May 2023

Modeling WLAN Received Signal Strengths Using Gaussian Process Regression on the Sodindoorloc Dataset

Fabian Hermann Josef Fuchs
University of Wisconsin-Milwaukee

MODELING WLAN RECEIVED SIGNAL STRENGTHS USING GAUSSIAN PROCESS REGRESSION ON THE SODINDOORLOC DATASET

by

Fabian Fuchs

A Thesis Submitted in
Partial Fulfillment of the
Requirements for the Degree of

Master of Science
in Mathematics

at
The University of Wisconsin-Milwaukee
May 2023

ABSTRACT

MODELING WLAN RECEIVED SIGNAL STRENGTHS USING GAUSSIAN PROCESS REGRESSION ON THE SODINDOORLOC DATASET

by

Fabian Fuchs

The University of Wisconsin-Milwaukee, 2023
Under the Supervision of Professor David Spade

While any wireless technology can be used for indoor localization purposes, *WLAN* has the advantage of having a huge existing infrastructure.

A radio map that matches specific locations to received signal strength is needed, to enable most of these indoor localization methods. To create these radio maps, with enough detail to achieve sufficient localization accuracy, is expensive and time consuming. Therefore, methods to interpolate and extrapolate more detailed maps from sparse radio maps are being developed.

One recent approach is to use Gaussian process regression. Even though some papers already studied Gaussian process regression, most studied only the basic model with zero mean and squared exponential kernel. In addition, when the model fit was evaluated in more detail, the experimental area was of limited complexity.

Hence, this thesis evaluates the fit of Gaussian process regression, in a more complex indoor environment, based on adequate model metrics and analysis of the plots of the predicted mean and standard deviation functions. As a conclusion, the most suitable model is presented, as well as the reasoning why it was chosen.

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

Indoor positioning systems (*IPS*) are not only a growing research topic but also an increasing market. While theoretically any wireless technology can be used for localization purposes, *WLAN* has the advantage of already being widely used and therefore having an existing infrastructure.

Although there are a lot of different wireless localization techniques, as seen in [11], a lot of these require a radio map that matches specific locations to received signal strength (*RSS*). The process of creating these radio maps in sufficient detail is expensive and time consuming. Therefore, sparse radio maps are generated instead and then interpolated and extrapolated.

One more recent approach for this is to use Gaussian process regression as shown in [14]. Even though other papers researched Gaussian process regression to model *WLAN* received signal strengths before, a lot of them [16] [7] [6] [1] just considered the standard model with zero mean and a squared exponential covariance function. When different Gaussian processes were studied to find the most appropriate model [14], the experiment area was of limited complexity. Even though one of the most interesting features of *WLAN* based positioning is that it can be used indoors, the experiment area of [14] was mostly outdoors and they described the two buildings in it as having outdoor-like properties. Consequently, this thesis tries to build upon [14] and test Gaussian process regression in a more complex indoor environment. Additionally, some modifications of the process are proposed and evaluated.

1.1 Dataset

This thesis uses a subset of the *SODIndoorLoc* [4] data set. It was chosen because it provides dense and uniformly distributed reference points with an average distance between adjacent ones of less than 1.2 m. The original data set covers three buildings from three different universities in separate cities, covering a total area of about 8000 square meters. The chosen subset focuses on only one of these buildings, namely the *SYL* building of the Shandong Jianzhu University in Jinan, China. This building was chosen because it has the second largest measured area of 2600 m^2 and has a more complex layout in comparison to the building with the largest measured area, which consists only of corridors. In contrast, the former has office rooms, meeting rooms, and corridors. A blueprint of the fourth floor, where the measurements took place, is given in Figure 1. Of special interest is the colored section because it is the area where the training and test points are located.

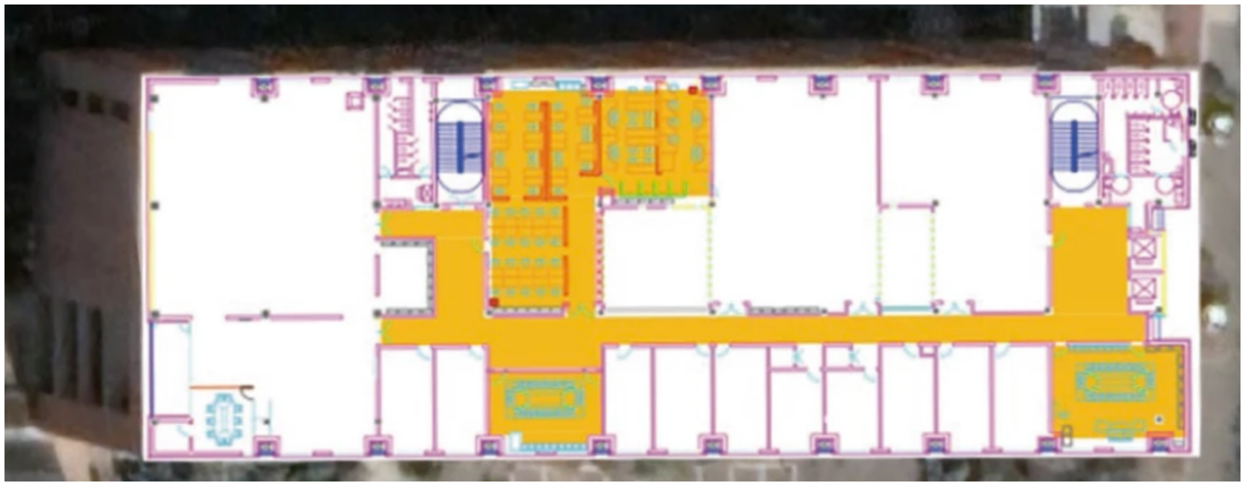


Figure 1: Building plan of the *SYL* Building. The training and test points are in the orange area.

Distributed on the floor are 296 different training points, visible in blue in Figure 2. For each training point 30 measurements took place, for a total of 8880 training measurements. Additionally, there are 102 different test points, visible in red. For each of these points 10 measurements were taken for a total of 1020 test measurements. Not only are the training and test points visible in the scatter plot but also the colored area of the floor plan in Figure 1 is recognizable.

Each measurement consists of the location of the point and the received signal strength for each of the 46 access points (*AP*s). These 46 *APs* are originally 23 dual band *AP*s. That means that every individual *AP* functions simultaneously in the 2.4GHz frequency band and in 5GHz frequency one. The distinction is important because while a 5GHz connection is

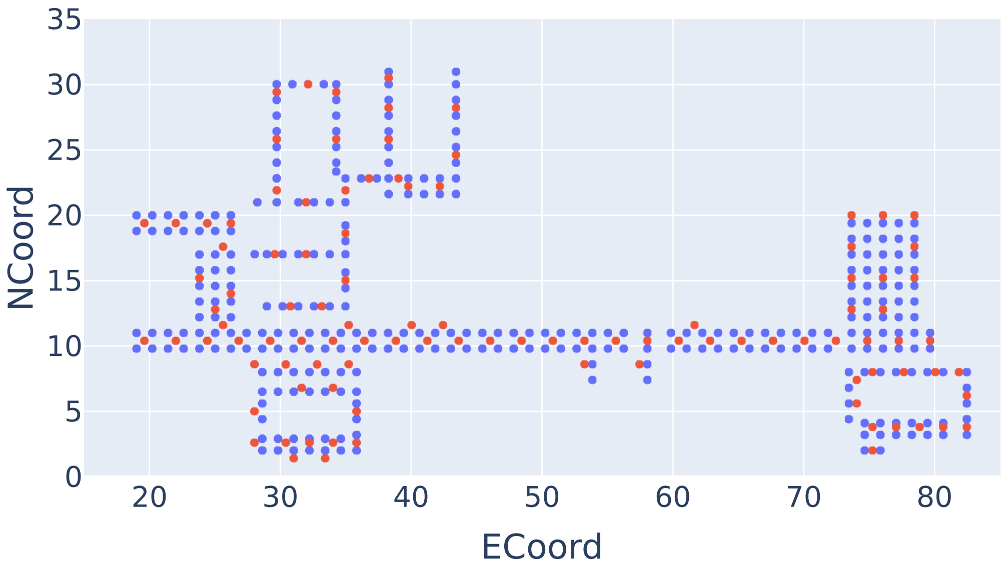


Figure 2: Scatter plot of the test points in red and training points in blue.

faster, it has a lower range. Therefore, the measured RSS can differ between the frequency bands, even if the physical location of the APs is the same. The locations of the APs is displayed in Figure 3.

Due to the distinct locations and bandwidths of the APs, each one reaches a different number of training and test points. The AP with the best coverage, (around 85%) is *MAC56*, which is centrally located in Figure 3 and uses the 5GHz bandwidth. Its 2.4GHz counterpart on the other hand only covers 45 percent of the points. The worst coverage has *MAC340*, where not even all 30 measurements for one training point were successful. Meanwhile the corresponding 2.4GHz AP still connects to around 30 percent of points.

The distribution of the different APs and their unsuccessful measurements, indicated with a value of 100 in the data set, is shown in Figure 4. On the left , Figure 4a is the histogram of the training measurements. On the right, Figure 4b gives the histogram for the test measurements. It can be seen that the distribution is similar, as expected, because the locations of the test and training points share their distribution. Most APs failed between

