Spring 2024

Investigating Uncertainty in Gaussian Process Models

Wilson Strasilla
San Jose State University

Investigating Uncertainty in Gaussian Process Models

Wilson Strasilla
Author
wilson.strasilla@sjsu.edu

Martina Bremer
Academic Advisor
martina.bremer@sjsu.edu

A Writing Project Submitted to
the Department of Mathematics in
Partial Fulfilment of the Requirements for
the MS Degree in Data Science

San Jose State University
May 2024

Contents

1 Introduction

Regression tasks are fundamental in various fields, ranging from machine learning to engineering, where the objective is to predict a continuous output variable based on various continuous or discrete input features. Traditional parametric regression models, such as linear regression, rely on fixed functional forms and make assumptions about the data distribution. However, these models often struggle to capture the intricate relationships present in real-world data, leading to sub-optimal performance in many cases.

Non-parametric models provide a flexible alternative by allowing the underlying function to adapt to the data without imposing rigid assumptions on the "shape" of the data. Gaussian Process Models (GPMs) represent a prominent class of non-parametric models that have shown practical use due to their flexibility and accuracy. Unlike traditional parametric models, GPMs define a distribution over functions, enabling uncertainty quantification and robust predictions even in the presence of noise or limited data.

This deep dive into the workings of GPMs will begin with an overview of the regression task and the rationale behind adopting non-parametric approaches. Subsequently, this paper delves into the fundamental concepts of GPMs, including their mathematical formulation, critical properties, and practical considerations, while largely following the framework of Rasmussen and Williams [3]. Furthermore, the role of uncertainty estimation and marginal likelihood in GPMs will be investigated through experiments conducted on simulated datasets.

The overarching goal of this paper is to provide an in-depth understanding of the inner workings of GPMs, in particular, exploring the effects of making modifications to the model's hyper-parameters. Concurrently, this research will look to provide a better understanding of uncertainty in GPMs, and explore some perhaps unforeseen benefits of using the marginal likelihood as an objective function for optimization.

2 Background

2.1 Regression

Regression analysis is a technique used for predicting the value of some dependent variable given some set of independent variables. There are many techniques for approaching this problem, but one of the most intuitive and well-known is least squares linear regression. The linear regression model gets its name from the property that there is a linear relationship between the dependent variable and the independent variables. This class of model can result in all sorts of shapes in their prediction surfaces, as transformations can be applied to the independent or dependent variables. These models have many excellent properties, such as ease of use and interpretation, but require the user to define the type of function the linear regression model will adhere to before fitting the data. If you have worked with these models, you will know that this can require

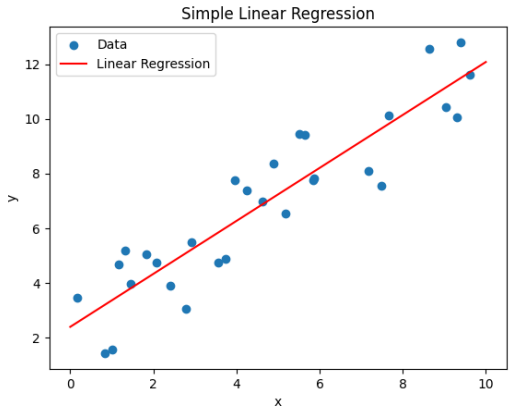

Figure 1: A simple example of linear regression. The regression hyper-plane (in this case, a line) is fit to minimize the sum of squared errors of prediction at all observations.

a lot of experimenting to find an appropriate set of transformations on the data, and even then, the fit may be questionable. This is the primary restriction that non-parametric models are designed to circumvent.

