



**Sonja Tripkovic, BSc**

**Construction of Mobile Performance Maps  
using Clustered Crowdsourced Measurements**

**Master's Thesis**

to achieve the university degree of

Master of Science

Master's degree programme: Telecommunications

submitted to

**Vienna University of Technology**

Supervision:

Univ.Prof. Dipl.-Ing. Dr.techn. Markus Rupp  
Senior Scientist Dipl.-Ing. Dr.techn. Philipp Svoboda

Institute of Telecommunications

Vienna, May 2020

---

## Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

May 22, 2020

Date

*Stripkovic*

Signature

---

## Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Markus Rupp, who provided me an opportunity to join their team and work on this diploma thesis. Without his valuable assistance, it would not be possible to conduct this study.

I am forever grateful to my advisor Dr. Philipp Svoboda for the continuous support of my study, his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and a mentor.

Furthermore, I am indebted to my many colleagues at the Institute of Telecommunications for their collaborative effort during data collection and consistent support. Thank you for the many laughs and for making the time of writing this thesis enjoyable.

Finally, I owe my deepest gratitude to my family and friends for their continuous and unparalleled love, help, and support. This thesis would not have been possible without them.



# Abstract

The goal is to create forecasts of network performance maps based on collected measurements from different data sources. Even though we focus on Reference Signal Receive Power (RSRP) as the measurement metric, the underlying model must be extendable to other parameters, such as data-rate or service quality. Additionally, we aim to exploit the underlying spatial properties of the data. We employ a Gaussian Process Regression (GPR). The superiority of the GPR against other regression models is its ability to provide a distribution of the prediction value rather than just a single value. This uncertainty can be exploited to determine the optimal location for the next measurement, such that the prediction error of the entire map is minimized. The task of collecting measurements can be outsourced to end-user devices, thus saving the network operators valuable time. We employ the clustering of the data, which offers a possibility of computational time reduction for the GPR prediction and the measurement noise averaging effect. We design three different measurement distribution scenarios and analyze the MSE degradation over different measurement point densities when clustering is applied. We investigate different cluster identification methods, applicable in the real-data measurements, where the nature of clusters is unknown in advance. By comparing their influence on the MSE of the GPR prediction, we conclude that the K-Means method is best suited for our purposes. Finally, we apply the K-Means cluster identification method in two experiments, using measurements collected by the XY-positioning table and a drone. We show that we can considerably reduce the number of GPR training points by clustering measurements, without significant loss of the prediction quality. The experimental results also confirm our simulation results on the number of required clusters for an accurate GPR prediction.



# Contents

<b>Abstract</b>	<b>v</b>
<b>1. Introduction</b>	<b>1</b>
1.1. Thesis Overview . . . . .	3
<b>2. Gaussian Process Regression</b>	<b>5</b>
2.1. Motivation . . . . .	5
2.2. State of the Art . . . . .	6
2.3. GPR Overview . . . . .	8
<b>3. Clustering Data for Efficient Performance Map Reconstruction</b>	<b>15</b>
3.1. Sampling from Multivariate Gaussian Distribution . . . . .	16
3.2. Evenly Spaced Cluster Centers with Evenly Spaced Cluster Points (S1)	20
3.3. Evenly Spaced Cluster Centers with Randomly Spaced Cluster Points (S2) . . . . .	27
3.4. Non-Uniform Clusters generated using Thomas Cluster Process (S3)	34
3.5. Comparison of S1, S2 and S3 . . . . .	37
3.5.1. Evenly Spaced clusters with Evenly Spaced Cluster Points (S1) vs Randomly Spaced Cluster Points (S2) . . . . .	37
3.5.2. Evenly Spaced Clusters (S2) vs Non-Uniform Clusters (S3)	39
3.6. Identifying Clusters . . . . .	44
3.6.1. Different Clustering Methods in Python . . . . .	45
3.6.2. Comparison of Clustering Methods based on the GPR pre- diction . . . . .	47
3.6.3. Performance Comparison of the Complete, Clustered and Identified Training Sets . . . . .	50
<b>4. Experimenting on Real Data</b>	<b>55</b>
4.1. XY-Table Measurements . . . . .	55

## Contents

---

4.2. Drone Measurements . . . . .	59
4.3. Sampling Strategy . . . . .	62
<b>5. Summary and Future Work</b>	<b>65</b>
5.1. Summary . . . . .	65
5.2. Possible Extensions . . . . .	66
<b>Appendices</b>	<b>67</b>



# 1. Introduction

A rising number of smartphone subscriptions worldwide, as well as an increase in average data volumes, push the mobile traffic growth forward day by day. According to CISCO, the annual run rate of mobile data traffic will grow almost 7-fold between 2017 and 2022. Additionally, the number of smart devices will grow 2-fold in the same time range, reaching 9 billion in number [1]. User experience is a critical factor in establishing and securing an advantage in the highly competitive telecom industry. This growing demand requires constant improvement of network performance while pressuring mobile broadband operators to enhance the quality of their services continuously.

Evaluation of network performance is based on the fusion of measurements from different data sources, and forecasting of performance metrics at locations with no measurements at hand. To reduce cost- and time-consuming conventional drive test measurements, the 3rd Generation Partnership Project (3GPP) defined a solution in their Release 10 specification under the name Minimization of Drive Tests (MDT). The solution employs the user User Equipment (UE) to collect cell-related metrics as well as position information and report them back to the central system [2]. With the number of smartphones growing, this concept of outsourcing performance measurements towards the end-user devices gains substantial importance in network performance development.

For data collection from end-users in Austria, a system called RTR-Nettest is deployed, operated by the Austrian Regulatory Authority for Broadcasting and Telecommunications (RTR) [3]. RTR-NetTest utilizes the transmissions from the smartphone to the RTR server to determine the quality parameters of users' Internet access. Additionally, to containing performance results of each user test, it also provides further meta information, such as signal strength of the UE throughout the measurement. Even though the RTR collected data is publicly available, we could not use it for this thesis purposes, as there are no areas yet with measurement point densities high

## 1. Introduction

---

enough to exploit clustering. Instead, for collecting data in our experiments, we first focus on the XY-positioning table, with a receiving mobile shielded in a box and connected to an external antenna for measuring the strength of the received LTE signal [4]. Secondly, we employ the measurements of the signal strength in LTE, collected using four UEs mounted on the drone [5].

We create forecasts of network performance maps based on collected measurements from different data sources. Even though we focus on Reference Signal Receive Power (RSRP)<sup>1</sup> as the measurement metric, the underlying model must be extendable to other parameters, such as data-rate or service quality. Additionally, we aim to exploit the underlying spatial properties of the data. As a solution to both previous matters, we employ the Gaussian Process Regression (GPR). The superiority of the GPR against other regression models is its ability to provide a distribution of the prediction value rather than just a single value. Thus providing a level of certainty of its predicted mean. We can exploit this uncertainty to determine the optimal location for the next measurement, such that the prediction error of the entire map is minimized. On the other hand, GPR does not scale well with large measurement sets since its computational complexity grows cubically with the number of measurements.

We are also faced with a problem of location accuracy, since the Global Positioning System (GPS), with moderately high inaccuracy, is used for UE positioning. GPR, however, requires exact locations of the measurement data to be known and has no parameter that could account for such inaccuracy. To solve this, we propose clustering of the measurement data, intending to locate the clusters, and find a single representative point for each group, that is then used as an input to the GPR. Such an approach has the advantage of reducing the GPR complexity on the one hand and embracing the presence of the location uncertainty on the other. We also answer the research question of which density of the measurements allows a sufficiently accurate reconstruction of network performance maps.

---

<sup>1</sup>Reference signal received power (RSRP), is defined as the linear average over the power contributions (in [W]) of the resource elements that carry cell-specific reference signals within the considered measurement frequency bandwidth [6].

## 1.1. Thesis Overview

In Chapter 2, we give an overview of the state of the art methods for GPR complexity reduction as well as a theoretical introduction to the GPR method. Intending to identify which density of points is necessary to achieve a specific MSE goal of the reconstructed map, we study how various types of data clustering impact the performance of map reconstruction. As the data collected in real-world scenarios is unknown, we first rely on data generated within simulations in Chapter 3, and based on these results, we strive to recover real-data performance maps in Chapter 4. The empirical analysis in Chapter 3, using regular and non-regular cluster distribution, as well as clusters created using statistical geometry, is extended to a real-world scenario by including the process of cluster identification. We investigate the impact of deriving clusters from the data on the prediction MSE and benchmark the performance of different clustering methods. In Chapter 3, we conclude on the optimal DD as well as the required density of measurement points. Chapter 4 applies the results from Chapter 3 in several experiments on real-data that include measurements collected using the XY-positioning table and a drone. Finally, we discuss the results and offer concluding remarks in Chapter 5.



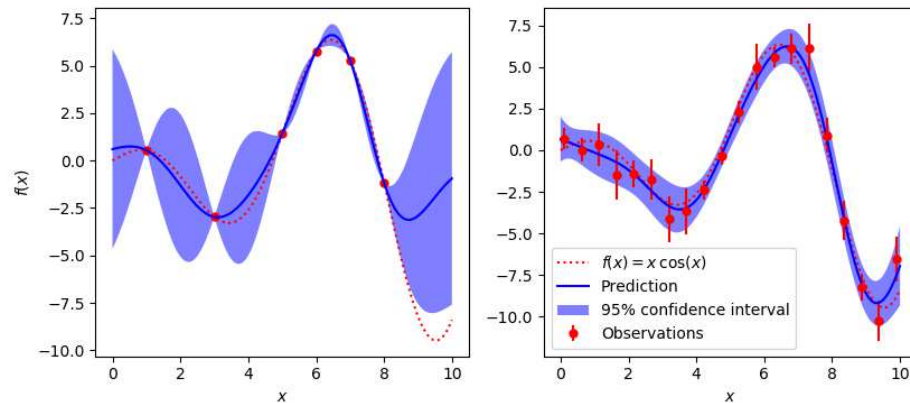
## 2. Gaussian Process Regression

### 2.1. Motivation

It is in network-operators interest to be aware of network performance at all locations, so they would know how and where the performance of their network can be improved. Therefore, there is a need to be able to precisely know network performance measures even at those locations where no measurements are at hand. Since we are distributing the measurements towards the end-user devices, the uncertainties in location and possibly the measurement noise level are going to be higher compared to traditional operator employed drive-test-measurements. While there are no regression models that could account for uncertainty in measurement location, one for accounting for the measurement noise does exist- the Gaussian Process Regression model.

A Gaussian Process (GP) model is a Bayesian probabilistic model for non-linear regression. As such, it can directly capture the model uncertainty by providing a distribution of the prediction value, rather than just one value as the prediction. An example of predicting a function  $f(x) = x \cos(x)$  from a fixed number of observations  $x$  is shown in Figure 2.1. The left plot depicts prediction from observations without measurement noise, while the right plot depicts the prediction from a noisy but a larger set of observations. Notice how in the noiseless case, the prediction goes exactly through the observations, while with the measurement noise, an uncertainty level in observations is allowed. Furthermore, with observations closer nearby one another, the 95% confidence interval between subsequent measurement decreases, making the prediction more accurate. This uncertainty is not directly captured in neural networks nor linear regression models.

## 2. Gaussian Process Regression



**Figure 2.1.:** GPR example: A noise-free case (left), a noisy case with known noise-level per datapoint (right) [7]

By selecting various covariance functions or kernels, that characterize correlations between different points in the process, Gaussian Process Regression (GPR) can add prior knowledge about the shape and spatial correlation of the underlying non-linear model. Under the assumption of a specific parametrized covariance structure underlying the data and given a sufficient density distribution of the training points, we can estimate the covariance function parameters based on the locations at which we have observed data, and use the inferred structure to make predictions at new locations. Because of the GPs' probabilistic structure, we can estimate variances at unsampled locations and use this information for the design of targeted sampling strategies - thus achieving the desired outcome for the network operator.[8]

### 2.2. State of the Art

Regression is a statistical method utilized for learning input-output mappings from empirical data (the training dataset) for continuous outputs. There is a vast choice between various forms of regressions, depending on the scenario and the purpose one aspires to accomplish.

The simplest and most extensively used type of regression is the linear regression, utilized for benchmarking of more complex regression forms. Unfortunately, its

simplicity results in an inadequacy of dealing with complex non-linear functions we want to infer. A simple approach to model such non-linear relationships is polynomial regression, based on merely adding different polynomial terms.

Polynomial regression only captures a certain amount of curvature in a non-linear relationship. But adding too many polynomial terms forces a polynomial curve to become overly flexible and take on some bizarre shapes for polynomial orders higher than 3 or 4. An alternative, and often superior, approach for modeling of non-linear relationships is to use splines [9].

Spline regression involves dividing the dataset into  $K$  distinct regions. Within each region, a polynomial function is fit to the data. However, these polynomials are constrained so that they join smoothly at the regions' boundaries or knots. Provided that the interval is divided into enough regions, this can produce an extremely flexible fit [10]. So instead of fitting a high-degree polynomial for the entire dataset, splines or piecewise polynomial regression with lower degree polynomials can be fit in separate dataset regions. But even splines are unable to account for various measurement noise levels, nor infer the underlying structure of the data since they always depend on the specified segment.

GPR models are typically the ones in use when exploiting the underlying structure and correlation of the data. In addition to the predictive mean, the GP treatment also yields a precise estimate of the noise level and predictive error bars [11]. While they can be rather powerful predictors with confidence intervals as important side information, they do require high accuracy in the training locations, and their computational complexity becomes hard to grasp with growing training data sets.

With high-accuracy of 5G New Radio (5gnr) technology, the continuous localization and user tracking are achievable. Di Taranto et al.[12] present in their publication how this accurate localization, together with signal strength, can be used to reduce signaling overhead in mobile communications. Furthermore, they propose the utilization of a spatial regression method, e.g., GPR, for generating signal strength maps from the collected measurements. Liao et al.[13] combine a non-zero mean autoregressive process with a zero-mean multivariate Gaussian process, therefore exploiting both spatial and temporal correlation for prediction of average channel gain.

The main limitation of GPR is the lack of online update possibility, since with each new training point GPR goes back to square one and recomputes the resource

## 2. Gaussian Process Regression

---

exhausting  $N \times N$ <sup>1</sup> matrix inverse. Since its computational time depends on the size of the training data set, many authors propose various state of the art solutions, that tackle the problem of GPR scalability from multiple angles. Liu et al.[14] summarize in their publication various proposed frameworks for GPR computational improvement. They categorize it into two main classes: a) global approximations that achieve sparsity of the full kernel (i) using a subset of the training data (subset-of-data); (ii) removing the entries of the covariance matrix with low correlations (sparse kernels); or (iii) employing a low-rank representation (sparse approximations) and b) local approximations exploiting (i) naive-local-experts which directly employs the pure local experts for prediction; (ii) mixture-of-experts and (iii) product-of-experts both boosting the predictions through model averaging.

Since we focus our measurements on the 4G networks, we still have to take GPS location uncertainty [15] into account. Muppirisetty, Svensson, and Wymeersch propose a framework for a GPR that can account for location uncertainty during learning/training and prediction/testing. Nevertheless, after the first tests, this method did not provide us with sufficiently accurate performance. Therefore we concentrate on different clustering methods of the training data set, intending to embrace existent location uncertainty while simultaneously reducing the GPR computational time.

### 2.3. GPR Overview

GPR is a broadly applied tool thoroughly explained in various books and publications [11] [17]. In this section, we summarize the GPR framework and assumptions under which we deploy it.

Let us suppose an exact training set is given, meaning the exact locations are known, and for each of those locations, a noisy observation has been measured. Each location is specified with  $N$ -dimensional real vector  $\mathbf{x} \in \mathbb{R}^N$  (e.g., coordinates in space), and a collection of those train point vectors is summarized in matrix  $\mathbf{X}$ . The noisy measurements are denoted by real values  $y_i$ , and consist of scalar real valued function  $f(\mathbf{x}_i)$  and observation error  $n_i$ :

$$y_i = f(\mathbf{x}_i) + n_i \quad (2.1)$$

---

<sup>1</sup> $N$  is the size of the training data set



where  $\mathbf{x}_i$ ,  $i = 1, \dots, I$ , denotes different locations in the training data set. We model  $f(\mathbf{x})$  as a Gaussian process, fully parametrized by a mean function

$$m(\mathbf{x}) := \mathbb{E}\{f(\mathbf{x})\} \quad (2.2)$$

and a covariance function

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &:= \text{cov}\{f(\mathbf{x}), f(\mathbf{x}')\} \\ &= \mathbb{E}\{(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))\}. \end{aligned} \quad (2.3)$$

Therefore any set of  $I$  random variables  $f(\mathbf{x}_i)$ ,  $i = 1, \dots, I$  are jointly Gaussian. Combining them into a random vector  $\mathbf{f} := (f(\mathbf{x}_1) \cdots f(\mathbf{x}_I))^T$ , we can write their joint probability density function as:

$$\begin{aligned} p(\mathbf{f}) &= p(f(\mathbf{x}_1), \dots, f(\mathbf{x}_I)) \\ &= \frac{1}{\sqrt{(2\pi)^I \det(\mathbf{K})}} \exp\left(-\frac{1}{2} (\mathbf{f} - \mathbf{m})^T \mathbf{K}^{-1} (\mathbf{f} - \mathbf{m})\right), \end{aligned} \quad (2.4)$$

where

$$\mathbf{m} := \mathbb{E}\{\mathbf{f}\} = (m(\mathbf{x}_1) \cdots m(\mathbf{x}_I))^T \quad (2.5)$$

and

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) \triangleq \text{cov}\{\mathbf{f}\} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_I) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_I) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_I, \mathbf{x}_1) & k(\mathbf{x}_I, \mathbf{x}_2) & \cdots & k(\mathbf{x}_I, \mathbf{x}_I) \end{pmatrix}. \quad (2.6)$$

Assuming we write the function  $\mathbf{f}$  as a very long vector, our goal is to be able to predict function value  $f(\mathbf{x}_*)$  (e.g. RSRP) at any given test location  $\mathbf{x}_*$ , or learn the function behind the model. We would like to consider every possible function that matches our data, independent of the number of function parameters involved. GPR is often referred to as a non-parametric method as if there are no parameters in our function. However, there are theoretically infinitely many, as we do not need to specify them upfront.

Covariance function carries our assumptions about the function we wish to infer, and as such, represents the central component of the GP predictor. In supervised learning,

## 2. Gaussian Process Regression

the concept of similarity between data points is a crucial ingredient. It reflects that points with data inputs  $\mathbf{x}$  in close vicinity are inclined to have similar target function values  $f(\mathbf{x})$ . Therefore training points that are near to the test point are likely to be more informative regarding prediction at that point, compared to those training points further away from it. In the Gaussian process framework, the *covariance function* defines this notion of data points similarity, and is interchangeably used with the term *kernel*.

A kernel is a general name of a function mapping a pair of inputs  $\mathbf{x}$  and  $\mathbf{x}'$  into the set of real numbers. A real kernel is symmetric if  $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ , therefore the covariance function is symmetric per definition. Valid covariance function is also always positive semi-definite, meaning all the eigenvalues of covariance matrix  $\mathbf{K}$  in 2.6 are non-negative.

covariance function	expression
constant	$\sigma_0^2$
linear	$\sum_{d=1}^D \sigma_d^2 x_d x'_d$
polynomial	$(\mathbf{x} \cdot \mathbf{x}' + \sigma_0^2)^p$
squared exponential	$\exp\left(-\frac{r^2}{2\ell^2}\right)$
Matérn	$\frac{1}{2^{v-1}\Gamma(v)} \left(\frac{\sqrt{2v}}{\ell} r\right)^v K_v\left(\frac{\sqrt{2v}}{\ell} r\right)$
exponential	$\exp\left(-\frac{r}{\ell}\right)$
$\gamma$ -exponential	$\exp\left(-\left(\frac{r}{\ell}\right)^\gamma\right)$
rational quadratic	$\left(1 + \frac{r^2}{2\alpha\ell^2}\right)^{-\alpha}$
neural network	$\sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^\top \Sigma \tilde{\mathbf{x}}')}}\right)$

**Table 2.1.:** Summary of several commonly-used covariance functions. The covariances are written either as a function of  $\mathbf{x}$  and  $\mathbf{x}'$ , or as a function of  $r = \|\mathbf{x} - \mathbf{x}'\|$  [11]

Assuming prior knowledge about the underlying function  $f$ , in terms of smoothness, symmetry, periodicity, etc., we can utilize it by choosing a one of in literature commonly-used kernels, summarized in Table 2.1, or create a new kernel via multiplication and/or addition of the existing ones.

Samples from a GP with the linear kernel are straight lines. Considering that a product of valid kernels is a valid kernel, the product of two linear kernels is a

quadratic kernel, giving rise to quadratic functions. This drift is further generalized to the polynomial kernel of order  $p$ .

