a)
(b)
Locations of WOW stations
Grid points
(c)
(d)
3.5°E
4°E
4.5°E
5°E
5.5°E
6°E
6.5°E
7°E
53.5°N
53°N
52.5°N
52°N
51.5°N
51°N

Figure 2.1: The figures 2.1a, 2.1b and 2.1c show the locations of the weather stations from each of the three parties. The KNMI has 10 more stations in the sea that are not shown here, however we will mask the sea because we are interested in the predictions on land. In Figure 2.1d the grid used for the predictions is shown.

2.2 Quality control

In the implementation of the experiments data from the KNMI, Rijkswaterstaat and WOW is used. It is assumed that the measurements from the KNMI are up to the WMO's standards, i.e. it is assumed that the KNMI measurements contain no bias or noise. In addition it is assumed that Rijkswaterstaat's data can be considered trustworthy enough for our use, in other words the bias and noise in the Rijkswaterstaat data are assumed to be acceptable [1]. However, WOW is a platform that allows everyone to share their weather data. These observations come from devices of varying quality and are often placed at urban locations that do not adhere to the WMO's standards, therefore it is unknown how accurate their data is. To ensure that the used WOW data is of sufficient quality, a quality control is carried out. The quality control procedure used is based on the work of Napoly et al., 2018 and consists of 4 steps M1 up to M4 [2].

Let n be the number of stations and m be the number of measurements per station. Let T be a matrix of size $n \times m$ that represents the data set. Each row represents one station and each column is one time step. Missing values or values flagged by the quality control procedure are filled in as 'not a number' (NaN).
The first level $M1$ makes use of the metadata, for example abnormalities in locations are taken into account. In some cases the PWS are improperly installed and were by default assigned a location based on the IP address of the wireless network, which led to stations having the same longitude and latitude. The data from these stations are set to NaN, because they are not properly set up [2].
In $M2$ the outliers are detected and masked. The detection is done by using a modified version of the z-score. The exact workings of this procedure is beyond the scope of this thesis so for more information we refer to [2]. After these suspicious measurements are detected and masked, we move on to M3.
During this step it is checked if there is enough daily and monthly coverage. To verify if there is enough coverage the percentage of measurements that were flagged in step M2 is taken into account, when 20% or more measurements from one station in a specified time frame are flagged it is assumed that the station is too erroneous to use and therefore all data from that station is set to NaN.
In the end at step M4, we aim to filter out the stations that are indoors. The measurements from these stations need to be removed, because the aim is to model the weather and measurements from indoor locations might not capture the weather well. For example, during the winter the temperature measurements could be substantially higher indoors due to heating. The filtering is done by computing the Pearson correlation coefficient P_c between each station and the median of the WOW data for each month. If this coefficient is lower than 0.9 for a station in a given month, then the measurements for that station are set to NaN for that month. This approach allows to filter out the indoor systems, because it is assumed that the indoor stations are less correlated to the outdoor stations and thus less correlated to the median of the WOW data [2].

After the quality control is carried out, there are only stations left that can be used for the computations. One of the limitations of this quality control is that it does not apply any type of bias correction, thus the observations received for the multi-fidelity analysis will have some errors. Dealing with these observational errors is an important aspect of this research project.

2.3 Synthetic data

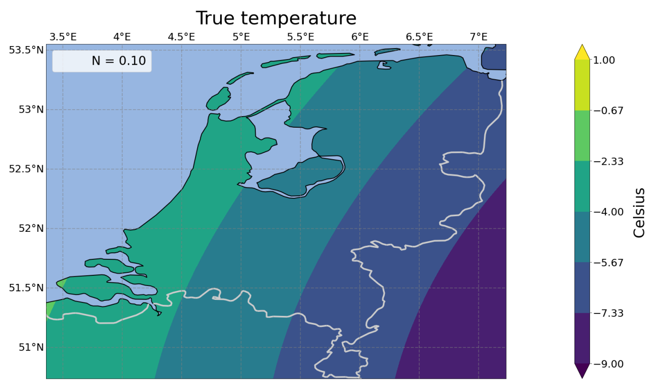
In addition to real data, synthetic data was also used for some experiments. This was done because it allows us to look at what happens to the predictions for temperature fields $g(\mathbf{x})$ with different spatial variability. With the real data, the model is tested for one day, but by generating synthetic weather patterns it can be seen how robust the procedure is. If the model is shown to be robust and that it works for functions $g(\mathbf{x})$ with high spatial variability, the model could also work for fine-grained weather phenomena such as wind or rain.
Synthetic data is used to test the robustness, because not only temperature values for the stations can be generated, but for the grid points as well. Therefore, the prediction accuracy can be checked by looking at the difference between the prediction and the true values for grid points.

Let $\mathbf{y}^s$ and $g(\mathbf{x}^g)$ denote the temperatures generated for the stations and the grid points ,respectively, and let $\mathbf{x}^s$ and $\mathbf{x}^g$ denote the station and grid point locations ,respectively, given by their longitude and latitude,

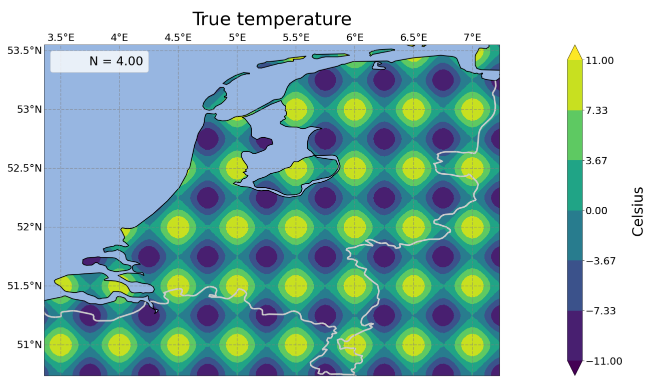
$$\mathbf{y}^s = a(\cos N\pi\mathbf{x}_1^s + \cos N\pi\mathbf{x}_2^s) + \epsilon$$
$$g(\mathbf{x}^g) = a(\cos N\pi\mathbf{x}_1^g + \cos N\pi\mathbf{x}_2^g).$$

This $g(\mathbf{x}^g)$ results in a temperature field that follows a lattice pattern, where the size of the lattice fields depend on N, and N is a parameter that determines how many oscillations per degree longitude and latitude there are. Furthermore, we multiply everything by amplitude a that was determined by looking at the amplitude of real measured temperature values, so that the minimum and maximum values of $\mathbf{y}^s$ and $g(\mathbf{x}^g)$ follow more closely to the maxima and minima of real temperature data. In addition, a synthetic error ϵ is added to $\mathbf{y}^s$. In consists of a systematic error value and random error value for the second and third party stations. These factors are determined by running the Kriging procedure that is described in section 3.4 and observing what values were predicted for the bias and noise. For the bias -3.22°C and -2.46°C were added to the second and third party stations respectively and 0.57°C and 1.62°C for the noise. Note that these bias and noise values were determined using data from the 25th of January, these values might differ for other days.

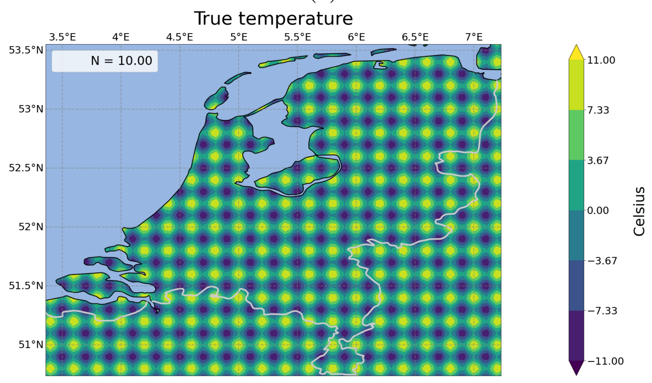
The resulting temperature fields $g(\mathbf{x}^g)$ are shown for various N's in Figure 2.2. Temperature data has a low spatial variability, so it is expected to be similar to Figure 2.2a and wind data is expected to show a higher spatial variability like Figure 2.2b.



(a)



(b)



(c)

Figure 2.2: The temperature maps for the synthetic data follow a lattice like pattern. When N increases lattice becomes smaller and the pattern shows a higher spatial variability.

The goal of using synthetic data is to show that for weather patterns that show low spatial variability, the 1PD stations alone suffice to make accurate predictions, but when the weather pattern has higher spatial variability, using 2PD and 3PD data is beneficial. To use all of this data for making a prediction, we need a method that is able to predict values of new data points given a set of known data. In the next chapter a few possible methods will be addressed and one of them will be discussed more thoroughly.

Chapter 3

Methodology

In order to analyse the data discussed in chapter 2, a suitable prediction method is necessary. In this section, multiple prediction methods are compared, and the most suitable interpolation method is extended to work similar to a regression method. This is necessary to account for the errors that we assume are present in the measured data.

3.1 Interpolation and regression

Interpolation and regression are both methods that help us predict the value of new data points when given a discrete set of known data points. Although they both provide a continuous approximation to discrete data, there is an important difference. For interpolation, all data points need to fit exactly, whereas this is not necessarily required for regression [3, 4]. For interpolation, a set of data points is given and it is assumed that this is perfect data, i.e. it contains no errors. The goal of interpolation is to find a single-valued curve that passes through all of those given data points. For example, if $(n + 1)$ data points are given, polynomial interpolation can be used to find a polynomial of order n that passes through all $n + 1$ points exactly [4]. On the other hand, if one tries to fit a polynomial of $n - 1$ it is over-constrained and finding a best fit leads to regression. So in this way interpolation methods can be extended to regression methods when much data is available.
Regression is a method that estimates the relationships between a dependent variable and independent variables; it computes a curve or surface that fits best to the known data points. For regression it is assumed that the known data is noisy, therefore regression does not aim to find a perfect fit for the known data. It also assumes some knowledge of the functional relationship, for example if the relationship is linear or exponential. An example of a linear method is ordinary least squares, this method computes a straight line that minimises the sum of squared errors [3].

A spatial prediction needs to be made, so we are interested in spatial interpolation methods. Spatial interpolation is a method that uses known data to estimate unknown values at locations where there is no known data available. Spatial interpolation assumes that the surface that needs to be modelled is continuous over space, which means that the surface does not have well-defined boundaries and can be seen in the entire area of interest and smoothly transitions from value to value. Examples of continuous fields could be temperature or relative humidity [5]. As a result, we can estimate the values at any location within our spatial boundaries, in our case any location within the Netherlands. Something else that is assumed, is that the attribute we are looking at is locally correlated. In other words, values that are spatially close together are more likely to be similar than values that are spatially far apart [6]. If we look at temperature, this means that we expect two locations that are several meters apart to have more similar temperatures than two locations that are several kilometres apart. At the end of the process, we want to have estimated temperature values for an area, such that the error with respect to the measured temperature is small and uniformly bounded. To reach this, we need to select an interpolation method that fits the problem at hand best. In the next section we will briefly discuss several popular spatial interpolation methods, such as splines, Inverse Distance Weighting (IDW), and Kriging.

Splines

A spline is a function defined by a piecewise polynomial $f(\mathbf{x})$, hence we do not fit one single high degree polynomial through our data, but we fit low degree polynomials through smaller subsets of our data to approximate $g(\mathbf{x})$ [6].

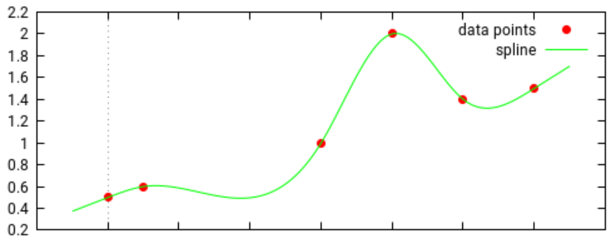


Figure 3.1: Example of a spline. Source: [7].

The fitting is done by minimising the total surface curvature of $f(\mathbf{x})$ which results in a smooth surface. In 3D we can imagine this as trying to bend a piece of paper such that it passes through all known points [6, 8, 9]. This is also how the technique got its name, originally a spline was a flexible piece of wood or metal that was used to draw smooth curves between points [10]. The method works the best for surfaces with low variance, hence it does not work well when points that are relatively close together have a large difference in values. This is a result of spline using slope calculations to compute the shape of our 'paper sheet' [8].
Advantages of this method are that with only a few known values it can generate a sufficiently accurate surface and it can preserve small details. A disadvantage is that it may have different extreme values than the data set and that it is sensitive to outliers. These disadvantages are true for all exact interpolators, but can result in more serious difficulties for this method because it works best for low variance surfaces [6].

Inverse Distance Weighting

As the name 'Inverse Distance Weighting' suggests, this method assigns weights to given data points based on their distance from the point $\mathbf{x}^*$ for which the value needs to be predicted. The known points that are nearest to $\mathbf{x}^*$ are assigned the highest weights and thus contribute more to the value of $\mathbf{x}^*$ that needs to be predicted [6]. This is different from the spline approach, because there a point either contributes to the prediction or it does not and with Inverse Distance Weighting (IDW) there is a distinction in how much each point contributes. In the example shown in Figure 3.2 the purple point in the middle represents $\mathbf{x}^*$ and its value needs to be predicted using the three closest red points whose values are written above the red dots. The left point is the closest with a distance of 350 meters, so it is assigned the highest weight out of the three points.

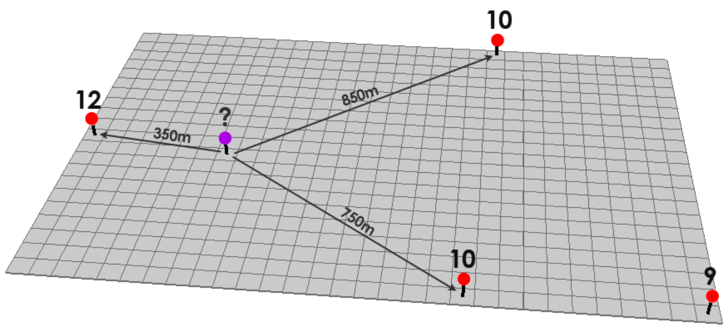


Figure 3.2: Example of a situation where IDW can be used. Source: [11]

To determine the interpolated values, IDW uses a linear-weighted combination of known sample points [8, 9]. Let us call the value of $\mathbf{x}^*$ that needs to be predicted y^* and $y_1, ..., y_n$ the values of the known points, then

$$y^* = \frac{\sum_{i=1}^{n} w_i y_i}{\sum_{i=1}^{n} w_i}.$$

The weights w_i are computed as

$$w_i = \begin{cases} \frac{1}{d_i^p}, & \text{if } d_i > d \\ 0, & \text{else} \end{cases}$$

with d_i the distance from the $\mathbf{x}^*$ to known point $\mathbf{x}_i$, d is the radius of the area around $\mathbf{x}^*$ within which the points used for the prediction must be located, and p is a constant that can be chosen as well to influence the weights. Constant p regulates how fast the weights decay with distance to the point $\mathbf{x}^*$ [11].

This method may be used when the data set is dense and sufficiently uniformly distributed enough to be able to capture local surface variation [8]. An advantage of this method is that it is very intuitive to make predictions this way. Another advantage is that it can capture extreme changes well, and dense evenly distributed points are well approximated. Due to working with weights associated with distance, the amount of points used to computed the new values can be controlled by changing the radius d [6, 8].
A disadvantage is that IDW is also sensitive to outliers. Furthermore, if the data is unevenly distributed we could get significant errors. For example when there is a data cluster, we could for example be using 50 known points to make an estimation for 1 unknown point, but if we try to estimate a point outside of that cluster we could only have 5 known points available for the estimation.

Kriging

Kriging is a method that is similar to IDW. Kriging also makes use of local weighting, but in addition to that it uses statistical properties of the known data points. So it is a probabilistic method while splines and IDW are deterministic methods.
A benefit of using statistics is that the covariance and correlation between any two points is estimated, which makes it possible to compute an error map for the entire surface. This can allow us to quantify the accuracy of the temperature prediction $f(\mathbf{x})$. The statistical properties also allow kriging to work with data containing a bias and noise, and it is able to make estimations for the bias and noise using maximum likelihood estimators when they are unknown. Some disadvantages are that it is more computationally expensive and it needs more user input than the other methods [6, 12]. A more detailed description of Kriging will be given in subsection 3.4.1.