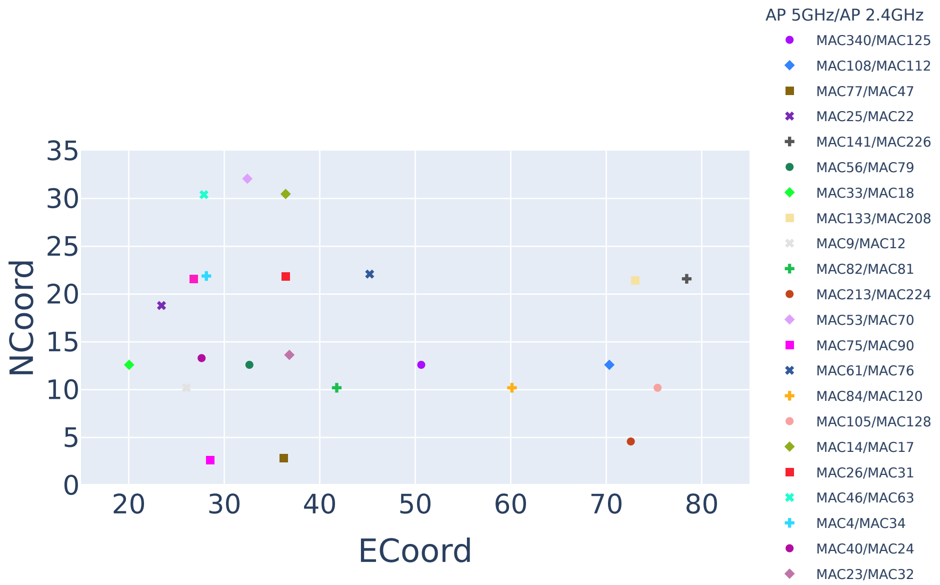
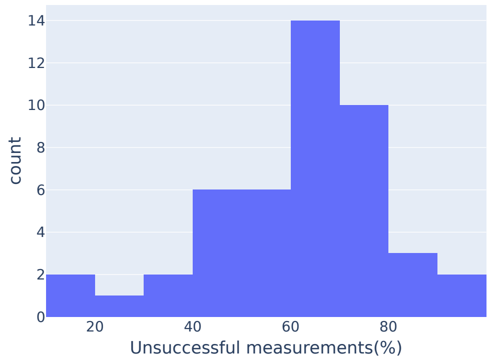
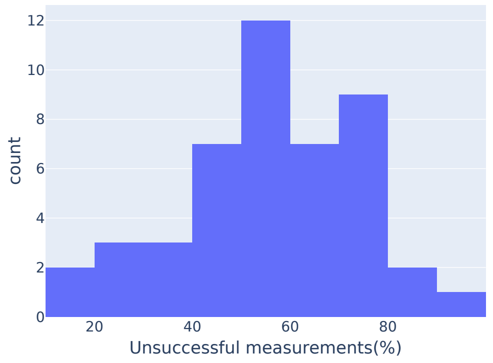


Figure 3: Scatter plot of the *AP* locations.

40 and 80 percent of their measurements which means they were successful in between 20 and 60 percent of cases.



(a) Histogram of the number of APs and their unsuccessful training measurements



(b) Histogram of the number of APs and their unsuccessful test measurements

Figure 4: Histogram of the number of APs and their unsuccessful measurements.

2 Introduction to Gaussian Process Regression

This chapter reviews the Bayesian linear regression model in section 2.1 and introduces the Gaussian regression model in section 2.2.

2.1 Bayesian Linear Model

The standard linear regression model with Gaussian noise has the following formula $f(\vec{x}) = \vec{x}^T\vec{w}$, where $\vec{x}$ is the input vector and $\vec{w}$ a vector of weights. The model assumes that the observations are defined as $y_i = f(\vec{x_i}) + \epsilon_i$ with the noise identically and independently Gaussian distributed with mean 0 and variance σ_n^2 [13]. The distribution of the observations $\vec{y}|X, \vec{w}$ is $\mathcal{N}(X^T\vec{w}, \sigma_n^2 I)$ and follows directly from the definition of the model and the noise assumptions, see equation (1). The matrix X has d rows and n columns, where d is the length of the input and weight vectors, and n is the number of training examples.

$$\begin{aligned} p(\vec{y}|X,\vec{w}) &= \prod_{i=1}^{n} p(y_i|\vec{x_i},\vec{w}) \\ &= \prod_{i=1}^{n} \frac{1}{\sigma_n\sqrt{2\pi}} \exp\left(-\frac{(y_i - \vec{x_i}^T\vec{w})^2}{2\sigma_n^2}\right) \\ &= \left(\frac{1}{\sigma_n\sqrt{2\pi}}\right)^n \exp\left(-\frac{\|\vec{y} - X^T\vec{w}\|_2^2}{2\sigma_n^2}\right) \end{aligned} \tag{1}$$

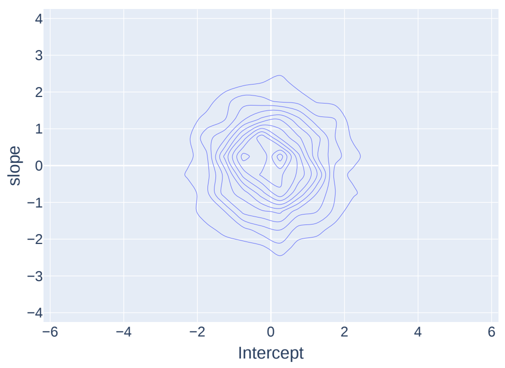
The Bayesian part of the model is that the parameters have a prior which enables inference. In this case the prior of the weight vector $\vec{w}$ is a Gaussian distribution with mean $\vec{0}$ and variance Σ_p. Then the posterior distribution can be calculated with Bayes‘ theorem, see equation (2). The marginal likelihood $p(\vec{y}|X)$, can be calculated by integrating the likelihood $p(\vec{y}|X,\vec{w})$ over the weights $\vec{w}$.

$$p(\vec{w}|\vec{y},X) = \frac{p(\vec{y}|X,\vec{w})p(\vec{w})}{p(\vec{y}|X)}, \qquad p(\vec{y}|X) = \int p(\vec{y}|X,\vec{w})p(\vec{w})d\vec{w} \tag{2}$$

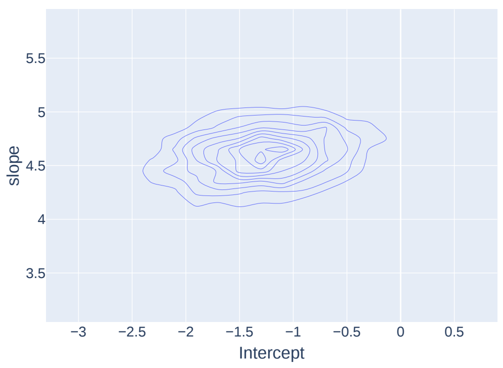
Considering only the parts of the posterior that depend on the weights, it is possible to calculate its conditional distribution, see equation (3).

$$
\begin{aligned}
p(\vec{w}|\vec{y},X) &= \frac{p(\vec{y}|X,\vec{w})p(\vec{w})}{p(\vec{y}|X)} \\
&\propto \exp\left(-\frac{1}{2\sigma_n^2}(\vec{y}-X^T\vec{w})^T(\vec{y}-X^T\vec{w})\right)\exp\left(-\frac{1}{2}\vec{w}^T\Sigma_p^{-1}\vec{w}\right) \\
&\propto \exp\Bigg(-\frac{1}{2}(\vec{w}-(\sigma_n^{-2}(\sigma_n^{-2}XX^T+\Sigma_p^{-1})^{-1}X\vec{y}))^T \\
&\qquad (\sigma_n^{-2}XX^T+\Sigma_p^{-1})\,(\vec{w}-(\sigma_n^{-2}(\sigma_n^{-2}XX^T+\Sigma_p^{-1})^{-1}X\vec{y}))\Bigg) \\
\Rightarrow \vec{w}|\vec{y},X &\sim \mathcal{N}\left(\sigma_n^{-2}(\sigma_n^{-2}XX^T+\Sigma_p^{-1})^{-1}X\vec{y},(\sigma_n^{-2}XX^T+\Sigma_p^{-1})^{-1}\right)
\end{aligned}
\tag{3}
$$

The difference between the prior distribution and posterior distribution is given in Figure 5, where the left plot 5a shows a Gaussian prior distribution with mean $\vec{0}$ and variance of I and the right plot 5b shows the posterior distribution after incorporating the training data. The prior distribution should be circles, but due to a limited sample size of 3000 the circles are crooked. However, the difference between the prior and the posterior distribution is still apparent, because the posterior is more of an ellipse, with a better defined slope than intercept.



(a) Contours of the prior distribution $\vec{w} \sim \mathcal{N}(\vec{0}, I)$.



(b) Contours of the posterior distribution $\vec{w}|\vec{y}, X$.

Figure 5: Comparison of prior and posterior distribution.

By averaging over all possible parameters, weighted by their posterior distribution, the predictive distribution can be computed, see equation (4).

$$\begin{aligned} &p(f(\vec{x_*})|\vec{y}, X, \vec{x_*}) = \int p(f(\vec{x_*})|\vec{x_*}, \vec{w})p(\vec{w}|\vec{y}, X)d\vec{w} \\ &\Rightarrow f(\vec{x_*})|\vec{y}, X, \vec{x_*} \sim \mathcal{N}\left(\sigma_n^{-2}\vec{x_*}^T(\sigma_n^{-2}XX^T + \Sigma_p^{-1})^{-1}X\vec{y}, \vec{x_*}^T(\sigma_n^{-2}XX^T + \Sigma_p^{-1})^{-1}\vec{x_*}\right) \end{aligned} \tag{4}$$

With the predictive distribution it is possible to predict the target value for new features which can be seen in Figure 6. The red line in the plot is the true regression line, with an intercept of -1 and a slope of 5. The red dots are the noisy observations which were generated by adding a Gaussian distribution with mean 0 and variance 1 to the noise free targets. The blue line shows the predicted regression line, by taking the mean of the predicted values with a sample size of 3000. Additionally, a confidence interval using the mean plus minus to times the standard deviation is represented by the light blue shaded area. To implement the Bayesian linear regression model, the python library *PyMC* [15] was used.