The gamma-exponential family of covariance functions includes both the exponential and squared exponential kernels. While the squared exponential function is smooth, the exponential kernel is only continuous and not differentiable. Since the function approximations produced by kernel methods inherit the smoothness of the kernel, a smooth kernel, like the squared exponential, is suitable for fitting smooth functions. In contrast, a non-differentiable kernel, like the absolute exponential, is a better alternative for fitting non-differentiable functions.

The SE kernel is widely most used with GP. The reason for this is that functions drawn from a GP with a SE kernel are infinitely differentiable, which on the other hand, makes these functions very smooth. The class of Matern kernels proposed by Stein is a generalization of the squared exponential kernel and the absolute exponential kernel parameterized by an additional parameter  $\nu$ <sup>2</sup>. Since it is using the absolute exponential kernel, the Matern kernel is better suited to capture less smooth physical processes due to its finite differentiability compared to using SE kernel alone.

Though some recent publications [19] tackle the problem of selecting the appropriate kernel from the data, there is yet no established way of deciding which covariance function is the best fit for a particular application, and people mostly rely on trial and errors of different ones.

For the purpose of this thesis, we are using the *squared exponential* covariance function multiplied with a *constant* kernel.

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2l^2} \|\mathbf{x} - \mathbf{x}'\|^2\right) \quad (2.7)$$

The hyper-parameter  $l > 0$  of the SE kernel is the characteristic length scale, encapsulating the distance at which function values do not correlate any more, or in other words, how far can we reliably extrapolate from the training data. Small lengthscale value indicates that function changes quickly, while large values portray functions that change slowly. The  $\sigma_f^2 \geq 0$  of the constant kernel accounts for the signal variance that scales the SE kernel.

<sup>2</sup>For  $\nu = \inf$  the Matern coincides with SE kernel.

## 2. Gaussian Process Regression

Going back to equation (2.1), we model the observation error  $n_i$  as statistically independent of  $f(\mathbf{x}_i)$ , statistically independent of  $f(\mathbf{x})$  for all other  $\mathbf{x} \in \mathbb{R}^D$ , and Gaussian with zero mean and variance  $\sigma_n^2$ . In general, the measurement error does scale with the function value  $f(\mathbf{x})$  (signal strength). However, since we assume shadow fading alone, without path loss component, this dependency is rather small and can be neglected for calculation simplicity. Furthermore, the  $n_i$  are assumed statistically independent, so that

$$\text{cov}\{n_i, n_j\} = \sigma_n^2 \delta_{i,j} = \begin{cases} \sigma_n^2, & i = j \\ 0, & i \neq j. \end{cases} \quad (2.8)$$

Therefore, defining  $\mathbf{n} := (n_1 \cdots n_I)^T$ , we have  $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \mathbf{I}_I)$ , where  $\mathbf{I}_I$  denotes the  $I \times I$  identity matrix. Furthermore, since the observation noise is assumed zero mean, we have

$$\mathbb{E}\{y_i\} = \mathbb{E}\{f(\mathbf{x}_i)\} = m(\mathbf{x}_i) \quad (2.9)$$

$$\begin{aligned} \text{cov}\{y_i, y_j\} &= \text{cov}\{f(\mathbf{x}_i), f(\mathbf{x}_j)\} + \text{cov}\{n_i, n_j\} \\ &= k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 \delta_{i,j} \end{aligned} \quad (2.10)$$

Using vector notation  $\mathbf{f} := (f(\mathbf{x}_1) \cdots f(\mathbf{x}_I))^T$  and  $\mathbf{y} := (y_1 \cdots y_I)^T$ , we can rewrite (2.1) as  $\mathbf{y} = \mathbf{f} + \mathbf{n}$  with

$$\mathbf{m} \triangleq \mathbb{E}\{\mathbf{y}\}, \quad (2.11)$$

$$\Sigma \triangleq \text{cov}\{\mathbf{y}\} = \mathbf{K} + \sigma_n^2 \mathbf{I}_I \quad (2.12)$$

In other words equations 2.11 and 2.12 define the prior distribution  $\mathbf{y} \sim \mathcal{N}(\mathbf{m}, \Sigma)$ .

By incorporating the knowledge that the training data provides about the function, we are able to derive the posterior distribution from which we can sample. We first write the joint prior distribution of the training outputs,  $\mathbf{y}$ , and the test outputs  $\mathbf{f}^*$ :

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & \mathbf{K}^* \\ \mathbf{K}^{*T} & \mathbf{K}^{**} \end{bmatrix}\right) \quad (2.13)$$

With  $\mathbf{K} = \mathbf{K}(\mathbf{X}, \mathbf{X})$ ,  $\mathbf{K}^* = \mathbf{K}(\mathbf{X}, \mathbf{X}^*)$ ,  $\mathbf{K}^{**} = \mathbf{K}(\mathbf{X}^*, \mathbf{X}^*)$ , where  $\mathbf{X}$  and  $\mathbf{X}^*$  represent the collections of training and test points. By conditioning the joint Gaussian prior distri-

bution on the observations, the posterior distribution is derived, and characterized by following mean and covariance functions:

$$\begin{aligned}\mathbf{f}^* | \mathbf{y}; \mathbf{X}, \mathbf{X}^* &\sim \mathcal{N}(\mathbf{m}^*, \text{cov}(\mathbf{f}^*)), \\ \mathbf{m}^* &\triangleq \mathbb{E}[\mathbf{f}^* | \mathbf{y}; \mathbf{X}, \mathbf{X}^*] = \mathbf{K}^{*T} [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y} \\ \Sigma^* &\triangleq \text{cov}(\mathbf{f}^*) = \mathbf{K}^{**} - \mathbf{K}^{*T} [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}^*\end{aligned}\quad (2.14)$$

Therefore, using this posterior distribution, we can sample points for any given test location  $\mathbf{x}^*$ , and not only can we get the expected value of the function at that location, but also its level of uncertainty.

In the case we do not know the hyperparameters of the covariance function in advance, we can learn them by maximizing the log marginal function [11].

So the GPR consists of 2 stages, parameter learning from the noisy training dataset and prediction at known test locations. As discussed, the relevant hyperparameters are signal variance  $\sigma_f^2$ , decorrelation distance (DD)  $l$  and noise variance  $\sigma_n^2$ .

In Chapter 3, we assume these hyperparameters are precisely known, and investigate the influence of different training locations distributions on the GPR prediction of network parameters, e.g., Reference Signal Receive Power (RSRP).



### 3. Clustering Data for Efficient Performance Map Reconstruction

Using GPR and a sufficient number of crowdsourced measurements in the area of interest, we can recreate the performance map of that area. Additionally, the distribution of those measurements strongly influences performance quality. Assuming equally distributed measurements over the entire area of interest will bring the best performance while drawing them together reduces our knowledge of the underlying map. In reality, the distribution of mobile users differs across rural and urban areas. In rural areas, we experience a low density of crowdsourced measurement points, whereas, in urban areas, we perceive much higher density. Since people tend to gather in specific locations more than others, we still face a problem of heterogeneous distribution of measurement points. Therefore it is expected to have large densities of crowdsourced measurements at train stations, stadiums, etc. Since people move in "clusters", it makes sense to model measurement points using a random cluster process, rather than equidistant sampling. Furthermore, we can use this naturally given clusters to our advantage, by finding a representative measurement point for each group and feeding it as a single GPR input. In addition to reducing the GPR complexity, we can even lessen the GPS induced location errors for each of those cluster points. In the following, we will start by explaining how we obtain exact performance map values at both test and train locations using sampling from multivariate Gaussian distribution in 3.1. We will proceed with the introduction of different cluster types we generate and use as train points (see Figure 3.1):

- Scenario 1 (S1): evenly spaced cluster centers with evenly spaced cluster points in 3.2,
- Scenario 2 (S2): evenly spaced cluster centers with randomly spaced cluster points in 3.3,

### 3. Clustering Data for Efficient Performance Map Reconstruction

- Scenario 3 (S3): non-uniform clusters generated using Thomas Cluster Process in 3.4.

Under the assumption of known GPR hyper-parameters, we average over cluster values and cluster locations to get a single representative training point for each cluster. Finally, we perform GPR prediction using equation (2.14) and representative cluster points as predictor input and discuss the influence of different cluster parameters on predictions. To analyze and compare predictions using different cluster types with varying cluster parameters as training dataset, we introduce distance metric Mean Squared Error (MSE), that measures the average of the squares of the errors according to:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (3.1)$$

where  $y_i$  is the exact value at test location  $i$ ,  $\hat{y}_i$  is the predicted value at test location  $i$  and  $N$  is the number of test locations.

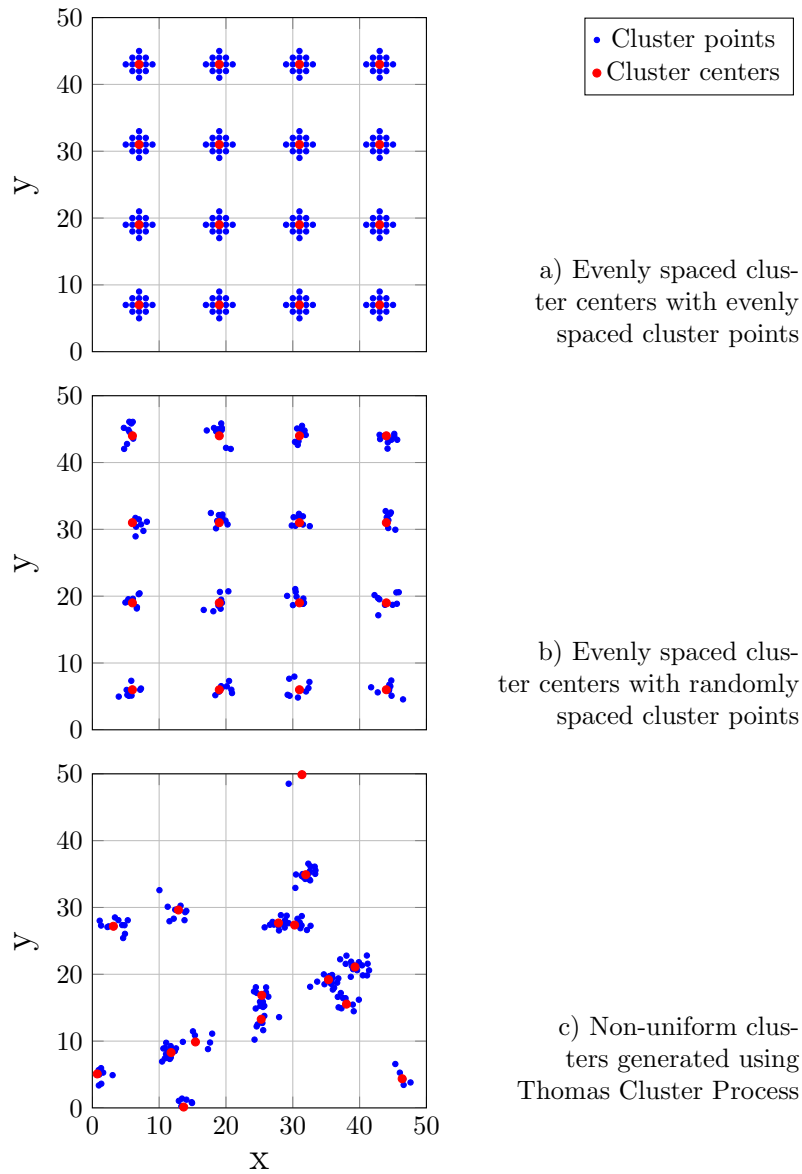
Nevertheless, the comparison of previously mentioned scenarios is not straightforward. For this purpose, we define the density of points as the number of measurement points in the area of interest that are available for training. For a fair comparison, we need to keep the density of points constant in all 3 cases, while working with random distribution in S3. Therefore, a few cluster centers may lie on the edge of the area of interest while their cluster points remain outside of it discarded as training data (see Figure 3.1 (c)). A solution to this problem is discussed in Section 3.5, where we first create the random clusters in Scenario 3, and then based on their density of points, we specify the clusters in Scenarios 2 and 1. The definition of point density is needed for real-world scenarios as well, as it makes different measurement layouts such as crowdsourcing and drive-tests comparable.

#### 3.1. Sampling from Multivariate Gaussian Distribution

To understand how the sampling from multivariate Gaussian distribution works, we first must understand the sampling from the normal distribution  $\mathcal{N}(0, 1)$  (Figure 3.2).



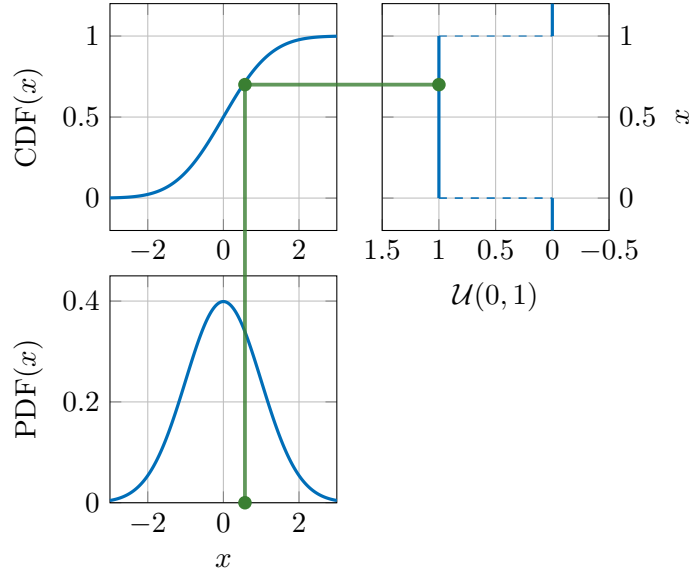
### 3.1. Sampling from Multivariate Gaussian Distribution



**Figure 3.1.:** Different cluster types, with parent cluster points as cluster centers in red, and cluster daughter points in blue.

### 3. Clustering Data for Efficient Performance Map Reconstruction

We can simply draw a random sample from a uniform distribution  $U(0,1)$ , project it onto the CDF<sup>1</sup> of the normal distribution, and when it meets the curve, project back onto the PDF<sup>2</sup> of the normal distribution. The cross-section with the x-axis is a sample from the given normal distribution.



**Figure 3.2.:** Sampling from normal distribution  $\mathcal{N}(0,1)$  by the projection of a sample from the uniform distribution  $\mathcal{U}(0,1)$  over the cumulative density function of  $\mathcal{N}(0,1)$ .

Furthermore, if we want to now to sample from a Gaussian distribution, that has arbitrary mean and variance,  $\mathcal{N}(\mu, \sigma^2)$ , all we need to do is sample again from a normal distribution, and then apply the transformation:

$$x_i \sim \mu + \sigma \mathcal{N}(0,1) \quad (3.2)$$

Thereby multiplying the sample from the normal distribution with the square root of variance, and shifting it according to the distribution mean.

In the multivariate case we can sample with the help of uniform distribution. Let us assume the random vector  $\mathbf{f} = (f(x_1), \dots, f(x_I))^T$  has a multivariate normal (or

<sup>1</sup>Cumulative Density Function

<sup>2</sup>Probability Density Function

### 3.1. Sampling from Multivariate Gaussian Distribution

Gaussian) distribution. That implies that every linear combination

$$\sum_{k=0}^I a_k f(x_k), a_i \in \mathbb{R}$$

is normally distributed with probability density function 2.4. From this  $I$ -dimensional Gaussian distribution  $\mathcal{N}(\mathbf{m}, \mathbf{K})$ , we wish to sample.

Sampling steps in the multivariate case:

1. Given set of locations  $I$  at which we want to sample, we compute the covariance matrix as 2.7, with free choice of kernel parameters  $\sigma_f$  and  $l$
2. Perform Cholesky decomposition of covariance matrix  $\mathbf{K} = \mathbf{L}\mathbf{L}^T$ , with lower triangular matrix  $\mathbf{L}$
3. From normal distribution  $\mathcal{N}(0, 1)$  generate a vector  $\mathbf{u} = (u_1, \dots, u_I)^T$  of independent standard normal random variables.
4. Compute  $\mathbf{f} = \mathbf{m} + \mathbf{L}\mathbf{u}$

By verifying the expectation value of  $\mathbf{f}$

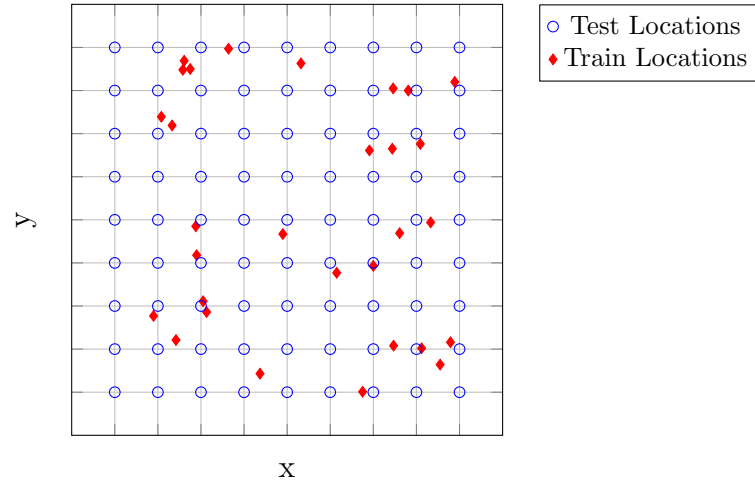
$$\mathbb{E}\{\mathbf{f}\} = \mathbb{E}\{\mathbf{m} + \mathbf{L}\mathbf{u}\} = \mathbb{E}\{\mathbf{m}\} + \mathbb{E}\{\mathbf{L}\mathbf{u}\} = \mathbf{m}$$

and covariance as mean of squares minus the square of means

$$\begin{aligned} \mathbb{E}\{(\mathbf{f} - \mathbf{m})(\mathbf{f} - \mathbf{m})^T\} &= \mathbb{E}\{\mathbf{f}\mathbf{f}^T\} - \mathbb{E}\{\mathbf{f}\}^T = \mathbb{E}\{\mathbf{m} + \mathbf{L}\mathbf{u}\}(\mathbf{m}^T + \mathbf{u}^T \mathbf{L}^T) - \|\mathbf{m}\|^2 \\ &= \mathbb{E}\{\mathbf{m}\mathbf{m}^T\} + \mathbb{E}\{\mathbf{m}\mathbf{u}^T \mathbf{L}^T\} + \mathbb{E}\{\mathbf{L}\mathbf{u}\mathbf{m}^T\} + \mathbb{E}\{\mathbf{L}\mathbf{u}\mathbf{u}^T \mathbf{L}^T\} - \|\mathbf{m}\|^2 \\ &= \|\mathbf{m}\|^2 + \mathbf{L}\mathbf{L}^T - \|\mathbf{m}\|^2 = \mathbf{K} \end{aligned}$$

the multivariate normal distribution of  $\mathbf{f}$  is proven. One of the problems when generating the simulated map on a regular grid is that random training locations rarely coincide with this grid, and therefore interpolation between the original points is necessary to get the values at exact training locations (see Fig 3.3). Using a method of sampling from multivariate Gaussian distribution makes it possible to get the exact values at both train and test locations of the simulated map simultaneously, so that interpolation error can be avoided. In the following subsections we show how different types of clusters can be generated and their values sampled using this method.

### 3. Clustering Data for Efficient Performance Map Reconstruction



**Figure 3.3.:** Test locations (blue circles) coincide with a regular grid, train locations (red diamonds) arbitrarily scattered around the grid.

## 3.2. Evenly Spaced Cluster Centers with Evenly Spaced Cluster Points (S1)

In an ideal world, we would have equally spaced clusters, with cluster points spread out throughout the whole region, within we want to predict the network performance. However, since we do not live in an ideal world, we can simulate regularly spaced clusters and use them as a benchmark for the comparison with our "reality" - non-uniform clusters.

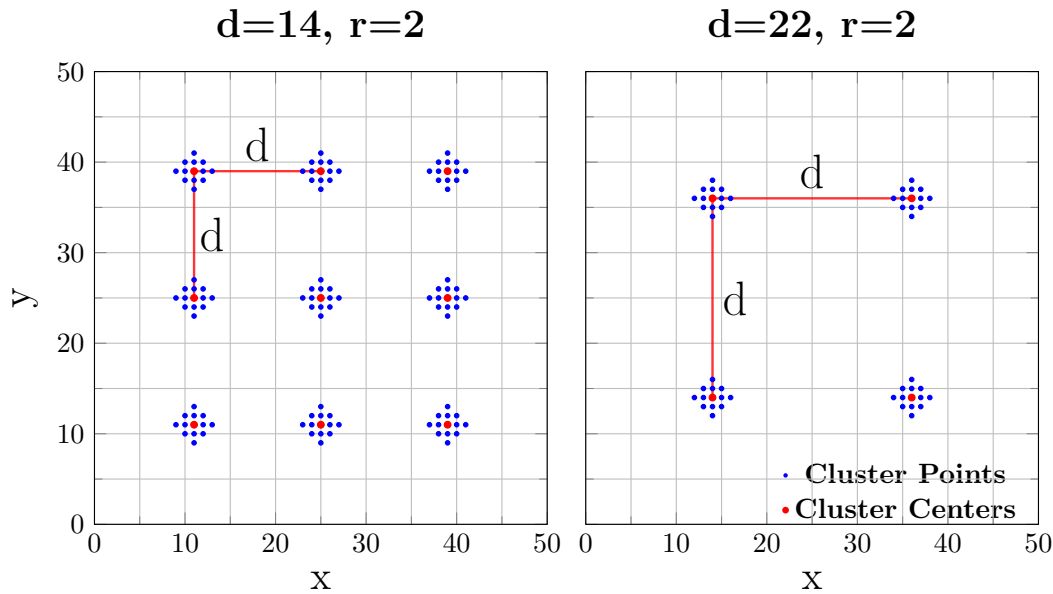
Let us assume we want to create accurate enough prediction of the network performance at each location of the equally spaced regular grid of test points, as shown in Figure 3.3 (blue circles). To do so, we assume our measurement locations form evenly spaced clusters with evenly spaced cluster points.