Comparing methods

For our purpose, not just the distance between two points matters, the statistical correlation between two stations depends on other factors as well. For example, their surroundings are important too, two stations on a wide open field correlate differently than two stations with the same distance from each other in a crowded city. Kriging would allow us to incorporate these differences in correlation due to its use of statistical properties unlike the other methods. Another benefit of the statistical description is that we can compute the standard deviations of the temperature prediction $f(\mathbf{x})$, which can be used to make an uncertainty map of $f(\mathbf{x})$. This map is very interesting for us, because it shows us how accurate the prediction is at every location. Therefore, if we plot the uncertainty map of $f(\mathbf{x})$ that was predicted using first party data and the uncertainty map of $f(\mathbf{x})$ when first, second and third party data were used to predict it, it not only shows us what the improvements are numerically, but also where those improvements are spatially.
Finally, Kriging is also able to deal with bias and noise that is present in the data. Doing so is also very important for us, because we are dealing with multiple data sources with different fidelities [6, 12]. Kriging is based the concept of 'Gaussian Processes', which we will first explain before diving into Kriging.

3.2 Multivariate Gaussian distributions

A random vector $X = (X_1, ..., X_n)^T$ follows a n-variate normal distribution if each random variable $X_i, i \in \{1, ..., n\}$ follows a Gaussian distribution and every linear combination of the n random variables follows a Gaussian distribution as well. Figure 3.3 shows an example of what such a distribution can look like.
Just like with the Gaussian distribution, its multivariate version is defined by a mean vector μ and a covariance matrix $\mathbf{\Sigma}$. For random vector X and its ith component X_i

$$X = (X_1, ..., X_n)^T \sim \mathcal{N}(\mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{\Sigma}|}} \exp(-\frac{1}{2}(X-\mu)^T \mathbf{\Sigma}^{-1} \exp(X-\mu))$$
$$X_i \sim \mathcal{N}(\mu_i, \mathbf{\Sigma}_{ii}).$$

The component μ_i is the mean of i-th coordinate X_i and the covariance matrix $\mathbf{\Sigma}$ tells how the random variables are correlated. It is defined as $\mathbf{\Sigma}_{i,j} = \mathrm{Cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)^T]$, with $i, j \in \{1, ..., n\}$. The covariance between X_i and X_j is the same as between X_j and X_i, therefore $\mathbf{\Sigma}$ always is symmetric [13, 14].

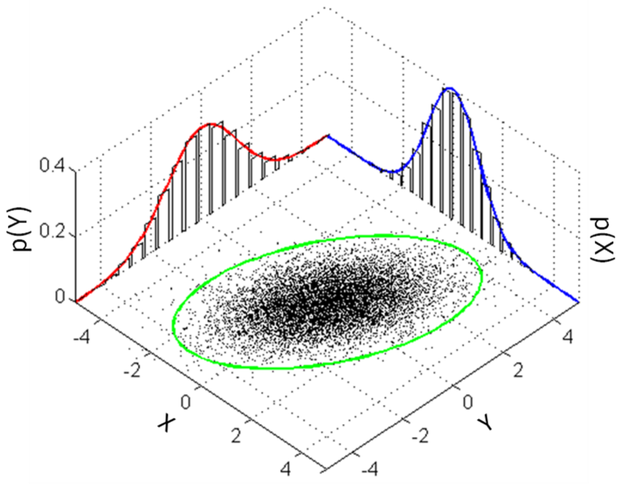


Figure 3.3: Example of a multivariate normal distribution. $\mu = (0, 0)^T$ and $\mathbf{\Sigma} = \begin{pmatrix} 1 & 3/5 \\ 3/5 & 2 \end{pmatrix}$. The distribution is centred around the mean and the shape of the ellipse in the xy-plane is determined by the covariance matrix. It also shows that the two marginal distributions, shown in red and blue, follow a univariate Gaussian distribution. Source: [15]

A nice property of Gaussian distributions is that they are closed under marginalisation and conditioning. This means that when we apply marginalisation or conditioning, the result is again a Gaussian distribution. Let us assume

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim \mathcal{N}\Big(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} \\ \mathbf{\Sigma}_{21} & \mathbf{\Sigma}_{22} \end{pmatrix} \Big),$$

where X_1 and X_2 are random vectors with no components in common. Due to the marginalisation property, we know

$$X_1 \sim \mathcal{N}(\mu_1, \mathbf{\Sigma}_{11})$$
$$X_2 \sim \mathcal{N}(\mu_2, \mathbf{\Sigma}_{22}),$$

see [13]. Consequently the distribution of existing random variables will not be influenced by the addition of new random variables. For example, the distribution of the measurements of the KNMI stations will not change when measurements from Rijkswaterstaat are added.

Conditioning is computing the probability distribution of a variable depending on another variable, denoted as $p(Y = y|X = x)$. This is very important for Gaussian processes (GPs), because it allows us to use Bayes' theorem to compute the posterior probability. The conditional distribution for a multivariate Gaussian is defined as

$$X_1|X_2 \sim \mathcal{N}(\mu_1 + \mathbf{\Sigma}_{11}\mathbf{\Sigma}_{22}^{-1}(X_2 - \mu_2), \mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}\mathbf{\Sigma}_{21}), \tag{3.1}$$

see [13, 14, 16]. Equation 3.1 can alternatively be written as,

$$\mathbb{E}[X_1|X_2] = \mu_1 + \mathbf{\Sigma}_{11}\mathbf{\Sigma}_{22}^{-1}(X_2 - \mu_2)$$
$$\mathrm{var}(X_1|X_2) = \mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}\mathbf{\Sigma}_{21}$$

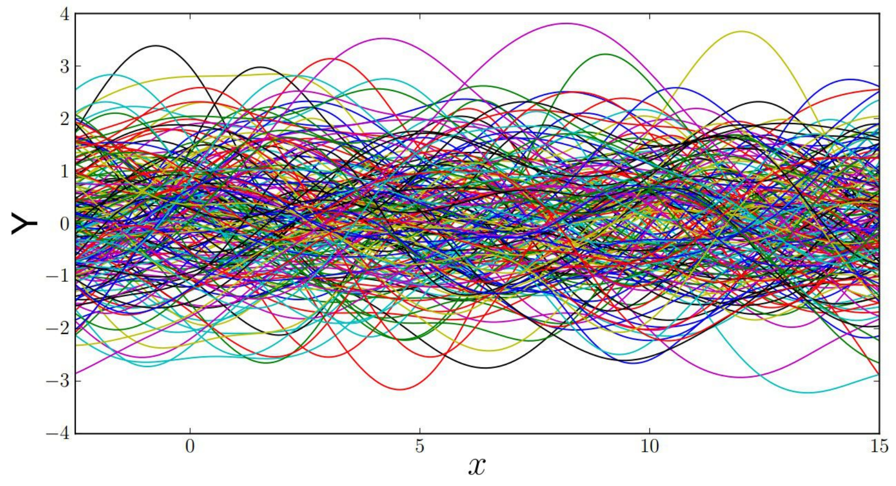
3.3 Gaussian Processes

For a given set of known $(\mathbf{x}, y)$ pairs that satisfy $g(\mathbf{x}) = y$, there are many classes of functions that fit the given data. Therefore it can be difficult to find the right approximation $f(\mathbf{x})$ of the true function $g(\mathbf{x})$. A Gaussian process (GP) provides an elegant solution by providing each of these possible functions with a probability. The mean of this distribution gives the most likely description of the known data [17].

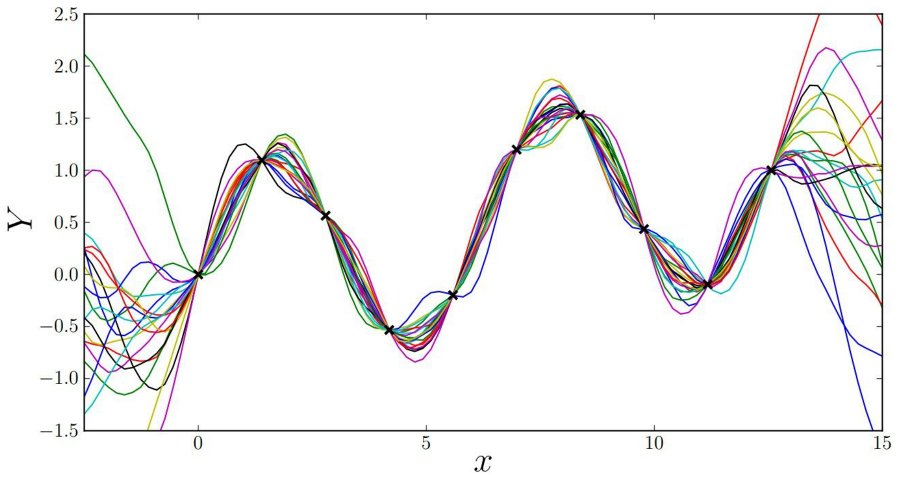
A GP is a stochastic process that consists of a (possibly infinite) collection of random variables, such that every finite collection of those random variables follows a multivariate Gaussian distribution [16, 18].
The goal of the method that uses a GP is to find the distribution of the collection of random variables by learning from training data D and predicting the data Y by modelling the distribution $P(Y|D)$ as a multivariate Gaussian distribution. The training data consists of input and output pairs, $D = \{(\mathbf{x}_i, y_i)|i = 1, ..., n\}$, where the input vector $\mathbf{x}_i$ is d dimensional. The output is a real-valued scalar $y_i = g(\mathbf{x}_i)$ with an unknown function $g : \mathcal{D} \to \mathbb{R}$. If we write this in matrix form we get $D = \{\mathbf{X}, \mathbf{y}\}$ with $\mathbf{X} \in \mathbb{R}^{d\times n}$ and $\mathbf{y} \in \mathbb{R}^n$. The goal is to predict the output vector $\mathbf{y}^*$ for previously unseen test input $\mathbf{X}^*$ using D. This can be done by looking at the probabilities of a certain function $f(\mathbf{X})$ given D that approximates $g(\mathbf{X})$ and the probability of observing $\mathbf{y}^*$ given f and $\mathbf{X}^*$, i.e. determining $p(f|D)$ and $p(\mathbf{y}^*|f, \mathbf{X}^*)$, for all possible functions f. The probability $p(\mathbf{y}^*|\mathbf{X}^*, D)$ is then computed as follows:

$$p(\mathbf{y}^*|\mathbf{X}^*, D) = \int p(\mathbf{y}^*|f, \mathbf{X}^*)p(f|D)df \tag{3.2}$$

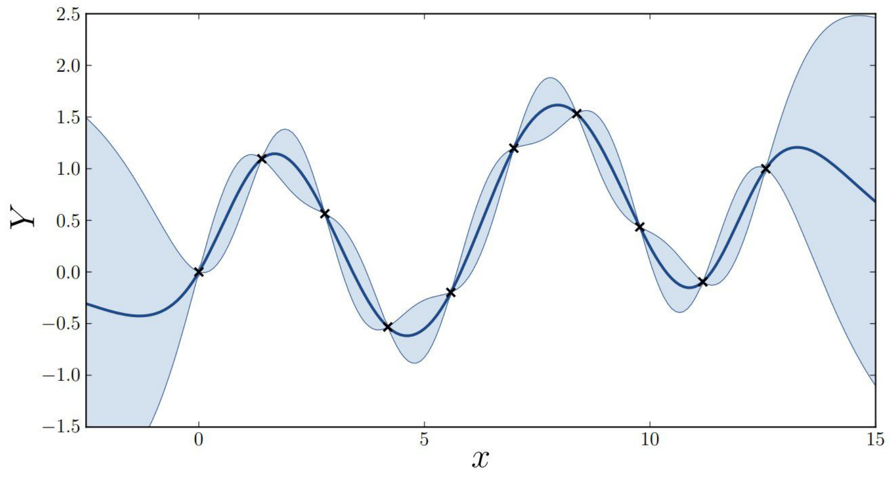
, see [18, 19]. As mentioned in section 3.2, Gaussian distributions are closed under conditioning, which means that $p(\mathbf{y}^*|\mathbf{X}^*, D)$ also follows a Gaussian distribution. From the resulting distribution function $p(\mathbf{y}^*|\mathbf{X}^*, D)$ values for $\mathbf{y}^*$ can be predicted by drawing samples. If we want to know the estimated value for a specific point i, we have to look at the i-th component of the drawn vector. Equation 3.2 can be solved by using Bayesian inference. This means that the hypothesis is updated when we get new information. In other words the distribution changes when the set D becomes larger. This can be represented visually as shown in Figure 3.4. In Figure 3.4a a number of samples are drawn from the prior distribution of Y. This is the distribution that is assumed for Y before any observations are made. When observations become available this prior distribution can be adjusted so that the samples comply with the observations, as shown in Figure 3.4b. From these samples, the mean of the posterior distribution can be computed with a 95% confidence interval that is defined as two times the standard deviation for each input value of x [17, 20].



(a)



(b)



(c)

Figure 3.4: In Figure 3.4a a lot of samples drawn from the prior distribution are shown, after observations are made the samples that do not reflect the observations are removed and we are left with the samples shown in Figure 3.4b. The result can be summarised by a mean function shown by a dark blue line in Figure 3.4c and a 95% confidence interval around it, displayed by the light blue shaded area. Source: [20]

To find this distribution $p(\mathbf{y}^*|\mathbf{X}^*, D)$, we need to find its mean vector μ and covariance matrix $\mathbf{\Sigma}$. μ is often assumed to be 0 to simplify the computations. If μ is not equal to 0, the bias can be corrected after the prediction is made. Estimating $\mathbf{\Sigma}$ is a bit more interesting, because $\mathbf{\Sigma}$ determines the shape of the distribution. This means that the covariance matrix tells us which type of functions from all possible functions are the most likely. The components of $\mathbf{\Sigma}$ are given by $\mathbf{\Sigma}_{i,j} = \kappa(p, p')$, where κ is the covariance function, also known as the kernel, and p and p' are points in $\mathbf{X}$ or $\mathbf{X}^*$. The covariance matrix Σ tells us how much influence p and p' have on each other. There are a lot of different kernel functions possible and we will go into more detail about them in subsection 3.4.4 [14].

Thus, Gaussian Processes model distributions over functions which means that they can be used to build a regression model. This is a model that estimates the relationships between one or more independent variables and a dependent variable by providing a function. In Kriging, the Gaussian Process will be our prior distribution function, this in combination with some data will allow us to compute a posterior distribution. Which in turn can be used to predict the expected value of the output variable given the input variables.

3.4 Kriging

Kriging is a Bayesian interpolation method that originates in geostatistics and is based on GPs. There are a few versions of Kriging, such as simple Kriging, ordinary Kriging and simple Kriging with local error estimates (Kriging LE) [20, 21, 22]. These first two methods are very similar, except for how they view the mean μ of the prior GP. If we want to predict the unknown output for a point $\mathbf{x}_j$ using Kriging, we get an estimate $f(\mathbf{x}_j)$ of the true function $g(\mathbf{x}_j)$ in the form $f(\mathbf{x}_j) = \mu + h(\mathbf{X}, \mathbf{x}_j)$. For ordinary Kriging, μ is seen as an unknown constant and for simple Kriging, μ is seen as a known constant. As the name suggests, Kriging LE is a version of simple Kriging that uses local error estimates instead of one error estimate that is used for all the stations. In the following sections we will discuss these methods more thoroughly.

3.4.1 Ordinary and simple Kriging

Kriging is a non-parametric approach for estimating a function $g(\mathbf{x})$. This means that when the number of data points increases, the number of parameters increases too, while in a parametric model the number of parameters is fixed, in contrast with common approaches such as Bayesian linear interpolation [23]. Kriging tries to estimate a distribution over all possible functions $f(\mathbf{x})$ that fit the observed data. Kriging lets the prior distribution on $f(\mathbf{x})$ be a GP,

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')),$$

where the mean $m(\mathbf{x})$ is defined as $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and covariance function κ by $\kappa(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))^T]$ [16, 18, 19, 24]. In our case, $\mathbf{x}$ is an $n \times 2$ matrix containing the locations of n weather stations and $\mathbf{x}'$ is the matrix that contains the grid points for which we want to predict the temperatures as defined in Figure 2.1d.