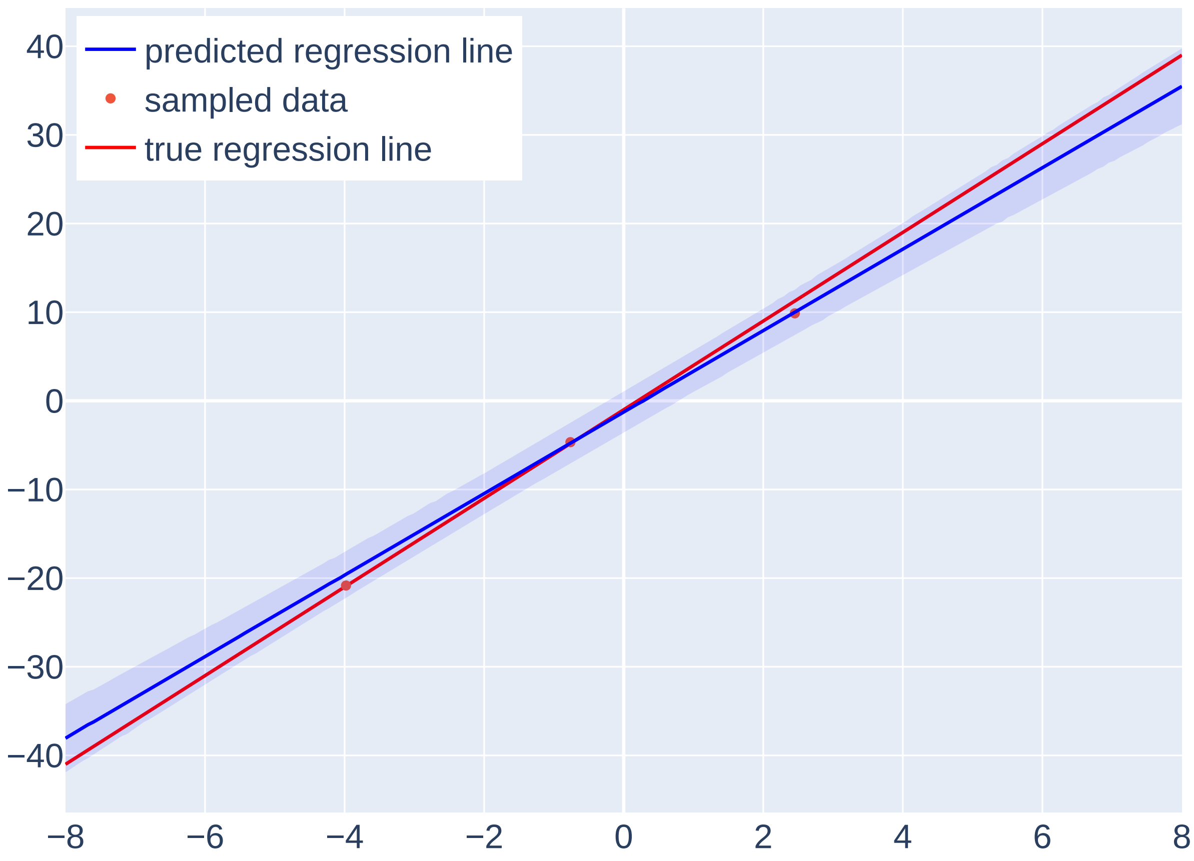


Figure 6: True regression line, sample data, predicted regression line with a confidence area of two standard deviations.

2.2 Gaussian Process Regression Model

Instead of predicting the target values $\vec{y}$, it is also possible to directly predict the function $f(x)$ by using a stochastic process to describe a distribution over a function space. Of particular interest are Gaussian processes (GPs), because Gaussian distributions are closed under conditioning, which means that the result of conditioning on a combined Gaussian distribution is still Gaussian.

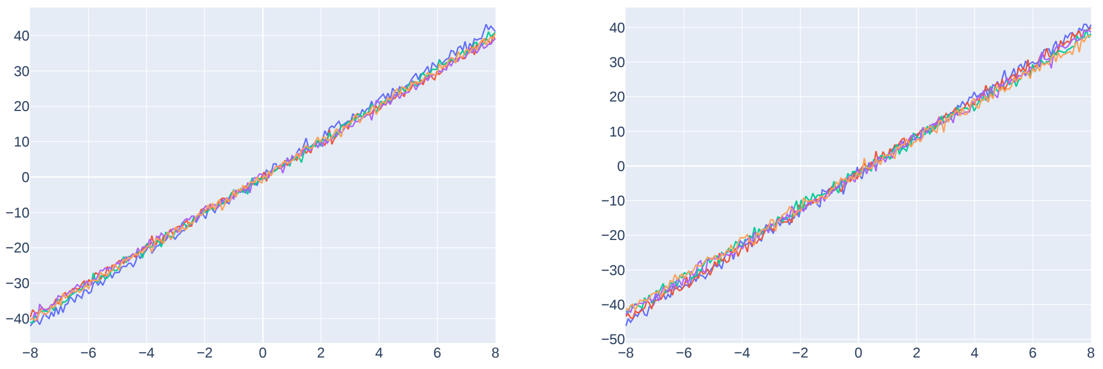
While Gaussian distributions are completely defined by their mean and variance, Gaussian processes are uniquely determined by their mean function $m(\vec{x})$ and covariance function $k(\vec{x}, \vec{x'})$, see equation (5).

$$\begin{aligned} m(\vec{x}) &= \mathbb{E}[f(\vec{x})] \\ k(\vec{x}, \vec{x'}) &= \mathbb{E}[(f(\vec{x}) - m(\vec{x}))(f(\vec{x'}) - m(\vec{x'}))] \end{aligned} \tag{5}$$

Noisy observations in the GP model are defined as $y_i = f(\vec{x_i}) + \epsilon_i$, where $f(x)$ is the GP defined by a its mean and covariance function ($f(\vec{x}) \sim \mathcal{GP}(m(\vec{x}), k(\vec{x}, \vec{x'}))$) and the noise is independently and identically Gaussian distributed ($\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma_n^2)$). Due to the marginalization property of stochastic processes the likelihood of $\vec{y}|X, f$ is a Gaussian likelihood, i.e. ($\vec{y}|X, f \sim \mathcal{N}(f(\vec{X}), \sigma_n^2 I)$), with mean $f(\vec{X})$ and variance $\sigma_n^2 I$. To calculate the marginal likelihood the likelihood is integrated over the function space f, similar to integrating over the parameter space $\vec{w}$ in the Bayesian model. Also related is the computation of the posterior distribution with Bayes theorem, see equation (6).

$$\begin{aligned} p(f|\vec{y}, X) &= \frac{p(\vec{y}|X, f)p(f)}{p(\vec{y}|X)} \\ f|\vec{y}, X &\sim \mathcal{GP}(m_{posterior}(\vec{x}), k_{posterior}(\vec{x}, \vec{x'})) \end{aligned} \tag{6}$$

With the posterior distribution, which is a GP, it is possible to predict functions that would fit the training data. See Figure 7, where the left plot, Figure 7a, shows 5 samples from the prior Gaussian process. The right plot, Figure 7b, shows 5 samples of the posterior distribution after 30 training iterations of a GP model with constant mean function and a


(a) Samples of the prior Gaussian process.

(b) Samples of the posterior Gaussian process.

Figure 7: Comparison of prior and posterior Gaussian processes.

linear covariance function. The model was created with the python library *GPyTorch* [8].

Instead of predicting some random functions that fit the training data, it is often preferable to predict the targets $f(X_*)$ of new features X_*. This is possible by calculating the predictive distribution. To do this, the first step is setting up the joint distribution, given in equation (7).

$$\begin{bmatrix} \vec{y} \\ f(\vec{X}_*) \end{bmatrix} \sim \mathcal{N}\left(\vec{0}, \begin{pmatrix} k(X,X) + \sigma_n^2 I & k(X,X_*) \\ k(X_*,X) & k(X_*,X_*) \end{pmatrix}\right) \tag{7}$$

Then, as discussed in the beginning, the conditional of a joint Gaussian distribution is still Gaussian distributed, which results in the predictive distribution, defined in equation (8).

$$\begin{aligned} f|X,\vec{y},X* &\sim \mathcal{N}\left(\overline{f(X_*)}, cov(f(X_*))\right) \\ \overline{f(X_*)} &\triangleq \mathbb{E}[f|X,\vec{y},X*] = k(X_*,X)\left(k(X,X)+\sigma_n^2 I\right)^{-1}\vec{y} \\ \text{cov}(f(X_*)) &= k(X_*,X_*) - k(X_*,X)\cdot\left(k(X,X)+\sigma_n^2 I\right)^{-1}k(X,X_*) \end{aligned} \tag{8}$$

With the predictive distribution it is possible to predict target values for new features, as shown in Figure 8. This plot is similar to the plot in Figure 6. In both plots the red line shows the true regression line, the red dots the sampled training data, the blue one the

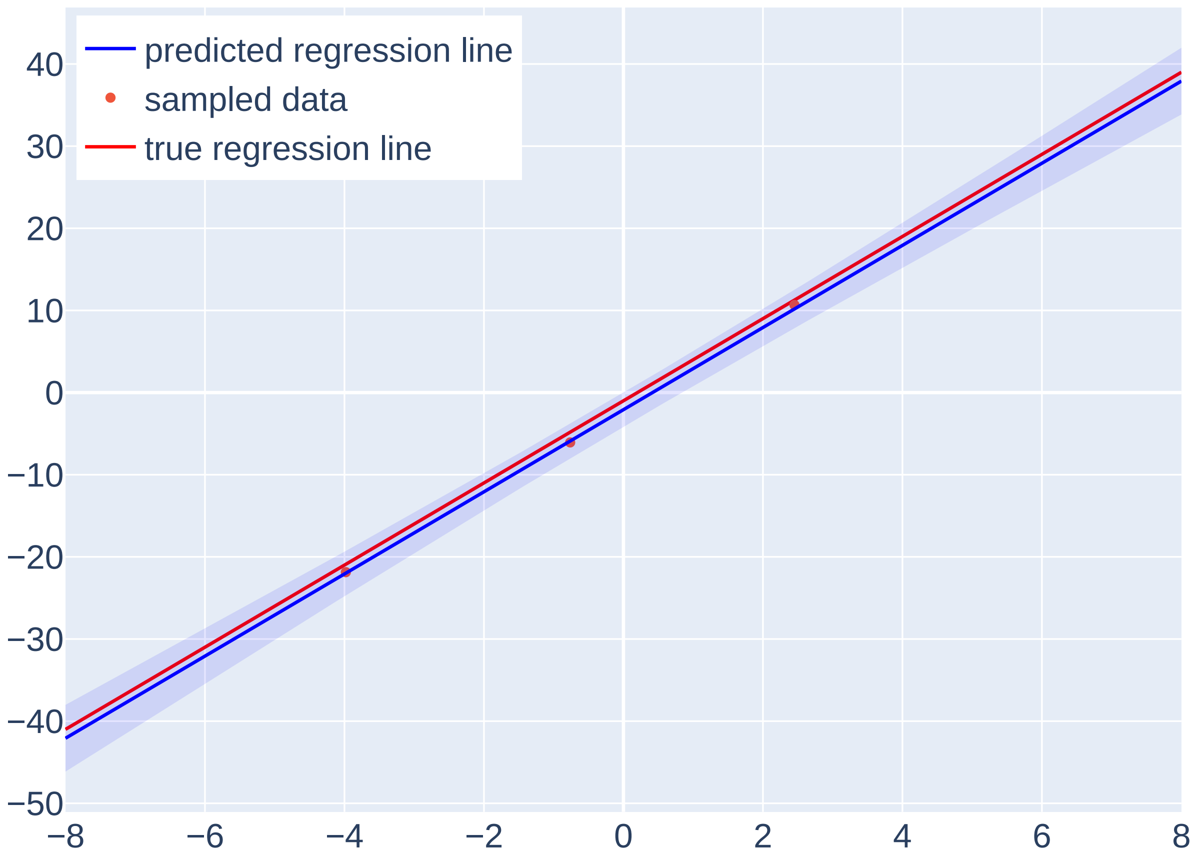


Figure 8: True regression line, sample data, predicted regression line with a confidence area of 2 standard deviations.

predicted regression line and the light blue shaded area the predicted regression line plus and minus two standard deviations. Clearly visible in plot 8 is that the variance is lower in the middle, where the training data is located, and increases at both sides, which was also discernible under the Bayesian model.