When generating this training dataset, there are two parameters to be considered:

- Distance between cluster centers denoted by  $d$  and
- Cluster radius denoted by  $r$

### 3.2. Evenly Spaced Cluster Centers with Evenly Spaced Cluster Points (S1)

Since there is no need to specify a unit of length upfront, we define a general length unit and call it a pixel. Each pixel can represent an arbitrary length segment of the map dimension, or in other words, 1 pixel can be equal 1m, 5m, 10m, and so on.

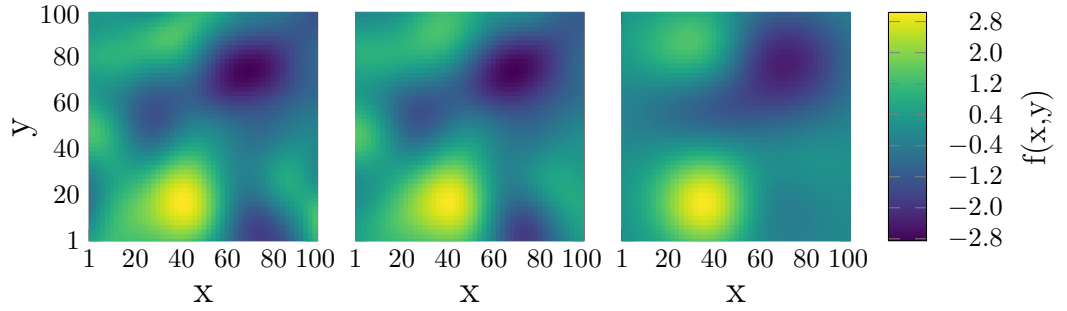


**Figure 3.4.:** With higher  $d$ , the number of clusters inside the area of interest decreases.

For cluster demonstration purposes, we use  $50 \times 50$  pixel grid (Figure 3.4), while for actual predictions further on a  $100 \times 100$  pixel grid is used with a test point at each pixel (Figure 3.5). We chose a  $100 \times 100$  map dimension, as it provides sufficient space to generate the number of clusters sufficient for performance assessment while keeping the out of memory problem at bay.

Figure 3.4 depicts how the cluster appearance in the area of interest changes with different distance between cluster centers. Assuming underlying hyper-parameters needed for the GPR prediction are known, and attaining the representative cluster centers by averaging across cluster values in each cluster, we can see the influence parameter  $d$  has on GPR prediction quality in Figure 3.5.

### 3. Clustering Data for Efficient Performance Map Reconstruction



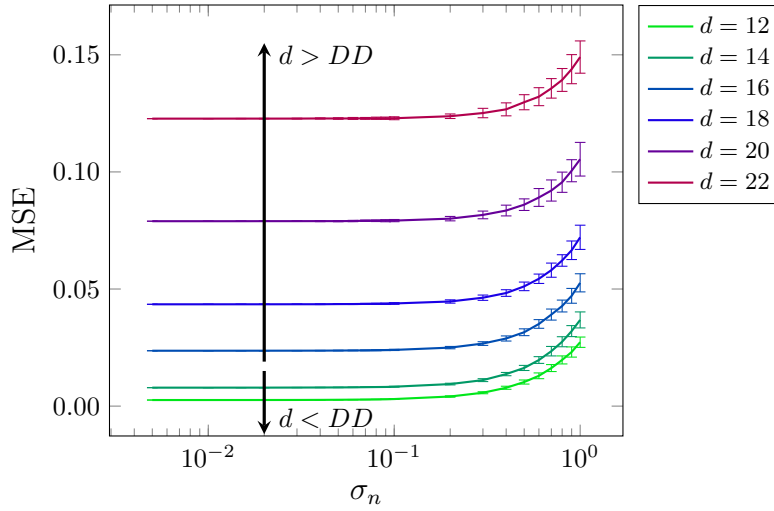
**Figure 3.5.:** True performance map(left), predicted performance map with  $d = 14$  and  $r = 3$  (middle), predicted performance map with  $d = 22$  and  $r = 3$  (right)

Distance between cluster centers (distance between GPR training locations)) determines how well we can capture the underlying structure of the performance map defined by the decorrelation distance. The original map depicted in the left plot of Figure 3.5, can be sufficiently well reconstructed if the neighboring GPR training points are at a maximum distance below DD of the underlying map. In the modeled scenario of  $100 \times 100$  pixel map we assumed DD of 15 pixels (pixels represent units of length, for example 1 pixel = 1m, or 1 pixel = 1cm). This implies that distance between neighboring training point locations must be below 15 pixels for optimal reconstruction. Therefore, when comparing two predicted maps from Figure 3.5, the degradation in prediction performance is obvious.

This trend is also observed in MSE comparison (Figure 3.6). Using different cluster distances shows the significant jump in performance degradation as soon as the cluster distance  $d$  exceeds the DD or characteristic length scale of the underlying map. Therefore, to be able to reproduce the original underlying map sufficiently well, we need clusters at distances smaller than the characteristic length scale. Important to note here is that the number of clusters in the area of interest reduces with higher cluster distance  $d$ . At a constant cluster radius  $r$ , a different number of clusters (different  $d$ ) corresponds to distinct point densities. Therefore to compare the performance between diverse scenarios, we must keep the point density constant<sup>3</sup>.

<sup>3</sup>or approximately constant, as in S3, we work with a random process

### 3.2. Evenly Spaced Cluster Centers with Evenly Spaced Cluster Points (S1)

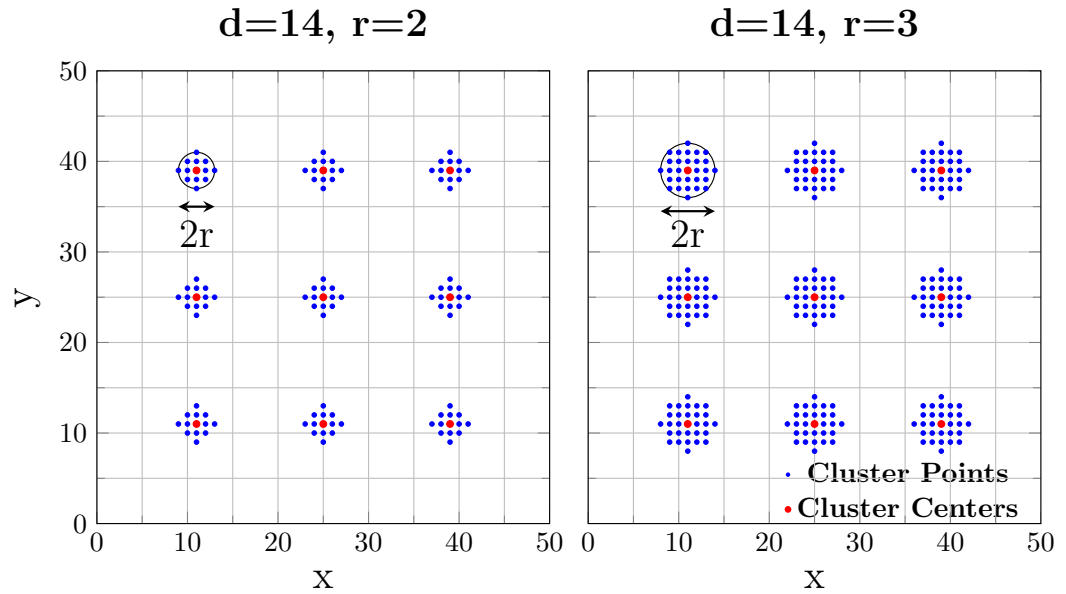


**Figure 3.6.:** MSE with errorbars, derived from 100 different performance map realizations. The map dimension is  $100 \times 100$  pixels,  $DD = 15$ ,  $\sigma_f = 1$ , cluster radius  $r = 3$ .

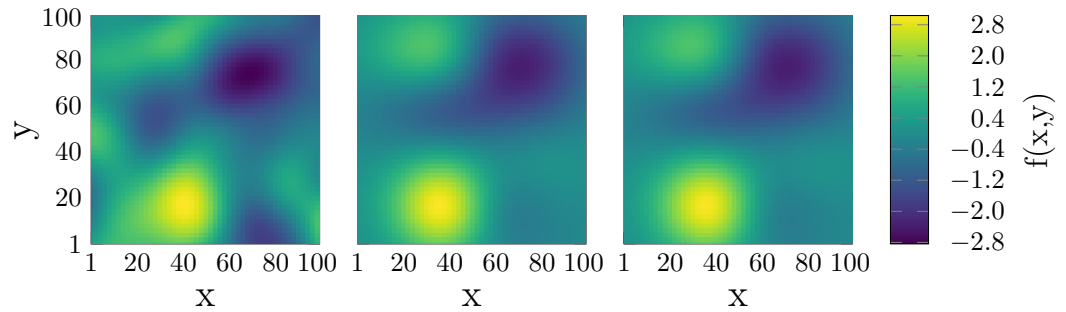
Cluster radius  $r$  determines the number of data points per cluster. For instance  $r = 2$  generates  $N = 13 \frac{1}{\text{cluster}}$ , while  $r = 3$  produces  $N = 29 \frac{1}{\text{cluster}}$  (see Figure 3.7). Since we do not feed all of those as training points to the GPR, but single representative averaged data point per cluster, in both cases, we will have the same number of training points that are fed as the GPR input.

In the case of very low measurement noise, both predictions will perform equally well. In contrast, if measurement noise variance is high, the larger number of points per cluster makes it easier to average the present noise out so that the prediction will be better when the noise level is unknown (see Figure 3.9). However, for low measurement noise, an error floor is hit. This fact indicates that adding additional measurements to existing clusters produces no additional gain. Therefore even if the cluster size increases, the distance between averaged cluster centers stays the same, and the function variation between those centers remains equally unexplored (see Figure 3.8).

### 3. Clustering Data for Efficient Performance Map Reconstruction



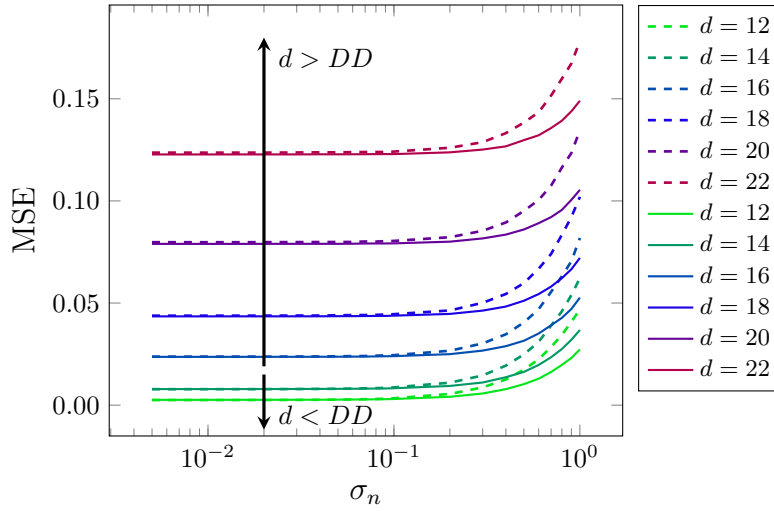
**Figure 3.7.:** Cluster radius  $r$  determines the number of points inside each cluster.



**Figure 3.8.:** True performance map (left), predicted performance map with  $d=22$  and  $r=2$  (middle), predicted performance map with  $d=22$  and  $r=3$  (right)



### 3.2. Evenly Spaced Cluster Centers with Evenly Spaced Cluster Points (S1)



**Figure 3.9.:** MSE with errorbars, derived from 100 different performance map realizations, map dimension is  $100 \times 100$  pixels,  $DD = 15$ ,  $\sigma_f = 1$ , cluster radius  $r = 2$  (dashed),  $r = 3$  (solid).

When averaging over the cluster values, the standard deviation of the cluster representative point is going to drop accordingly,  $\sigma'_n = \frac{\sigma_n}{N}$ , where  $N$  is the number of data points inside each cluster. This formula is characteristic of repeated measurements and can be looked up in [17]. We use it as an approximation since these are not measurements repeated at the same location, but rather in close vicinity one from another. Therefore, the inputs to the GPR prediction are averaged cluster centers as training dataset and the known hyper-parameter tuple  $[\sigma_f, l, \sigma'_n]$ .

We have seen that averaging over a higher number of cluster points brings advantage when higher noise is present even though in both cases, the same amount of training points is used. This raises the question of how this method compares with merely using the cluster center point while discarding all other cluster points. Another possibility is to perform GPR locally at each cluster to find the GPR predicted cluster center point and then use these predicted central points as the input for the GPR prediction of the entire area of interest. While we reduce the GPR computational time by finding a representative training point per cluster, we do have losses in prediction accuracy compared to using all cluster points. Therefore, we compared the performance of previous methods with a GPR that uses all cluster points as training

### 3. Clustering Data for Efficient Performance Map Reconstruction

---

dataset. And lastly, the best performance of GPR is expected when the training points are randomly scattered across the area of interest.

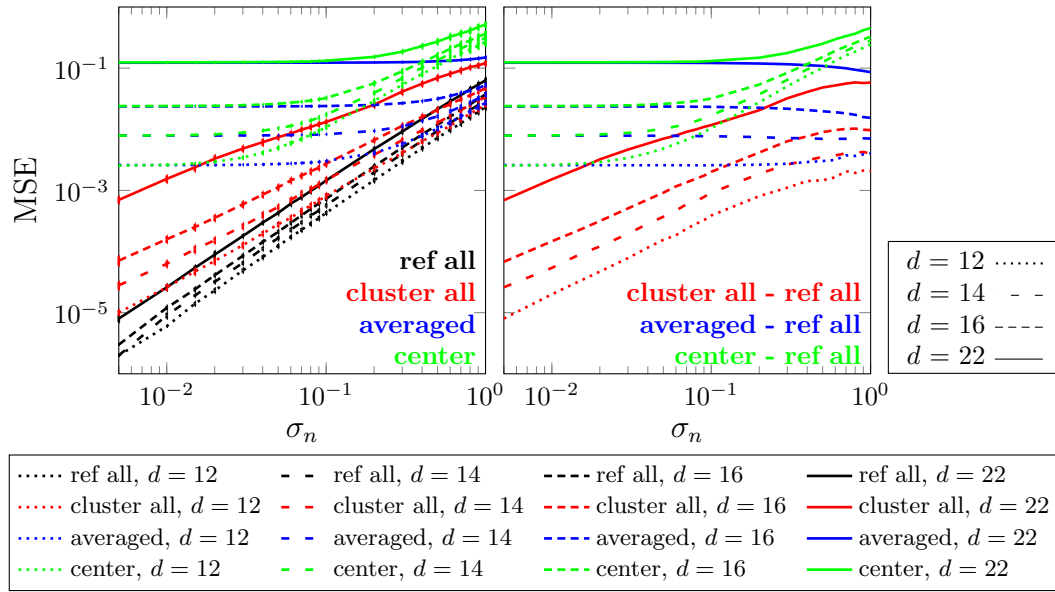
These methods were compared using  $100 \times 100$  pixel grid as a testing dataset. The MSE is calculated using Equation 3.1 and plotted over different measurement noise levels (Figure 3.10). Values in plots are averaged over 100 different performance map realizations with equal hyper-parameters, and the errorbars represent the standard deviation of the MSE value.

We distinguish the following cases:

- ref all** Randomly scattered points across an area of interest, while keeping the same density of training points as is available in the clustered version. We use this setup as a reference for all others.
- cluster all** All measurement points in clusters are used as the training dataset.
- averaged** Averaging over location and value in each cluster, and using averaged cluster centers as a training dataset.
- block-wise** Local GPR inside each cluster to predict the cluster center point, subsequently using these predicted centers as training dataset for the GPR prediction of the entire area of interest.
- center** Only cluster center point is used, while the rest of the cluster points is discarded.

As expected, we notice in Figure 3.10 that two cases where all available data is used for training outperform those where the training set is reduced by a factor of  $N$  (whether by averaging, using local GPR or dumping). Averaging over clusters method performs equally well as the block-wise GPR, with less computational time. As for using the center cluster point while dumping the rest, we see the prediction deterioration with higher measurement noise. A likely scenario considering that the measurement noise of that single point grows accordingly and is not reduced by averaging.

### 3.3. Evenly Spaced Cluster Centers with Randomly Spaced Cluster Points (S2)



**Figure 3.10.:** MSE with errorbars, derived from 100 different performance map realizations, map dimension is  $100 \times 100$  pixels,  $DD = 15$ ,  $\sigma_f = 1$ , cluster radius  $r = 3$ . Note that the averaged and block-wise case overlap in the figure.

### 3.3. Evenly Spaced Cluster Centers with Randomly Spaced Cluster Points (S2)

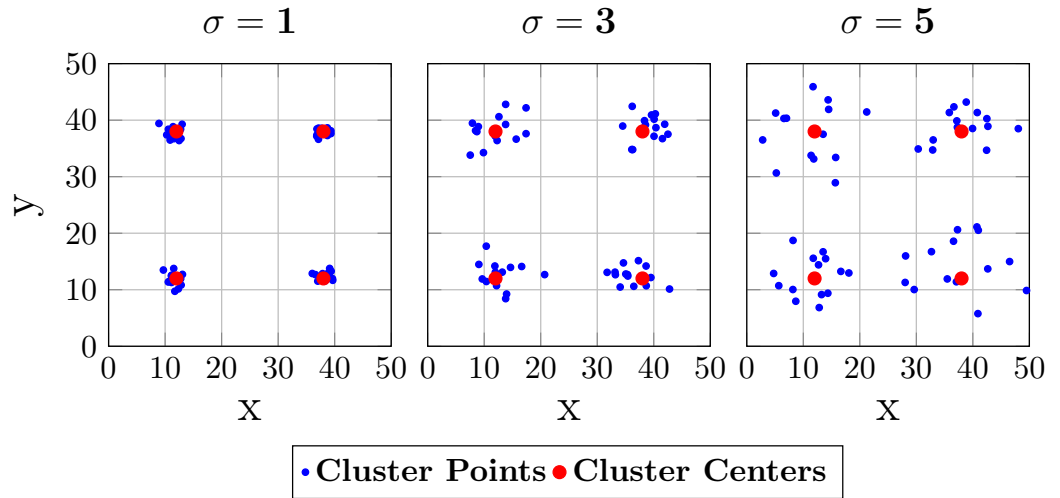
Imagine we want to predict a network parameter in a modern city, with an orthogonal grid of streets, such as Barcelona or New York. In such a scenario, our assumption of evenly spaced cluster centers corresponds to the street crossings. However, the users on those crossings are not standing in organized formations, but somewhat randomly scattered across the sidewalk and street crossing. To account for this randomness, we need to relax our assumption from S1 and assume users scattered around the cluster center or, in this case, the crossing center.

In the previous section, we have seen how the clusters with training data that coincide with the testing grid behave. Now we discuss clusters that have parent points (cluster

### 3. Clustering Data for Efficient Performance Map Reconstruction

centers) that match with the testing grid, and cluster daughter points randomly scattered around parent points (Figure 3.11). Since they are generated using a Gaussian distribution in x- and y-direction, we now have the number of points per cluster as a parameter that is independent of the cluster radius. Therefore instead of two, we now consider three different cluster parameters:

- Distance between cluster centers denoted by  $d$
- Standard deviation  $\sigma$  of Gaussian distribution (with a mean at the cluster center) that determines the cluster spread or cluster diameter.
- The number of points per cluster  $N$ . This parameter is now independent of the cluster radius, as the training cluster points do not coincide with the testing grid, comparing to Section 3.2.