We need to define a distribution over function values at the points in $\mathbf{x}$. Kriging assumes that the distribution $p(f(\mathbf{x}^*))$ is jointly Gaussian with mean μ and covariance matrix $\mathbf{P}$, in other words we assume the prior distribution to be $Y^* \sim \mathcal{N}(\mu, \mathbf{P})$. Even though this distribution depends on $\mathbf{x}$ and $\mathbf{y}$, this is still called a prior distribution, because it is assumed a priori that Y^* can be modelled using a GP. In ordinary kriging, μ is an unknown constant while in simple Kriging μ is assumed to be a known constant.
Furthermore we also assume the normal likelihood for the observations,

$$\mathbf{y}|\mathbf{Y}^* \sim \mathcal{N}(\mathbf{H}\mathbf{Y}^*, 0) \tag{3.3}$$

where $\mathbf{H}$ is an observation matrix. Let n be the number of observations and p be the number of points we want to predict, in case of the temperature prediction n is the number of weather stations and p is the number of grid points from Figure 2.1d. Then the observation matrix $\mathbf{H} \in \mathbb{R}^{n \times p}$ tells us where the observations were taken relative to the points we want to predict. An example of what a matrix $\mathbf{H}$ may look like is this:

$$\mathbf{H} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Here $\mathbf{H}$ has two rows and four columns so we have two stations and the grid used to make a prediction consists of four points. The elements in the first row tell us that the first station can be found right between the first two grid points and the second row shows that the second station is exactly at the second grid point.

The components of covariance matrix $\mathbf{P}$ are given by $\mathbf{P}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, where κ is a positive definite kernel, just like for a GP, the kernel defines which type of functions from all of the possible functions are the most likely under the prior. κ must be positive definite, to make sure that $\mathbf{P}$ is positive (semi) definite [16]. There are several different kernels, but here we will be using the kernel defined as follows

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \psi_{i,j} \tag{3.4}$$

where σ^2 is the product of the standard deviations of $\mathbf{y}_j$ and $\mathbf{y}_i$, but since they are both from the same set $\mathbf{y}$, this is just the variance of $\mathbf{y}$. Our choice for this kernel will be more thoroughly discussed in subsection 3.4.4. $\psi_{i,j}$ is the basis function corresponding to the correlation between the known data point $\mathbf{x}_i$ and the data point $\mathbf{x}_j$ that needs to be predicted.

$$\psi_{i,j}(\mathbf{x}_i, \mathbf{x}_j) = \exp\Big(- \sum_{k=1}^{d} \frac{||(\mathbf{x}_i)_k - (\mathbf{x}_j)_k||^2}{2\hat{\theta}_k^2} \Big), \tag{3.5}$$

with d being the number of dimensions. $\hat{\theta} \in \mathbb{R}^d$ is a spatial parameter that we can think of as an indicator that tells us how quickly the function changes when $\mathbf{x}_j$ moves closer or further away from $\mathbf{x}_i$. A high $\hat{\theta}_k$ value tells us that a certain function is active along dimension k, and a low value that it is inactive [25].

To obtain $\hat{\theta}$, we use a maximum likelihood estimator with respect to θ, which is equivalent to minimising [26]

$$L(\theta) = \log(|\mathbf{HPH}^T|) + (\mathbf{y} - \mathbf{H}\mu)^T(\mathbf{HPH}^T)^{-1}(\mathbf{y} - \mathbf{H}\mu). \tag{3.6}$$

The product $\mathbf{HPH}^T \in \mathbb{R}^{n\times n}$ can be interpreted as the matrix of correlations between sample data.
Besides $\hat{\theta}$, another MLE is needed for μ, which we will define as

$$\hat{\mu} = \frac{\mathbb{1}^T(\mathbf{HPH}^T)^{-1}\mathbf{y}}{\mathbb{1}^T(\mathbf{HPH}^T)^{-1}\mathbb{1}}, \tag{3.7}$$

where $\mathbb{1}$ is an $n \times 1$ vector of ones. Using all of this information, we would like to make the prediction. Our prediction is the value at desired point $\mathbf{x}_j$ that maximises the likelihood. Given the sample data and MLE's of θ and μ, this translates to

$$f(\mathbf{x}_j) = \hat{\mu} + \sum_{i=1}^{n} c_i \psi_{i,j}(\mathbf{x}_i, \mathbf{x}_j).$$

The constants c_i are given by column vector $\mathbf{c} = (\mathbf{HPH}^T)^{-1}(\mathbf{y} - \mathbb{1}\hat{\mu})$[27].

As mentioned in section 3.1 an advantage of Kriging compared to the other methods is its ability to produce an error map. This is due to known statistical properties, because this allows us to construct correlation matrices. To make the error map we are interested in the variance. The variance can be computed using the prior $\mathbf{Y}^* \sim \mathcal{N}(\mu, \mathbf{P})$, Equation 3.3 and Bayes' rule to find the Kriging posterior

$$\mathbf{Y}^*|\mathbf{y} \sim \mathcal{N}(\mathbf{m}, C)$$

where $\mathbf{m}$ is the posterior mean and C the posterior covariance. Recall that in section 3.2 for two random vectors X_1, X_2 with

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim \mathcal{N}\Big(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} \\ \mathbf{\Sigma}_{21} & \mathbf{\Sigma}_{22} \end{pmatrix} \Big),$$

their conditional distribution was defined as

$$X_1|X_2 \sim \mathcal{N}(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(X_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}). \tag{3.8}$$

Equation 3.8 can be applied to Equation 3.9 to find $\mathbf{Y}^*|\mathbf{y}$,

$$\begin{pmatrix} \mathbf{Y}^* \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}\Big(\begin{pmatrix} \hat{\mu} \\ \hat{\mu} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{PH}^T \\ \mathbf{HP} & \mathbf{HPH}^T \end{pmatrix} \Big). \tag{3.9}$$

We have our covariance matrix $\mathbf{P}$, which is Σ_{11}. To get from stations to grid points, we used matrix $\mathbf{H}$, so Σ_{22} is $\mathbf{HPH}^T$, $\Sigma_{12} = \mathbf{PH}^T$ and $\Sigma_{21} = \mathbf{HP}$. These are plugged into Equation 3.8 and results in

$$\begin{aligned} \mathbb{E}[\mathbf{Y}^*|\mathbf{y}] &= \hat{\mu} + \mathbf{PH}^T(\mathbf{HPH}^T)^{-1}(\mathbf{y} - \hat{\mu}) \\ &= \hat{\mu} + \mathbf{K}(\mathbf{y} - \hat{\mu}) \\ \mathrm{var}(\mathbf{Y}^*|\mathbf{y}) &= \mathbf{P} - \mathbf{PH}^T(\mathbf{HPH}^T)^{-1}\mathbf{HP} \\ &= (I - \mathbf{KH})\mathbf{P} \end{aligned} \tag{3.10}$$

Where $\mathbf{K} = \mathbf{PH}^T(\mathbf{HPH}^T)^{-1}$ is the Kalman gain. Something like this is not possible for the other methods, because we are unable to produce a matrix that tells us something meaningful about the correlation between grid-points or about the correlation between grid-points and stations [28].

3.4.2 Kriging as regression method

The methods discussed in subsection 3.4.1 work as an interpolation method, but we would like to use Kriging as a regression method. This is because our data will be noisy, that is $\mathbf{y} = g(\mathbf{x}) + \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\sigma}_\epsilon^2)$ the irreducible error, therefore we do not want to force our prediction through our measured data points. We can adjust our method by taking

$$\begin{aligned} \mathbf{P}_{i,j} &= \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ &= \kappa(\mathbf{x}_i, \mathbf{x}_j) + \boldsymbol{\sigma}_\epsilon^2 \mathbb{I}_{i=j}, \end{aligned}$$

i.e. we add the variance of the error to the diagonal of the covariance matrix [18]. We only add this to the diagonal because we assume that the errors between stations are uncorrelated.
Now we also need to adjust the correlation matrix $\mathbf{HPH}^T$, because it forces our prediction to pass exactly through our sampled points. However, when we have noisy data, we do not need the prediction to pass through our sample data exactly, therefore we add a regularisation constant σ_n^2 to the diagonal of $\mathbf{HPH}^T$. Let $\mathbf{R} = \sigma_n^2 I$, then we get $\mathbf{HPH}^T + \mathbf{R}$. The assumed likelihood for the observations is also changed to $\mathbf{y}|\mathbf{Y}^* \sim \mathcal{N}(\mathbf{HY}^*, \mathbf{R})$, which also changes the Kriging posterior. Instead of applying $\Sigma_{22} = \mathbf{HPH}^T$ in (3.8), we use $\Sigma_{22} = \mathbf{HPH}^T + \mathbf{R}$. We do not need to add $\mathbf{R}$ to Σ_{12} and Σ_{21} because $\mathbf{R}$ only adds noise to the variance of a location and not the covariance between two locations. This results in a Kalman gain of $\mathbf{K} = \mathbf{PH}^T(\mathbf{HPH}^T + \mathbf{R})^{-1}$.

To be able to find the right $\hat{\theta}$, $\hat{\mu}$ and σ_n^2, we need to adjust Equation 3.6 and Equation 3.7 too,

$$\begin{aligned} L_r(\theta, \sigma_n^2) &= \log(|\mathbf{HPH}^T + \mathbf{R}|) + (\mathbf{y} - \mathbf{H}\mu)^T(\mathbf{HPH}^T + R)^{-1}(\mathbf{y} - \mathbf{H}\mu) \\ \hat{\mu}_r &= \frac{\mathbb{1}^T(\mathbf{HPH}^T + \mathbf{R})^{-1}\mathbf{y}}{\mathbb{1}^T(\mathbf{HPH}^T + \mathbf{R})^{-1}\mathbb{1}} \end{aligned}$$

where the subscript r stands for regression. However, in our implementation we used $\hat{\mu}_r$ equal to the mean of $\mathbf{y}$ for lower computational speed. Now that the parameters are optimised for the new setting, the new theoretical prediction can be defined as

$$f_r(\mathbf{x}_j) = \hat{\mu}_r + (\mathbf{PH}^T)^T(\mathbf{HPH}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbb{1}\hat{\mu}_r) \tag{3.11}$$

[27].
Equation 3.11 tells us that when we have little noise, the prediction $f_r(\mathbf{x}_j)$ depends more on the measurements $\mathbf{y}$ and less on the mean $\hat{\mu}_r$ than when we have a lot of noise. We can easily demonstrate this for the case with just one measurement by using a Taylor expansion.

$$\begin{aligned} \hat{y_r}(\mathbf{x}_j) &= \hat{\mu}_r + \sigma^2(\sigma^2 + \sigma_n^2)^{-1}(y - \hat{\mu}_r) \\ &= \hat{\mu}_r + \frac{1}{1 + \frac{\sigma_n^2}{\sigma^2}}(y - \hat{\mu}_r) \\ &= \Big(1 - \frac{\sigma_n^2}{\sigma^2}\Big)y + \frac{\sigma_n^2}{\sigma^2}\hat{\mu}_r + \mathcal{O}\Big(\frac{(\sigma_n^4}{\sigma^4}\Big). \end{aligned}$$

We see that when the regularisation constant σ_n^2 increases, the factor in front of y decreases, while the factor in front of $\hat{\mu}_r$ increases.

To see why it is important to incorporate noise into the computation, we look at an example discussed in Forester et al. (2006). They compare the results of using Kriging as interpolation and Kriging as regression. They considered an airfoil where the drag coefficient C_D had to be minimised. The airfoil was defined by five orthogonal shape functions. The first of these functions f_1 was kept constant and represented the shape of a NASA supercritial SC(2) airfoil. The next function, f_2, is a least squares fitting to the preceding fit, and each airfoil of the SC(2) series can be added to f_1 with weight w_2. In Figure 3.5 the difference between the two methods is clearly visible. When we do not force the prediction for C_D to go through all the sample points, we get a much better prediction than when we do [27].

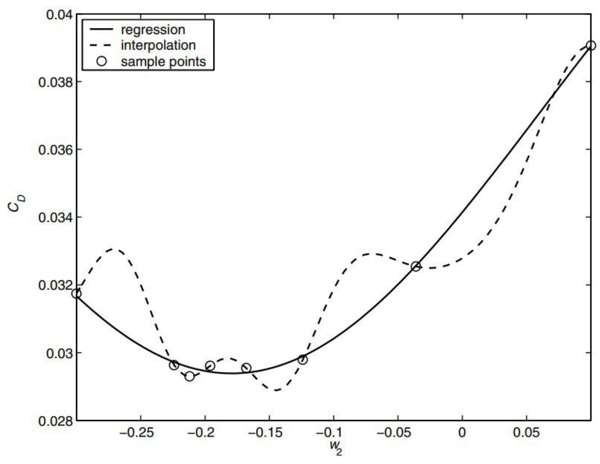


Figure 3.5: The results of using Kriging as a regression or as an interpolation method. It shows that when we do not force the prediction through the measured data points, we get a much more natural looking result. Source: [27]

3.4.3 Simple Kriging using a local error estimate

The idea behind Simple Kriging using a local error estimate (Kriging LE) is that we would like to be able to assign each observation $\mathbf{y}_i$ its own systematic error (bias) $\mathbf{b}_i$ and random error (noise) with standard deviation $(\sigma_n^2)_i$. To achieve this we modify the likelihood,

$$\mathbf{y}|\mathbf{Y}^* \sim \mathcal{N}(\mathbf{H}\mathbf{Y}^* + \mathbf{b}, \mathbf{R})$$

to have $\sigma_n^2 \in \mathbb{R}^n$ instead of a scalar. These changes translate into the posterior mean and likelihood function

$$\begin{aligned} \mathbb{E}[\mathbf{Y}^*|\mathbf{y}] &= \hat{\mu}_r + \mathbf{K}(\mathbf{y} - \hat{\mu}_r - \mathbf{b}) \\ L_{LE}(\theta, \sigma_n^2, \boldsymbol{b}) &= \log(|\mathbf{HPH}^T + \mathbf{R}|) + (\mathbf{y} - \mathbf{H}\hat{\mu}_r - \mathbf{b})^T(\mathbf{HPH}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{H}\hat{\mu}_r - \mathbf{b}) \end{aligned} \tag{3.12}$$

Note that the likelihood function is now used to find θ, σ_n^2 and $\mathbf{b}$. In Equation 3.12 it might look like a bias is subtracted from all the data, but recall that for first party data a bias and noise of 0 were assumed. Using a vector σ_n^2 brings us a benefit that we do not have to use the same σ_n^2 for all of the stations, but can make distinctions between their accuracy. This is beneficial for us, because we are working with high fidelity data where the difference in accuracies can vary several degrees Celsius.
In the implementation we do not give each station its own bias and noise value, but we did make the distinction between first, second and third party stations. We decided this because it is a very costly process to determine $(\sigma_n^2)_i$ and $\mathbf{b}_i$ for all individual stations.

In the table below there is a short overview of the equations in which Kriging LE differs from Simple and Ordinary Kriging.