3 Model Selection

This chapter discusses the model selection procedure. Section 3.1 considers the paper *Revisiting Gaussian Process Regression Modeling for Localization in Wireless Sensor Networks* [14] and its model selection process. Furthermore, it explains what compromises model selection for Gaussian processes and elaborates about the model that was finally chosen. Section 3.2 explores the optimization process of that model with different hyper-parameters and the training and test scores. Finally, section 3.3 looks at and compares plots of the predicted mean and predicted standard deviations of the fitted models.

3.1 Possible Models

As discussed in chapter 2.2, a Gaussian process regression model is completely defined by its mean and covariance function. The paper [14] considered the following basic covariance functions and their combinations: the squared exponential, the Matérn with three different values (0.5, 1.5, 2.5) of its smoothness parameter v, the rational quadratic, and the independent noise kernel. Furthermore, three different mean functions: the zero-mean, the constant mean, and the linear mean, were assessed.

The conclusion of the paper is that the most suitable and general model consists of a constant mean function combined with a *Matérn kernel*, especially a *Matérn kernel* with smoothness parameter v equal to 1.5. The formula for this covariance function is given in equation (9), where K_v is a modified Bessel function of the second kind [13] and Θ is an additional parameter, called the length-scale. Both parameters are positive and a Gaussian process with *Matérn kernel* is $\lceil v \rceil - 1$ times differentiable.

$$k_{Matérn}^{v,\Theta}\left(\vec{x}, \vec{x`}\right) = \frac{2^{1-v}}{\Gamma(v)} \left(\sqrt{2v} \cdot d\left(\vec{x}, \vec{x`}\right)\right)^{v} K_v\left(\sqrt{2v} \cdot d\left(\vec{x}, \vec{x`}\right)\right) \tag{9}$$

$$d^{\Theta}\left(\vec{x}, \vec{x`}\right) = \left(\vec{x} - \vec{x`}\right)^{T} \Theta^{-2} \left(\vec{x} - \vec{x`}\right) \tag{10}$$

Even though the general formula of the *Matérn* covariance function (9) uses the modified Bessel function K_v, in practice it is not. Since when v is a half integer, $v = d + 0.5, d \in \mathbb{N}$, the *Matérn kernel* can be written as a combination of a polynomial function of order d and an exponential function, and the Bessel function vanishes. The specific formulas for the *Matérn kernel* with the three considered values for v are given in equation (11).

$$
\begin{aligned}
k_{Matérn}^{0.5,\Theta}\left(\vec{x}, \vec{x'}\right) &= \exp\left(-\sqrt{d\left(\vec{x}, \vec{x'}\right)}\right) \\
k_{Matérn}^{1.5,\Theta}\left(\vec{x}, \vec{x'}\right) &= \left(1 + \sqrt{3} \cdot \sqrt{d\left(\vec{x}, \vec{x'}\right)}\right) \exp\left(-\sqrt{3} \cdot \sqrt{d\left(\vec{x}, \vec{x'}\right)}\right) \\
k_{Matérn}^{2.5,\Theta}\left(\vec{x}, \vec{x'}\right) &= \left(1 + \sqrt{5} \cdot \sqrt{d\left(\vec{x}, \vec{x'}\right)} + \frac{5}{3} \cdot d\left(\vec{x}, \vec{x'}\right)\right) \exp\left(-\sqrt{5} \cdot \sqrt{d\left(\vec{x}, \vec{x'}\right)}\right)
\end{aligned}
\tag{11}
$$

Also of note is, that for $v \to \infty$ the *Matérn kernel* converges to the *squared exponential kernel*, which appears to be the second best kernel in [14]. The formula for the *squared exponential kernel* is given in equation (12). A *GP* with it as covariance function is infinitely differentiable.

$$
k_{SE}^{\Theta}\left(\vec{x}, \vec{x'}\right) = \exp\left(-\frac{1}{2} \cdot d\left(\vec{x}, \vec{x'}\right)\right) \tag{12}
$$

Notice, that all the above mentioned equations for the covariance functions do not include a scaling parameter θ_{scale}. This is because, instead of adding the scale parameter directly to the individual kernels, a special *ScaleKernel*, that is used in combination with the *Matérn* or *squared exponential kernel*, is defined in equation (13).

$$
k_{scale}^{\theta_{scale}}\left(\vec{x}, \vec{x'}\right) = \theta_{scale} \cdot k_{original}\left(\vec{x}, \vec{x'}\right) \tag{13}
$$

Even though [14] concluded $v = 1.5$ to best the best overall smoothness parameter, it should be distinguished between different circumstances. For the joined radio map $v = 0.5$

turned out to be the best. This could be because in order to deal with the differences between how the signal travels indoor and outdoor, a rougher function is needed. On the other hand, the best value of v for the indoor map in [14] is 2.5. Considering that they mentioned that the building has outdoor like properties regarding to the WiFi signal propagation due to the building having soft partitions and open windows most of the time, a smoother function would make sense. In contrast, $v = 0.5$ could be a better fit for the radio map of the 4th floor of the *SYL* building, due to its more complex layout. Therefore, this thesis reconsiders the three examined values of (0.5, 1.5, 2.5) for v.

The book [13] recommends the values 1.5 and 2.5 for the smoothness parameter v, due to a value of 0.5 leading to a very rough process. In addition, it explains why these three values were the only ones considered for v. Namely because from finite noisy training examples a value greater than 3.5 is hard to distinguish from a value that converges to infinity, ergo the *squared exponential kernel*.

One additional hyper-parameter that exists due to the problem having a two-dimensional input that the prior mentioned paper [14] did not test, is *ard_num_dims* [8], which controls the number of length-scales parameters of the kernel. The length-scale parameter influences the smoothness of the fitted function and the range of reliable extrapolation. A small value means the function can change relatively quickly and a large value results in slow changes. In contrast to the number of length-scales the length-scale itself is learnable from the data and therefore not part of the model selection process. The possible values for *ard_num_dims* in this case are 1, which would mean the same length-scale for the x and y directions or 2 which would result in different length-scales.

Overall, the constant mean function performed better than the zero-mean function in [14], which fits the reality of the data, because the mean of an *RSS* function should never be zero. While an *RSS* value of 0 is theoretically possible, in practice the measured range of *RSS* is between -30 and -90.

Meanwhile, the linear mean function is not considered because it not only adds an additional hyper-parameter, but also because as [14] explains, a linear mean function can only approximate one slope well and the others poorly. The constant mean function on the other hand only assumes an eventual convergence to that constant value, which again fits the reality of the circumstance because in theory every point in space that is outside the detection range of an *AP* should have an *RSS* less than -120, which is the lowest possible *RSS* value.

Therefore, the only models left to consider are *GP*s with the *Matérn kernel* and its different parameter combinations combined with the constant mean function.

3.2 Optimization and Results

Hence, the remaining question is what values to choose for the non-learnable hyper-parameters v and *ard_num_dims* of the *Matérn kernel*. For that reason, each possible combination of these was tested on all 46 *AP*s and individually optimized.

The *Adam* [10] optimizer, which is a stochastic gradient descent algorithm that computes individual learning rates for different parameters, was used with 100 training iteration and a learning rate of 0.1. The algorithm was initialized with the mean value of the training targets for the constant mean parameter, and zeros for the rest of the parameters.

After optimization, the test scores, consisting of the *Bayesian information criterion* (*BIC*) and the *root mean square error* (*RMSE*), were calculated. Additionally, the R^2 score was calculated on the training data. The formula for the *BIC* is given in equation (14) [12], where k is the number of parameters of the Gaussian process model. The *BIC* is a model selection measure, where lower values are better similar to the *RMSE*. It is based on the likelihood of the predictions with an added penalty term for the number of parameters k to prevent over-fitting.

$$BIC(\vec{y}, X, f) = -2\log(p(\vec{y}|X, f)) + k \cdot log(n) \tag{14}$$

The averages over almost all the *AP*s were taken and the results listed in Table 1. Since the optimization of *MAC340* resulted in an R^2 over 1, which signals a failure, this *AP* had to be left out. This result though is not unexpected, because as described in section 1.1 *MAC340* does not even reliably cover one training point and therefore has fewer than 30 training measurements. Excluding that *AP*, the means of the remaining ones lead to some interesting discoveries.

First, it appears that the value of v has more influence than the number of length-scales at least one the $RMSE$.

Second, because having two length-scales adds one additional parameter, the BIC of the models with one length-scale is considerably less than of the ones with two.

ard_num_dims	v	$RMSE$	BIC	R^2
1	0.5	7.145231	24.679359	0.978565
1	1.5	7.571736	26.370824	0.978437
1	2.5	7.721295	27.124973	0.978422
2	0.5	7.139899	30.691593	0.978569
2	1.5	7.559488	32.390402	0.978451
2	2.5	7.710614	33.134540	0.978431

Table 1: Mean scores.

Third, while the R^2 scores of the different models are close they reflect the same ranking as ordering the models by $RMSE$. Namely (2, 0.5), (1, 0.5), (2, 1.5), (1, 1.5), (2, 2.5), and (1, 2.5), where the combination with two length-scales is better with the same smoothness parameter. But because the differences between the $RMSE$s are so slight the ranking with regard to the BIC is considerably different. All models with one length-scale are better than the ones with two and a higher value of v leads to a higher BIC.

Looking at the worst score for each parameter combination instead, see Table 2, shows that except for *MAC340* the chosen models work reasonably well for all the other *AP*s. Furthermore, the worst scores reflect the trends of the average scores. For example, the maximum $RMSE$ and the minimum R^2 scores of the models have the same order than the mean scores, with the only exception that the model with two length-scales and v equal to 2.5 has a higher $RMSE$ than its counterpart with one length-scale. If the models are ranked by the BIC instead, the order is also similar except that this time the worst model with one length-scale has a higher BIC than the best one with two length-scales.

ard_num_dims	v	$RMSE$	BIC	R^2
1	0.5	12.215887	29.460987	0.931803
1	1.5	12.603783	34.156258	0.931385
1	2.5	12.709380	36.287029	0.931362
2	0.5	12.183849	35.358818	0.931851
2	1.5	12.590466	40.029758	0.931526
2	2.5	12.729073	42.082603	0.931371

Table 2: Maximum scores.