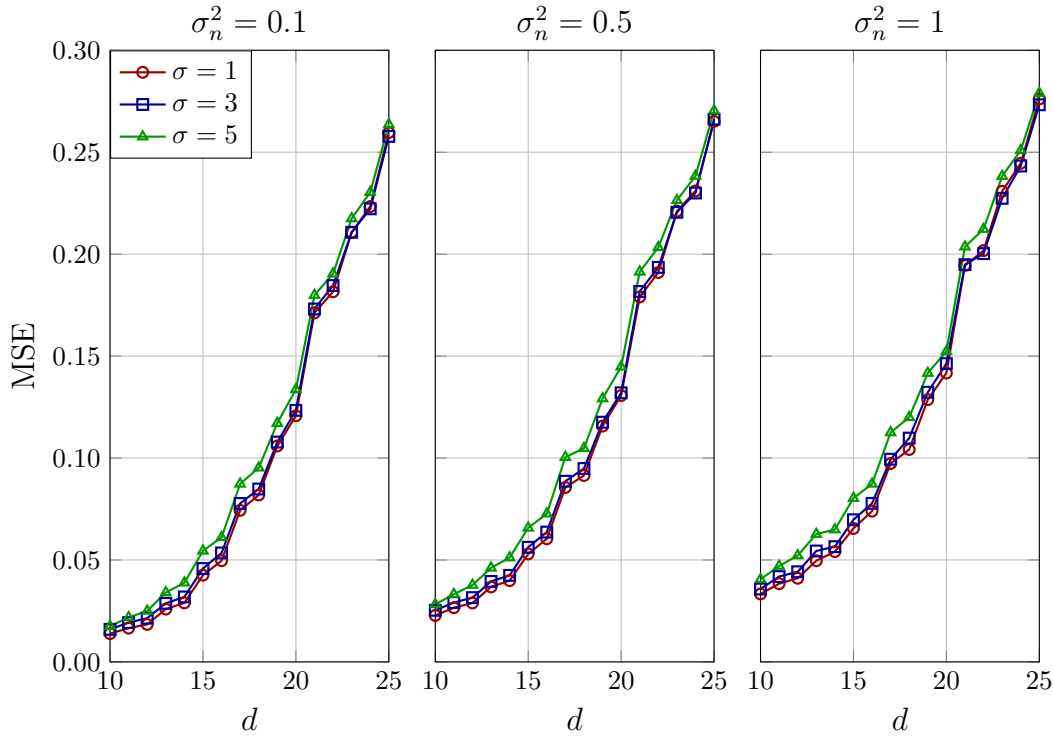
**Figure 3.11.:** Influence of Gaussian distribution standard deviation  $\sigma$  on the cluster spread. Fixed parameter values are:

- Number of points per cluster  $N = 15$
- Distance between cluster centers  $d = 22$

For the following analysis, we used a  $200 \times 200$  pixel map, with  $DD=20$ . Figure 3.12 shows the influence of  $d$  (distance between cluster centers) on the MSE under different  $\sigma$ . The number of points per cluster for this case was fixed,  $N = 30$ , but even for smaller or larger  $N$ , we observed the same trend. There are only slight variations in MSE when using different  $\sigma$ . On the other hand, increasing distance between

### 3.3. Evenly Spaced Cluster Centers with Randomly Spaced Cluster Points (S2)

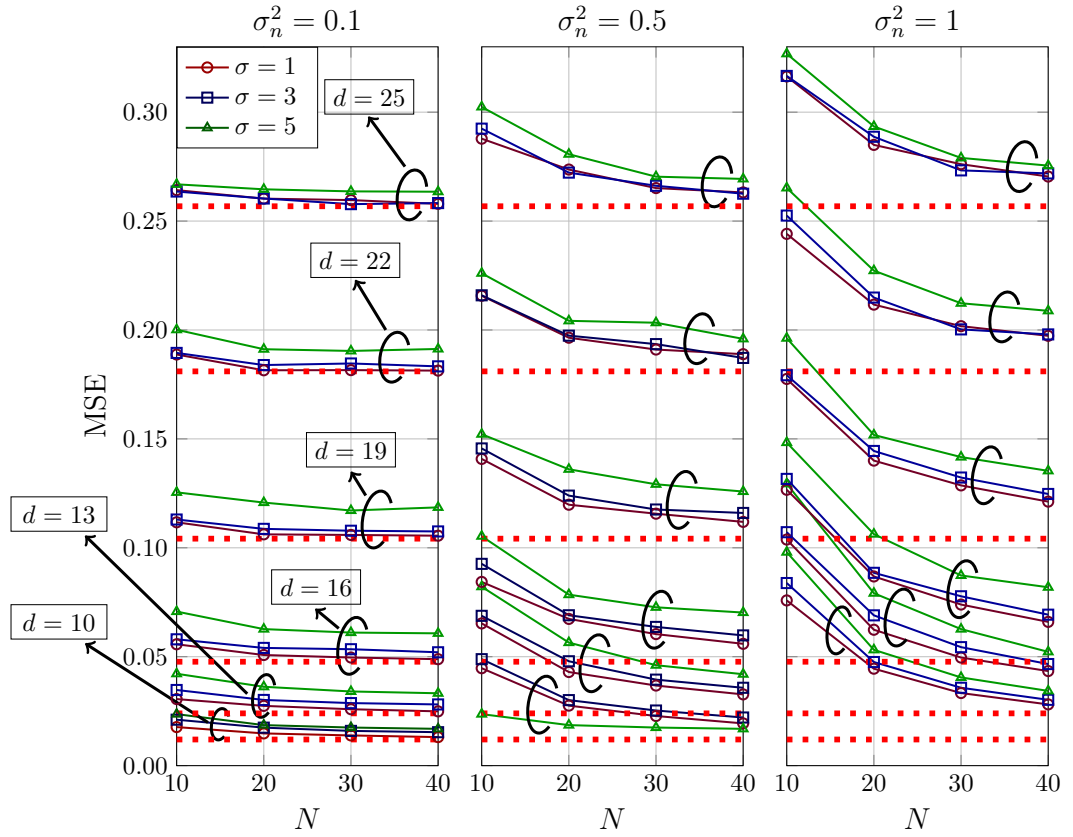
cluster centers ( $d$ ) has the strongest influence on MSE deterioration, independent of how large the measurement noise variance ( $\sigma_n^2$ ) is. We notice a significant jump after  $d = 20$ , where the distance between training points exceeds the decorrelation distance. In these cases, the GPR framework cannot predict the function variations between the points at a distance larger than the characteristic length scale.



**Figure 3.12.:** MSE averaged over 100 different performance map realizations. Fixed number of cluster points is  $N = 30$ .

Without measurement noise, the MSE almost does not change with increasing  $N$ . A similar trend is observed for the low measurement noise level (Figure 3.13 left). When comparing three plots in Figure 3.13, we notice that the higher the  $N$  is, the the MSE is smaller, or in other words, the higher the noise level is, the larger  $N$  is required for better performance improvement. Additionally, with decreasing  $d$ , we obtain more clusters in the same area of interest, therefore more training points altogether, which again improves the overall MSE.

### 3. Clustering Data for Efficient Performance Map Reconstruction



**Figure 3.13.:** MSE means over 100 different realizations, for different combinations of  $N$ ,  $d$ ,  $\sigma_n^2$  and  $\sigma$ .

Table 3.1 shows the MSE values for  $N = 100$  for various combinations of parameters  $\sigma_n$ ,  $\sigma$  and  $d$ . Bold values representing the minimum MSEs are then used as asymptotic approximations in Figure 3.13 (dashed lines in red represent the asymptotes for  $d \in \{10, 13, 16, 19, 22, 25\}$ , and have been calculated for  $N = 100$  and  $\sigma = 1$ ). As for the remaining  $\sigma$  (3 and 5) values, MSEs are slightly higher at  $N = 100$ , but will approach the same asymptote for higher  $N$ .

### 3.3. Evenly Spaced Cluster Centers with Randomly Spaced Cluster Points (S2)

	$\sigma_n^2 = 0.1$			$\sigma_n^2 = 0.5$			$\sigma_n^2 = 1$		
	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$
$d = 10$ :	<b>0.0122</b>	0.0143	0.0127	0.0124	0.0152	0.0150	0.0172	0.0182	0.0259
$d = 13$ :	<b>0.0243</b>	0.0244	0.0277	0.0336	0.0292	0.0325	0.0255	0.0267	0.0408
$d = 16$ :	<b>0.0477</b>	0.0506	0.0593	0.0503	0.0537	0.0625	0.0538	0.0575	0.0667
$d = 19$ :	<b>0.1042</b>	0.1070	0.1153	0.1070	0.1093	0.1192	0.1115	0.1134	0.1225
$d = 22$ :	<b>0.1810</b>	0.1811	0.1886	0.1820	0.1840	0.1913	0.1863	0.1859	0.1937
$d = 25$ :	<b>0.2568</b>	0.2559	0.2608	0.2598	0.2588	0.2630	0.2626	0.2605	0.2648

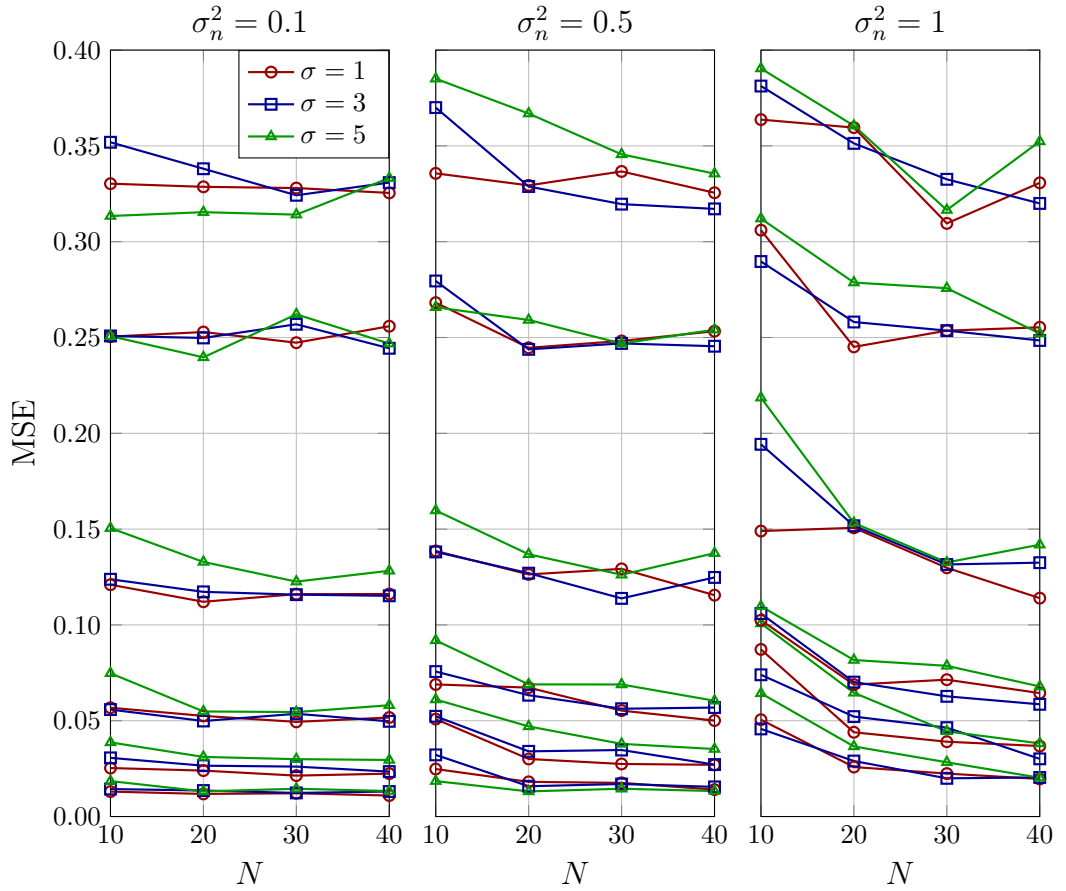
**Table 3.1.:** MSE for  $N = 100$ , used as asymptotic approximation. Bolded values are represented by red dashed lines in Figure 3.13

Since the values plotted in Figure 3.13 are the means over 100 realizations, it is also interesting to see how are the values spread around this mean. For this, we calculated the 5 and 95 percentile<sup>4</sup> values for all parameter combinations. Figure 3.14 shows the difference between 95 and 5 percentiles, or in other words, where 90% of the realizations lie.

We notice that for smaller cluster distances, the mean and 90 percentile do not differ much, as for larger  $d$  the difference between them increases. This is easily seen when comparing Figure 3.13 and Figure 3.14. Reason is the greater number of outliers for larger cluster distances.

<sup>4</sup>Percentile is the value below which a given percentage of observations in a group of observations falls

### 3. Clustering Data for Efficient Performance Map Reconstruction



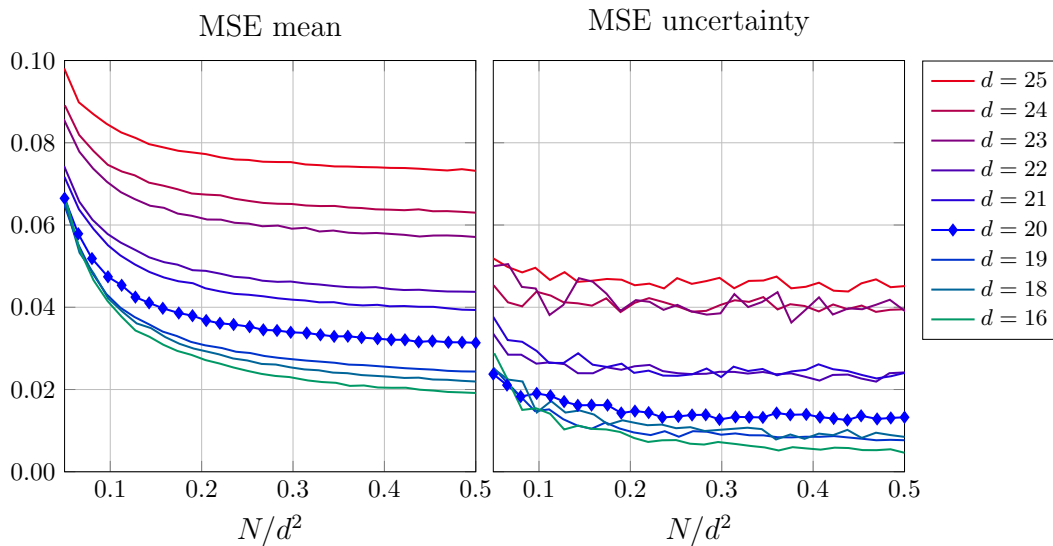
**Figure 3.14.:** MSE 95%-5% (percentile) over 100 different realizations, for different combinations of  $N$ ,  $d$ ,  $\sigma_n^2$  and  $\sigma$ .

Formerly we examined how the different parameters influence the MSE independently of each other. Now we wonder whether an optimal balance between the different number of points per cluster ( $N$ ) and distances of the clustered centers ( $d$ ) can be found. For this, we examine how the MSE varies across different point densities  $\frac{N}{d^2}$  for various distances between neighboring cluster centers. Assuming a specific point density in a designated area, we would like to know if rearranging of the measurement locations by switching to another  $d$ , would improve the prediction performance. For this purpose, a map of dimension  $200 \times 200$  pixels was created, with  $DD = 20$ . We considered three values of  $\sigma$  (1, 3 and 5), but the achieved



### 3.3. Evenly Spaced Cluster Centers with Randomly Spaced Cluster Points (S2)

MSEs do not differ as much between these three cases, so only one is shown in the figure, for easier readability. We used thirty values of linearly spaced density in the range  $[0.05, 0.5]$ , and then by multiplying with different  $d^2$ , we obtained the number of points per cluster  $N$ . Clustered points were placed randomly in clusters around the given grid ( $200 \times 200$ ), and then using interpolation, the values of RSRP at those points were acquired. These calculated points are further accepted as known cluster points. By adding random noise on top, with  $\sigma_n^2 = 1$ , the training points with measurement noise were produced. These cluster points are then averaged in value and location to get the averaged single training point per cluster. Lastly, these cluster centers represent the training dataset for the GPR. Figure 3.15 shows the MSEs and its 90-percentiles for  $200 \times 200$  pixel map, averaged over 250 realizations.



**Figure 3.15.:** Mean MSE (left) and 95%-5% percentile MSE (right) over 100 realizations,  $\sigma = 1$ ,  $\sigma_n^2 = 1$ ,  $DD = 20$ ,  $\sigma_f = 1$ .

What we can see from these plots is that there is no switch point to be found since the lines never intersect. The conclusion is rather that at a constant density, it is always better to choose more spread out measurements, suggesting smaller cluster distance  $d$ . Additionally, we also notice a jump in MSE after  $d = 20$ , since it gets harder to predict the map if we have cluster distances larger than map decorrelation distance (here  $DD = 20$ ) as previously mentioned.

## 3.4. Non-Uniform Clusters generated using Thomas Cluster Process (S3)

In previous chapters, we discussed scenarios that are not applicable in most circumstances in reality. They are idealized versions of user clusters, and we use them for the performance comparison with natural user clusters, which can be rather messy and random. Stochastic geometry models are employed to attain a tractable analysis of wireless network nodes and capture their spatial randomness.

To create authentic natural clusters for our study, we need to think of the regions where we gather the measurements since the user distribution is different in rural and urban areas. We can assume that in the rural areas, we encounter zones with barely any users, while in urban areas, users are nearly omnipresent in all regions. Additionally, in those areas with users, we can have various user densities. To account for the randomness of the accessible areas and randomness of the number of users in them, it makes sense to use two separate point processes. For this reason, a family of cluster point processes, known as Neyman-Scott point processes, is widely used in spatial statistics for telecommunications.

The parameters of the Neyman-Scott point process are:

- $\lambda_{\text{parent}}$  - The intensity of the Poisson point process, which forms the cluster centers.
- $\lambda_{\text{daughter}}$  - The mean number of points per cluster.
- $\sigma$  - The size of the clusters.
- $p(\cdot, \sigma)$  is a probability density function parameterized by  $\sigma$ , which determines the spread of daughter points around the cluster center.

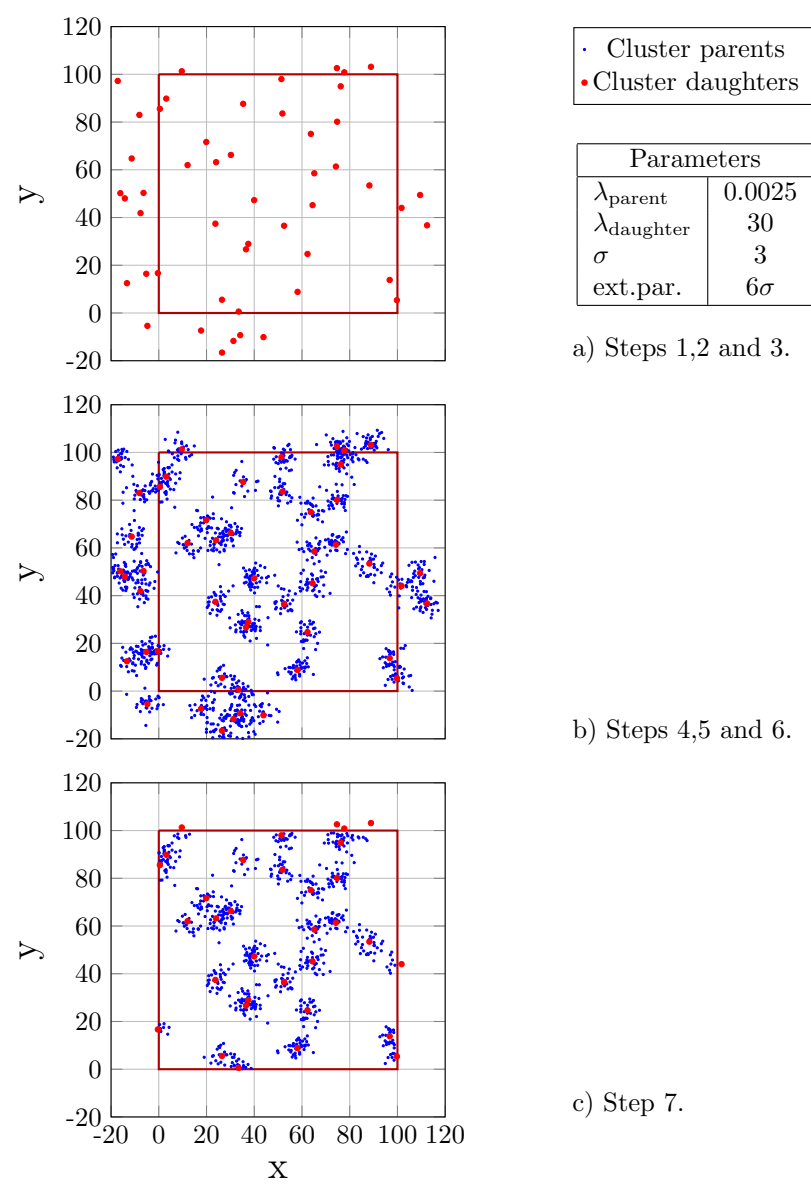
If  $p(\cdot, \sigma)$  is Gaussian distribution around the cluster center at the origin, then the process is called Thomas cluster point process [20].