Quantity	Simple and Ordinary Kriging	Kriging LE
Assumed likelihood for observations	$\mathbf{y}\vert\mathbf{Y}^* \sim \mathcal{N}(\mathbf{HY}^*, \mathbf{R})$	$\mathbf{y}\vert\mathbf{Y}^* \sim \mathcal{N}(\mathbf{HY}^* + \mathbf{b}, \mathbf{R})$
Noise covariance matrix	$\mathbf{R} = \sigma_n^2 I$, $\sigma_n^2 \in \mathbb{R}$	$\mathbf{R} = \sigma_n^2 I$, $\sigma_n^2 \in \mathbb{R}^n$
posterior distribution mean	$\mathbb{E}[\mathbf{Y}^*\vert\mathbf{y}] = \mu + K(\mathbf{y} - \mathbf{H}\hat{\mu}_r)$	$\mathbb{E}[\mathbf{Y}^*\vert\mathbf{y}] = \mu + K(\mathbf{y} - \mathbf{H}\hat{\mu}_r - \mathbf{b})$
Minimising for hyper parameters	$\log(\vert\mathbf{HPH}^T + \mathbf{R}\vert) + (\mathbf{y} - \mathbf{H}\hat{\mu}_r)^T (\mathbf{HPH}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{H}\hat{\mu}_r)$	$\log(\vert\mathbf{HPH}^T + \mathbf{R}\vert) + (\mathbf{y} - \mathbf{H}\hat{\mu}_r - \mathbf{b})^T (\mathbf{HPH}^T + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{H}\hat{\mu}_r - \mathbf{b})$

3.4.4 Kernels

The kernel function is very important for Kriging, because it defines which functions are most likely under the prior, which in turn induces the generalisation properties [1] of the model [30]. For example, it determines whether a function is stationary or non-stationary. The stationary kernel functions are translation invariant, i.e. they only depend on the relative positions of their input. Non-stationary kernels depend on the absolute positions of their input, which means an origin needs to be specified [16, 17]. Another property that is decided by the kernel function is the variation speed. For example if the functions in Figure 3.4 vary too quickly, we can slow it down by adjusting the parameter θ in the kernel function. Note that finding the right hyperparameters for the kernel function such that we end up with the right properties for prediction $f(\mathbf{x})$, such as the variation speed, is exactly the numerical challenge in applying Kriging [17].

In Kriging a kernel κ must be positive-definite and is used to define the prior covariance between any two function values

$$\text{Cov}[f(\mathbf{x}_i), f(\mathbf{x}_j)] = \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

, see [30]. There are many kernels available, but three commonly used kernels are the squared exponential kernel, periodic kernel and the linear kernel. These three will be discussed briefly to motivate why the squared exponential kernel was chosen.

The squared exponential kernel is also known as the Gaussian kernel and allows one to model functions with local variations. In d dimensions the kernel is defined as

$$\kappa_{SE}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp\Big(-\sum_{k=1}^{d} \frac{||(\mathbf{x}_i)_k - (\mathbf{x}_j)_k||^2}{2\theta_k^2}\Big),$$

and is shown in the leftmost panels in Figure 3.6. κ_{SE} depends on the relative positions of $\mathbf{x}$ and $\mathbf{x}'$ and is therefore a stationary kernel. A nice property of this kernel is that it is infinitely differentiable, which means that $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$ is infinitely continuously differentiable, i.e. $f(\mathbf{x})$ is smooth [17, 30, 31]. The periodic kernel can be used to model periodic functions with a repeating structure and in d dimensions it is defined as

$$\kappa_{PER}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp\Big(-\sum_{k=1}^{d} \frac{2}{\theta_k^2} \sin^2\Big(\frac{\pi||(\mathbf{x}_i)_k - (\mathbf{x}_j)_k||}{p}\Big)\Big),$$

where p is the period of the function and it is shown in the middle figures of Figure 3.6. This kernel is also stationary, because it depends on $||\mathbf{x}_i - \mathbf{x}_j||$. The linear kernel is defined as

$$\kappa_{LIN}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2(\mathbf{x}_i - c)(\mathbf{x}_j - c)$$

where c is the offset which tells us the x-coordinate of the point that all posterior lines must go through. In the rightmost figures of Figure 3.6 we see that that point falls just out of the frame. κ_{LIN} is non-stationary because it does not depend on the distance between $\mathbf{x}$ and $\mathbf{x}'$ but rather on the value of the input.

[1]Generalisation is a term used to describe the ability of a trained model to adjust to previously unseen data [29].

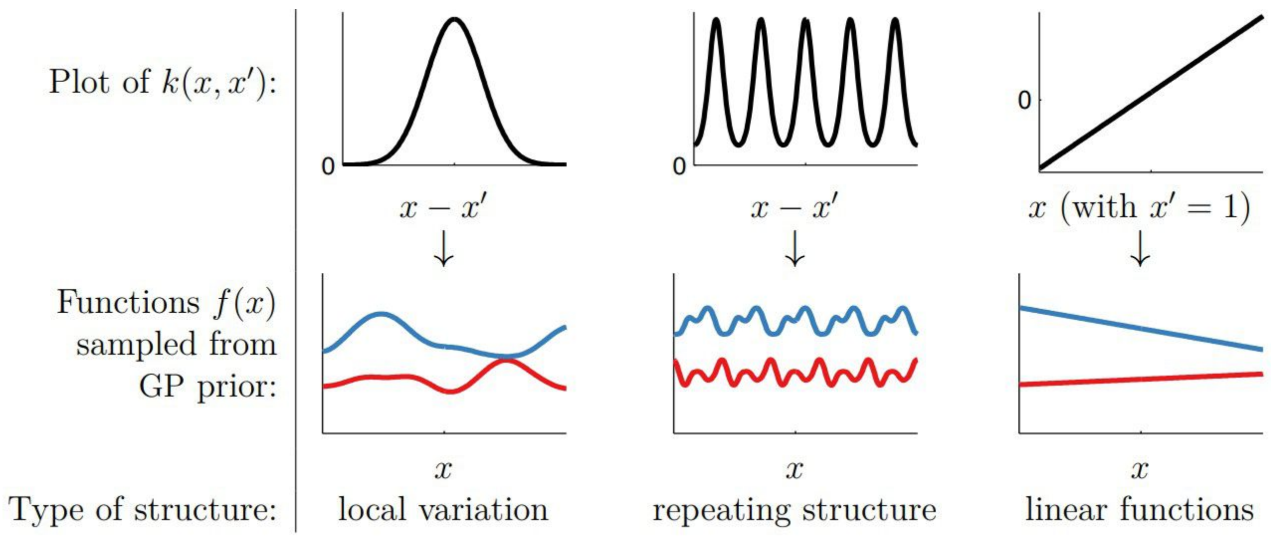


Figure 3.6: Visualisation of the squared exponential kernel, the periodic kernel and the linear kernel, from left to right respectively. On the first row the plot of the kernels is shown and on the second row the types of functions they model are shown. Source: [30]

Overall, the squared exponential kernel seems to fit our needs best. This kernel works well when you try to model smooth functions, which is exactly what we are trying to do with a temperature field [30]. However, this does not mean it is the best kernel for other weather phenomena, for example for wind it is recommended to use the exponential kernel [32].

Chapter 4

Implementation

Now that the theoretical background has been discussed, KrigingLE can be put in to practice. The implementation of Kriging LE is made in Python and uses the data from the 25th of January 2019 provided by the KNMI (1PD), Rijkswaterstaat (2PD) and WOW (3PD) to make a temperature prediction for the Netherlands. The code and data used to make the predictions can be found on Github[1].
To be able to make a prediction using Kriging LE, the data that is used for training and the points for which the temperature should be predicted need to be defined. The points for which temperature needs to be predicted are defined as the grid points as shown in Figure 2.1d. Our training data will be the data provided by the KNMI, Rijkswaterstaat and WOW, and for each party this will consist out of a vector with station locations and their corresponding average daily temperature in a vector. So we have three vectors consisting of the locations and three vectors of the temperatures. If we want to work with data from multiple parties we concatenate their data in numerical order. The station locations vector is denoted $\mathbf{x}$ and the temperature vector is denoted $\mathbf{y}$, however we work with a normalised version of $\mathbf{y}$ denoted $\mathbf{y}_n$.
The kriging process can roughly be divided in three steps; hyperparameter estimation, temperature prediction and an error prediction. Before we can dive into these, we need to discuss how we construct our correlation matrix $\mathbf{HPH}^T$, noise covariance matrix $\mathbf{R}$ and how we solve systems $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{x}$.

4.1 The correlation matrix, noise covariance matrix and solving linear systems

Correlation matrix

We defined our correlation matrix as $\mathbf{HPH}^T$, where $\mathbf{H}$ is an observation matrix and $\mathbf{P}$ the covariance matrix. To build these, we need three things. The parameter θ and two sets of locations $\mathbf{x}_1$ and $\mathbf{x}_2$ between which we want to know the correlation.
First we construct a zero matrix D of size $|\mathbf{x}_1| \times |\mathbf{x}_2|$ and then for each entry of D we compute $\psi_{i,j}(\mathbf{x}_i, \mathbf{x}_j)$ as in Equation 3.5. Note that we do not multiply by σ^2 here as in Equation 3.4, because we work with normalised data, which implies that σ^2 is equal to 1

Noise covariance matrix

To construct this matrix we need to know the number of stations n, the noise standard deviation parameter σ_n and a proxy which tells us which noise parameter belongs to which party. Each party that is used has its own noise parameter, for 1PD 0 noise is assumed and for the other parties we estimate σ_n as explained in section 4.2. This means that σ_n is a vector of length 1 if we only use 1PD, first and second party data (12PD) or first and third party data (13PD) and, length 2 if we use data from the first, second and third party (123PD). To assure that we use the right noise value for each station a proxy array is used, this array contains zeroes and ones where the ones indicate which of the noise values to use for which station. We construct a zero matrix $\mathbf{R}$ of size $n \times n$ and add the noise to the diagonal of $\mathbf{R}$. Note that we add σ_n^2 to each station, because σ_n is expressed as a standard deviation and this is a noise covariance matrix.

[1] `https://github.com/daniellevanb/KrigingLE`

Solving linear systems

To solve linear systems such as $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{x}$, we first use a Cholesky decomposition instead of using $\mathbf{x} = \texttt{np.linalg.solve}(\mathbf{A}, \mathbf{y})$ right away because the Cholesky decomposition is faster for repeated solves.
For a Cholesky decomposition, we decompose a Hermitian positive (semi) definite matrix $\mathbf{A}$ in $\mathbf{A} = \mathbf{LL^T}$, where $\mathbf{L}$ is a lower triangular matrix. After $\mathbf{L}$ is computed, we solve $\mathbf{Ly} = \mathbf{b}$ for $\mathbf{y}$ and lastly $\mathbf{L^Tx} = \mathbf{y}$ for $\mathbf{x}$ [33].
The proof of why we are allowed to use this decomposition here can be found in Appendix B.

4.2 Hyperparameter estimation

Now that we have introduced our basis, we can proceed to the first step of Kriging; making a prediction for the hyperparameters b, σ_n and θ. The hyperparameter used for bias is b, σ_n for the standard deviation, also referred to as noise, and θ is the spatial parameter. The bias tells us how big the systematic error is from a certain party and the noise indicates what the random error is. Both b and σ_n are measured in degrees Celsius and they can be a scalar values or a two dimensional vector depending on how many parties we use data from. Spatial parameter θ is always a two dimensional vector that contains two parameters, one for the north-south direction and one the east-west direction and it is measured in degrees.

To estimate these hyperparameters we start by entering hyperbounds. These entered values provide the program with an upper and lower bound for each hyperparameter. We determine the kernel parameter θ for all data used together therefore we always have two lower and two upper bounds for θ, one set of bounds for the longitude and one set of bounds for the latitude. Bias b and noise σ_n are determined for each party individually, consequently the number of bounds depend on the number of parties we use data from. Note that we assume 0 bias and 0 noise for first party data. The hyperbounds are defined in two lists, one for the upper bounds and one for the lower bounds. They both always start with the bounds for b, followed by σ_n and ended with θ. For the computation we will work with the logarithmic values of the bounds for σ_n and θ, because using a logarithmic representation ensures uniform treatment across all scales.

To make sure the right hyperparameter is used for the right party, we use proxies. These are arrays consisting of ones and zeroes that indicate weather a certain station has a bias or noise, and which hyper parameter to use for it. For example if we are using first and second party data consisting of n_1 first party stations and n_2 second party stations, our proxy array will be a one dimensional array of length $n_1 + n_2$ that starts with n_1 zeroes, to show that those stations do not have a bias or noise , followed by n_2 ones to indicate that those stations do have a bias and noise factor in their measurements. In the case we are using first, second and third party data we have two bias and noise parameters. In that case our proxy will be a two dimensional array of length $n_1 + n_2 + n_3$ where n_3 is the number of third party stations. Again the first n_1 rows will only be zeroes, followed by n_2 rows with entries $[1, 0]$ and finally n_3 rows of $[0, 1]$. The rows used for the second and third party indicate whether the first or second element of b or σ_n should be used for the stations. To normalise the proxy array, we divide all ones by the standard deviation of $\mathbf{y}$.

Now that we have defined the bounds between which we need to search for the optimal parameters and the proxy to know which parameter belongs to which party we can look into finding the optimal hyper parameters. To find these parameters, we use `scipy.optimize.differential_evolution`. This scipy library function is a stochastic method used for global search optimisation problems. The algorithm needs a function and bounds as input. It returns the parameters for which the function reached its minimum [34].
In our case, the input is a likelihood function with respect to the hyper parameters, and our bounds. The output will be the optimal values of the parameters b, n and θ.

The maximum likelihood estimator (MLE) with respect to b, σ_n and θ from Equation 3.12, is:

$$L(b, \sigma_n^2, \theta) = -\log|\mathbf{HPH^T} + \mathbf{R}| - \mathbf{y}_{nb}^T A^{-1}\mathbf{y}_{nb}.$$

Where $\mathbf{A} = \mathbf{HPH^T} + \mathbf{R}$ and

$$y_{nb}^{(i)} = \begin{cases} y_n^{(i)}, & \text{if } i \in S_1 \\ y_n^{(i)} - b_k, & \text{if } i \in S_k \ , \ k = 2, 3 \end{cases}. \tag{4.1}$$

Note that since `scipy.optimize.differential_evolution` looks for a minimum and this is a *maximum* likelihood estimator, we will need to use $-L(b, \sigma_n^2, \theta)$ as input.

To construct $L(b, \sigma_n^2, \theta)$, the optimiser provides the likelihood function with a guess for the hyperparameters. From this guess, we separate the bias, noise and spatial parameters from each other and we take the exponent of the noise and spatial parameter, because we are working with their logarithm. After this we adjust our $\mathbf{y}_n$ to incorporate the estimated bias as is done in Equation 4.1. Subsequently $\mathbf{HPH^T} + \mathbf{R}$ is computed as the sum of the correlation and noise covariance matrices. Instead of computing the determinant of $\mathbf{HPH^T} + \mathbf{R}$, we use the fact that the determinant of $\mathbf{HPH^T} + \mathbf{R}$ is equal to the product of the eigenvalues of $\mathbf{HPH^T} + \mathbf{R}$, so instead of $\log|\mathbf{HPH^T} + \mathbf{R}|$, $\sum \log(\text{eigenvalues } \mathbf{HPH^T} + \mathbf{R})$ is computed [35].
For the next part lets say $\mathbf{x} = (\mathbf{HPH^T} + \mathbf{R})^{-1}\mathbf{y}_{nb}$, then $\mathbf{x}$ is solved from

$$\mathbf{x} = (\mathbf{HPH^T} + \mathbf{R})^{-\mathbf{1}}\mathbf{y}_{nb}$$
$$(\mathbf{HPH^T} + \mathbf{R})\mathbf{x} = \mathbf{y}_{nb}$$

using the Cholesky decomposition, which was the last component that was needed to compute $L(b, \sigma_n^2, \theta)$.

4.3 Temperature and uncertainty prediction

Temperature prediction

Now that the optimal parameters are computed, the predictions can be made.
Firstly $\mathbf{y}_{nb}$ is computed using the optimal estimate for b. Now we can compute our initial prediction $\mathbf{y}_0$, by using

$$\mathbf{HPH}^T\mathbf{y}_0 + \mathbf{R}\mathbf{y}_0 = \mathbf{A}\mathbf{y}_0 = \mathbf{y}_{nb}$$

where $\mathbf{HPH}^T$ is the correlation matrix between stations, which results in a normalised prediction for the stations. To get from $\mathbf{y}_0$ to a prediction $\mathbf{y}_n^*$ for each grid point, $\mathbf{y}_0$ is multiplied with the correlation matrix between stations and grid points, i.e. $\mathbf{y}_n^* = \mathbf{y}_0\mathbf{PH}^T$. However, $\mathbf{y}_n^*$ is computed using normalised data and needs to be denormalised in order to obtain the final temperature prediction $\mathbf{y}^* = \mu + \sigma\mathbf{y}_n^*$, where σ is the standard deviation of $\mathbf{y}$.