Finally, the standard deviations of the different test scores, given in Table 3, are acceptably small, which means that the GP model with constant mean and a *Matérn* covariance function in general is robust against different amounts of training data. This is one advantage of Gaussian process regression in general. It specializes in smaller data regimes and is robust against over-fitting even when given little data. Furthermore, this Table shows the superiority of the model with one length-scale and v equal to 0.5 again, because it has the smallest standard deviation of BICs and the second smallest for the $RMSE$s and R^2 scores.

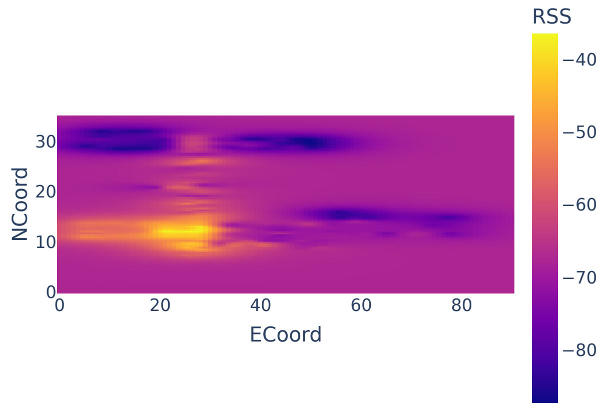
ard_num_dims	v	$RMSE$	BIC	R^2
1	0.5	1.708749	2.148543	0.014306
1	1.5	1.769104	2.898622	0.014356
1	2.5	1.787766	3.298446	0.014358
2	0.5	1.695102	2.371022	0.014278
2	1.5	1.758351	3.041898	0.014347
2	2.5	1.780547	3.399242	0.014354

Table 3: Standard deviation of the scores.

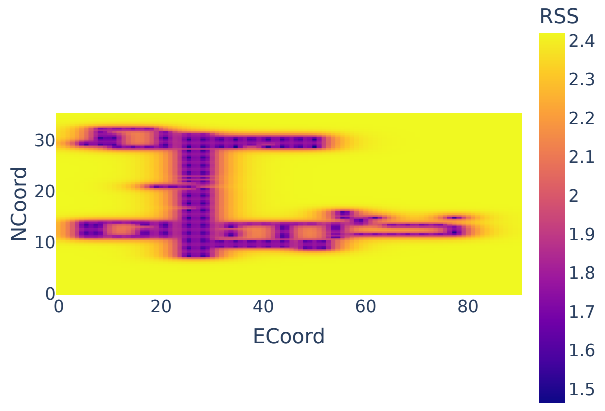
In conclusion the model with one length-scale and v equal to 0.5 appears to be the best fit for the data, due to having the lowest mean BIC as well as almost achieving the lowest $RMSE$ and highest R^2. Moreover, the worst scores and the standard deviations follow the same trend as the mean scores, with the mentioned model being the best in regard to the BIC and almost being the best in regard to the other two scores.

3.3 Graphical Results

Looking at plots of the predicted means and predicted standard deviations confirms these conclusions. All the plots in this chapter were generated for *MAC56*, with all the different value combinations for the length-scale and smoothness parameter. The first two plots in Figure 9, show the predicted means and standard deviations for a $v = 0.5$ and one length scale.



(a) Predicted mean values for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.

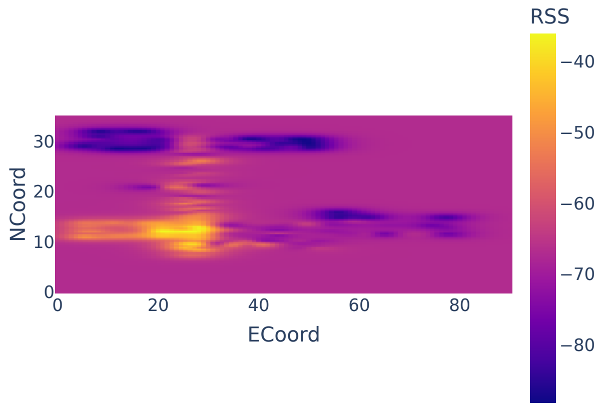


(b) Predicted standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.

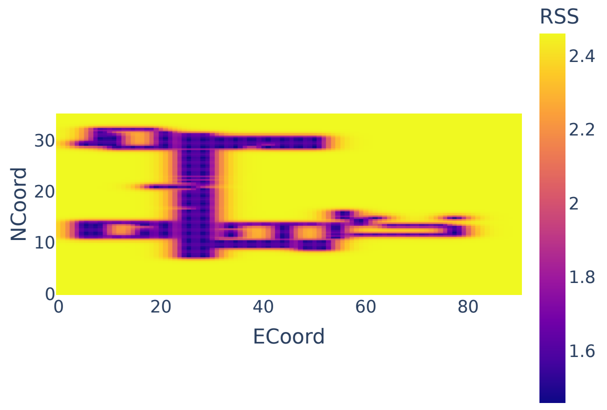
Figure 9: Predicted means and standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.

On the left in Figure 9a the predicted means are shown. Noticeable is the yellow region in the lower left section, which corresponds roughly to the location of *MAC56*, like one would expect. Additionally, we see yellow regions transitioning to purple and then blue, along the horizontal and vertical hallway, where the signal travels relatively uninterrupted. The two rooms in the top left section are shaded blue, which indicates a rather bad signal. In contrast the area outside the measured region is colored purple, which corresponds to values around -70, which is better than some values predicted in the top left and bottom right region. This is interesting, because the *AP* probably does not even reach some of these areas, like for example the top right corner, which should therefore have a way lower predicted *RSS*. This is due to the constant mean function the current model uses, and removing the 100 values in the dataset, that indicate an unsuccessful connection.

On the right plot 9b the standard deviations are shown. The difference between the areas, where the model has training data and where the model does not, is evident. For the first case the standard deviations are around 1.5, indicated in purple, and for the second case the standard deviations are around 2.5, indicated in yellow. The higher standard deviations reflect the uncertainty of the model regarding these points, which makes sense, because it has no training data for them and therefore has to extrapolate.



(a) Predicted mean values for *MAC56* with $v = 1.5$ and $ard\_num\_dims = 1$.

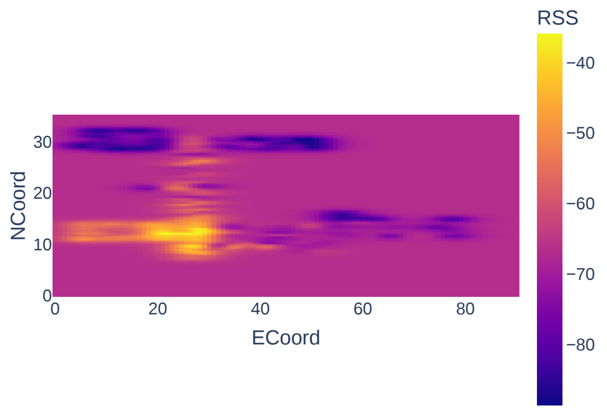


(b) Predicted standard deviations for *MAC56* with $v = 1.5$ and $ard\_num\_dims = 1$.

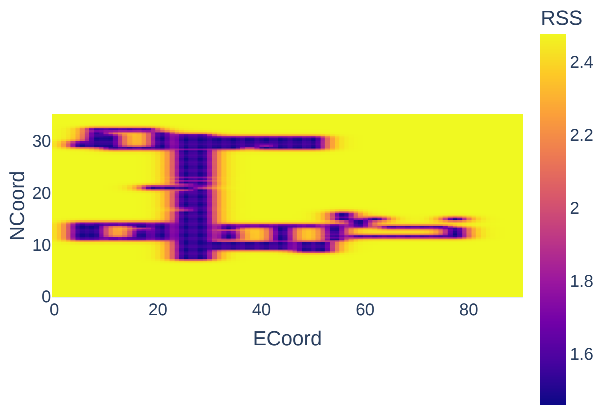
Figure 10: Predicted means and standard deviations for *MAC56* with $v = 1.5$ and $ard\_num\_dims = 1$.

Figure 10 shows the predicted means and standard deviations for $v = 1.5$ and one length-scale. The most noticeable difference to Figure 9 is that in the left plot 10a the yellow and blue regions, where the function has not converged to the mean, overall are smaller than before. This means that the now smoother functions converges faster to the mean. The standard deviation plots also show some differences. Whereas in Figure 9b the transition between the confident areas, standard deviation around 1.5, and and less confident areas, standard deviation around 2.5, was gradual the transition is now faster. This is especially noticeable in the are of the vertical hallway, ECoord of 22 and 34 and NCoord between 15 and 30.

The difference between Figure 10 and Figure 11 is less pronounced. What is noticeable is



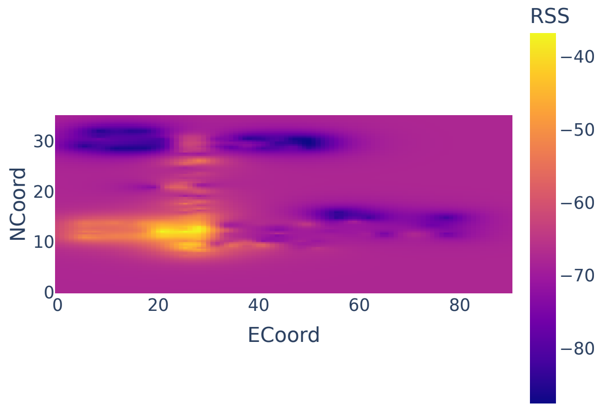
(a) Predicted mean values for *MAC56* with $v = 2.5$ and $ard\_num\_dims = 1$.



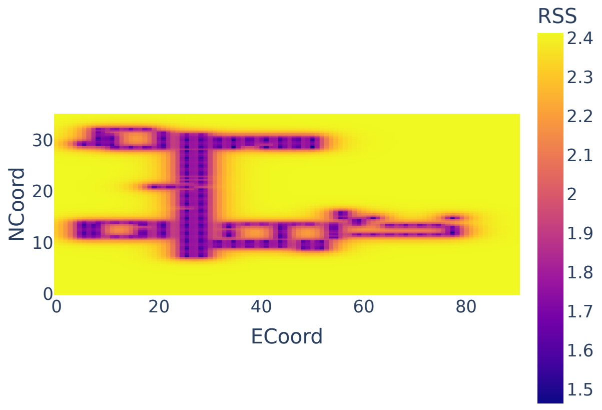
(b) Predicted standard deviations for *MAC56* with $v = 2.5$ and $ard\_num\_dims = 1$.