### 3.4. Non-Uniform Clusters generated using Thomas Cluster Process (S3)

Construction of user clusters using Thomas cluster point process shown in Figure 3.16 can be described in the following steps:

1. Since we are simulating a part of "reality" in a chosen simulation window, it is expected that some of the daughter points in the simulation window originate from parents that lie outside of it. To overcome these edge effects, we simulate the parent points in an extended simulation window.
2. Poisson point process with intensity  $\lambda_{\text{parent}}$  is then used to generate number of parent points in the extended area ( $N_{\text{parent}}$ ).
3. Both x and y coordinates of those  $N_{\text{parent}}$  points are then drawn from a uniform distribution over the extended area.
4. Poisson point process with intensity  $\lambda_{\text{daughter}}$  is used to generate number points for each cluster or the number of points that will be distributed around each of the parent locations.
5. Daughter locations are then drawn from the normal distribution  $\mathcal{N}(0, \sigma)$ , with a standard deviation that determines the spread of the clusters.
6. Shift of daughter points to their corresponding parent location.
7. Simulation window is cut to its original size.

### 3. Clustering Data for Efficient Performance Map Reconstruction



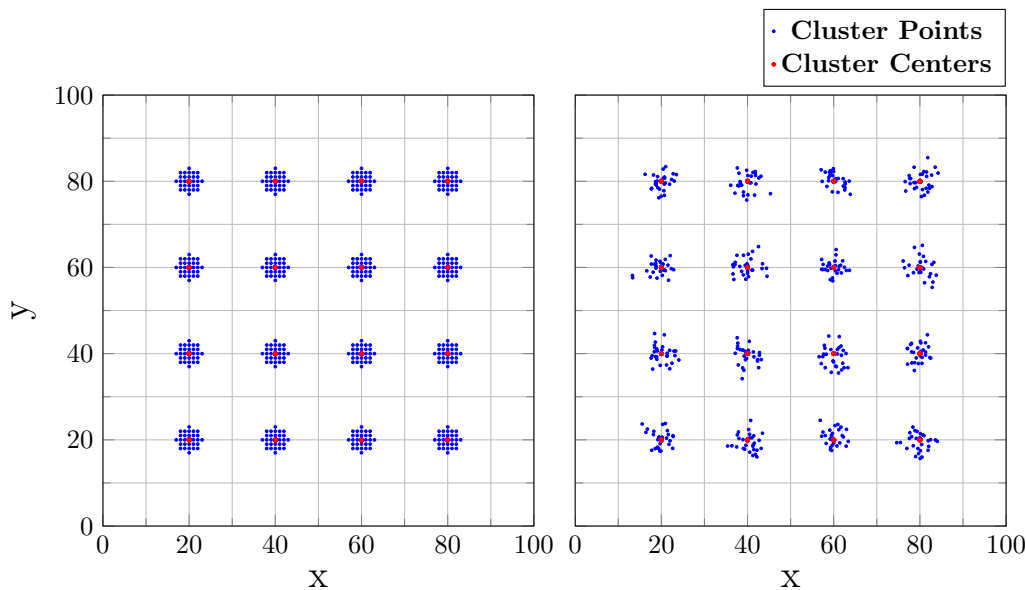
**Figure 3.16.:** Thomas Cluster Process for cluster generation in S3.

### 3.5. Comparison of S1, S2 and S3

We have discussed three different scenarios of user cluster generation in previous chapters. In order to adequately compare them, we keep the user density in the area of interest constant in all three situations. For demonstration simplicity, we first make a comparison between scenarios S1 and S2 and afterward, between S2 and S3.

#### 3.5.1. Evenly Spaced clusters with Evenly Spaced Cluster Points (S1) vs Randomly Spaced Cluster Points (S2)

To compare GPR prediction from evenly spaced cluster centers with evenly spaced cluster points in S1 and evenly spaced cluster centers with randomly spaced cluster points in S2 we used the setup shown in Figure 3.17.

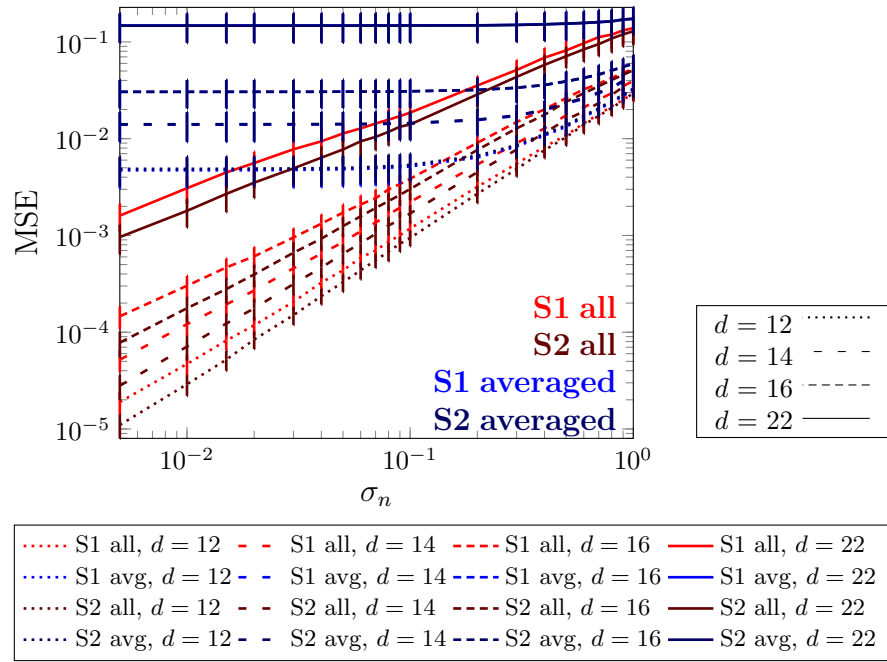


**Figure 3.17.:** Training points in clusters in S1 (left) and S2 (right), distance between clusters  $d = 20$ ,  $r = 3$  and  $\sigma = 2$  is chosen for demonstration purposes.

Using clusters at varying distances  $d$ , we predicted the map of  $100 \times 100$  pixels. The

### 3. Clustering Data for Efficient Performance Map Reconstruction

number of points per cluster, as well as the total number of clusters in both scenarios, are equal, which makes two cases comparable by having the same number of training points. To keep the cluster radii similar, we choose  $r = 3$  for S1 and  $\sigma = 2$  for S2. The MSE of the GPR predictions with varying measurement noise levels is shown in Figure 3.18. We see a slight difference between S1 and S2 when using all available cluster points (red lines). The reason is that random cluster points in S2 manage to gather more information about the areas without measurements, compared to more compact clusters in S1. On the other hand, when averaging over clusters in both scenarios (blue lines), the performance difference between them disappears. Since the number of cluster points is even, averaging over clusters reduces the noise level in both cases equally, while keeping the number of training points identical.



**Figure 3.18.:** Comparison of S1 and S2: Prediction MSE of  $100 \times 100$  pixel map with  $DD = 15$  and  $\sigma_f = 1$ , over various noise levels and different distances between cluster centers.

### 3.5.2. Evenly Spaced Clusters (S2) vs Non-Uniform Clusters (S3)

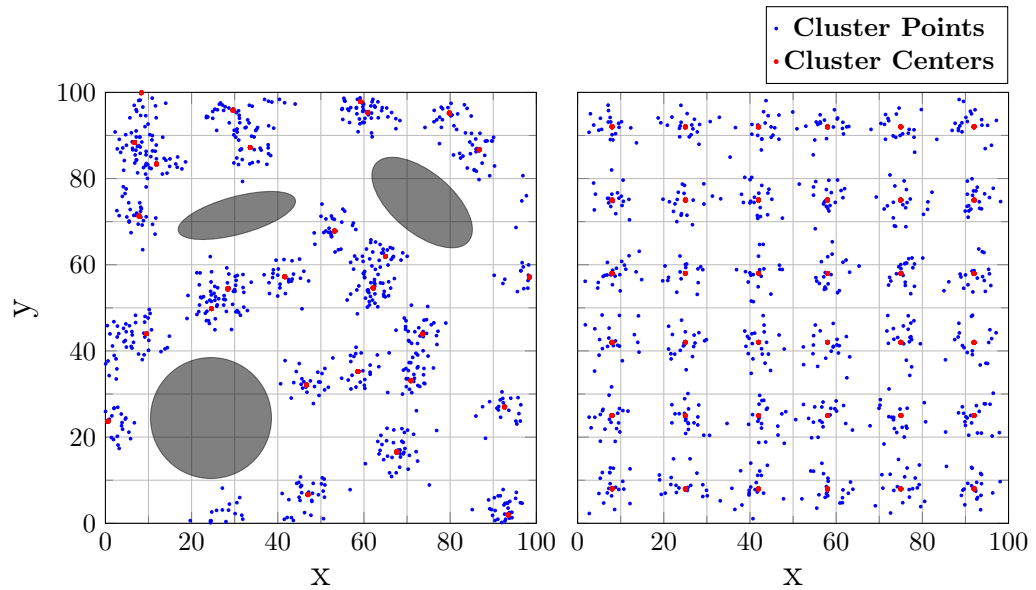
To compare GPR prediction from evenly spaced cluster centers with randomly spaced cluster points in S2 with non-uniform clusters in S3, we used the setup shown in Figure 3.19. Since we are working with random point processes for cluster generation, we first generate the non-uniform clusters in S3 and, consequently, the evenly spaced clusters in S2. In order to keep the point density constant<sup>5</sup>, certain adapting of clusters in S3 is also required after their generation. The procedure of calculating the cluster parameters in S2 based on clusters from S3 is summarized in the following steps:

1. **Family tree generation:** Generate non-uniform clusters using Thomas Cluster Process.
2. **Counting parents:** Find the number of parent points  $N_{S3}$  that have daughters inside the area of interest<sup>6</sup>.
3. **Parents adjustment:** Calculate the number of clusters (parent points) in S2  $N_{S2} = \text{int}(\sqrt{N_{S3}})$ .
4. **Parents pruning:** Randomly remove  $m = N_{S3} - N_{S2}$  clusters from S3, so that  $N_{S2} = N_{S3}$ .
5. **Parents spreading:** Based on the number of clusters  $N_{S2}$  find the distance between cluster centers  $d$  in S2 such that the clusters are evenly spread out across the area of interest.
6. **Offspring custody:** To keep the densities approximately equal, determine the number of points per cluster in S2 by dividing the total number of points in S3 with the number of clusters in S2.
7. **Offspring distribution:** Use the same  $\sigma$  to create the point distribution inside clusters in S2 as you used in S3.

<sup>5</sup>or approximately the same in S1 in S2

<sup>6</sup>It is likely to have daughters in the area of interest that originate from parents outside of it.

### 3. Clustering Data for Efficient Performance Map Reconstruction



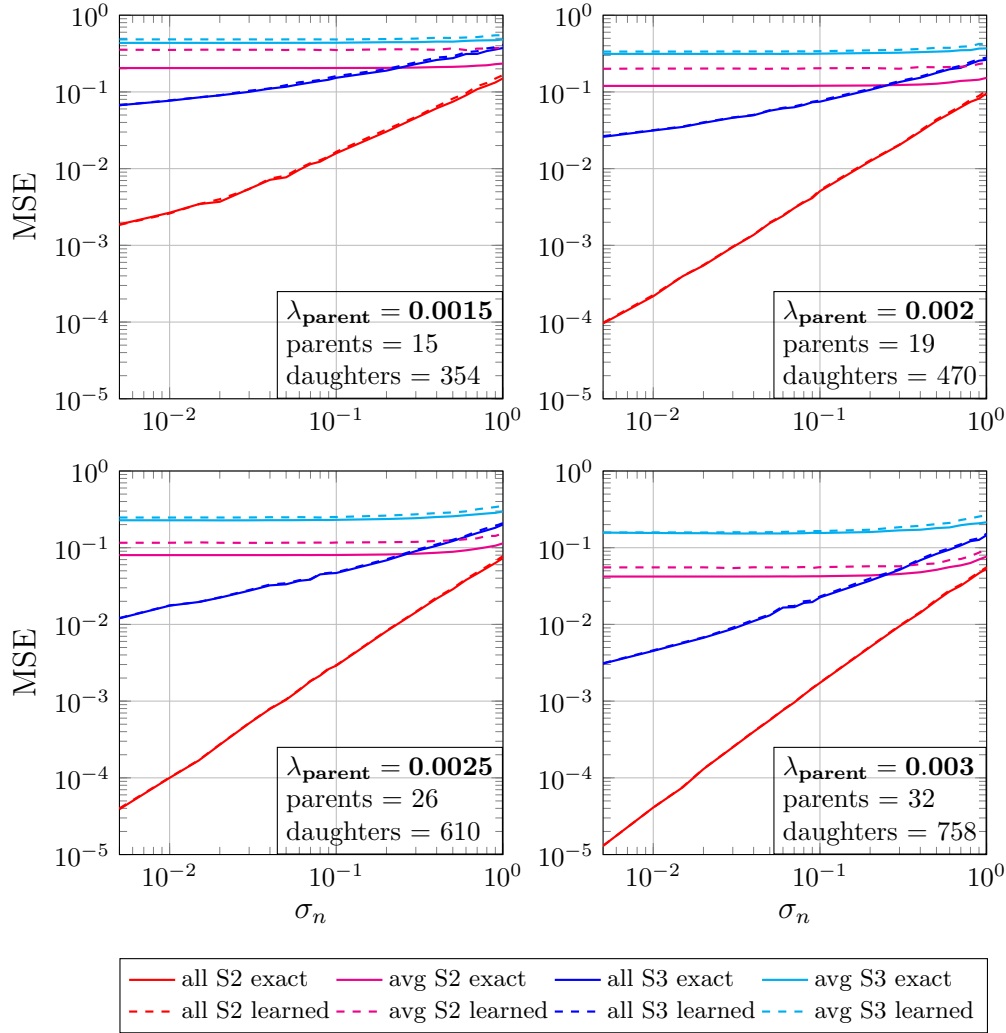
**Figure 3.19.:** Training points in clusters in S3 (left) and S2 (right), with equal point densities over the area of interest. Gray areas represent the black holes.

Prediction results from such cluster generation in S3 and S2 are shown in Figure 3.20. The MSE is averaged over 100 various map realizations. Since with each non-uniform cluster setup, the MSE is quite different, the errorbars are relatively high and therefore left out from the plots. We show the results for four different parent point counts, with an average of 30 daughters per each cluster. Solid lines represent predictions from exactly known hyper-parameters, while dashed lines are the predictions from hyper-parameters learned using the same training set as for the prediction.

The MSE drops down with higher parent point count, e.g., cluster count, as expected. We notice the cluster averaging loss is smaller in S3 than in S2. Due to the cluster randomness in S3, some clusters may be generated close together or even overlapping, leaving the areas of the map empty of training points. We call these empty areas *the black holes* (see Figure 3.19). In the case of evenly spread out clusters as in S2, we keep the black holes at a minimum, and therefore achieve better prediction performance.



### 3.5. Comparison of S1, S2 and S3



**Figure 3.20.:** Comparison of S2 and S3: Prediction MSE of  $100 \times 100$  pixel map with  $DD = 15$  and  $\sigma_f = 1$ , over various noise levels, averaged over 100 realizations. For each plot different cluster count was used.

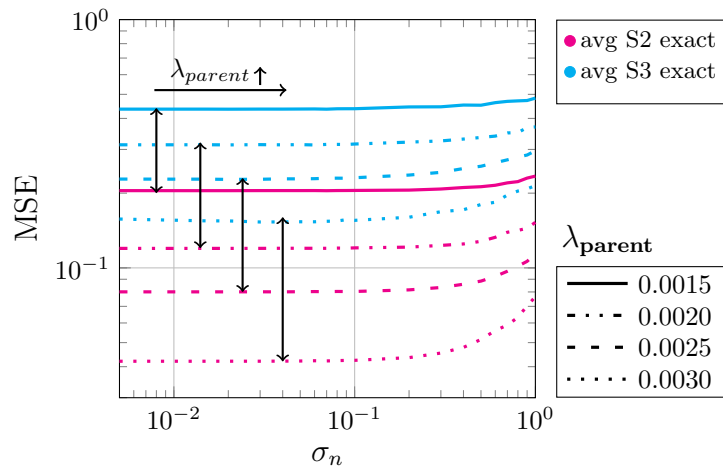
Scenario:	all S2	all S3	avg S2	avg S3
GPR train point count:	758	758	32	32

**Table 3.2.:** Training points for different scenarios with  $\lambda_{\text{parent}} = 0.003$ .

### 3. Clustering Data for Efficient Performance Map Reconstruction

Table 3.2, provides the number of training points used as GPR input in each of the cases, for  $\lambda_{\text{parent}} = 0.003$ . The reduction of computational complexity is straightforward, as the complexity scales cubically  $O(n^3)$  with the number of GPR training points  $n$ .

Zooming in on the averaged scenarios in Figure 3.20 we notice a constant offset between two averaged cases (black line arrows in Figure 3.21). that increases with higher parent point count. An increase in offset can be explained by a faster improvement in S2 compared to S3, as the new clusters we introduce are equally spaced. The introduction of new random clusters in S3 may or may not bring improvement dependent on the placement of those additional clusters.

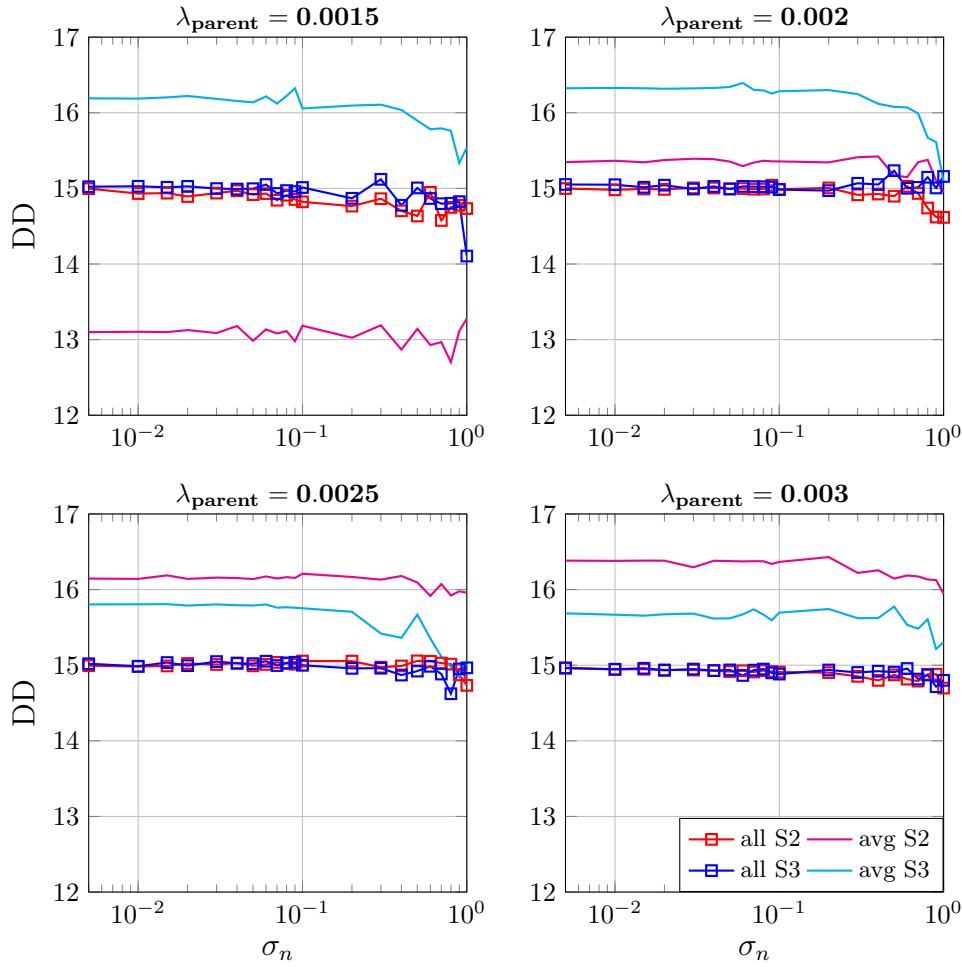


**Figure 3.21.:** Comparison of averaged S2 and averaged S3: Prediction MSE of  $100 \times 100$  pixel map with  $DD = 15$  and  $\sigma_f = 1$ , over various noise levels, averaged over 100 realizations, for four various number of parent points. Black lines represent the gap between S2 and S3 for equal cluster counts.

The offset between averaged S2 and S3 cases cannot be compensated by introducing new points inside existing clusters since the number of training points in GPR will stay the same. A possibility to reduce that difference is to introduce new clusters in black hole areas with the highest GPR prediction uncertainty. This way, we can add new information into the GPR instead of accumulating additional cluster points at areas about which we already know enough.

### 3.5. Comparison of S1, S2 and S3

Notice the offset between prediction using the exact versus learned hyper-parameters (solid vs dashed lines) in Figure 3.20. A closer inspection into parameter learning under four previously mentioned cases with various parent count, reveals why this offset is the highest in the case of using averaged cluster centers in S2.