Uncertainty prediction

The uncertainty prediction is simply the standard deviation per grid point. In subsection 3.4.1 the following was shown

$$\operatorname{var}(\mathbf{y}^*|\mathbf{y}) = \mathbf{P} - \mathbf{PH}^T(\mathbf{HPH^T} + \mathbf{R})^{-1}\mathbf{HP}$$

However, we are only interested in the diagonal of $\operatorname{var}(\mathbf{y}^*|\mathbf{y})$, because this contains the expected error within a grid point. Therefore, to lower the computation time instead of subtracting the entire matrices $\mathbf{P}$ and $\mathbf{PH}^T(\mathbf{HPH^T} + \mathbf{R})^{-1}\mathbf{HP}$ we can subtract their diagonals from each other. However, since we are working with normalised data, the diagonal of P contains only ones. Consequently, the uncertainty can be computed as

$$\text{normalised uncertainty} = \sqrt{\mathbf{I} - \operatorname{diag}\Big(\mathbf{PH}^T(\mathbf{R} + \mathbf{HPH}^T)^{-1}\mathbf{H})\mathbf{P}\Big)}$$
$$\text{uncertainty} = \sigma\sqrt{\mathbf{I} - \operatorname{diag}\Big(\mathbf{PH}^T(\mathbf{R} + \mathbf{HPH}^T)^{-1}\mathbf{H})\mathbf{P}\Big)}$$

where $\mathbf{I}$ is a vector of ones of length n and σ the standard deviation of $\mathbf{y}$.
The temperature prediction will be quantitatively verified in section 5.1, but the quantitative verification of the uncertainty prediction is more complicated and beyond the scope of this research.

4.4 Cross validation

Besides the uncertainty map that shows the standard deviation of our prediction, it would be nice to have one value to describe how accurate our map is. To get this value we use cross-validation (cv). cv is a method that splits the known data into two disjointed sets; a training set and a test set. The training set is used to build the model and the test set is used to compute how accurate the prediction is [17]. For example, we would like to estimate a function $f(x)$ and we are given 10 points that are on $f(x)$. We can take one of those 10 points to be our test set, and use the other 9 to make an estimation $\hat{f}(x)$ of what $f(x)$ looks like. Then we can evaluate the performance of our prediction by computing the loss between $\hat{f}(x)$ and the test point. We can use any loss function with cv, but the squared error loss, $(\hat{f}(x)-f(x))^2$, is the most common and the one we will be using.

However, when we follow this cv procedure, not all data is used for training and if our test set is small there is a possibility that the computed error has a large variance. Therefore, we will use k-fold cross validation. Instead of splitting our data in to two disjointed sets, we now split it into k equal sized disjointed sets. We pick one of those k sets to be our test set and use the other $k-1$ sets for the training. We repeat this process k times, such that each set is the test set exactly once. This way all our data is used for testing once, and all data is used for training at least once. The downside to this approach is that instead of having to only compute one model, we now must compute k models. If we take k equal to the number of given data points, we have a special case, namely leave-one-out cross-validation (LOO-CV) [17].

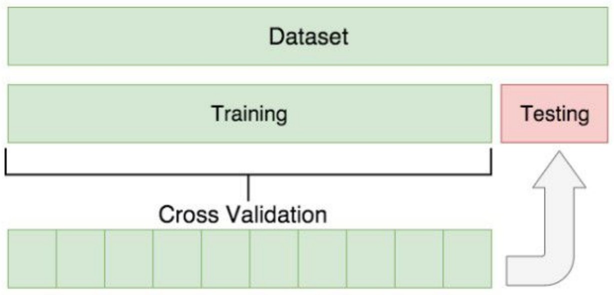


Figure 4.1: Visualisation cross validation. Source: [36]

Before we can start the cross-validating, the hyper parameters need to be determined. Because computing hyperparameters is a very expensive procedure, the hyperparameters were determined before we performed the k-fold cross validation. The same set of hyperparameters was used throughout, instead of computing new ones for each new training set. These hyperparameters were computed by running the hyperparameter estimation part of the Kriging procedure using all test and training data together.

The first step after determining the hyperparameters is defining the training and test sets. In our situation, we have to deal with three data sources, and since we do not know how accurate the 2PD and 3PD is, we decided to only use the 1PD to make the k test sets. This means that for our test set we had 1 1PD set, and the training set consisted of the other $k-1$ 1PD sets supplemented by the 2PD and 3PD when needed. Before the test sets are made, the 1PD is shuffled. Let us say that $\mathbf{x}^1$ and $\mathbf{y}^1$ are the arrays containing only the 1PD station locations and measurement. When these are shuffled the same permutation is applied to $\mathbf{x}^1$ and $\mathbf{y}^1$. The shuffling is necessary because the $\mathbf{x}^1$ contains the stations in location-wise order. If we do not shuffle, our test sets would contain stations located in the same area and we want them to be distributed all over the country.

For each test group we are going to compute the error and in the end take the average over those errors. When we compute the error we pick one test group j out of all the test groups. The remaining test groups are added to the training data and used to make a prediction. Lets call the vector that predicts the temperatures for our test stations $\mathbf{p}$, and the vector with the measured temperatures for the test stations $\mathbf{m}$. Then the error $\mathcal{E}_j$ for test set j is computed as

$$\mathcal{E}_j = \sum_{i=0}^{s} (\mathbf{p}_i - \mathbf{m}_i)^2$$

with s being the size of the test group. To get to the final error $\mathcal{E}$, we repeat this for every test set we made and add their errors up, take the average and compute the square root of it. The average of this is taken by dividing by the number of 1PD stations n_1, i.e.

$$\mathcal{E} = \frac{1}{n_1}\sqrt{\frac{1}{k}\sum_{j=1}^{k}\mathcal{E}_j}.$$

Chapter 5

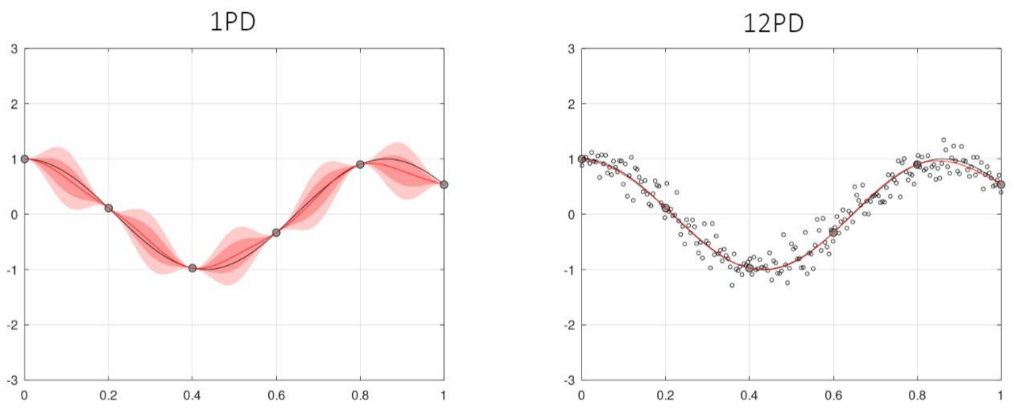
Results

5.1 Results from the synthetic data

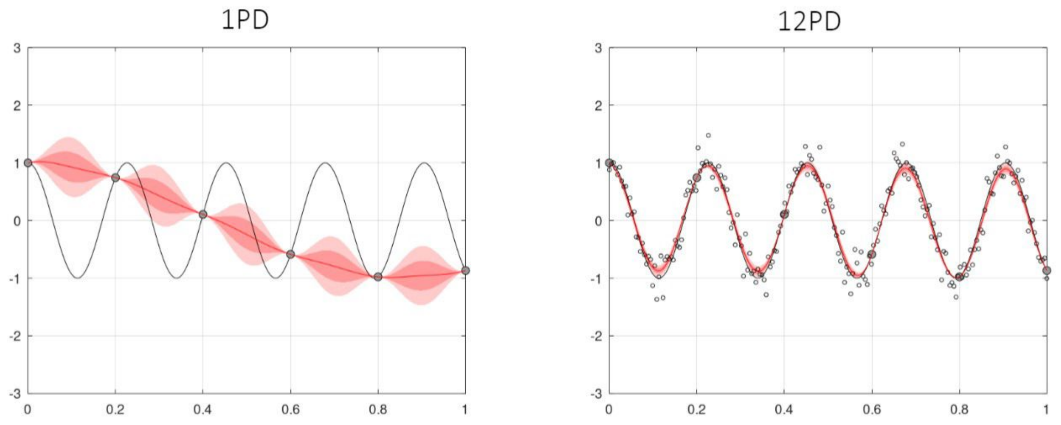
Recall that in section 2.3 synthetic data was introduced, real station locations were used which were all assigned a synthetic temperature defined as follows

$$\begin{aligned} \mathbf{y}^s &= a(\cos N\pi\mathbf{x}_1^s + \cos N\pi\mathbf{x}_2^s) + \epsilon \\ g(\mathbf{x}^g) &= a(\cos N\pi\mathbf{x}_1^g + \cos N\pi\mathbf{x}_2^g) \end{aligned}$$

here a stands for the amplitude is determined by real temperature data, the superscript g stands for 'grid' and s for 'station'. To $\mathbf{y}^s$ a synthetic error ϵ was added consisting out of the bias and noise. The result of the true temperature field $g(\mathbf{x}^g)$ was a lattice pattern shown in Figure 2.2, where the size of the pattern was determined by parameter N. The maps produced by $g(\mathbf{x}^g)$ are going to be predicted using $\mathbf{y}^s$. These true temperature fields can be used to evaluate the accuracy of our predictions.
In our results, we expect to see that the predictions using 1PD, first and second party data (12PD) and first, second and third party data (123PD) all perform well if N is low, but we expect 12PD and 123PD to perform better than 1PD when N is higher. When N is high there are a lot more oscillations and more measurement points are needed to be able to identify them, for visualisation see Figure 5.1. Note that no bias was added to the 2PD, only noise.



(a)



(b)

Figure 5.1: Here we see the comparison between the performances of 1PD (on the left) and 12PD (on the right). In Figure 5.1a N is small and in Figure 5.1b N is large. The values of the true function are shown in black, first party measurements as big circles, second party measurements as smaller circles and the predictions are shown as a red line. The shaded red area around the line is a 95% confidence interval, computed as twice the standard deviation for the input value.
We see that when N is small, first party data is enough. However, when N is large first party data provides too few measurements for an accurate prediction but adding second party data ensures that an accurate estimation can be made

We can see that these expectations are indeed true when we look at the root mean squared errors (RMEs) that were computed between the prediction made for each grid point and $g(\mathbf{x}^g)$, the true temperature value for each grid point, for various combinations of first, second and third party data and various values of N. The results are shown in Figure 5.2. For a small value of N we see that all combinations involving first party data perform equally well. Between the values of $N = 0.6$ and $N = 1.5$ 1PD starts to perform slightly worse than 12PD, 13PD and 123PD, but beyond $N = 1.5$ the difference becomes more significant. At $N = 1.5$ we hit the Nyquist rate, which means that when the frequency of measurements per oscillation becomes lower than at that point, we will not be able to make an accurate reconstruction of the original temperature field because high frequencies are aliased to low frequencies as seen in the left panel of Figure 5.1b [37]. In that figure it can be seen that the measurements for 1PD are not frequent enough, therefore we can not model all the oscillations that are there.
Figure 5.2 also nicely shows why 2PD and 3PD are not used on their own: there is precise data needed, such as the 1PD set, to compute the bias and noise variables. The 2PD and 3PD contain bias and noise therefore they do not have the precise data needed to make accurate predictions for b and σ_n which leads to inaccurate predictions.

This figure tell us that for weather with low spatial variability such as temperature fields 1PD, 12PD, 13PD and 123PD are all performing equally well. However, after a certain level of spatial variability is reached 12PD, 13PD and 123PD perform better than 1PD. Therefore we can use 2PD and 3PD in addition to our 1PD to obtain a higher resolution weather prediction than the current one. Being able to perform at a higher resolution will lead to a nationwide reduction of uncertainty in the predictions, especially for the weather with a higher spatial variability such as rain or wind.

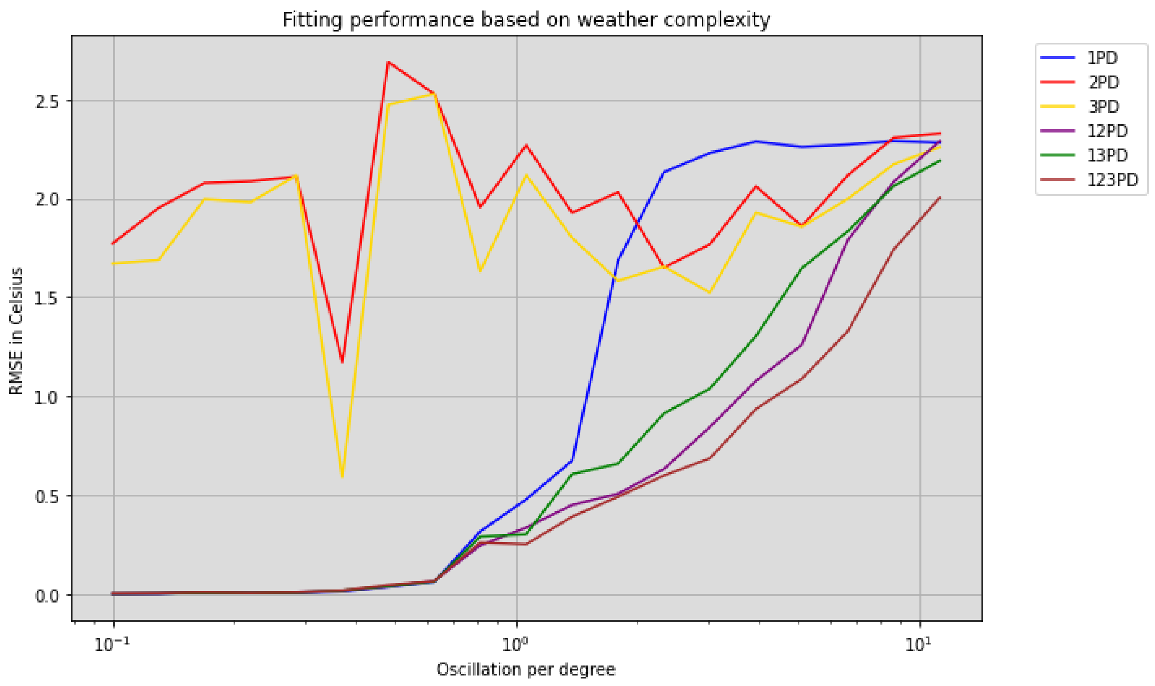


Figure 5.2: This figure shows the RMSE values for different combinations of first, second and third party data for various levels of spatial variability of the weather data. This RMSE was computed between the predicted values for each grid point and the true value of each grid point. It shows that for low spatial variability 1PD, 12PD, 13PD and 123PD perform equally well, but after 1.5 oscillation per degree 1PD performs significantly worse than the other options.

These findings are also reflected in the individual maps that were created using Kriging LE and $\mathbf{y}^s$. For a low N, such as $N = 0.1$ in Figure 5.3, the maps look the same for 1PD, 12PD and 123PD. When the uncertainty is computed for the three predictions as in section 4.3, all three end up an uncertainty of 0.0 degree Celsius.