2.2 Non-Parametric Models

Non-parametric models do away with the traditional parameters used in models such as linear regression. Instead, the shape of the response surface is determined by the data itself, making the parameters the target values of the data. Perhaps the most intuitive non-parametric model is kernel regression. In this class of model, making a prediction $\hat{y}_i$ at x equates to taking the weighted average of the observed values y_j for all observations.

$$\hat{y}_i = \sum_{j=1}^{n} w_{ij} y_j \tag{1}$$

The weights w_{ij} are determined by a parameter called the bandwidth, as well as some kernel function $K()$. A kernel function is essentially a measure of the distance between two observations. In this application, the kernel is used to determine weights, which must sum to one to be used for prediction through a weighted average. To this end, the kernel evaluations are normalized, as seen in (2). The bandwidth represents how much impact observations close to the predictive point in the domain space should have on prediction in comparison to farther observations. With a larger bandwidth, all observations will be treated more equally than with a smaller bandwidth. In a simple case, such as with a uniform kernel, the bandwidth can represent an interval where observations are considered, with the weights outside this interval set to zero.

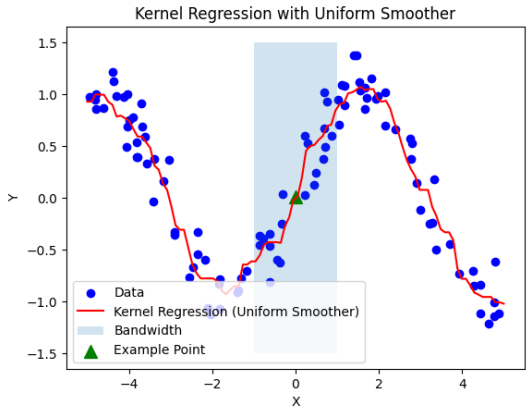

Figure 2: An example of kernel regression on data generated from a sine function with randomness. Unlike linear regression models, the non-parametric model can capture this shape without knowledge of the function type. Observe the example point in the center of the shaded region. To make a prediction at $X = 0$, a weighted average of the target values Y is taken over all observations with $-1 \leq X \leq 1$ (the shaded area), which represents the bandwidth (for other choices of smoother all of the data may be considered but with different weights). As this example uses a uniform smoother, this is just a simple average of the points within the interval determined by the bandwidth.

$$w_{ij} = \frac{K(\frac{x_i - x_j}{b})}{\sum_{j=1}^{n} K(\frac{x_i - x_j}{b})} \tag{2}$$

As can be seen in figure 2, this method can be very effective at capturing the shape of the data without much experimentation. There are, however, some drawbacks to non-parametric models. Perhaps most importantly, the lack of traditional parameters can make drawing conclusions about the effects of one variable on another difficult. In a method such as least-squares linear regression, one can make a statement about what can be expected to happen to the target value when each variable changes by using the trained weights. However, in the case of Kernel Regression, no relation between individual variables and the response is obtained, and every point on the response surface is simply a function of the training data. In addition to the lack of interpretability, one can very easily over-fit the data using this class of model. By lowering the bandwidth, one can create a model that gives a disproportionate amount of weight to the most similar observations in terms of predictive variables. This will result in a model that fits the dataset very well but likely will not represent the true relation between the variables. Interestingly, in the case of a uniform kernel, the resulting model may even fail to make predictions at some points if there is no data within the domain defined by the bandwidth. Finally, while the reduced assumptions of non-parametric models add flexibility, this comes at the cost of

some of the power granted through those assumptions. The assumptions of parametric models generally provide some additional confidence in predictions, meaning non-parametric models often require more data to achieve this same confidence.

3 Gaussian Process Models

Definition 3.1 *A Gaussian process is a collection of random variables that is completely specified by its mean and covariance functions [3].*

In this paper, the Bayesian approach to Gaussian Process modeling will be discussed. That is, a prior GPM is constructed with basic assumptions. This prior distribution is then conditioned on the observed data to get a posterior distribution for the GPM. This will be further discussed in the following sections, but for now, it is perhaps most effectively introduced graphically.