Figure 11: Predicted means and standard deviations for *MAC56* with $v = 2.5$ and $ard\_num\_dims = 1$.

that the purple area surrounded by the blue region in the top left, around ECoord of 18 and NCoord of 30, is getting bigger. Which again indicates that the smoother functions converge faster to the mean. This purple area is reflected in the plot of the standard deviations on the right 11b, where the confidence of the predictions sinks drastically. The area of low confidence is also in the earlier plots, but where the area was orange before like in plot 9b, the area is now yellow, which indicates an even lower confidence.



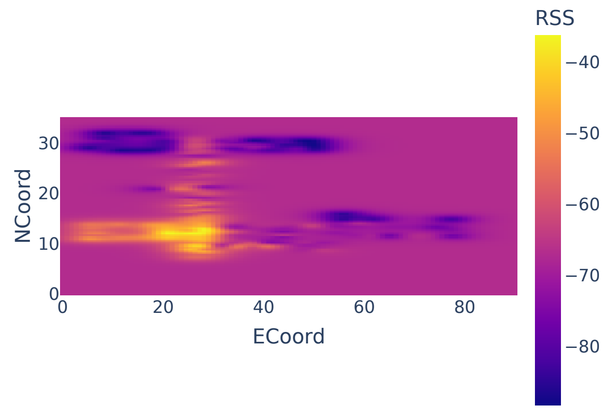
(a) Predicted mean values for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 2$.



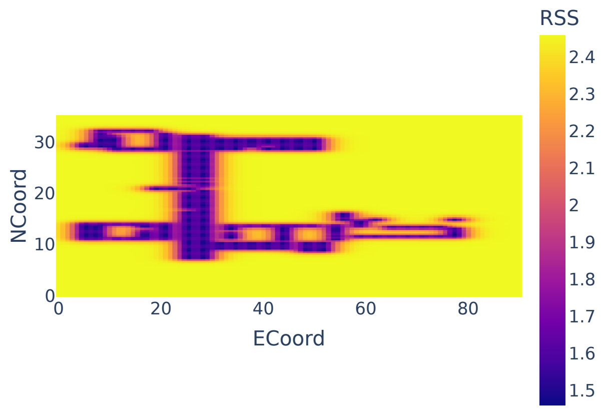
(b) Predicted standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 2$.

Figure 12: Predicted means and standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 2$.

Comparing Figure 12 to Figure 9 shows the difference between one and two length-scales. One obvious one is that the yellow and blue areas, where the function has not yet converged to the mean, are smaller along the horizontal axis. Looking at the blue region, in the top left corner, in plot 9a the area almost touched the left side of the plot, but in plot 12a there is a small but noticeable region between the purple area and the left side of the plot. The same phenomenon is also clearly visible for the yellow region in the lower left corner and still there but less evident for the other shaded areas. This contraction along the horizontal line is also detectable when comparing the plots of the standard deviation in Figure 9b and 12b. The reason being, that the function now converges faster along the horizontal axis than before.



(a) Predicted mean values for *MAC56* with $v = 1.5$ and $ard\_num\_dims = 2$.

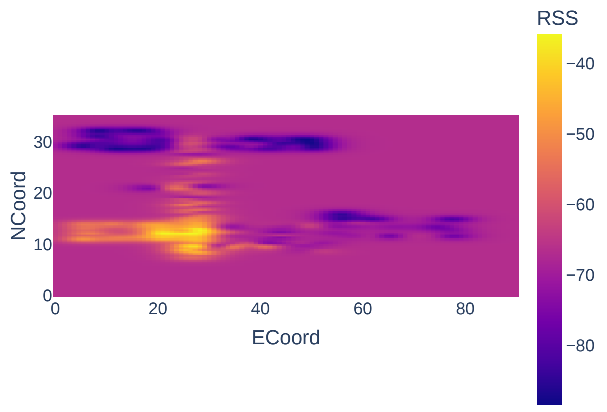


(b) Predicted standard deviations for *MAC56* with $v = 1.5$ and $ard\_num\_dims = 2$.

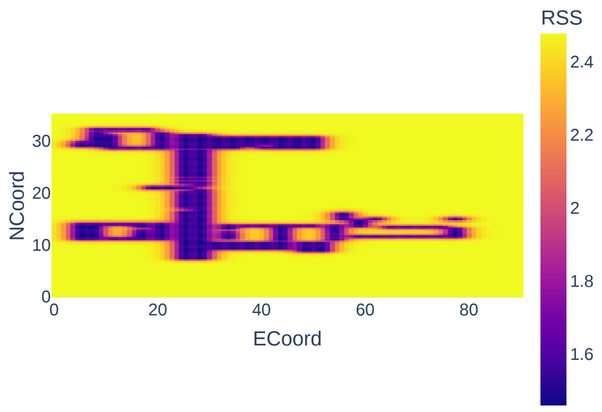
Figure 13: Predicted means and standard deviations for *MAC56* with $v = 1.5$ and $ard\_num\_dims = 2$.

Looking at Figure 13 and contrasting it with 10, the same contraction along the horizontal line is also perceptible but smaller in scope. While the yellow regions in the lower left corner stay almost the same, the blue regions in the top left corner still show the differences, both in the plots of the predicted means and in the plots of the predicted standard deviations.

Figure 14 shows the predicted means and predicted standard deviations for two length-scales and a $v = 2.5$. When comparing it to Figure 11, the same trend of a faster convergence to the mean along the horizontal axis is still visible, but it is getting less noticeable the



(a) Predicted mean values for *MAC56* with $v = 2.5$ and $ard\_num\_dims = 2$.



(b) Predicted standard deviations for *MAC56* with $v = 2.5$ and $ard\_num\_dims = 2$.

Figure 14: Predicted means and standard deviations for *MAC56* with $v = 2.5$ and $ard\_num\_dims = 2$.

smoother the function is. While the plots of the predicted means look almost identical, the plots of the standard deviations still reflect the trend, especially when looking at the transitions between the experiment area and the area without training points.

In conclusion, the difference in the predicted means and standard deviations are similarly small to the differences in $RMSE$ and R^2 scores. Therefore, the model with one length-scale and v equal to 0.5 still seems like the best fit.

4 Modifications

4.1 Normalizing Features and Standardizing Targets

One simple way to improve the performance of most machine learning algorithms is to normalize the features and standardize the targets.

Normalizing the features means scaling the features so that they lie in the unit hypercube. The process is described in equation (15), where x is a feature vector with d rows, with d being the number of features, two in this case. The vector $\vec{x}_{min}$ contains the minimum value of every feature and the vector $\vec{x}_{max}$ the maximum values.

$$\vec{x}_{norm} = \frac{(\vec{x} - \vec{x}_{min})}{(\vec{x}_{max} - \vec{x}_{min})} \tag{15}$$

The goal of feature normalization is to transform features to be of similar scale and thereby improve performance and stability of the model.

The next step is standardizing the targets, which means transforming the targets so that they have a mean of zero and unit variance. This is achieved by subtracting the mean and dividing by the standard deviation. In equation (16), $\vec{y}_{mean}$ is a vector, with every entry being the mean of the targets, and y_{std} is the standard deviation of the targets.

$$\vec{y}_{norm} = \frac{\vec{y} - \vec{y}_{mean}}{y_{std}} \tag{16}$$

Both steps are important, because the default parameters used in *GPyTorch* are optimized with normalized features and standardized targets in mind.

The results of the optimized models after normalization and standardization are listed in Table 4. The process is mostly the same as in the chapter before, with a few important distinctions. First, the model changed from a constant mean function to a zero-mean function. Second, for some combination of hyper-parameters additional *AP*s fail their optimization now, not only *MAC340*. Third, a new column was added, which contains the re-scaled $RMSE$, so that the results can still be compared between the transformed and non-transformed model.

ard_num_dims	v	$RMSE$	BIC	R^2	$RMSE_{rescaled}$
1	0.5	0.587836	13.665744	0.957087	6.804479
1	1.5	0.575901	13.825740	0.772233	6.715705
1	2.5	0.584674	13.894095	0.686079	6.877352
2	0.5	0.587733	19.706101	0.956976	6.803491
2	1.5	0.575603	19.881312	0.772549	6.715268
2	2.5	0.584571	19.961693	0.686996	6.875908

Table 4: Mean scores.

One drastic change in the results are the lower BIC scores, which are influenced by the model having one less parameter, due to the zero-mean function replacing the constant mean function. In comparison the $RMSE$ only changed slightly. The best score in chapter 3.2 was 7.139899, belonging to the model with two length-scales and $v = 0.5$. Now the model with the same parameters has a score of 6.803491, which is around 5 percent lower. The $RMSE$ of the other models decreased by around a similar amount, which is a notable performance improvement. Hampering that performance improvement, is the fact the R^2 scores sunk drastically, except for the models with a smoothness parameter of 0.5. In the worst case, the R^2 value fell to about 0.69 for the models with $v = 2.5$. This could be a result of replacing the constant mean function with the zero-mean function. While the constant mean function learns its mean and therefore gains an additional degree of freedom to better fit the data, the zero-mean function expects zero to be the best fit.

Thinking about the shape of the *RSS* function, the function should in theory fall from 0, the best *RSS* value possible, in every direction till it converges against -120, the worst value possible, which indicates no connection. This is not the case with the current model. After standardizing the targets, the targets have a mean of zero, which means positive and negative targets. But the GP converges to zero, due to the chosen zero-mean function. In addition to the decreased R^2 scores, this could also explain why some additional APs failed the optimization process. Table 5 lists the combination of APs and parameters that failed their optimization. All these *AP*s are not only missing at over half their measurements, but the remaining measurements also differ drastically. For example, *MAC128* contains values between -11 and -73, even though it is missing 65 percent of the possible training data. As seen in Table 5, it seems like the smoother functions cannot handle these kinds of circumstances.

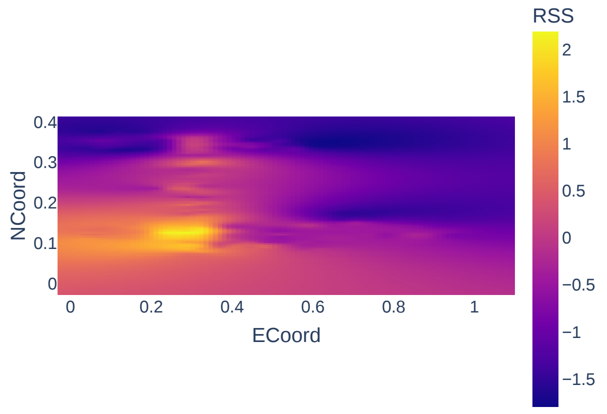
ard_num_dims	v	*AP*s
1	0.5	MAC340
1	1.5	MAC340, MAC224, MAC128
1	2.5	MAC340, MAC208, MAC224, MAC128
2	0.5	MAC340
2	1.5	MAC340, MAC224, MAC128
2	2.5	MAC340, MAC208, MAC224, MAC128

Table 5: *AP*s and parameter combinations that failed the optimization.