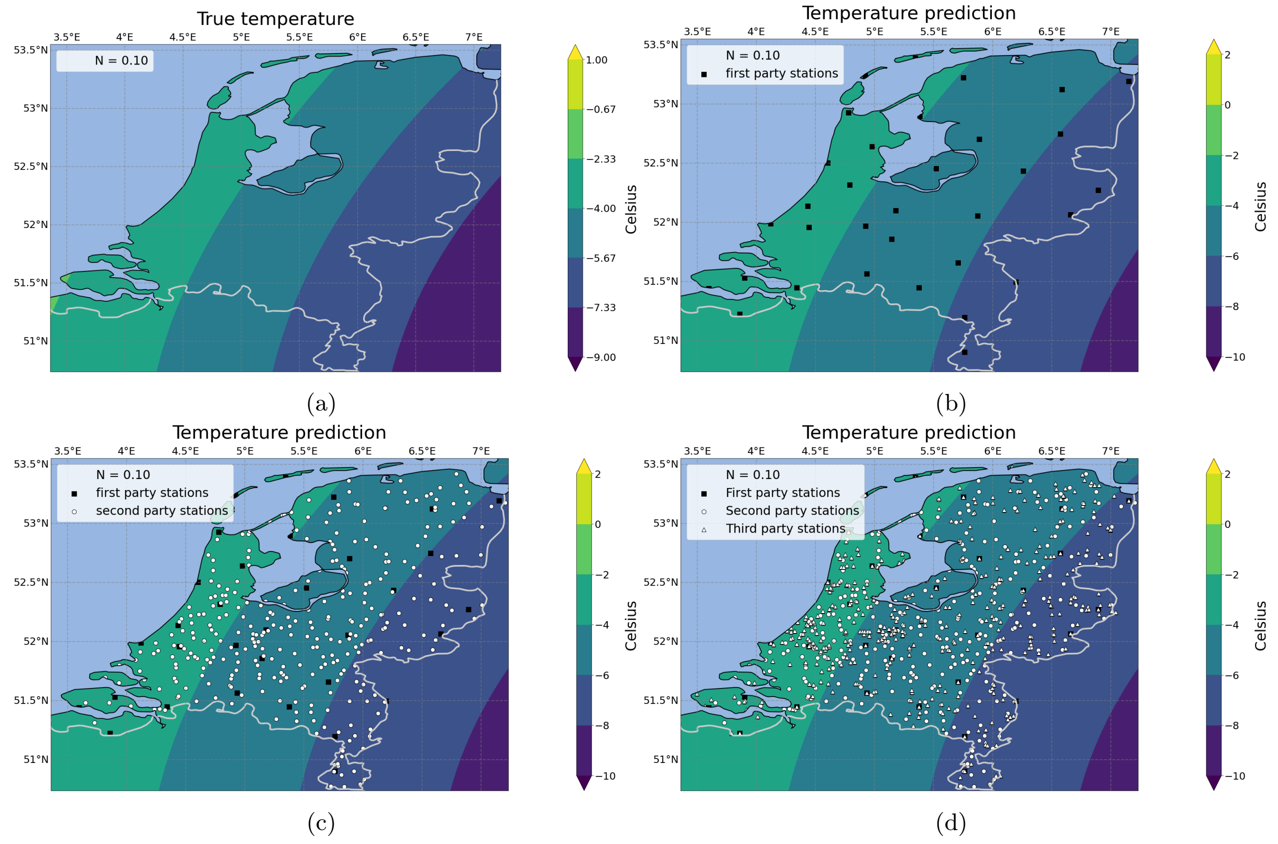
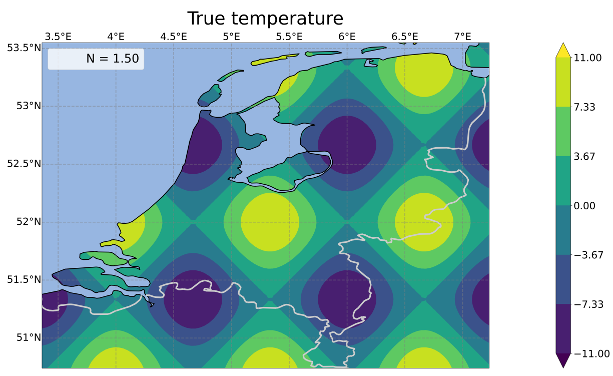
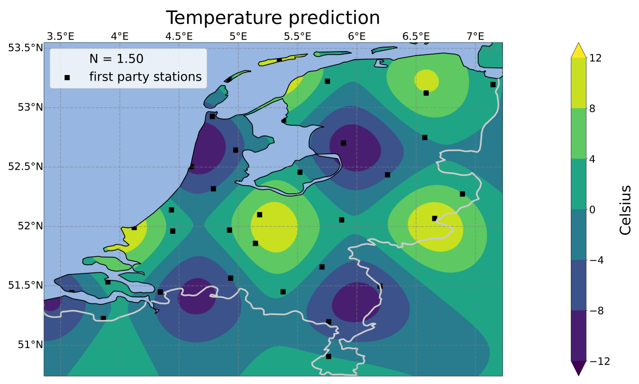


Figure 5.3: In Figure 5.3a the true temperature field is shown for $N = 0.1$. In the other figures the predictions of Figure 5.3a are shown using 1PD, 12PD and 123PD.

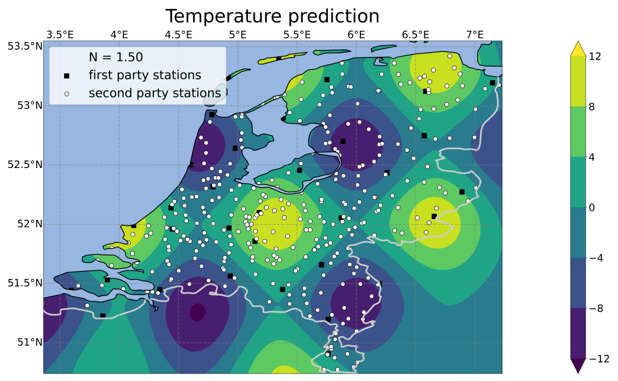
When we look at the results for $N = 1.5$ in Figure 5.4 the differences between the three options are starting to show. In Figure 5.4a the true temperature field is shown and in the following three images the predictions are shown, the predictions are similar but there are some improvements visible between going from 1PD to 12PD or 123PD. For example the sharp corners in the contours are better resolved where the density of stations is high. The improvement from going from 1PD to 12PD is also visible when we look at the uncertainty and difference maps. In the uncertainty maps in Figure 5.4e and Figure 5.4f we can see that the average uncertainty goes down from 1.232° Celsius to 0.168° Celsius and that for areas with little data available such as in the northern provinces of Friesland and Groningen the uncertainty also decreases significantly. Note that while 12PD and 123PD perform significantly better, 1PD is still able to produce a map that resembles the true temperature field. When we look at the differences between 12PD and 123PD Figure 5.4f and Figure 5.4g show what we also saw in Figure 5.2. 12PD and 123PD perform similarly but 123PD performs slightly better.



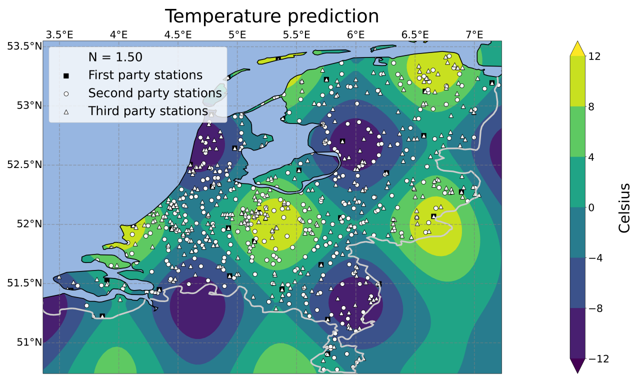
(a)



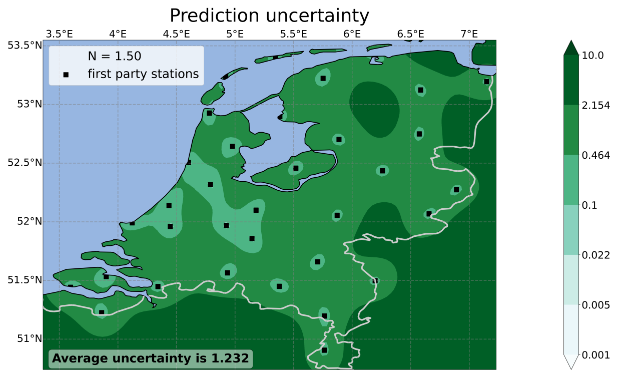
(b)



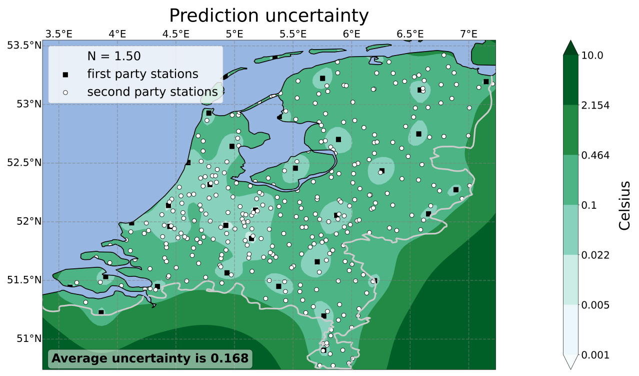
(c)



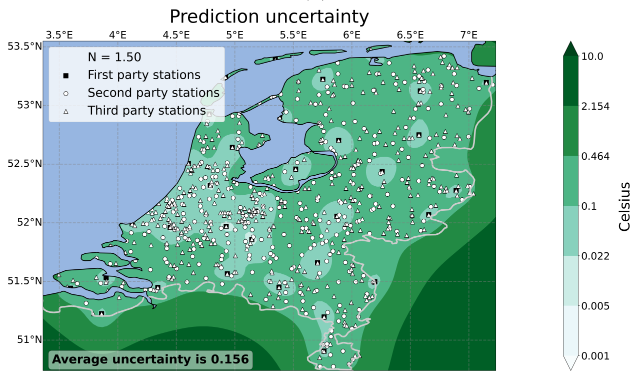
(d)



(e)



(f)

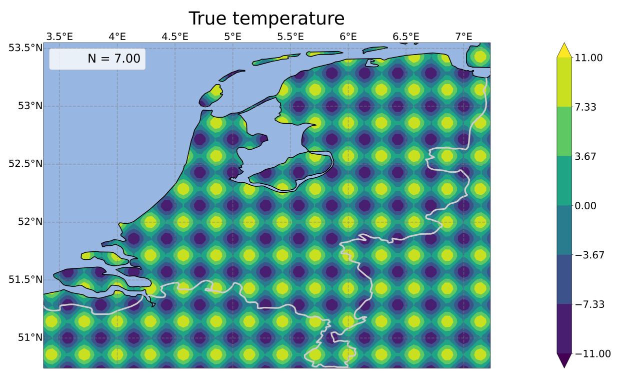


(g)

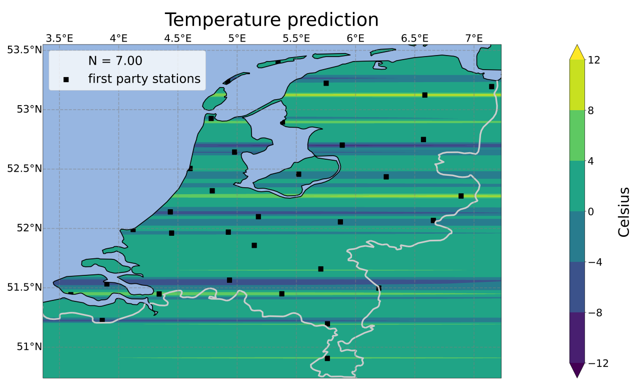
Figure 5.4: In Figure 5.4a the true temperature field is shown for $N = 1.5$. In the figures Figure 5.4b to Figure 5.4d the predictions of Figure 5.4a are shown using 1PD, 12PD and 123PD. The following three figures display the improvement from going from 1PD to 12PD to 123PD by showing the uncertainty of the predictions made.

Now we have seen two cases where 1PD made reasonable predictions, but let us look at a more extreme case of $N = 7$ shown in Figure 5.5. Here we can see that using only 1PD is not enough anymore to model the true

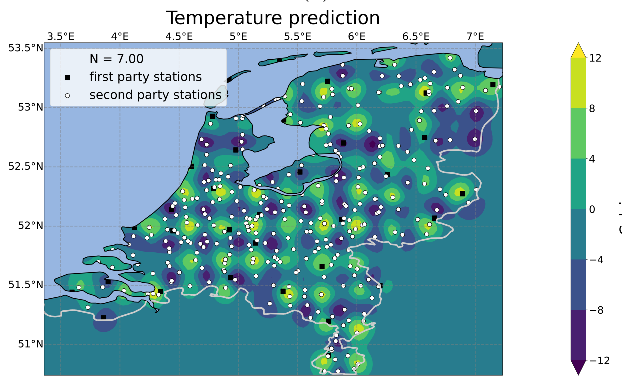
temperature field, in Figure 5.5b we can see a pattern that does not resemble Figure 5.5a at all. The prediction in Figure 5.5c resembles the true temperature map more, but we still see some holes in the pattern such as in the north of the country around Friesland and Groningen. This is also visible when we look at the uncertainty and difference maps of 12PD in Figure 5.5e and Figure 5.5g. Because there are so many oscillations the uncertainty is only low when close to stations, in areas where there are not many stations we therefore see larger errors in the prediction. In Figure 5.5g this is clearly visible as dark red or dark blue spots in the areas where there are holes in the pattern in Figure 5.5c. Figure 5.5g and Figure 5.5h display the differences between the prediction and the true temperature field, so the darker the red or blue the larger the difference between the predicted temperature and the true temperature is for a point.
When the third party data is added, the average uncertainty is reduced by 0.9°C and these spots with large uncertainty and large differences are reduced in size too as shown in Figure 5.5f and Figure 5.5h.



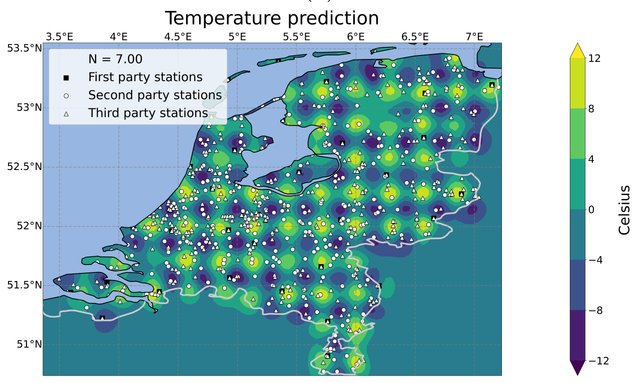
(a)



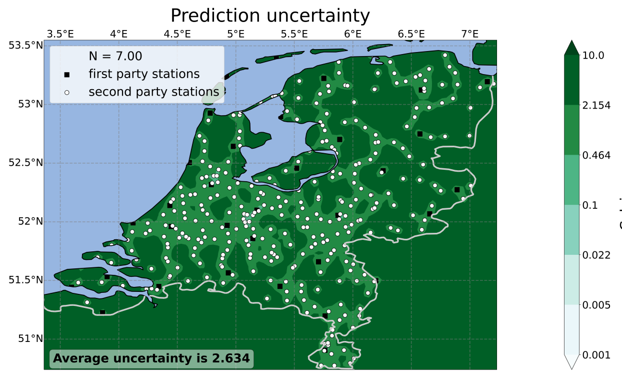
(b)



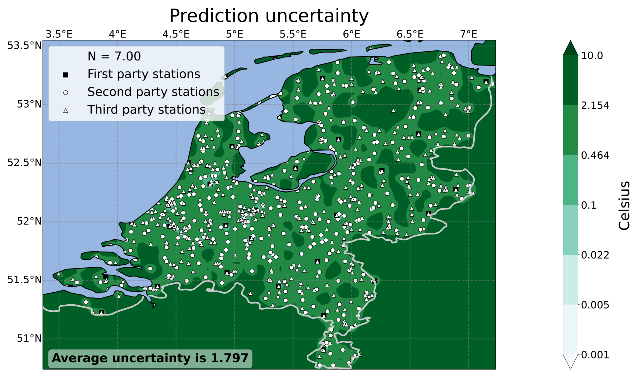
(c)