**Figure 3.22.:** Learning of hyper-parameter length-scale ( $DD = 15$ ), using various parent counts.

Parent count has the most substantial influence on the learning of the length-scale (DD) hyper-parameter. Figure 3.22 shows that learning it is problematic when the

### 3. Clustering Data for Efficient Performance Map Reconstruction

---

parent count is too low, and parents are equidistantly spaced. As parents are at a distance greater than the DD of the map, they can no longer accurately capture and infer the underlying structure of the performance map. On the contrary, equally low parent count achieves better performance when parents are randomly placed, e.g., S3. As here some parents might be in close vicinity from each other, the length-scale can be estimated with higher accuracy on average.

We have seen the performance comparison of S1, S2, and S3. Notice that the switching from S1 to S2 brings no MSE degradation when the averaging in both cases is applied. On the other hand, under no measurement noise present utilizing all available points, can boost the performance by two to four orders of magnitude (depending on the number of clusters). (see Figure 3.18). A shift from averaged S2 to averaged S3 introduces a half order of magnitude MSE degradation, independent of the level of measurement noise. While the boost when using all available points in S3 amounts between one and two orders of magnitude (see Figure 3.20).

For previous predictions, we used known random clusters that we generated. Natural clusters, in reality, will have a similar appearance, but we will not have information on which point is belonging to which cluster. Therefore to find the natural clusters, we employ one of the various clustering algorithms. We provide their overview and performance evaluation in section 3.6.

### 3.6. Identifying Clusters

Mobile user distribution is random in nature and a priori unknown to the experimenter. Since people tend to gather at specific locations more than others, user clusters naturally arise. Along with them, the black holes with no user measurements can also be a rather frequent occurrence. Therefore the closest emulation of such a scenario is non-uniform clusters, e.g., S3. Such clusters, we now consider as unknown and need to identify using various clustering algorithms.

Impact on the resulting MSE of the GPR prediction in the case when cluster centers are unknown is analyzed in this section. We first discuss various clustering algorithms widely used and described in the literature and apply them in S3. Lastly, we compare the GPR prediction error from averaged values of extracted clusters and discuss their performance compared to the exact clusters used in the previous chapter.

### 3.6.1. Different Clustering Methods in Python

The quality of the clustering method is hard to define since it is very application-specific. In the following, we introduce four categories following the notation of [21]:

- Partitioning algorithms,
- Hierarchical algorithms,
- Model-based algorithms,
- Density-based algorithms.

Many variations of these four algorithms are presented in the literature. For our purposes, we will discuss and apply one from each previously mentioned categories.

**Partitional clustering** consists of the construction of various data groups and their iterative improvement based on a specific partitioning criterion. One of the fundamental and widely implemented clustering algorithms proposed in the literature of data clustering is *K-Means* algorithm. It begins with choosing  $K$  representative points from the data as the initial cluster centroids. Next, each point is assigned to the closest centroid based on a proximity measure<sup>7</sup> selected, and thus clusters arise. The new cluster centroid is recomputed in each iteration by averaging the location of data points assigned to it. Lastly, all data points are reassigned to the new centroids. These two steps are iteratively computed until a convergence criterion is met. In practice, it follows the rule that the iterative procedure must be continued until only 1% of the points change their cluster memberships. The convergence speed of *K-Means* strongly depends on the choice of the initial centroids. To improve its speed, instead of selecting random initial centroids, the *K-Means++* algorithm is utilized. Here only the first centroid is selected at random, while all others are chosen such that they are farthest from all previously selected centroids. After finding initial centroids, regular *K-Means* is employed. Nevertheless, a substantial downside of *K-Means* algorithm is the necessary knowledge of the number of clusters in advance. It is also having trouble with data with outliers since all data points end up allocated to one of the clusters.

**Hierarchical clustering** uses a distance matrix as the clustering criteria. This method has the advantage of not requiring the number of clusters as the input but needs a termination condition. Hierarchical methods are categorized into agglomerative

<sup>7</sup>Euclidean distance metric is the most popular choice.

### 3. Clustering Data for Efficient Performance Map Reconstruction

and divisive clustering methods. *Agglomerative clustering* regards each data point as a singleton cluster at the bottom level and then merges the clusters bottom-up. The algorithm computes the dissimilarity matrix at each level, based on the chosen criteria<sup>8</sup>. The closest sets of clusters are merged, and the dissimilarity matrix updated correspondingly for the next level. This process continues until the final maximal cluster (that contains all the data objects in a single cluster) is obtained. The cluster hierarchy is interpreted using the standard binary tree terminology called a dendrogram. The tree root represents all data objects to be clustered (singleton points), and this is referred to as level 0 in the hierarchy. At each level, the data objects which are subsets of the entire dataset correspond to the clusters. The clusters entries are determined by traversing the tree from the current cluster node to the base singleton data points. The primary advantage of having a hierarchical clustering method is that it allows for cutting the hierarchy at any given level and obtaining the clusters correspondingly. Divisive methods, on the contrary, start at the top, regarding all data objects as a macro-cluster and splitting them continuously into two groups, hence generating a top-down hierarchy of clusters.

**Model-based clustering** techniques use a probabilistic approach to optimize the fit between the given dataset and some mathematical model. They operate under the assumption that a mixture of underlying probability distributions generated the data. In effect, each cluster can be represented mathematically by a parametric probability distribution, such as a Gaussian or a Poisson distribution. By modeling the entire dataset as a mixture of  $K$  (number of clusters) component distributions, the clustering problem is transformed into a parameter estimation problem. Data points that belong most likely to the same distribution can then easily be defined as clusters. Since the data is generated using Gaussian distributions around each cluster parent point in S3, using a *Gaussian Mixture (GM)* model should provide the best results in cluster identification. This method, however, has the same disadvantage as K-Means model, as it requires the number of clusters to be known beforehand.

**Density-based clustering** makes no assumptions about the number of clusters or their distribution and is therefore considered as a non-parametric method. Clusters represent dense areas in the data space separated from each other by sparser areas. *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)* estimates the density by counting the number of points in a fixed-radius neighborhood and

---

<sup>8</sup>Possible criteria: single link, complete link, group average, centroid similarity, Ward's criterion. In the following, we used the group average criterion.

considers two points as connected if they lie within each others neighborhood. A point  $a$  is called core point if the neighborhood of radius  $Eps$  contains at least  $MinPts$  points. A point  $b$  is directly density-reachable from a core point  $a$  if  $b$  is within the  $Eps$ -neighborhood of  $a$ . Two points  $a$  and  $b$  are called density-connected if there is a third point  $c$  from which both  $a$  and  $b$  are density-reachable. A set of density-connected points which is maximal with respect to density-reachability is defined as a cluster. *Ordering Points To Identify the Clustering Structure (OPTICS)*, unlike DBSCAN, keeps cluster hierarchy for a variable neighborhood radius  $Eps$ , and is better suited for usage on large datasets. [21]

The incomplete knowledge of the underlying cluster structure in S3 will contribute to a certain MSE increase of the GPR prediction. In the following, we look at these four clustering algorithms for cluster extraction in S3 and their impact on the MSE.

### 3.6.2. Comparison of Clustering Methods based on the GPR prediction

We now analyze how various methods for cluster extraction influence the GPR prediction in S3. Clustering algorithms we used for this inquiry are:

- K-Means
- Gaussian Mixture (GM)
- Agglomerative Clustering (AC)
- Ordering Points To Identify the Clustering Structure (OPTICS)

Since both for K-Means and GM algorithms, the number of clusters must be specified beforehand, we can either use the exact number of clusters generated in S3 or based on the dimension of the map and the known decorrelation distance determine the minimal number of clusters required for sufficiently accurate prediction. In case we choose the exact cluster count, the prediction results almost do not differ from prediction from exactly known clusters, as the number of training points remains the same in all three cases (exact, K-means, GM). However, if we consider we have a map of dimension  $100 \times 100$  pixels and a  $DD = 15$ , we know from previous discussions that the best-case scenario has training points equidistant spaced at least one DD one from another. This allows for seven training points in a dimension of 100 pixels, giving a total of  $7^2 = 49$  clusters. By using DD, we can thus extract the number of clusters required for optimal prediction. As we can see (3.23), the

### 3. Clustering Data for Efficient Performance Map Reconstruction

prediction error can thus be significantly lowered compared to using exact clusters. This method is of advantage, as in reality, the exact number of clusters is never known, but the DD can be determined from the data.

AC, on the other hand, does not require a cluster count to be known, but rather a termination condition. The termination condition defines the cluster distance above which clusters are no longer merged together. Hence, for the smaller termination distance, we obtain a larger number of identified clusters. The performance of the GPT prediction from such clusters is strongly dependent on the chosen termination distance. We surely do not want to merge clusters at distances greater than the DD, but we can set the termination condition to be less or equal to DD. We chose four different termination distances: DD/4, 3DD/4, DD/2, and DD.

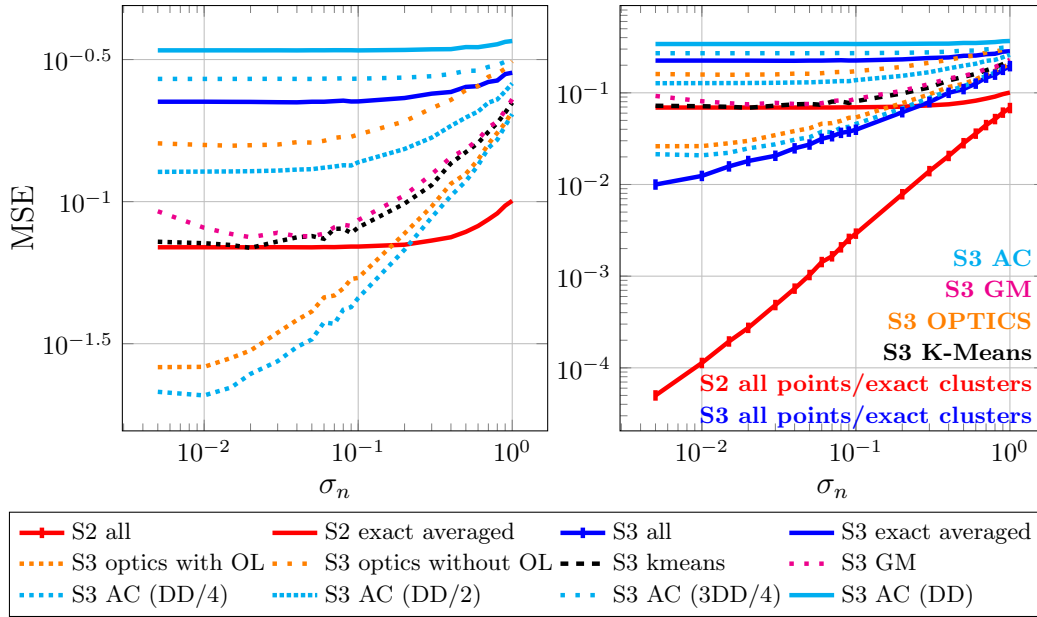
As for the OPTICS algorithm, we only need to specify the minimum number of points a cluster must contain. The average number of points per cluster is set to 30, but we likely have cluster points at edges originating from parents outside of the area of interest, making clusters with point count much lower than 30 probable. Therefore we choose  $MinPts = 5$ . This is the only algorithm than allows for the outliers not to be included in the clusters. Therefore, we analyzed how the prediction, both with and without outliers, influences the MSE.

Figure 3.23 shows how various clustering methods affect the GPR prediction, averaged over 250 different realizations. In addition to utilizing four previously mentioned clustering algorithms, we show how the prediction using exact clusters compares to them. While keeping the point densities equal in S3 and S2, their comparison is once again depicted. The number of extracted clusters is the most relevant parameter influencing the GPR prediction error. Table 3.3 summarizes the average number of clusters extracted using these various methods, as well as the number of outliers if present. The total number of points used for GPR prediction represents the sum of clusters and the outliers.

Algorithm:	exact	K-Means	GM	OPTICS	AC (DD/4)	AC (DD/2)	AC (3DD/4)	AC (DD)
Cluster count:	25	49	49	35	116	34	19	15
Outliers:	0	0	0	252	0	0	0	0
Used Train Points:	25	49	49	287	116	34	19	15

**Table 3.3.:** Cluster count for various clustering algorithms, averaged over 250 different realizations. Brackets in the AC algorithm denote various termination distances. The total number of measurements available on average is 622.





**Figure 3.23.:** Comparison of S2 and S3, while using various clustering methods in S3. Brackets in the AC algorithm denote various termination distances.

We notice that GM and K-Means algorithms have comparable performance since they both have the same number of identified clusters.

The AC algorithm strongly depends on the choice of the termination condition. Choosing a small termination distance (DD/4) results in much higher cluster count, providing better GPR prediction, as the number of training points is higher. We have to keep in mind that choosing termination distance too low reduces the computational speed, as the number of training points approaches the total number of measurements.

Optics algorithm results in larger cluster count compared to exact cluster count, therefore increasing the training point count for the GPR and the overall prediction performance. Including the outliers (OL), increases our training dataset up to three times, resulting in more accurate prediction, at the price of longer computation time.

### 3. Clustering Data for Efficient Performance Map Reconstruction

---

Under the scope of wireless performance map reconstruction, where the decorrelation distance order of magnitude is roughly known or can be estimated from measurements, we conclude that the K-Means method is best suited for cluster identification. Even though the AC method can provide better results for low termination conditions than K-Means, the clusters it identifies can take various shapes and highly uneven number of cluster points among them. Thus, resulting in rather high MSE errorbars, depending on the form of the original clusters generated in S3.

#### 3.6.3. Performance Comparison of the Complete, Clustered and Identified Training Sets

In this section, we want to summarize Chapter 3, by comparing performance when using different training sets we previously discussed. We started by working with a complete measurement set as the training dataset. We moved on to clustering the whole measurement set to known simulated clusters, which we then averaged to find a single representative point for each measurement group, thereby reducing the training set and the computational time significantly. Finally, we analyzed a more realistic scenario where cluster identification is required before averaging, as the real clusters are always unknown in advance. Additionally, the hyper-parameters of the underlying performance map can vary depending on the channel conditions and therefore require to be estimated as well.

Therefore as the final comparison, we provide the performance of the GPR learning as well as GPR prediction plotted against the *density of measurement points* for the following cases:

- All measurement points in S2,
- All measurement points in S3,
- Clustered and averaged points in S2,
- Clustered and averaged points in S3, using exact clusters,
- Clustered and averaged points in S3 - using K-Means method for cluster identification.

For the overview we summarize this list in the Table 3.4 bellow with line styles corresponding to Figure 3.24.







Scenario \ Train Set	all points	average exact cluster centers	average K-Means identified cluster centers
S2			
S3			

Table 3.4.

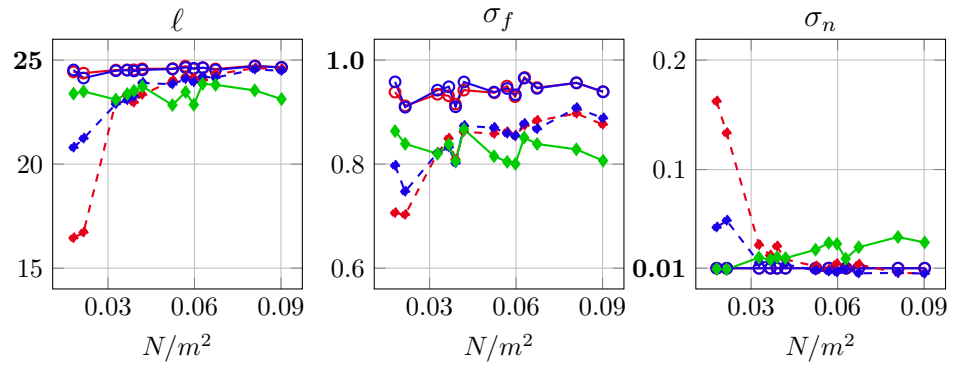
Figure 3.24a shows how the learning of hyper-parameters decorrelation distance ( $\ell$ ), signal standard deviation ( $\sigma_f$ ), and noise standard deviation ( $\sigma_n$ ) depends on the number of available measurement points in the area of interest. Note that  $N$  denotes the number of available measurement points in the area of interest and  $m^2 = \text{pixel}^2$ , as our simulation works with pixel points that can be interpreted as various distance units. As expected, learning using all available measurements as train points results in the closest estimate of the actual parameters. While in clusters identified using K-Means, learning from averaged cluster points has rather consistent performance over point density, as the number of training points is kept constant.

Under the assumption that hyper-parameters are known or accurately estimated from the measurements, we can compare the GPR prediction of previously mentioned cases, in Figure 3.24b.

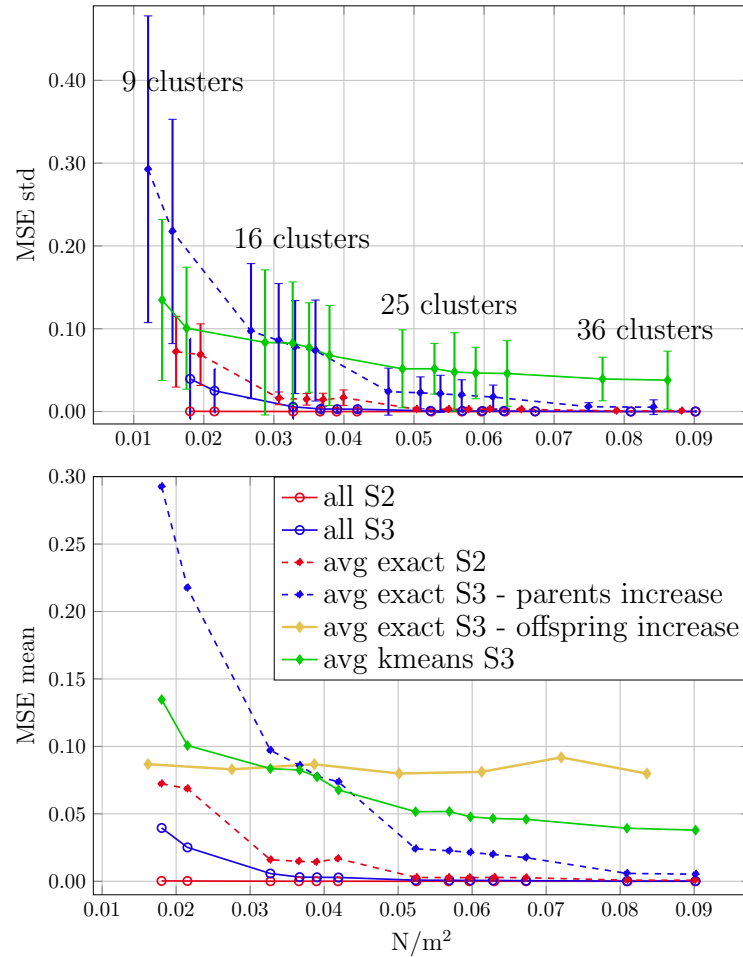
Point density represents the total number of measurements in the area of interest. The increase in density in our simulation of TCP can either be achieved by introducing new clusters (increasing  $\lambda_{\text{parent}}$ ) or adding new offsprings to the existing clusters (increasing  $\lambda_{\text{daughter}}$ ). Note that these two cases deliver a significant difference in the GPR performance. When we increase offspring density, we gain very little new information, as the areas where we add them already have measurements. While when adding new clusters by increasing the parent density, we can learn additional information about the black hole areas. This performance difference is also depicted in Figure 3.24. Notice how the yellow line is very steady, independent of the density we applied. As explained, the averaging over a higher number of points inside a cluster does not gain much new information, other than reducing the noise, which we here assumed to be known. On the contrary, if we increase the number of clusters, we see a significant drop in the MSE for higher densities.

Notice that the ascending point density (for all lines but ) used here comes from increasing the number of parents(clusters) in TCP, and smaller variations in offspring

### 3. Clustering Data for Efficient Performance Map Reconstruction



(a)



(b)

**Figure 3.24.:** (a) Learning of hyper-parameters. (b) Prediction using exact hyper-parameters (lines are shifted to show errorbars).

densities.

We chose the decorrelation distance  $DD = 25$ ,  $\sigma_f = 1$ ,  $\sigma_n = 0.01$  for a map dimension of  $100 \times 100$  pixels. As decorrelation distance is 25, and we need at least one training point per  $DD \times DD$  area for sufficient prediction quality, this would mean at least 16 training points in area  $100 \times 100$  are required. Therefore we observe rather high error bars, for the case only 9 clusters are available - while for 16, 25, and 36 clusters, the MSE errorbars significantly reduce.

Notice that the prediction MSE using exact averaged clusters, both in S2 and S3, actively improves with higher densities when the cluster number increases. On the other hand, it stays rather flat in the regions where cluster count is constant, but offspring density increases. This implies that in order to improve prediction, we must introduce new clusters in the black hole areas, rather than increasing the density by adding new points in already existing clusters. On the other hand K-Means, independently of the number of clusters generated by simulation, always groups all available points to a minimum number of clusters required by map dimension and DD - in this case, 16. Thus, keeping the MSE line rather flat over various point densities.