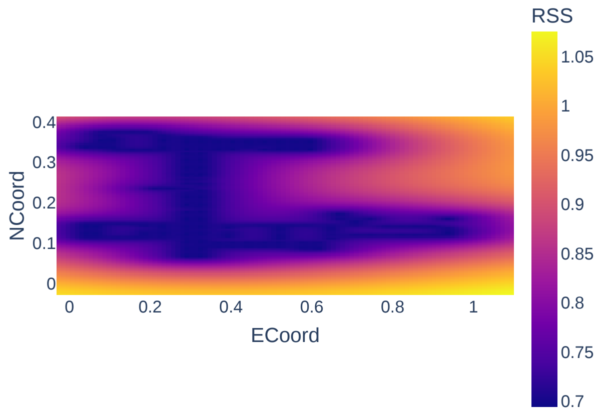
The optimizer used was the same *Adam* [10] optimizer, with 100 training iterations and a learning rate of 0.005. The learning rate is lower compared to the learning rate in chapter 3.2, because after normalizing the features and standardizing the targets, the parameters of the model are now significantly smaller. This is important, because the learning rate governs how fast the optimizer learns the hyper-parameters, a small learning rate results in small changes to the parameter and a large learning rate in large changes.

The results from Table 4 can be verified, by looking at some plots of the predicted mean and predicted standard deviations. Comparing Figure 15 to Figure 9 from section 3.3, with both models having $v = 0.5$ and one length-scale, it is possible to see the influence of the standardization and normalization.

First, there is a clear difference between the plot of the predicted means 15a after standardizing and normalizing and without 9a. Not only is the yellow area, of the room in the lower left corner, significantly stretched along the vertical direction in the current plot, but also the original blue region in the top left, is almost indiscernible from the blue region that replaced the purple region from before. Whereas before, everything with some distance from the experiment area converged to the mean relatively quickly, this time almost the whole top half of the plot, did not converge against the mean. This is not necessarily undesirable though, because the lower values in that region reflect the reality of an *RSS* function better.



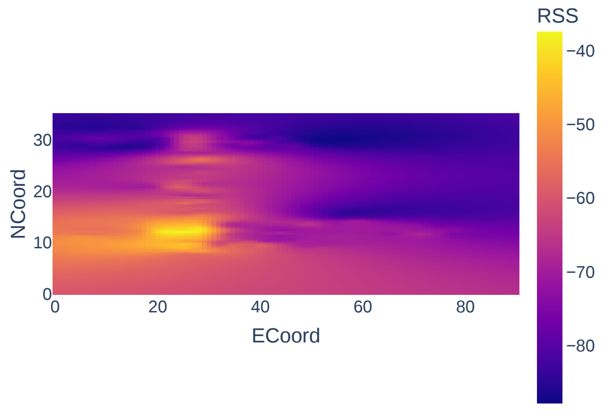
(a) Predicted mean values for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.



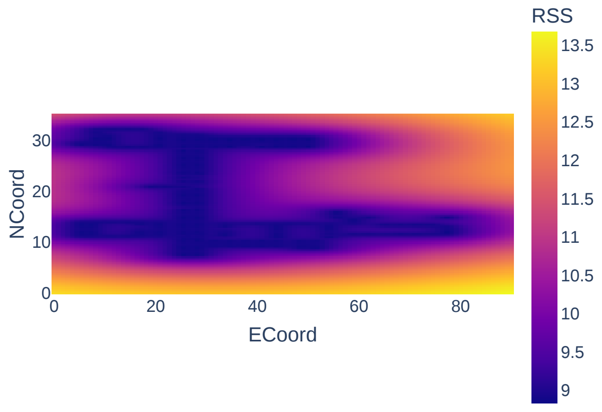
(b) Predicted standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.

Figure 15: Predicted means and standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.

The stretching along the vertical axis could be because while normalizing the features should bring them to the same scale, in this case, it does the opposite. Before both the horizontal and vertical axis used the same unit, but after normalizing them, the units are different, because the experiment area is more than twice as long as it is wide.



(a) Rescaled predicted mean values for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.



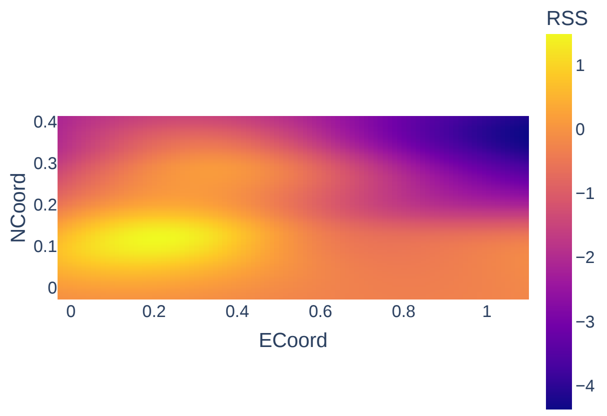
(b) Predicted standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.

Figure 16: Rescaled predicted means and standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.

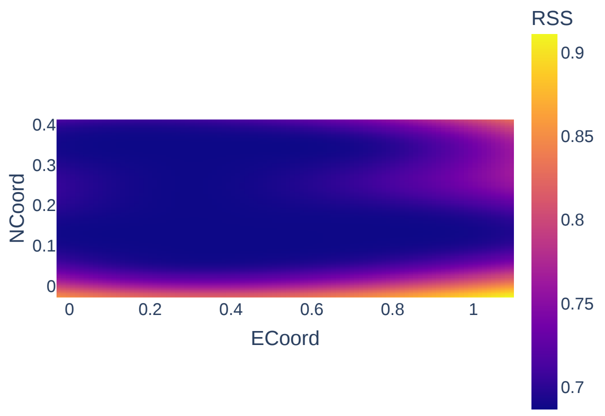
Second, the difference between the plots of the standard deviations is also clear. While the outline of the experiment area is clearly recognizable in 9b, it is harder to recognize it in the current plot 15b. Also, the standard deviations, while lower in absolute units, are higher, relatively to the mean, than before. In Figure 9b the standard deviation is between 1.5 and 2.5 for mean values between -40 and -90. Now the standard deviation is between 0.7 and 1.05 for a mean between -1.5 and 2.

This difference in the standard deviation gets clearer when looking at the scaled back predicted means and predicted standard deviations, which are displayed in Figure 16. The plots are identical to the plots in Figure 15, except for the color bars. The range of the predicted mean went from [-1.5, 2] to [-90, -40], which is the same range as for the predicted means in Figure 9a of the non-standardized model. On the other hand, the range of the standard deviations went from [0.7, 1.05] to [9, 13.5], which is significantly higher than in Figure 9b.

The reason for this decrease in confidence in the predictions could be two-fold. On the one hand it could be an artifact of the scaling issue. But it could also be because the model loses flexibility by not being able to learn a constant mean and instead having a zero-mean function.



(a) Predicted mean values for *MAC56* with $v = 2.5$ and $ard\_num\_dims = 1$.



(b) Predicted standard deviations for *MAC56* with $v = 2.5$ and $ard\_num\_dims = 1$.

Figure 17: Predicted means and standard deviations for *MAC56* with $v = 2.5$ and $ard\_num\_dims = 1$.

Figure 17a shows that for one length-scale and $v = 2.5$, all possible advantages of normalizing the features and standardizing the targets vanish completely. The higher v value results in an extremely smooth function, which is so smooth that any resemblance of the function to the building disappears, which is highly undesired. The standard deviation plot 17b is an other indictment of the standardized model, because in contrast to prior standard deviation plots no similarities to the building are visible either.

The drastic difference in the plots between Figure 15 and Figure 17 explains the similar large difference in R^2 visible in Table 4.

In conclusion, normalizing the features and standardizing the targets while slightly improving the average $RMSE$ and BIC, did not achieve the expected performance improvements, but instead made the models worse, especially when looking at the models with a higher value of the smoothness parameter.

4.2 Replacing Missing Values and Choosing Constant Mean

In chapter 3 and section 4.1 the measurements with a *RSS* value of 100, ergo the points that have no connection with the specific *AP*, were removed. In this section, instead of dropping these measurements, the *RSS* value gets replaced with -120. Furthermore, instead of learning the constant mean from the training data, it is fixed to -120.

These changes are made, so that the resulting *GP* stronger resembles a real *WiFi* signal. Due to the limited range of every *AP*, the domain of the *GP*, would have to be restricted to that exact range, because all extrapolations outside that domain are almost definitely false, which is visible in the plots of the predicted means in section 3.3.

Alternatively, the *GP* could converge to a value that indicates a missing connection, for example -120. Which should lead to all extrapolated values from outside the signal range, returning -120, which would be correct.

This section uses the model from chapter 3, with a constant mean function, because normalizing and standardizing did not improve the model, but made it worse instead, as

ard_num_dims	v	$RMSE$	BIC	R^2
1	0.5	20.728166	32.295786	0.925756
1	1.5	21.866363	35.199034	0.925427
1	2.5	22.248371	36.430018	0.925359
2	0.5	20.748607	39.285659	0.926519
2	1.5	22.272956	42.556297	0.941790
2	2.5	22.711738	43.552389	0.926161

Table 6: Mean scores

described in section 4.1.

For the hyper-parameter optimization the *Adam* algorithm was used, but this time with 150 iterations and a learning rate of 1. The training and test scores of the optimized models are shown in Table 6. Comparing these scores to the scores of the optimized models, from chapter 3.2, in Table 1 reveals how much worse the new scores are.

While the $RMSE$ scores before ranged between 7.14 and 7.72, they are now between 20.73 and 22.71, which is around three times worse. One notable difference though is that before the model with two length-scales always had a slightly lower $RMSE$ on average, which is no longer the case. On the contrary, the models with one length-scale now have the slight advantage. The BIC went from a range of [24.68, 33.13] to a range of [32.30, 43.55], which is around 76% worse. Furthermore, the lowest R^2 before was 0.9784 and is now 0.9254, which is about six percent worse.