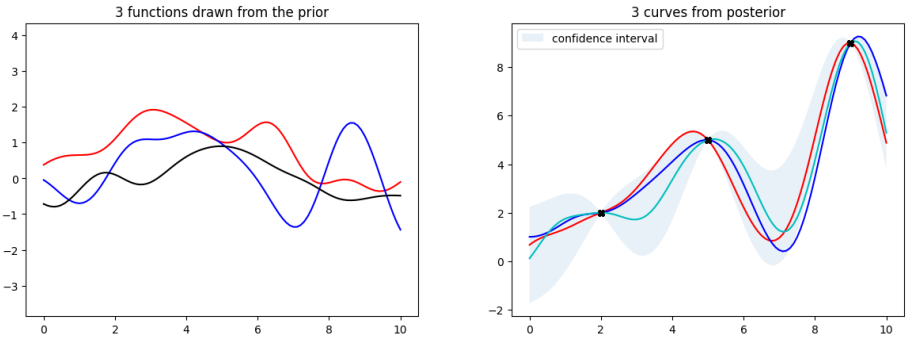

Figure 3: The first graph shows three functions drawn from the prior distribution with mean zero, which is not yet conditioned on the observations. The second graph shows three functions drawn from the posterior distribution, which is fit to the observations.

This process of conditioning the prior distribution to the observations can be imagined (in certain conditions) as weeding out functions that do not agree with the observations. This elimination of functions can be seen in the fact that all of the functions in the second panel of figure 3 perfectly pass through the points that represent the training observations. In section 3.3 the inclusion of noise in the GPM will be discussed, and this addition will allow for the functions to not align perfectly with the training observations. Information is also gained about how likely any function may be. In the figure, this is represented by the highlighted portion of the plot, which is the confidence interval of prediction. Notice that the shaded region tends to grow in width farther from the observations, meaning that the model is less certain in areas where it has less training information to work with.

This, however, is just a glimpse into the intuition of GPMs. The following sections will show how this Bayesian approach can be linked to linear regression

before finally turning to the non-parametric model known as GPM, which will be the focus of this paper.

3.1 Weight-Space View

Among the several ways to help understand Gaussian process models is the weight-space view. The weight space view is closely linked to linear regression in that the goal is to estimate weights $\mathbf{w}$, which will act as the slopes of the response surface, using the observations. The observed values of the dependent variable are represented by $\mathbf{y}$, while the independent variables are represented as X. σ_n^2 represents the uncertainty of the observed values $\mathbf{y}$. This uncertainty gives rise to the likelihood of $\mathbf{y}$ shown in (3).

$$p(\mathbf{y}|X, \mathbf{w}) = N(X'\mathbf{w}, \sigma_n^2 I) \tag{3}$$

It can be seen in (3) that the first term of the normal distribution (the mean) is simply the application of a weight vector to some dependent variables, which is how predictions are made in the technique least squares linear regression discussed in section 2.1. Initially, the model has no information about what the weights are, but is able to use the training data to find new weights, as well as confidence intervals for these weights. In addition to being able to make predictions, information is gained about the certainty of each prediction (which varies based on the input). As the steps are described, the process will also be described graphically for an example trained with the following simple data set, which has already been used to create the plots seen in figure 3.

$$\begin{aligned} X &= \begin{bmatrix} 1 & 1 & 1 \\ 2 & 5 & 9 \end{bmatrix} \\ \mathbf{y'} &= \begin{bmatrix} 2 & 5 & 9 \end{bmatrix} \end{aligned} \tag{4}$$

Since a prior distribution for the weights is needed, assumptions must be placed on the distribution of $\mathbf{w}$. Generally, those assumptions are that $\mathbf{w}$ has a mean of $\mathbf{0}$ and some covariance Σ_p. In the weight-space model, these weights can be interpreted as the slopes in simple linear regression and include the intercept. Thus there are a number of weights equal to one plus the number of variables, in this simple example that means that there are two slopes. In X the need for an intercept is reflected with an extra row of ones.

$$\mathbf{w} \sim N(\mathbf{0}, \Sigma_p) \tag{5}$$

The goal now is to find the posterior distribution of $\mathbf{w}$ using Bayesian principles.

$$posterior = \frac{likelihood \times prior}{marginal\, likelihood} \tag{6}$$

The marginal likelihood in (6) takes the form seen in (7).