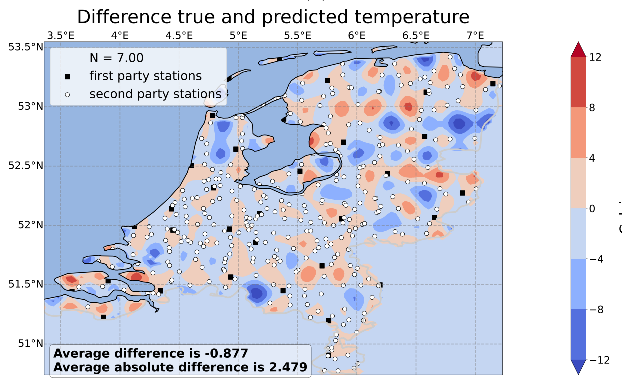
(d)



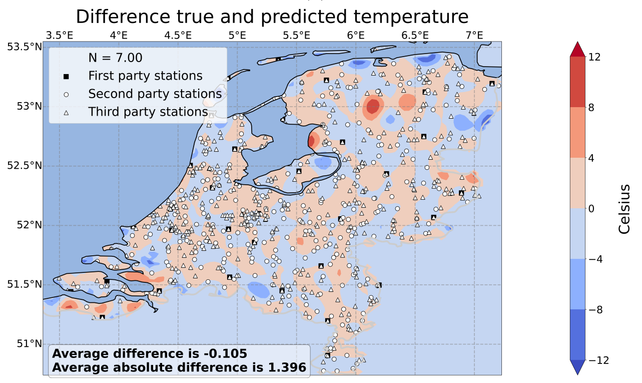
(e)



(f)


Difference true and predicted temperature
N = 7.00
first party stations
second party stations
Average difference is -0.877
Average absolute difference is 2.479
Celsius

(g)

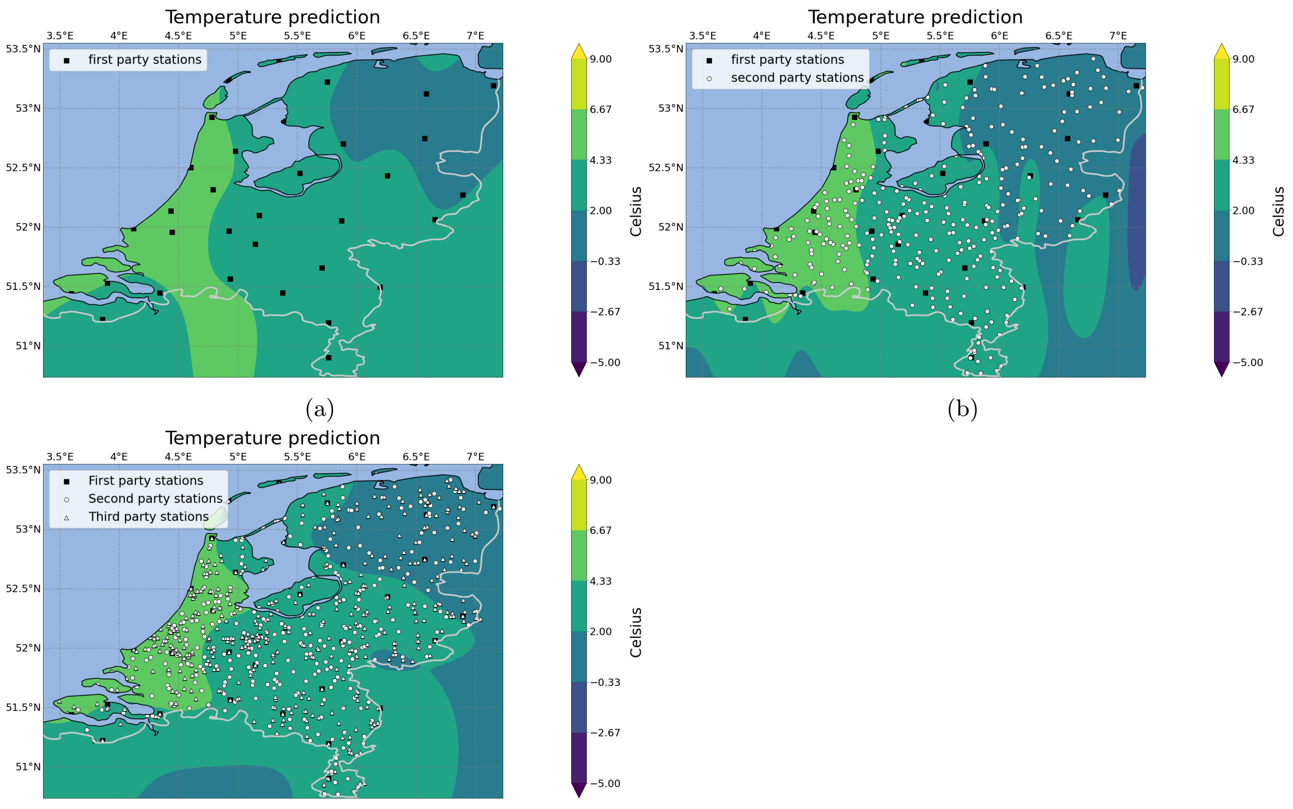

Difference true and predicted temperature
N = 7.00
First party stations
Second party stations
Third party stations
Average difference is -0.105
Average absolute difference is 1.396
Celsius

(h)

Figure 5.5: In Figure 5.5a the true temperature field is shown for $N = 7$. In the figures Figure 5.5b to Figure 5.5d the predictions of Figure 5.5a are shown using 1PD, 12PD and 123PD. We see that using just 1PD is not enough to make an accurate prediction. The following four figures show us the difference between the accuracy of using 12PD or 123PD. In Figure 5.5e and Figure 5.5f the uncertainties of the predictions are shown and in Figure 5.5g and Figure 5.5h the differences between Figure 5.5a and the predictions.

5.2 Results from the real data

In the previous section it was shown that for complex weather data it is beneficial to use 2PD and 3PD in addition to 1PD. However, the data used was synthetic and we would also like to see what happens when real data is used. However, temperature fields do not exhibit high spatial variability so we do not expect to see a large performance difference between using 1PD, 12PD and 123PD. In the predictions shown below in Figure 5.6 we see that there indeed are not any major differences between the three maps.

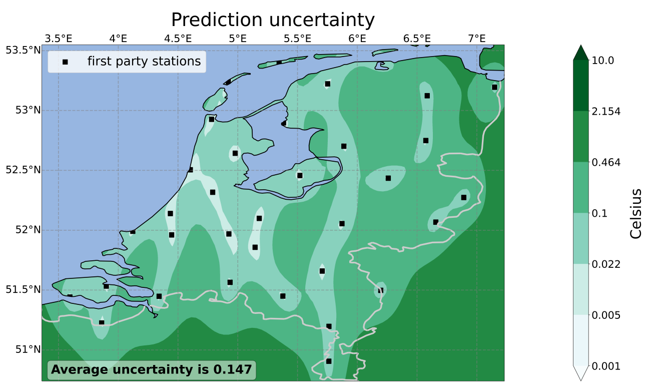


(a) (b)

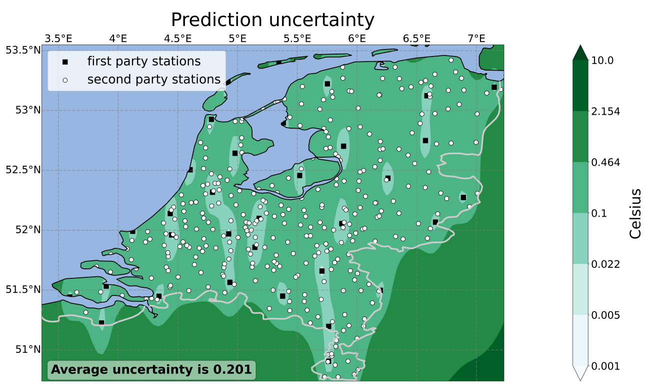
(c)

Figure 5.6: In these figures we see the predicted temperature maps for the Netherlands for the 25th of January 2019. In Figure 5.6a we see the prediction using just first party data, in Figure 5.6b first and second party data is being used and in Figure 5.6c first second and third party data was considered.

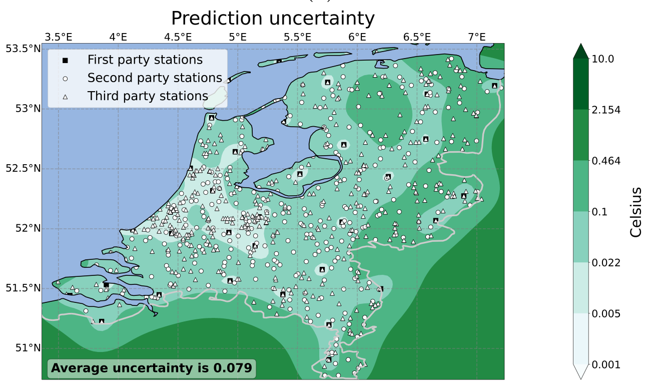
This is also reflected in the uncertainty of the predictions in Figure 5.7, there is a slight improvement in the uncertainty of the prediction when second and third party data are added to the model. However, we see that there is a slight decline in improvement when we only use 12PD. When cross-validation is used to compute the error similar results are obtained, 0.78 for 1PD, 1.71 for 12PD, 0.57 for 123PD.



(a)



(b)



(c)

Figure 5.7: The uncertainty for the predictions made in Figure 5.6.

Chapter 6

Conclusion

In this thesis we have introduced Kriging LE as a spatial regression method that allows us to deal with multi-fidelity data. This method made that possible because it is able to make estimations for spatial, bias and noise parameters. The adjustment of the parameters is necessary to make accurate predictions, since 2PD and 3PD are assumed not to be as precise as 1PD. Hence, in KrigingLE, the spatial parameter determines the spatial correlation between the stations, whereas the bias and noise levels are adjusted to the 2PD and 3PD observations.

KrigingLE was tested for two scenarios, using synthetic data and real data. The experiment with synthetic data shows that the current approach might be useful to model weather phenomena with high spatial variability when 1PD is combined with 2PD and 3PD. The experiment with real data shows that the predicted temperature fields average standard deviations decreases when 1PD is combined with 2PD and 3PD, which tells us there is a significant improvement when including 2PD and 3PD.
This means that second and third party data can be used to increase the spatial resolution of weather predictions and that they can be used to ensure that the nation wide prediction uncertainty is decreased significantly, especially for phenomena such as rain and wind.

An interesting topic for future studies would be to determine which second and third party stations are needed to improve the prediction and which stations are redundant. In this thesis we used all available second and third party stations that passed the quality control, but it might be possible that some stations that were used are redundant. This could happen when two stations are very close to each other, then it might not be necessary to use both stations. Especially with the increasing number of WOW stations it is interesting to investigate this, because eliminating redundant stations will decrease the computational cost. Besides looking at where some stations are redundant, it is also interesting to look at which areas need more stations. This can be especially promising for 3PD, because their stations are cheaper and can be placed where needed.
Another point for future research might be instead of using one noise variable for each party, using a noise variable for each station. This will allow the prediction to be more precise, but comes with increased computational cost.
Another point for future research might be looking into the number of noise and bias variables that are used. Instead of using one noise and bias variable for each party, one could, for example, look into the possibility of using a different noise and bias value for each sensor type. Increasing the number of noise and bias variables will ensure that they can be estimated more accurately, therefore the prediction will be more precise. However, since more variables need to be estimated this prediction will come with an increased computational cost.

6.1 Acknowledgements

I would like to thank my supervisors Irene and Jouke, who were always ready to help and offer suggestions and advice. Their knowledge and enthusiasm also made me more motivated and enthusiastic about the topic as well. Furthermore I would like to express my gratitude for my supervisor, Jason, who guided me throughout my thesis. I am also grateful to the KNMI for the opportunity to do this internship with them. I also appreciate the RDWD climate service team from the KNMI for being so welcoming and interested in my research. Thanks should also go to Palina Salanevich for being the second reader of my thesis.

Appendix A

Notation table

Quantity	Thesis	Code
data coordinates	$\mathbf{x}$	x
grid coordinates	$\mathbf{x}^*$	xi
quantity of interest	Y^*	$\cdots$
data	$\mathbf{y}$	y
drift	μ	mu
standard deviation	σ	st
observation matrix	$\mathbf{H}$	$\cdots$
prior covariance matrix	$\mathbf{P}$	$\cdots$
correlation matrix between sample data	$\mathbf{HPH}^T$	P
noise covariance matrix	$\mathbf{R}$	R
predictor matrix (correlation between stations and grid points)	ψ	b
correlation matrix between sample data with noise	$\mathbf{HPH}^T + \mathbf{R}$	A
posterior mean	$\mathbb{E}[\mathbf{Y}^*\|\mathbf{y}]$	yi
posterior covariance	$\mathrm{var}(\mathbf{Y}^*\|\mathbf{y})$	$\cdots$
posterior uncertainty	$\cdots$	ui
bias	$\mathbf{b}$	bias hyper
noise	σ_n	$\sqrt{\text{diagonal}(R)}$
correlation length	θ	theta
bias proxy	$\cdots$	eBiasProxy
noise proxy	$\cdots$	eStdProxy
bias proxy coefficient	$\cdots$	ebiahyper
noise proxy coefficient	$\cdots$	estdhyper

Appendix B

Why is the Cholesky decomposition allowed?

As mentioned section 4.1 a Cholesky decomposition is only allowed for matrices that are Hermitian positive (semi) definite. If a matrix is Hermitian positive definite the decomposition is unique, if it is Hermitian positive semi-definite, this is not necessarily the case. For our purpose the decomposition does not need to be unique, thus we are going to prove that $A = HPH^T + R$ is Hermitian and positive semi-definite.

Proof.

Definition B.0.1. A matrix X is Hermitian if and only if X is equal to its own conjugate transpose, i.e. $x_{ij} = \overline{x_{ji}}$ [38].

Let $\mathbf{A} = \mathbf{HPH}^T + \mathbf{R}$, $\mathbf{A}$ consists of a composition of real valued matrices, therefore $\mathbf{A}$ must be real too. Hence, to show $\mathbf{A}$ is Hermitian, we only need to show that $\mathbf{A}$ is symmetric.
$\mathbf{A} = \mathbf{HPH}^T + \mathbf{R}$, where $\mathbf{HPH}^T$ is the correlation matrix between the grid points and therefore symmetric, and $\mathbf{R}$ is a diagonal noise covariance matrix and thus also symmetric. The sum of two symmetric matrices is again symmetric and therefore $\mathbf{A}$ is a Hermitian matrix.

Now we need to prove that $\mathbf{A}$ is positive semi-definite.

Definition B.0.2. A matrix $\mathbf{A}$ is positive semi-definite if $\mathbf{x}^T\mathbf{A}\mathbf{x}$ is non-negative for every non-zero real column vector $\mathbf{x}$ [39].