The density of 0.03 indicates that, on average, we have 3 points in a  $10\text{m} \times 10\text{m}$  area. We note from the results that for densities lower than 0.03 averaging is counterproductive, as we are left with an insufficient number of training points after it is applied. However, if a sufficient number of new clusters is introduced in the black holes, as in the case of  $N/m^2 = 0.06$  (36 clusters), we see that averaging is beneficial for the computation reduction, as the prediction performance approaches the case where the complete measurement set is used for prediction (all S3). Assuming the DD in Vienna is in the order of 25m, and signal variance is of the order  $\sim 1$  as in our simulation, we can convert the problem statement from  $10\text{m}^2$  area to Viennas' 1st district of  $2.869\text{km}^2$ , or even the entire Vienna of  $414\text{km}^2$ . These areas would require 4590 clusters and 662400, respectively, to achieve an MSE of around 0.05.



## 4. Experimenting on Real Data

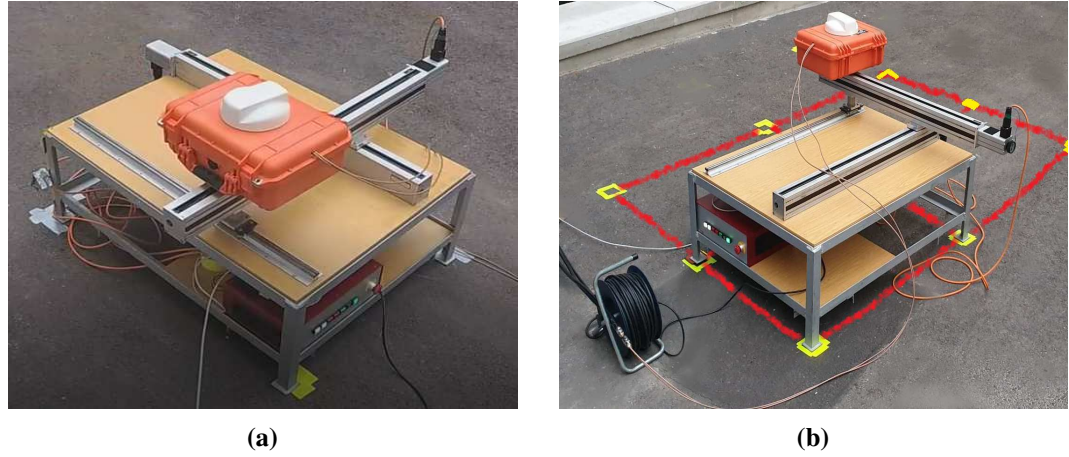
In the real world, we do not have automatically clustered datasets. We had to find the best way to reduce the GPR complexity without significant prediction performance reduction. Using a simulation of the real-world scenario in the 3, we concluded that the best method to achieve this is the K-Means algorithm. In this chapter, we focus on testing the results from Chapter 3 in two different real-world scenarios. We first look at a setup where a receiving cell phone is mounted on an XY positioning table and connected to an external antenna for measuring the strength of the received LTE signal [4]. Secondly, we exploit the measurements of the signal strength in LTE, collected using a UE mounted to the drone. The employed setup for collecting a dataset of the mobile cellular network's KPIs is given in Platzgummer et al.[5].

### 4.1. XY-Table Measurements

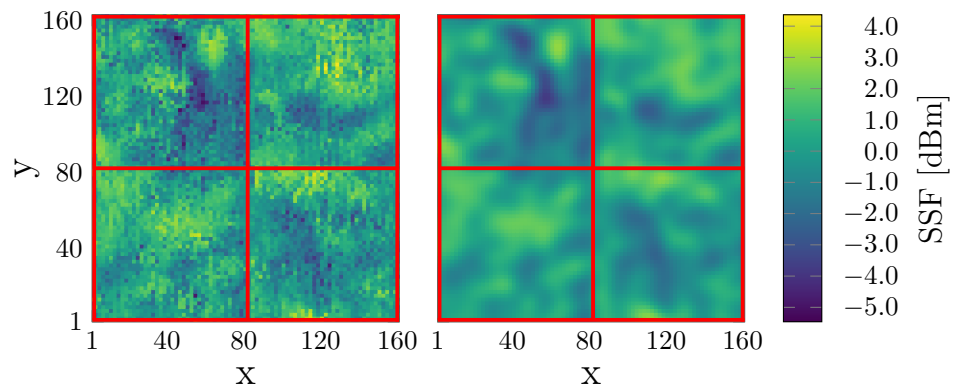
In this scenario, we used the setup from the reference [4], where the receiving cell phone is mounted on an XY-positioning table and connected to external antennas. The table covers  $81\text{cm} \times 81\text{cm}$  area, and the intervals at which it stops to take measurements can be adjusted. We took measurements at an interval of 2cm both in X and Y-direction. To get a map larger than  $81\text{cm} \times 81\text{cm}$ , we moved the table across four side-by-side tiles, denoted by red squares in Figures 4.1 and 4.2. Such setup produced the measurement map of dimension  $162\text{cm} \times 162\text{cm}$ . Notice that these measurements were taken over the course of multiple days. We processed these raw data, and filtered the measurements to LTE band 20 (800MHz frequency) and its ARFCN 6400. We average the measurements taken at each position to get a more accurate estimate of the signal strength. Each RSRP value is reduced by the mean of the entire measurement map. This way, instead of predicting RSRP of the received signal, we focus on the Small Scale Fading effect. Next, the hyper-parameters of the underlying SSF map are estimated using the complete measurement area. Parameter

#### 4. Experimenting on Real Data

estimates are:  $DD = 6.69\text{cm}$ ,  $\sigma_f = 1.05$ , and  $\sigma_n = 0.844$ . These estimates are further considered to be exact and are employed when predicting the map using reduced datasets.



**Figure 4.1.:** XY-positioning table measurement setup: (a) Measurement box moves across the rails both in X and Y direction. (b) Red chalk with yellow tape marks the four table positions next to each other, to measure a four times larger area.



**Figure 4.2.:** Original measurements on the left, predicted map on the right.

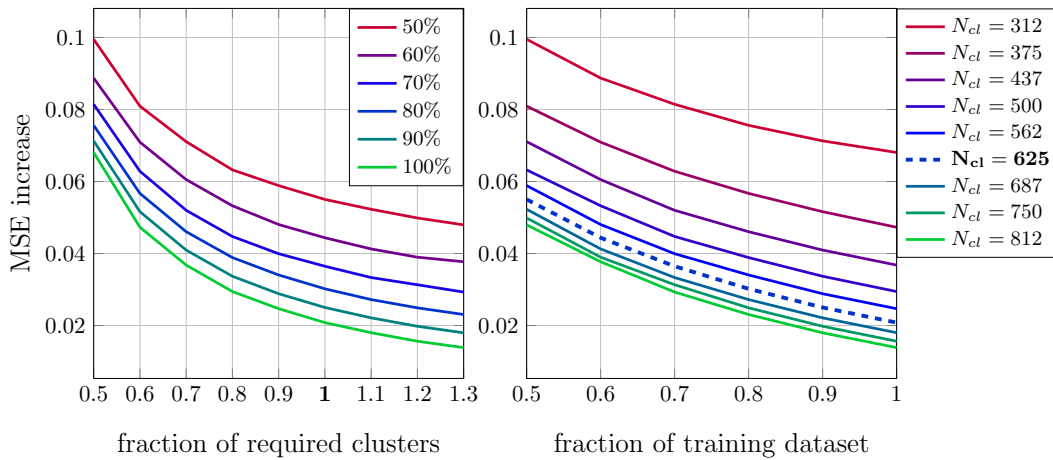
Figure 4.2 shows the measurements taken at distances of 2cm, on the left, and predicted map using all measurement points on the right. This predicted map is then



used as the ground truth reference for MSE increase calculation. The red squares denote the position of the XY-Table to measure at four adjoining areas.

We randomly sample from the entire dataset and use the subset as the training dataset. Next, the clustering using K-Means method is applied to reduce the computational time of the GPR. We have seen in Chapter 3 that the required number of clusters strongly depends on the dimension of the map and its decorrelation distance. Thus, using the parameter  $DD = 6.69\text{cm}$ , we determine the amount of 625 *required clusters*.

To affirm our conclusions from Chapter 3, using various point densities for the training dataset, we cluster them below and above the number of required clusters. Values from 312 to 812 clusters, corresponding to 0.5 to 1.3 fraction of required clusters, were implemented. We show the results in Figure 4.3, where an increase in MSE is depicted for various point densities and a different number of identified clusters. The hyper-parameters and the ground truth map are estimated using measurements at all positions. For this reason, when we speak of the MSE increase, we refer to the prediction performance degradation from using clustered values compared to the prediction from the entire dataset.



**Figure 4.3.:** MSE increase conditioned a various number of identified clusters and different point densities of the training dataset.

On the left plot, we show the MSE increase as a function of the fraction of the

## 4. Experimenting on Real Data

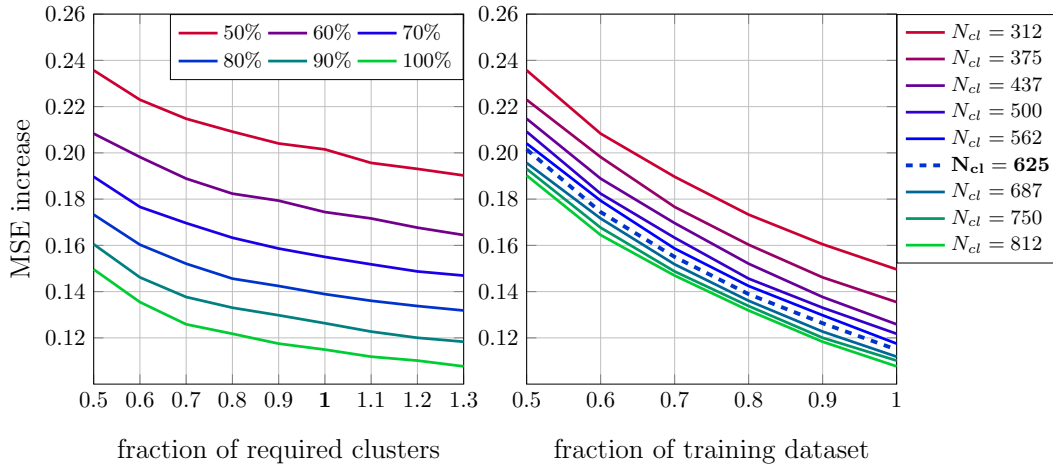
---

required number of K-Means identified clusters, while the colors of the lines represent percentages of the used training dataset. These percentages can be translated to point densities, where the range [50%...100%] corresponds to [1283...2566N/m<sup>2</sup>]. On the right plot, the MSE decay as a function of the fraction of the training dataset is depicted, while different colors represent a different number of K-Means identified clusters  $N_{cl}$ . Notice the dashed line outlining the required number of clusters, and the significant MSE increase if we use fewer clusters than needed. On the other hand, employing more clusters than needed does not improve the performance significantly.

Even though these two plots seem similar, notice that the lines with constant point densities (left) drop much faster than the lines with a constant number of clusters (right). We even see a saturation effect on the left plot, when the required number of clusters is achieved. Going from 0.5 to 1 fraction of required clusters, the MSE increase drops by half, while a further increase in the fraction of necessary clusters gains small MSE improvement.

If we use all available measurements and apply the clustering instead of using each measurement separately in the prediction step, we can reduce the size of the inversion matrix more than tenfold, from  $6724 \times 6724$  to  $625 \times 625$ , at the cost of only 2% MSE increase. If we can afford a further 0.2% MSE increase, the size of the inversion matrix can be further halved to  $312 \times 312$ . By reducing the training dataset from 6725 to 625 we reduce the computation time of the GPR prediction 66 times. Therefore, the clustering strategy will be beneficial for reducing computational efforts when working with large datasets of crowdsourced measurements, that have 10 000 samples or more.

## 4.2. Drone Measurements



**Figure 4.4.:** MSE increase conditioned a various number of identified clusters and different point densities of the training dataset.

We can also subsample the measurements and take one measurement at each 4cm length, instead of 2cm. This way, we come to a total of  $6724/4 = 1681$  measurement points, which can again be clustered to 625 clusters. The MSE increase for this case is shown in Figure 4.4. Note that 100% of measurements here denotes the 1681 measurements, comparing to 6724 measurements in Figure 4.3. Thus, by subsampling the map this way and grouping all measurement points to 625 clusters, we reach a 10% increase in MSE compared to the case where the measurements were made at 2cm distances.

## 4.2. Drone Measurements

In this section, we employ the measurement setup from the reference [5]. Four UEs were mounted on top of the drone that flew a line pattern shown in Figure 4.6, over a field area in Klingenbach, Austria, shown in Figure 4.5. After several days of experimentation, the raw data measurements collected using all four UEs were filtered to LTE band 20 (800MHz frequency) with ARFCN 6300 [22], leaving a total of 4725 measurement points.

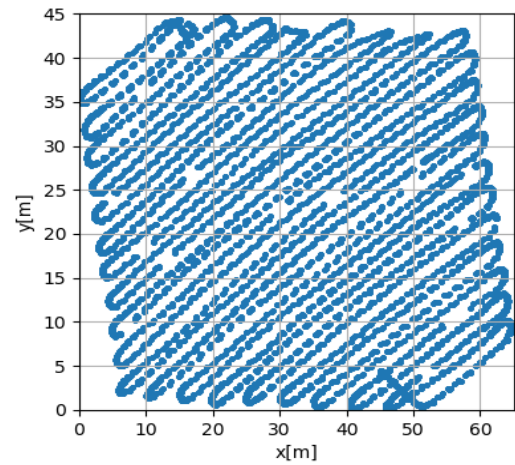
#### 4. Experimenting on Real Data



**Figure 4.5.:** Snapshot of a field in Klingensbach, Austria [23], where the drone measurements were taken.



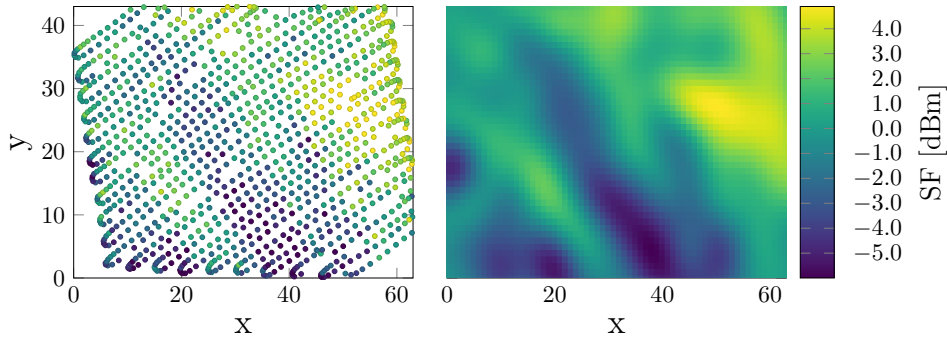
(a)



(b)

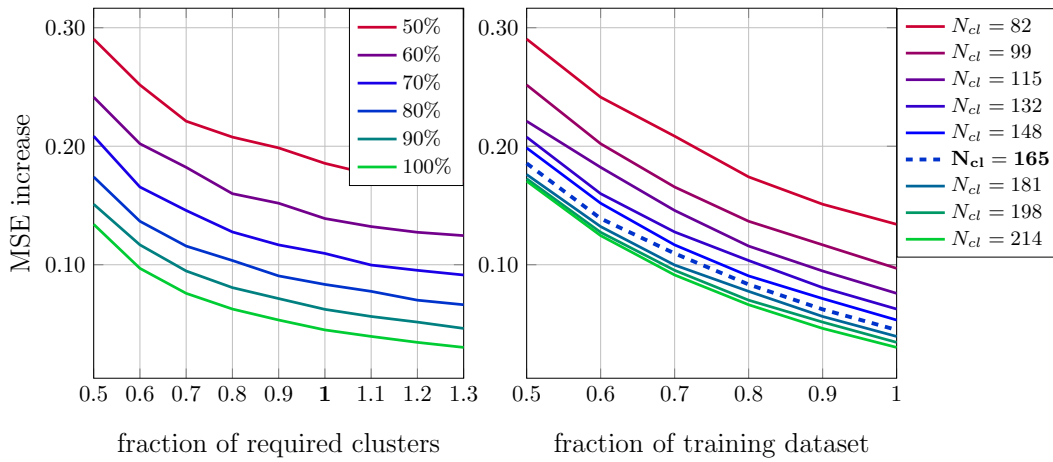
**Figure 4.6.:** (a) Drone with four UEs mounted on top. (b) Drone measurement pattern.

Next, the RSRP is reduced by its mean value, which accounts for the path loss of the given cell, leaving a Shadow Fading (SF) map as a result. Entire measurement set is then utilized to estimate the hyper-parameters  $DD = 4.42\text{m}$ ,  $\sigma_f = 1.41$  and  $\sigma_n = 1.57$  of the underlying SF map. Finally, using the entire dataset, the ground truth map is predicted and then used as a comparison to calculate the MSE increase when clustering is applied.



**Figure 4.7.:** Original measurements on the left, predicted map on the right.

Again as in the XY-Table scenario, we used point densities from 50% to 100% of the complete measurement set. Additionally, the number of clustered identified using the K-Means method was varied around the required number of clusters,  $N_{cl} = 140$ .



**Figure 4.8.**

**Figure 4.9.:** MSE increase conditioned a various number of identified clusters and different point densities of the training dataset.

We observe the same trend in Figure 4.8 as in the XY-Table scenario, with a slight increase in the MSE. This increase happens due to the higher signal variance in the

## 4. Experimenting on Real Data

---

shadow fading map compared to the small scale fading scenario, as well as the higher measurement noise level. As for the number of identified clusters, we again notice a jump in the MSE when the required number of clusters condition is not satisfied. The complete dataset we used here has 4725 measurement points. If we agree to accept an MSE increase of 10%, we can reduce the training dataset by 30%, thereby reducing our measurement time in the same amount. In addition to that, exploiting clustering allows us to shorten the time between the measurements. We now only need to fly and measure at those 625 cluster locations instead of flying the entire field in a line pattern.

### 4.3. Sampling Strategy

Based on the conclusion of Chapter 3 and the results we have shown in this chapter, we can derive the following sampling strategy for future reference.

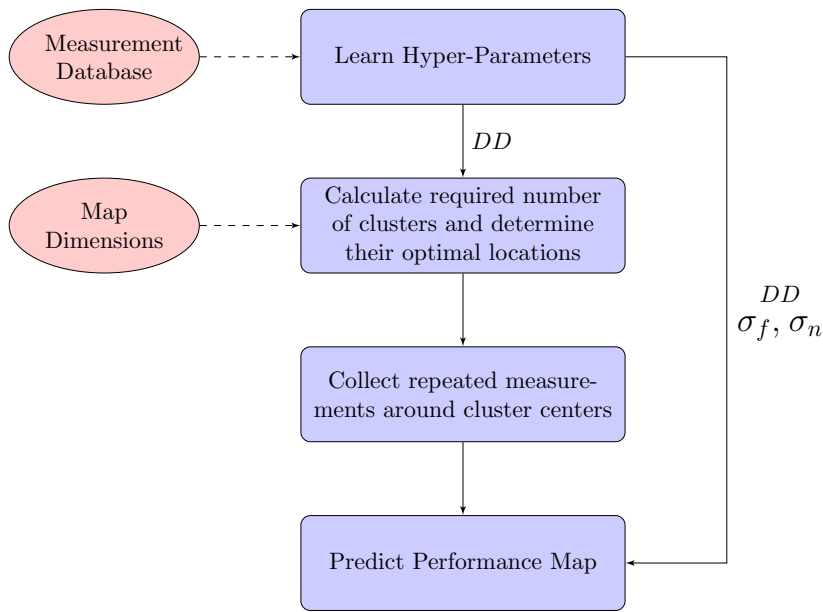
In the first step, we use all available samples to learn the underlying map hyper-parameters. When working in an application-specific domain, where we can roughly guess the parameters in advance, choosing narrower bounds for parameters  $DD$ ,  $\sigma_f$ , and  $\sigma_n$  can significantly speed up the learning process. This step is most time-consuming and does not need to be repeated often, as the hyper-parameters vary insignificantly over time and location.

Secondly, we fill the prediction area with clusters spaced by one  $DD$ , learned in step one. This way, optimal positioning of the training points is achieved, where we reduce the number of locations we need to sample at to 625 clusters in XY-Table and 165 clusters in the drone measurement scenario.

Thirdly, we collect repeated samples at previously determined cluster locations over which we can average. In the case of the drone, by measuring at one specific location for a longer time, we can reduce the position error, which is sensitive to drone velocity. This way, instead of extensively sampling the area in line pattern at a constant speed, we can optimize and speed up the measurements by only collecting data at the cluster locations.

Lastly, using the hyper-parameters and the cluster measurements we made, we can average over the clusters, and using those averaged number of clusters as the training dataset, calculate the underlying performance map.