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w} \tag{7}$$

applying Bayes' rule,

$$p(\mathbf{w}|X, \mathbf{y}) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)} \tag{8}$$

the posterior distribution of $\mathbf{w}$ can be found to be proportional to (9) by taking only the terms of (7) which depend on $\mathbf{w}$.

$$p(\mathbf{w}|X, \mathbf{y}) \propto exp(-\frac{1}{2}(\mathbf{w} - \overline{\mathbf{w}})'(\sigma_n^{-2}XX' + \Sigma_p^{-1})(\mathbf{w} - \overline{\mathbf{w}})) \tag{9}$$

where σ_n^2 is the noise of the target variable, and

$$\overline{\mathbf{w}} = \sigma_n^{-2}(\sigma_n^{-2}XX' + \Sigma_p^{-1})^{-1}X\mathbf{y} \tag{10}$$

making this a Gaussian of the form:

$$p(\mathbf{w}|X, \mathbf{y}) \sim N(\overline{\mathbf{w}} = \sigma_n^{-2}A^{-1}X\mathbf{y}, A^{-1}) \tag{11}$$

where

$$A = \sigma_n^{-2}XX' + \Sigma_p^{-1} \tag{12}$$

Using the mean and variance from (11), the posterior's confidence in the weights can be visualized by plotting the weights within one standard deviation of the mean. Notice in figure 4 that the variance for w_0 (the intercept) is far greater than that of w_1, as any slight change in the slope can lead to a more significant change in the intercept.

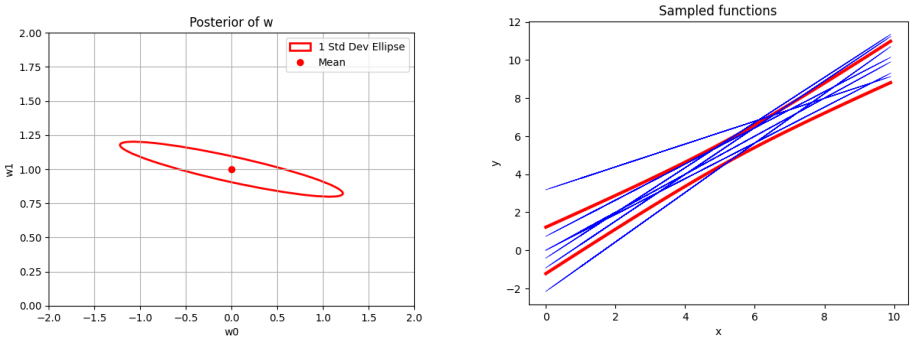

Figure 4: The dot in the first panel is the mean of the posterior distribution of $\mathbf{w}$. The ellipse represents the confidence interval of $\mathbf{w}$ under the posterior. The second panel shows multiple lines drawn from the posterior of $\mathbf{w}$. Observe that while the slopes do not vary too drastically, the intercept does.

The weights can finally be used to make predictions. In figure 5, the y-intercept is the first weight w_0, and the slope is the other weight w_1. Similarly to how figure 4 is constructed, the equation of the regression line can be found through the means of (11), and the confidence can be found using the variance of (11). Alternatively, one can get the same results through a posterior distribution on f_* as in (13) where $\mathbf{x}_*$ is a dense lawn between zero and ten where predictions are made.

$$p(f_*|\mathbf{x}_*, X, \mathbf{y}) = N(\sigma_n^{-2}\mathbf{x}_*' A^{-1} X\mathbf{y}, \mathbf{x}_*' A^{-1}\mathbf{x}_*) \tag{13}$$

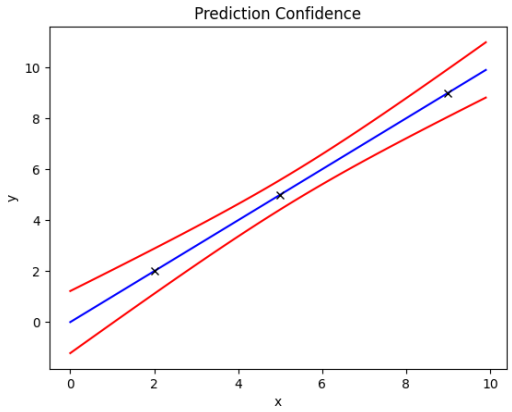

Figure 5: In this plot, the blue line represents the regression surface, and the thick curved red lines represent the one standard deviation bound on predictions. It can be seen that the certainty gets worse as x gets farther from the center of the plot due to the increase in the total distance to the data points.