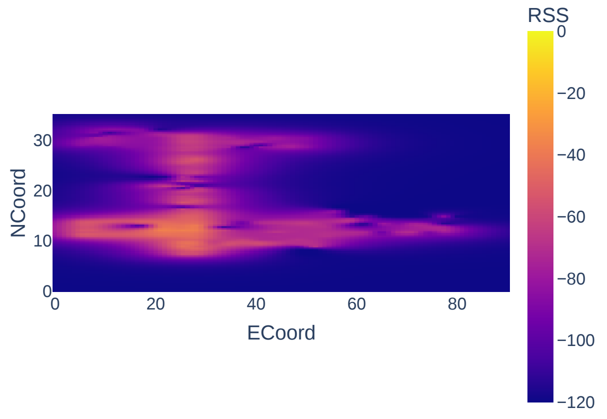
Looking at the plots of the predicted means and standard deviations confirms the above results. The first row of Figure 18 shows the predicted means on the left 18a and the predicted standard deviations on the right 18b. Both are from a model that was trained for this section. Therefore, the 100 values were replaced with -120 prior to training. For the plots in the second row, a model from chapter 3 was used, where the 100 values were removed prior to training. Moreover, the second row is almost identically with Figure 9, but the color-bar was re-scaled for easier comparison.

The most obvious difference between the plots of the predicted means is the color around the house. While the *GP* from chapter 3, converged to a learned mean of -67.6276, the current *GP* converges to -120, as intended.

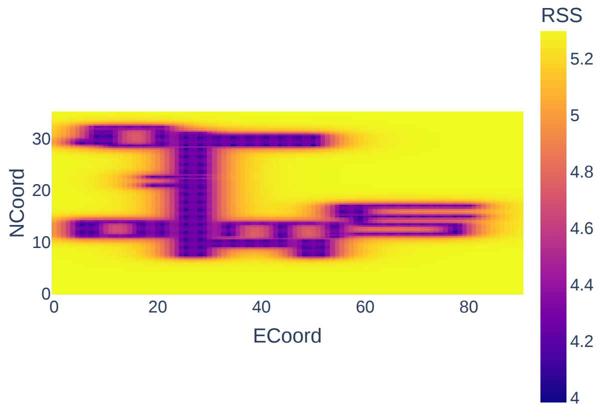
The second difference is that inside the experiment area the current model seems to generally predict higher values, which could explain the worse test scores. This is clearest when looking at the top half of the experiment area, as well as at the lower right side. While the plot in Figure 18c, shows predicted means below -80 for these areas, which would make sense, because they have a significant distance from the *AP*, the plot in Figure 18a shows values above -60 for most of these areas. In general, the mean function from the new model seems to predict less distinct values and converges faster, which seems to be an effect of a higher value for the output-scale θ_{scale} parameter. While the model of the second row has an θ_{scale} of 3.8 the model in the first row has a θ_{scale} of 14.11. Additionally, if the mean is -120 but all the observations are higher, then the distribution of the observations is heavily left skewed, even though Gaussian processes are symmetric.

The plots of the standard deviations look almost the same at first glance, but one big difference is the scale of the color bar. While the standard deviations in plot 18d are between 1.5 and 2.4, the standard deviations in plot 18b are between 4 and 5.2. This means that overall, the current *GP* is less confident in its predictions that the one from chapter 3. Another difference is that in plot 18d for the top section of the room in the right, *ECoord* between 60 and 80 and *NCoord* around 30, the standard deviations are indistinguishable from the area outside the experiment area. In contrast, the standard deviations in plot 18b for that section are the same as for the rest of the experiment area.

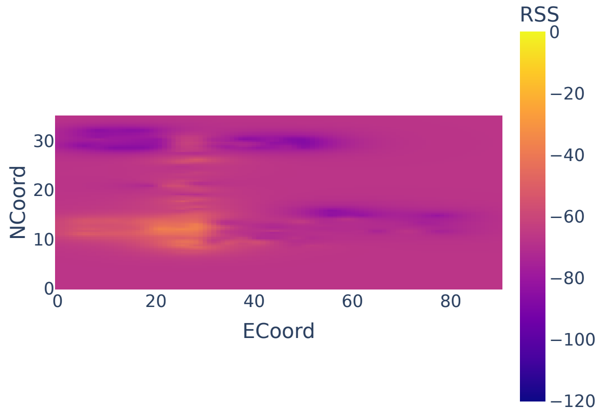
To sum up, the plots of the predicted means and predicted variances support the verdict, that replacing the 100 values with -120 and forcing the constant mean to -120, impaired the model instead of improving it.



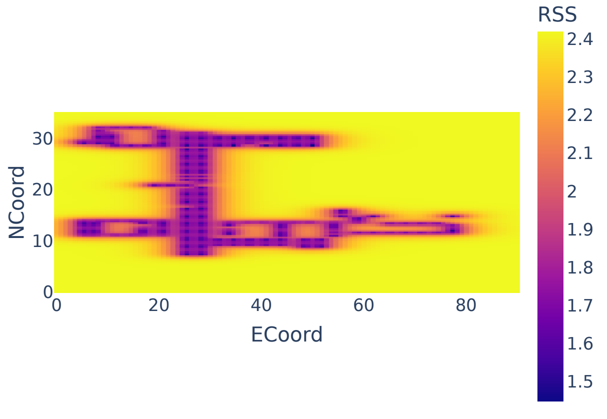
(a) Predicted mean values for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.



(b) Predicted standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.



(c) Predicted mean values for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.



(d) Predicted standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$.

Figure 18: Comparison of predicted means and standard deviations for *MAC56* with $v = 0.5$ and $ard\_num\_dims = 1$ of two different models.

5 Conclusion

After evaluating the model fitting process in [14], this paper decided to focus on a *GP* with a constant mean and *Matérn* covariance function. While [14] tested this combination, their experiment area was of limited complexity. Therefore, the model was reconsidered in a more complex indoor environment.

In total, two different length-scale options, one length-scale and two length-scales, and three different values for the smoothness parameter v (0.5, 1.5, 2.5) were tested. It can be concluded that a parameter combination of one length-scale and a smoothness parameter v of 0.5, is the best one to model the spatial structure of *RSS* in more complex indoor environments. This was confirmed by the different model measures, as well as by the analysis of the plots of the predicted means and standard deviations. This is distinct to the findings of [14], which chose 1.5 as the value of the smoothness parameter. The reason being that a more complicated environment requires a rougher function.

Furthermore, standardizing and normalizing prior to training decreased the overall performance of the model contrary to expectations.

Additionally, forcing the mean to -120, to better reflect the actual properties of an *RSS* function decreased the model performance as well. One potential way to incorporate a converge to -120 into the *GP* model, is to use a deep Gaussian process instead. A deep Gaussian process [5] is similar to a deep neural net, but instead of having multiple layers of neural networks, it consists of layers of Gaussian processes, where each layer contains one or more *GP*s.

Another option to model the left skewness of the resulting distribution, is to use a *SkewGP* [3] [2] instead of a standard Gaussian process. *SkewGP*s are an extension of *GP*s that inherit all the good properties of *GP*s, while overcoming some of its limitations. In particular the symmetry around the mean and that the Gaussian distribution is not heavily tailed.

References

[1] Mohamed M. Atia, Aboelmagd Noureldin, and Michael J. Korenberg. “Dynamic Online-Calibrated Radio Maps for Indoor Positioning in Wireless Local Area Networks”. In: *IEEE Transactions on Mobile Computing* 12.9 (2013), pp. 1774–1787. DOI: 10.1109/TMC.2012.143.

[2] Alessio Benavoli, Dario Azzimonti, and Dario Piga. *A unified framework for closed-form nonparametric regression, classification, preference and mixed problems with Skew Gaussian Processes*. 2021. arXiv: 2012.06846 [stat.ML].

[3] Alessio Benavoli, Dario Azzimonti, and Dario Piga. “Skew Gaussian processes for classification”. In: *Machine Learning* 109 (Sept. 2020), pp. 1–26. DOI: 10.1007/s10994-020-05906-3.

[4] Jingxue Bi et al. “Supplementary open dataset for WiFi indoor localization based on received signal strength”. In: *Satellite Navigation* 3 (Nov. 2022). DOI: 10.1186/s43020-022-00086-y.

[5] Andreas Damianou and Neil D. Lawrence. “Deep Gaussian Processes”. In: *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Carlos M. Carvalho and Pradeep Ravikumar. Vol. 31. Proceedings of Machine Learning Research. Scottsdale, Arizona, USA: PMLR, 2013, pp. 207–215. URL: https://proceedings.mlr.press/v31/damianou13a.html.

[6] Felix Duvallet and Ashley D. Tews. “WiFi position estimation in industrial environments using Gaussian processes”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 2216–2221. DOI: 10.1109/IROS.2008.4650910.

[7] Brian Ferris, Dirk Hähnel, and Dieter Fox. “Gaussian Processes for Signal Strength-Based Location Estimation”. In: Aug. 2006. DOI: 10.15607/RSS.2006.II.039.

[8] Jacob R Gardner et al. “GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration”. In: *Advances in Neural Information Processing Systems*. 2018.

[9] Plotly Technologies Inc. *Collaborative data science*. 2015. URL: https://plot.ly.

[10] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014).

[11] Fen Liu et al. “Survey on WiFi-based indoor positioning techniques”. In: *IET Communications* 14.9 (2020), pp. 1372–1383. DOI: https://doi.org/10.1049/iet-com.2019.1059. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-com.2019.1059. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-com.2019.1059.

[12] James Lloyd et al. “Automatic Construction and Natural-Language Description of Nonparametric Regression Models”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 28.1 (2014). DOI: 10.1609/aaai.v28i1.8904. URL: https://ojs.aaai.org/index.php/AAAI/article/view/8904.

[13] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006, pp. I–XVIII, 1–248. ISBN: 026218253X.

[14] Philipp Richter and Manuel Toledano-Ayala. “Revisiting Gaussian Process Regression Modeling for Localization in Wireless Sensor Networks”. In: *Sensors* 15.9 (2015), pp. 22587–22615. ISSN: 1424-8220. DOI: `10.3390/s150922587`. URL: `https://www.mdpi.com/1424-8220/15/9/22587`.

[15] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. “Probabilistic programming in Python using PyMC3”. In: *PeerJ Computer Science* 2 (Apr. 2016), e55. DOI: `10.7717/peerj-cs.55`. URL: `https://doi.org/10.7717/peerj-cs.55`.

[16] Jaehyun Yoo and Hyoun Jin Kim. “Target Tracking and Classification from Labeled and Unlabeled Data in Wireless Sensor Networks”. In: *Sensors* 14.12 (2014), pp. 23871–23884. ISSN: 1424-8220. DOI: `10.3390/s141223871`. URL: `https://www.mdpi.com/1424-8220/14/12/23871`.