### 4.3. Sampling Strategy



**Figure 4.10.:** Flow chart of the sampling strategy.





## 5. Summary and Future Work

### 5.1. Summary

We exploited the concept of clustering measurements with the goal of reducing measurement noise, both in KPI value and GPS location. By introducing the Gaussian Process Regression as a possible solution to the prediction of network KPI performance maps, we were able to leave room for slight measurement noise uncertainties in the data itself. As measurement noise is allowed in the training locations, we are able not only to retrieve the prediction value at a given test location but also a level of certainty that predicted value is the true one. The standard deviation in the test values can then be utilized to find black hole areas, and discover at which locations the new clusters of measurements should be introduced next. We investigated how different distributions of training datasets influence the GPR prediction with and without applying clustering of the training dataset. Going from clusters at regular distances and uniform cluster points distribution (S1), we moved to those with random cluster points distribution (S2), finally coming to a close real-world scenario, where both cluster and cluster points locations are randomized (S3). By analyzing the MSE degradation through these scenarios for various point densities, we concluded that the density increase achieved by adding extra data points into existing clusters does not bring much improvement. Instead, we must increase the number of clusters, and not the offspring cluster points, to gain new information.

As the real-world data clusters are unknown in advance, we reviewed different types of clustering algorithms that could be exploited for our purposes. We concluded that the best option is the K-Means algorithm, with a downside of a need to fix the number of clusters that the algorithm distinguishes. This downside, however, can be overcome by first estimating the decorrelation distance of the underlying KPI performance map. Based on the dimension of the test area and the estimated DD, we can easily calculate the number of clusters that would suffice for an accurate prediction. We

## 5. Summary and Future Work

---

compared the K-means cluster identification method with the scenarios where the simulated clusters are precisely known, over varying point densities. We had seen that injecting new clusters in our simulation brings improvement, even when the number of K-Means identified clusters stays the same.

In Chapter 4, we examined how the K-Means clustering method works in two different real-world scenarios. First, by exploiting SSF measurements collected by four XY-positioning tables provided with exact locations of the training dataset, we investigated how using different point densities with varying cluster count influences the MSE. We remarked a significant jump in the MSE when the required number of clusters for this specific map had not been satisfied. While on the other hand, increasing this number did not bring much improvement compared to the computational time that would be lost at its cost. Secondly, we analyzed the shadow fading measurements collected by four UEs mounted on a drone. We noticed a similar trend as before, with a slight change in the MSE level, present due to the different signal variance. Finally, we derived a sampling strategy for the collection of measurements.

### 5.2. Possible Extensions

We can now understand the influence of the map decorrelation distance on the GPR, and how to choose and place the clusters appropriately when possible. We also know which clustering method is best suited in our scenario when the clusters are unknown. This thesis provides the ground, and there are several things to be considered as further applications. Exploiting the RTR crowdsourced measurements would be one of them, conditioned that the available RTR database expands sufficiently. On the one hand, the influence of the location uncertainty in the training data set is yet to be examined. On the other, a cost function specifying the best location for the next measurement in the black hole areas could be inquired.

# Appendix A.

## Source Code Implementation

The source code implementation of the simulations and real data experiments discussed through this Thesis is available online at <http://squid.nt.tuwien.ac.at/gitlab/stripkov/MasterThesis>. In this Appendix, we show only a few segments of the source code.

### A.1. Sampling from Multivariate Gaussian Distribution

```
1 import numpy as np
2 import scipy as sp
3
4 def sampling_fromGPR(xMin, yMin, xMax, yMax, TrainPoints, step, DD, sigma_f):
5     """
6     Returns [TrainPoints, TrainValues], [TestPoints, TestValues]
7
8     Input Parameters:
9     xMin, yMin, xMax and yMax define the dimensions of the testing grid;
10    step defines how dense the grid is (if step=2 we have a grid 0,2,4...);
11    DD is decorrelation distance of the map we are sigma_f is signal standard variation
12    """
13    # regular grid of TestPoints
14    x = np.arange(xMin, xMax+1, step, dtype=int)
15    y = np.arange(yMin, yMax+1, step, dtype=int)
16    xrr = np.repeat(x, y.size); yrr= np.tile(y, x.size)
17    TestPoints = np.stack((xrr, yrr), axis = 1)
18
19    # compute squared exponential kernel
```

## Appendix A. Source Code Implementation

```
20 AllPoints = np.append(TestPoints, TrainPoints,axis=0)
21 sq_dist = np.sum(AllPoints**2, 1).reshape(-1, 1) + np.sum(AllPoints**2, 1) - 2 * np.dot(
    ↳ AllPoints, AllPoints.T)
22 kernel = sigma_f**2 * np.exp(-0.5 / DD**2 * sq_dist)
23
24 # perform Cholesky decomposition
25 A = sp.linalg.cholesky(np.add(kernel, 1e-10*np.eye(kernel.shape[0])), lower= True)
26
27 # generate vector of independent, standard normal variables
28 Z = np.random.normal(0.0, 1.0, AllPoints.shape[0])
29
30 # compute samples as X=AZ
31 X = A.dot(Z)
32
33 # separate train and test datasets
34 TestPoints_values = X[0:TestPoints.shape[0]]
35 TrainPoints_values = X[TestPoints.shape[0]:]
36
37 return np.concatenate((TestPoints,TestPoints_values.reshape(TestPoints_values.shape[0],1)),
    ↳ axis = 1), np.concatenate((TrainPoints,TrainPoints_values.reshape(TrainPoints_values
    ↳ .shape[0],1)),axis = 1) # return TestData, TrainData
```

### A.2. Generating Clusters using TCP

Using TCP implementation from [24], we created a function that returns offspring locations, together with information about specific clusters they belong to.

```
1 import numpy as np
2 import pandas as pd
3
4 def thomas_cluster_process(xMin,yMin,xMax,yMax,lambdaParent,lambdaDaughter,
    ↳ sigmaDaughter, extension_parameter=6):
5     """
6     Returns the set of points, cluster they belong to, cluster centers they belong to.
7
8     Input parameters:
9     xMin,yMin,xMax,yMax define the simulation area;
10    lambdaParent*sim_area gives the average number of parent points (number of clusters);
11    lamdaDaughter gives average number of points in clusters;
12    sigmaDaughter is the standard deviation defining the daughter spread around the parents;
13    extension_parameter defines the extended area around the simulation area – for edge effects;
```

```

14  """
15
16  # Extended simulation windows parameters
17  rExt=extension_parameter*sigmaDaughter
18  xMinExt = xMin - rExt; xMaxExt = xMax + rExt
19  yMinExt = yMin - rExt; yMaxExt = yMax + rExt
20  # rectangle dimensions
21  xDeltaExt = xMaxExt - xMinExt; yDeltaExt = yMaxExt - yMinExt
22  # area of extended rectangle
23  areaTotalExt = xDeltaExt * yDeltaExt
24
25  # Poisson point process for the number of parents
26  nPointsParent = np.random.poisson(areaTotalExt*lambdaParent)
27  # x and y coordinates of Poisson points for the parents
28  xxParent=xMinExt+xDeltaExt*np.random.uniform(0,1,nPointsParent)
29  yyParent=yMinExt+yDeltaExt*np.random.uniform(0,1,nPointsParent)
30
31  # Poisson point process for the number of daughters in each cluster
32  nPointsDaughter = np.random.poisson(lambdaDaughter, nPointsParent)
33  numbPoints = sum(nPointsDaughter) # total number of points
34
35  # Generate the (relative) locations in Cartesian coordinates by
36  # simulating independent normal variables for relative coordinates
37  xx0 = np.random.normal(0, sigmaDaughter, numbPoints)
38  yy0 = np.random.normal(0, sigmaDaughter, numbPoints)
39
40  # replicate parent points (ie centres of disks/clusters)
41  xx = np.repeat(xxParent, nPointsDaughter)
42  yy = np.repeat(yyParent, nPointsDaughter)
43
44  # translate points (ie parents points are the centres of cluster disks)
45  xx = xx + xx0; yy = yy + yy0
46
47  # create df with all relevant information
48  groups = np.arange(nPointsDaughter.shape[0])
49  col3 = np.repeat(groups, nPointsDaughter, axis=0)
50  xParent = np.repeat(xxParent, nPointsDaughter, axis=0)
51  yParent = np.repeat(yyParent, nPointsDaughter, axis=0)
52  ALL = np.stack((xx,yy,col3,xParent,yParent),axis = 1)
53  df_all = pd.DataFrame(ALL)
54  df_all.columns = ['x', 'y', 'group', 'xParent', 'yParent']
55
56  # retain points inside simulation window
57  indexNames = df_all[ np.logical_or(np.logical_or(df_all.x<xMin,df_all.x>xMax),np.

```

## Appendix A. Source Code Implementation

```
58         ↪ logical_or(df_all.y<yMin,df_all.y>yMax)).index
59         df_all.drop(indexNames , inplace=True) # delete extended region
60     return df_all
```

### A.3. Hyper-Parameter Learning and GPR Prediction

Hyper-parameter learning can be omitted when the parameters are already known, by simply setting

```
optimizer=None.
```

For implementation reference see [25] and [26].

```
1 import sklearn.gaussian_process as gp
2
3 # choosing smaller hyper-paramrter bounds reduces computational time
4 # RBF is the squared exponential kernel
5 # it is multiplied with a constant kernel accounting for signal variance
6 # white kernel is added on the diagonal, accounting for noise variance
7 kernel = gp.kernels.RBF(10,(1,50)) * gp.kernels.ConstantKernel(1,(1e-5,1e5)) + gp.kernels.
8         ↪ WhiteKernel(noise_level=1, noise_level_bounds=(1e-5,1e5))
9 model= gp.GaussianProcessRegressor(kernel=kernel, alpha=1e-4, n_restarts_optimizer=100,
10         ↪ normalize_y=True)
11 model.fit(TrainLocations, TrainValues) # fit the model to training dataset
12 PredictedValues= model.predict(TestLocations) # predict using learned model
```

### A.4. Required Number of Clusters

```
1 # DD...learned decorrelation distance
2 # map dimensions
3 x_max = TestLocations[:,0].max()
4 y_max = TestLocations[:,1].max()
5 # requiered number of clusters
6 Ncl_required = np.arange(0,x_max,DD).shape[0]*np.arange(0,y_max,DD).shape[0]
```

## Appendix B.

### Measurement Dataset

The measurement datasets utilized in this Thesis, collected using XY-positioning table and the drone, can be found at

<http://squid.nt.tuwien.ac.at/gitlab/stripkov/MasterThesis/data/experiments>.

Matrix form of the XY-positioning table dataset we used:

$$XY_{\text{dataset}} = \begin{bmatrix} \text{x[cm]} & \text{y[cm]} & \text{RSRP[dBm]} \\ 1 & 1 & -120.42 \\ 1 & 3 & -120.73 \\ 1 & 5 & -121.01 \\ \vdots & \vdots & \vdots \\ 161 & 163 & -120.96 \\ 161 & 165 & -120.01 \end{bmatrix}$$

For our purposes we subtract the RSRP mean from the last column leaving the SSF:

$$XY_{\text{dataset}} = \begin{bmatrix} \text{x[cm]} & \text{y[cm]} & \text{SSF[dBm]} \\ 1 & 1 & 6.72788962e-01 \\ 1 & 3 & 3.64455629e-01 \\ 1 & 5 & 8.11222958e-02 \\ \vdots & \vdots & \vdots \\ 161 & 163 & 6.97788962e-01 \\ 161 & 165 & 1.37788962e-01 \end{bmatrix}$$

## Appendix B. Measurement Dataset

---

Matrix form of the drone measurement dataset we used:

$$\text{Drone}_{\text{dataset}} = \begin{bmatrix} \textcolor{blue}{x[m]} & \textcolor{blue}{y[m]} & \textcolor{blue}{RSRP[\text{dBm}]} \\ 45.09558954 & 4.42181968 & -76.825 \\ 45.26660148 & 4.31748209 & -76.95 \\ 45.88604408 & 3.85748815 & -74.1 \\ \vdots & \vdots & \vdots \\ 15.22734257 & 43.30085646 & -73.075 \\ 15.70447844 & 42.67995257 & -72.75 \end{bmatrix}$$

For our purposes we subtract the RSRP mean from the last column, leaving the SF:

$$\text{Drone}_{\text{dataset}} = \begin{bmatrix} \textcolor{blue}{x[m]} & \textcolor{blue}{y[m]} & \textcolor{blue}{SF[\text{dBm}]} \\ 45.09558954 & 4.42181968 & -1.93238536 \\ 45.26660148 & 4.31748209 & -2.05738536 \\ 45.88604408 & 3.85748815 & 0.79261464 \\ \vdots & \vdots & \vdots \\ 15.22734257 & 43.30085646 & 1.81761464 \\ 15.70447844 & 42.67995257 & 2.14261464 \end{bmatrix}$$



# Acronyms

**3GPP** 3rd Generation Partnership Project.

**5gnr** 5G New Radio.

**ARFCN** Absolute Radio-Frequency Channel Number.

**DBSCAN** Density-Based Spatial Clustering of Applications with Noise.

**DD** decorrelation distance.

**GM** Gaussian Mixture.

**GP** Gaussian Process.

**GPR** Gaussian Process Regression.

**GPS** Global Positioning System.

**KPI** Key Performance Indicator.

**MDT** Minimization of Drive Tests.

**MSE** Mean Squared Error.

**OPTICS** Ordering Points To Identify the Clustering Structure.

**RSRP** Reference Signal Receive Power.

**RTR** Austrian Regulatory Authority for Broadcasting and Telecommunications.

**SE** Squared Exponential.

**SF** Shadow Fading.

**SSF** Small Scale Fading.

**UE** User Equipment.



# Glossary

**cluster** A group of measurement data points around a cluster center, at a maximum distance from cluster center defined by cluster radius  $r$  in S1, or by standard deviation  $\sigma$  of normal distribution around the cluster center in S2 and S3.

**cluster center** A parent data point around which cluster of data points is formed. In case of S1 location of cluster center coincides with averaged location of its daughter cluster data locations.

**crowdsourcing** Obtain information by enlisting the services of a large number of people, either paid or unpaid, typically via the Internet, rather than having tasks done within a company by employees.

**decorrelation distance** Describes how far do you need to move (along a particular axis) in input space for the function values to become uncorrelated [11].

**S1** Scenario 1: evenly spaced cluster centers with evenly spaced cluster points.

**S2** Scenario 2: evenly spaced cluster centers with randomly spaced cluster points.

**S3** Scenario 3: non-uniform clusters generated using Thomas Cluster Process.



# List of Figures

2.1. GPR example . . . . .	6
3.1. Cluster types . . . . .	17
3.2. Sampling from normal distribution . . . . .	18
3.3. Test grid and random train locations. . . . .	20
3.4. Distance between cluster centers $d$ . . . . .	21
3.5. Predicted performance maps using different $d$ . . . . .	22
3.6. S1 MSE using different $d$ . . . . .	23
3.7. Cluster radius $r$ . . . . .	24
3.8. Predicted performance maps using different $r$ . . . . .	24
3.9. S1 MSE using different $d$ and different $r$ . . . . .	25
3.10. Prediction MSE in S1 . . . . .	27
3.11. Cluster spread in S2 . . . . .	28
3.12. Prediction MSE in S2 for fixed number of points per cluster but different $d$ . . . . .	29
3.13. Prediction MSE in S2, for various cluster parameter combinations. . . . .	30
3.14. Prediction MSE 90 percentile in S2, for various cluster parameter combinations. . . . .	32
3.15. Prediction MSE in S2 for different point densities . . . . .	33
3.16. Thomas Cluster Process . . . . .	36
3.17. Training points, S1 vs S2 . . . . .	37
3.18. Prediction MSE, S1 vs S2 . . . . .	38
3.19. Training points, S2 vs S3 . . . . .	40
3.20. Prediction MSE, S2 vs S3, for different cluster densities . . . . .	41
3.21. Prediction MSE, averaged S2 vs averaged S3 . . . . .	42
3.22. Decorrelation distance learning, S2 vs S3 . . . . .	43
3.23. Prediction MSE, S2 vs S3, using various clustering methods . . . . .	49
3.24. Parameter learning and prediction - final comparison . . . . .	52

## List of Figures

---

4.1. XY-Table setup . . . . .	56
4.2. XY-Table: SSF map . . . . .	56
4.3. XY-Table: MSE increase . . . . .	57
4.4. XY-Table: MSE increase . . . . .	59
4.5. Drone measurement area, Klagenbach, Austria . . . . .	60
4.6. Drone measurement setup . . . . .	60
4.7. Drone: SF map . . . . .	61
4.8. XY-Table . . . . .	61
4.9. Drone: MSE increase . . . . .	61
4.10. Sampling Strategy . . . . .	63

# Bibliography

- [1] CISCO. *Cisco Annual Internet Report (2018–2023)*. Mar. 2020. URL: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf>.
- [2] Wuri A Hapsari et al. “Minimization of drive tests solution in 3GPP.” In: *IEEE Communications Magazine* 50.6 (2012), pp. 28–36.
- [3] Austrian Regulatory Authority for Broadcasting and Telecommunications (RTR). *RTR Multithreaded Broadband Test (RMBT): Specification*. June 2017. URL: <https://www.netztest.at/doc/>.
- [4] Michael Rindler et al. “Swift indoor benchmarking methodology for mobile broadband networks.” In: *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. IEEE. 2017, pp. 1–5.
- [5] Valentin Platzgummer et al. “UAV-Based Coverage Measurement Method for 5G.” In: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE. 2019, pp. 1–6.
- [6] 3GPP. “LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer; Measurements (3GPP TS 36.214 version 10.1.0 Release 10).” In: *Etsi Ts 136 214 0* (2011), pp. 1–15.
- [7] Scikit-learn developers. *Gaussian Processes regression: basic introductory example*. URL: [https://scikit-learn.org/stable/auto\\_examples/gaussian\\_process/plot\\_gpr\\_noisy\\_targets.html](https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_noisy_targets.html).
- [8] Naren Ramakrishnan and Chris Bailey-Kellogg. “Gaussian Process Models in Spatial Data Mining.” In: *Encyclopedia of GIS*. Ed. by Shashi Shekhar and Hui Xiong. Boston, MA: Springer US, 2008, pp. 325–329. ISBN: 978-0-387-35973-1. DOI: [10.1007/978-0-387-35973-1\\_440](https://doi.org/10.1007/978-0-387-35973-1_440). URL: [https://doi.org/10.1007/978-0-387-35973-1\\_440](https://doi.org/10.1007/978-0-387-35973-1_440).

## Bibliography

---

- [9] Peter Bruce and Andrew Bruce. *Practical statistics for data scientists: 50 essential concepts*. " O'Reilly Media, Inc.", 2017.
- [10] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.
- [11] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.
- [12] Rocco Di Taranto et al. "Location-aware communications for 5G networks: How location information can improve scalability, latency, and robustness of 5G." In: *IEEE Signal Processing Magazine* 31 (2014), pp. 102–112.
- [13] Qi Liao, Stefan Valentin, and Slawomir Stanczak. "Channel gain prediction in wireless networks based on spatial-Temporal correlation." In: *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC 2015-August* (2015), pp. 400–404.
- [14] Haitao Liu et al. "When Gaussian Process Meets Big Data: A Review of Scalable GPs." In: *IEEE Transactions on Neural Networks and Learning Systems* (2020), pp. 1–19.
- [15] Christian Dussault et al. "Influence of satellite geometry and differential correction on GPS location accuracy." In: *Wildlife Society Bulletin* (2001), pp. 171–179.
- [16] L Srikar Muppirisetty, Tommy Svensson, and Henk Wymeersch. "Spatial wireless channel prediction under location uncertainty." In: *IEEE Transactions on Wireless Communications* 15.2 (2015), pp. 1031–1044.
- [17] Franz Hlawatsch. "Gaussian Process Regression." In: *J96-D* (2017), pp. 3068–3078.
- [18] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.
- [19] Shengyang Sun et al. "Differentiable compositional kernel learning for Gaussian processes." In: *arXiv preprint arXiv:1806.04326* (2018).
- [20] Chiranjib Saha, Mehrnaz Afshang, and Harpreet S. Dhillon. "3GPP-Inspired HetNet Model Using Poisson Cluster Process: Sum-Product Functionals and Downlink Coverage." In: *IEEE Transactions on Communications* 66 (2018), pp. 2219–2234.



- [21] Charu C Aggarwal and Chandan K Reddy. *Data Clustering: Algorithms and Applications*, ser. 2013.
- [22] CISCO. *RTR Channels and Technologies*. May 2020. URL: <https://www.rtr.at/en/tk/Channels>.
- [23] Google Maps. *Drone measurement area in Klingenbach, Austria*. May 2020. URL: <https://www.google.com/maps/place/47%C2%B046'03.4%22N+16%C2%B031'49.4%22E/@47.7678945,16.5303032,293a,35y,330h/data=!3m1!1e3!4m5!3m4!1s0x0:0x0!8m2!3d47.7676!4d16.5304>.
- [24] Paul Keeler. *Simulating a Thomas cluster point process*. May 2020. URL: <https://hpaulkeeler.com/simulating-a-thomas-cluster-point-process/>.
- [25] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [26] Lars Buitinck et al. “API design for machine learning software: experiences from the scikit-learn project.” In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.