If matrix $\mathbf{A}$ and matrix $\mathbf{B}$ both are positive semi-definite, their sum must be positive semi-definite too [39]. Then for $\mathbf{A}+\mathbf{B}$ we get for all non-zero real column vector's $\mathbf{x}$:

$$\mathbf{x}^T(\mathbf{A}+\mathbf{B})\mathbf{x} = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{B}\mathbf{x} \geq 0 \tag{B.1}$$

because $\mathbf{A}$ and $\mathbf{B}$ both positive semi-definite, thus $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ and $\mathbf{x}^T\mathbf{B}\mathbf{x} \geq 0$.
Hence it is enough to show that $\mathbf{HPH}^T$ and $\mathbf{R}$ are positive semi-definite to show that $\mathbf{A}$ is positive semi-definite. Lets start with $\mathbf{R}$. We know that $\mathbf{R}$ is a diagonal matrix with non-negative entries. This means all eigenvalues of $\mathbf{R}$ are larger or equal than 0 and therefore $\mathbf{R}$ is positive semi-definite [39].
Now we need to show that $\mathbf{HPH}^T$ is semi positive definite too. $\mathbf{P}$ is a covariance matrix, for which $\mathbf{P}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Now recall from subsection 3.4.1 that kernel function κ was chosen such that it was positive definite to ensure that $\mathbf{P}$ was positive (semi) definite. Now we apply H to get to the covariance matrix $\mathbf{HPH}^T$. Because matrix multiplication is associative and $\mathbf{P}$ positive (semi) definite,

$$\mathbf{x}^T\mathbf{HPH}^T\mathbf{x} = \mathbf{b}^T\mathbf{P}\mathbf{b} \geq 0 \tag{B.2}$$

Therefore $\mathbf{HPH}^T$ is also positive (semi)definite. Note that $\mathbf{P}$ may be (strictly) positive definite, but $\mathbf{HPH}^T$ only positive semi-definite if $\mathbf{H}$ has a nontrivial null-space.
This concluses that $\mathbf{HPH}^T + \mathbf{R}$ is a Hermitian positive semi-definite matrix. □

Appendix C

KrigingLE code

C.1 License

C.2 KrigingLE code

```python
import numpy as np
import scipy.optimize

def strd(x):
    out = str(np.round(x, decimals = 3))
    return(out)

def fit(x,y,eBiasProxy,eStdProxy ,hyperbounds, verbose=0):
    # normalise data
    ndim = x.shape[1]
    mu = np.mean(y)
    st = np.std(y)
    yn = (y-mu)/st
    ebian = eBiasProxy/st
    nebia = ebian.shape[1]
```

```
estdn = eStdProxy/st
#nestd = max(estdn.shape[1],1)
if estdn.shape[1] == 0:
    estdn = 0*yn + 0.01
# estimate hyperparameters
lowerbounds = []
for item in ('lb_bias', 'lb_noise', 'lb_theta'):
    if item == 'lb_bias':
        lowerbounds+=list(hyperbounds[item])
    else:
        lowerbounds+=list(np.log(hyperbounds[item]))

upperbounds = []
for item in ['ub_bias', 'ub_noise', 'ub_theta']:
    if item == 'ub_bias':
        upperbounds+=list(hyperbounds[item])
    else:
        upperbounds+=list(np.log(hyperbounds[item]))


lowerbounds = np.matrix(np.array(lowerbounds)).ravel()
upperbounds = np.matrix(np.array(upperbounds)).ravel()
nhyper = np.size(upperbounds)

def goalfunctionlocal(x0):
    print(ebian)
    out = goalfunction(x0,x,yn,ebian,estdn)
    out = out[0,0]
    return out

bounds=scipy.optimize.Bounds(lowerbounds.T, upperbounds.T,
   keep_feasible=False)
opt = scipy.optimize.differential_evolution(goalfunctionlocal,
   bounds)
transfhyper = np.matrix(opt.x)


nbiasproxy = ebian.shape[1]
nStdProxy = estdn.shape[1]
transfhyper = np.matrix(transfhyper)
nhyper = transfhyper.shape[1]
if nbiasproxy == 0:
    hyper = np.matrix(np.exp(transfhyper))
    ynb = yn.copy()
else:
    biashyper = np.matrix(transfhyper[:,0:nbiasproxy])
    hyper = np.matrix(np.exp(transfhyper[:,nbiasproxy:nhyper]))
    ynb = yn - ebian*biashyper.transpose()
hypernoise = hyper[:,0:nStdProxy]
hypertheta = hyper[:,nStdProxy:]
# compute initial state
A = buildA(x,estdn,hypernoise, hypertheta)
y0n = np.linalg.solve(A,ynb)

# output
model = {'x': x, 'estdn': estdn, 'mu': mu, 'st': st,
                'biashyper0': biashyper, 'hypertheta0': hypertheta, '
                   hypernoise0' : hypernoise,
                'y0n': y0n}
        return model


    def predictMean(model,xi):

        # predict value
        b = corrmatrix(xi,model['x'],model['hypertheta0'])
        yin = b*model['y0n']

        # denormalise prediction
        yi = model['mu'] + model['st']*yin
        # output
        return yi


    def predictStd(model,xi):

        #inflationFactor = len(model['y0n'])/(len(model['y0n'])-len(
           model['hyper0'])-len(model['biashyper0']))

        # predict variance
        b = corrmatrix(xi,model['x'],model['hypertheta0'])
        A = buildA(model['x'],model['estdn'],model['hypernoise0'],model
           ['hypertheta0'])
        #varyin = inflationFactor * ( 1. - np.diag(np.dot(b,np.linalg.
           solve(A,b.T))) )
        varyin = 1. - np.diag(np.dot(b,np.linalg.solve(A,b.T)))
        varyin = np.matrix(varyin).T
        uin = np.sqrt(varyin)

        # denormalise prediction
        ui = model['st']*uin

        # output
        return ui

    def corrmatrix(x1,x2,hyper):

        ndim = x1.shape[1]
        nhyper = hyper.shape[1]
        theta = hyper[:,(nhyper-ndim):(nhyper)]

        D = np.zeros((np.shape(x1)[0],np.shape(x2)[0]))

        for k in range(ndim):
            X2,X1 = np.meshgrid(np.squeeze(np.asarray(x2[:,k])),
                                np.squeeze(np.asarray(x1[:,k])))
            H = X2-X1
            if ndim==1:
                D += -0.5 * H**2 * np.squeeze(np.asarray(theta))**-2
            else:
                D += -0.5 * H**2 * np.squeeze(np.asarray(theta))[k]**-2
        C = np.exp(D)
        return C

    def buildA(x,estdn,hypernoise, hypertheta):
        nestd = estdn.shape[1]
        P = corrmatrix(x,x,hypertheta)
        R = np.zeros((np.shape(x)[0],np.shape(x)[0]))
        for k in range(np.shape(x)[0]):
            for n in range(nestd):
                R[k,k] = R[k,k] + (hypernoise[:,n]*estdn[k,n])**2
        A = P + R
        return A

    def goalfunction(transfhyper,x,yn,ebian,estdn):
        nbiasproxy = ebian.shape[1]
        nStdProxy = estdn.shape[1]
        transfhyper = np.matrix(transfhyper)
        nhyper = transfhyper.shape[1]
        if nbiasproxy == 0:
            hyper = np.matrix(np.exp(transfhyper))
            ynb = yn.copy()
        else:
            biashyper = np.matrix(transfhyper[:,0:nbiasproxy])
            hyper = np.matrix(np.exp(transfhyper[:,nbiasproxy:nhyper]))
            ynb = yn - ebian*biashyper.transpose()
        hypernoise = hyper[:,0:nStdProxy]
        hypertheta = hyper[:,nStdProxy:]
        A = buildA(x,estdn,hypernoise, hypertheta)
        goal = np.sum(np.log(np.linalg.eigvals(A))) + np.dot(ynb.T,np.
           linalg.solve(A,ynb))

        goal = np.real(goal)

        return goal
```

Appendix D

Code exchange log

26-10-2021, Jouke to Daniëlle This file contained the initial KrigingLE code. It also provided four example codes on how to use the kriging for estimating a one or two dimensional function using first party data without proxies and initial guesses, and examples on how to use first and second party data without proxies and initial guesses, with proxies and with proxies and initial guesses.

4-1-2021, Daniëlle to Xinrong Shared the KrigingLE code, that used a dictionary for the hyperbounds. Also shared 'realdata' code that puts the given data in the right format, calls the right krigingLE functions and eventually plots the predictions. Also shared 'syntheticdata' code, which uses synthetic data to simulate stations and use that for a polynomial regression interpolation to make a prediction.

3-3-2022, Irene to Daniëlle Irene sent me a code to produce a mask that could filter out grid points in Germany, Belgium or the sea. Unfortunately I was not able to install certain python packages needed for this program, so she sent me the already computed binary mask file instead.

Bibliography

[1] Irene Garcia-Martì et al. 'From proof-of-concept to proof-of-value: approaching third- party data to operational workflows of national meteorological services'. In: (2022).

[2] Adrien Napoly et al. 'Development and Application of a Statistically-Based Quality Control for Crowdsourced Air Temperature Data'. In: *Frontiers in Earth Science* 6 (2018). ISSN: 2296-6463. DOI: `10.3389/feart.2018.00118`. URL: `https://www.frontiersin.org/article/10.3389/feart.2018.00118`.

[3] Alan O Sykes. 'An introduction to regression analysis'. In: (1993).

[4] Johan Frederik Steffensen. *Interpolation*. Courier Corporation, 2006, pp. 1–4.

[5] *Discrete and Continuous Data*. [Online; accessed 16-5-2022]. URL: `http://wiki.gis.com/wiki/index.php/Discrete_and_Continuous_Data#Continuous_Data`.

[6] Marco A Azpurua and Karina Dos Ramos. 'A comparison of spatial interpolation methods for estimation of average electromagnetic field magnitude'. In: *Progress In Electromagnetics Research M* 14 (2010), pp. 135–145.

[7] ttk592. *Spline*. 2021. URL: `https://github.com/ttk592/spline`.

[8] GIS resources. *Types of interpolation methods*. [Online; accessed 9-02-2022]. URL: `https://gisresources.com/types-interpolation-methods_3/`.

[9] Daniel Kurtzman and Ronen Kadmon. 'Mapping of temperature variables in Israel: sa comparison of different interpolation methods'. In: *Climate research* 13.1 (1999), pp. 33–43.

[10] Wikipedia contributors. *Spline — Wikipedia, The Free Encyclopedia*. [Online; accessed 2-03-2022]. 2022. URL: `https://en.wiktionary.org/wiki/spline`.

[11] *Inverse Distance Weighting (IDW) Interpolation*. [Online; accessed 1-6-2022]. URL: `https://gisgeography.com/inverse-distance-weighting-idw-interpolation/`.

[12] Zaria Tatalovich, John P Wilson and Myles Cockburn. 'A comparison of thiessen polygon, kriging, and spline models of potential UV exposure'. In: *Cartography and Geographic Information Science* 33.3 (2006), pp. 217–231.

[13] Fabian Dablander. *Two properties of the Gaussian distribution*. [Online; accessed 31-03-2022]. Feb. 2019. URL: `https://fabiandablander.com/statistics/Two-Properties.html`.

[14] Jochen Görtler, Rebecca Kehlbeck and Oliver Deussen. 'A Visual Exploration of Gaussian Processes'. In: *Distill* (2019). https://distill.pub/2019/visual-exploration-gaussian-processes. DOI: `10.23915/distill.00017`.

[15] Wikipedia. *Multivariate normal distribution*. [Online; accessed 31-03-2022]. URL: `URL : https : / / en . wikipedia.org/wiki/Multivariate_normal_distribution`.

[16] Novi Quadrianto, Kristian Kersting and Zhao Xu. 'Gaussian Process'. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 428–439. ISBN: 978-0-387-30164-8. DOI: `10.1007/978-0-387-30164-8_324`. URL: `https://doi.org/10.1007/978-0-387-30164-8_324`.

[17] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006, p. 111.

[18] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012, pp. 515–542. URL: `http://noiselab.ucsd.edu/ECE228/Murphy_Machine_Learning.pdf`.

[19] Mohammad Shekaramiz, Todd K Moon and Jacob H Gunther. 'A Note on Kriging and Gaussian Processes'. In: (2019).

[20] Rodolphe Le Riche and Nicolas Durrande. 'An overview of kriging for researchers'. In: (2019).

[21] Noel Cressie. 'The origins of kriging'. In: *Mathematical geology* 22.3 (1990), pp. 239–252.

[22] Jouke HS de Baar et al. 'Kriging regression of PIV data using a local error estimate'. In: *Experiments in fluids* 55.1 (2014), pp. 1–13.

[23] Nick (https://stats.stackexchange.com/users/1913/nick). *Why are Gaussian process models called non-parametric?* Cross Validated. URL:https://stats.stackexchange.com/q/46624 (version: 2012-12-27). eprint: `https://stats.stackexchange.com/q/46624`. URL: `https://stats.stackexchange.com/q/46624`.

[24] Kilian Weinberger. *Lecture 15: Gaussian Processes.* Machine Learning for Intelligent Systems, Course page. July 2018. URL: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote15.html.

[25] Alexander IJ Forrester, András Sóbester and Andy J Keane. 'Multi-fidelity optimization via surrogate modelling'. In: *Proceedings of the royal society a: mathematical, physical and engineering sciences* 463.2088 (2007), pp. 3251–3269.

[26] Jouke HS De Baar, Richard P Dwight and Hester Bijl. 'Fast maximum likelihood estimate of the Kriging correlation range in the frequency domain'. In: *IAMG 2011: Proceedings of the International Association of Mathematical Geosciences" Mathematical Geosciences at the Crossroads of Theory and Practice", Salzburg, Austria, 5-9 September 2011* (2011).

[27] Alexander IJ Forrester, Andy J Keane and Neil W Bressloff. 'Design and analysis of" Noisy" computer experiments'. In: *AIAA journal* 44.10 (2006), pp. 2331–2339.

[28] Christopher K Wikle and L Mark Berliner. 'A Bayesian tutorial for data assimilation'. In: *Physica D: Nonlinear Phenomena* 230.1-2 (2007), pp. 1–16.

[29] *Machine learning crash course.* [Online; accessed 14-5-2022]. URL: https://developers.google.com/machine-learning/crash-course/generalization/video-lecture.

[30] David Duvenaud. 'Automatic model construction with Gaussian processes'. PhD thesis. University of Cambridge, 2014, pp. 8–10.

[31] Motonobu Kanagawa et al. 'Gaussian processes and kernel methods: A review on connections and equivalences'. In: *arXiv preprint arXiv:1807.02582* (2018). [Online; accessed 15-4-2022], pp. 6–12. URL: https://arxiv.org/abs/1807.02582.

[32] Yanting Li, Shujun Liu and Lianjie Shu. 'Wind turbine fault diagnosis based on Gaussian process classifiers applied to operational data'. In: *Renewable Energy* 134 (2019), pp. 357–366.

[33] Michael Parker. 'Chapter 13 - Matrix Inversion'. In: *Digital Signal Processing 101 (Second Edition).* Ed. by Michael Parker. Second Edition. Newnes, 2017, pp. 149–162. ISBN: 978-0-12-811453-7. DOI: https://doi.org/10.1016/B978-0-12-811453-7.00013-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128114537000135.

[34] *scipy.optimize.differential_ evolution documentation.* [Online; accessed 28-5-2022]. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html.

[35] Oliver Knill. *Math 19b: Linear Algebra with Probability, Lecture 28:eigenvalues.* [Online; accessed 6-04-2022]. URL: https://people.math.harvard.edu/~knill/teaching/math19b_2011/handouts/lecture28.pdf.

[36] *Train/Test Split and Cross Validation in Python.* Towards Data Science. [Online; accessed 11-04-2022]. URL: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6#:~:text=In%20K%2DFolds%20Cross%20Validation,it%20against%20the%20test%20set..

[37] Emiel Por, Maaike van Kooten and Vanja Sarkovic. 'Nyquist–Shannon sampling theorem'. In: *Leiden University* 1 (2019), p. 1.

[38] RA Horn and CR Johnson. 'Matrix analysis second edition'. In: (2013).

[39] Adriaan Van den Bos. *Parameter estimation for scientists and engineers.* John Wiley & Sons, 2007.

List of Figures

List of acronyms

cv	cross-validation. 23, 24
GP	Gaussian process. 11, 12, 15
IDW	Inverse Distance Weighting. 8–10, 45
LOO-CV	leave-one-out cross-validation. 24
MLE	maximum likelihood estimator. 16, 22
PWS	personal weather stations. 2, 6
RMSE	root mean squared error. 27
WMO	World Meteorological Organisation. 2, 6

List of terms

k-fold cross validation method that computes how accurate the prediction is. Split the first party data in K groups, and use K-1 of those groups, combined with the second and/or third party data to make a prediction. Test how accurate the prediction is using the group of first party data that was not used for the prediction [17]. 24

Kriging LE is Kriging regression using a local error estimate. 15, 18, 21, 28, 34

posterior distribution or a posterior, of an unknown parameter is the probability distribution of said parameter after taking observations into account. 18

prior distribution or a prior, of an unknown parameter is the probability distribution of said parameter before observations are taken into account. 14, 15

WOW Weather Observation Website of the Met Office, the UK's national weather service. This provides the data from weather stations people have at home. `https://wow.metoffice.gov.uk/`. 2, 4, 6, 21, 34