In summary, using this approach, the data is used to update a prior distribution of the weights. The mean of the posterior distribution can be interpreted as the slopes, and the variance of the posterior can be used to describe the certainty in both the predictions and the slopes. You likely noticed, however, that the number of slopes is defined in this case, meaning this approach has the same limitations as linear regression. In the next section a function space approach will be discussed, which removes this restriction.

3.2 No Noise Function-Space View

Alternatively to applying Bayesian techniques to weights, one can apply them to the function space. Similarly to in the weight-space view, the mean of the prior is initialized to a zero vector, with the difference being that the elements of this mean represent the target values $\mathbf{f}_*$. Accompanying this discussion will be an example using the same dataset as the previous section.

One of the keys to the GP is the kernel function, which can be manipulated greatly to best fit the data. In this case, the squared exponential kernel function seen in (14) will be used due to its simplicity and other nice properties. For example, every function in its prior has infinitely many derivatives.

$$k_y(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^2\right) \tag{14}$$

In (14) σ_f^2, and l are the signal variance and length-scale respectively, which will be discussed further in section 3.3. This leads to a prior distribution on the response where the mean is a 0 vector, and the covariance is determined by the kernel on all input observations. In section 2.2, the kernels were expressed as scalars. In (15), the covariance matrix $K(X_*, X_*)$ is a matrix where the kernel is evaluated for every pair of elements in X_* using the equation in (14), or any other appropriate kernel function.

$$\mathbf{f}_* \sim N(\mathbf{0}, K(X_*, X_*)) \tag{15}$$

Drawing from this prior distribution, of course, results in rather random looking functions such as in the first panel of figure 6. This is due to the fact that the covariance function does not consider the dependent variable.

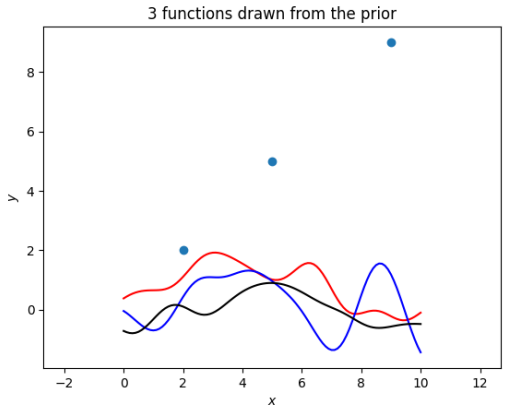

Figure 6: These curves show functions drawn from the prior distribution. See that the responses are more or less centered around zero and do not represent the observations.

This information on its own is clearly not very useful. However, if a restriction is placed on these random functions requiring that they pass through the known observations, these functions immediately become much more interesting. To start, observe the joint distribution of the training and test outputs given the prior as seen in (16). This is little different from the original prior, now simply expanding the covariance matrix and mean vector to accommodate the increased information.

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim N(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}) \tag{16}$$

All of the information above in (16) is known besides $\mathbf{f}_*$. The next step is to restrict this joint prior to functions that agree with the training data. This can be done by conditioning the joint prior to the observations. It can be shown that in the following scenario:

$$\begin{aligned} \mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{bmatrix} \sim N(\mu, \Sigma) \text{ where} \\ \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \end{aligned} \tag{17}$$

the conditional distribution of $\mathbf{X}_1$ given $\mathbf{X}_2 = \mathbf{x}_2$ is normal with

$$\begin{aligned} \text{Mean} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mu_2) \\ \text{Covariance} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \end{aligned} \tag{18}$$

This leads to the distribution seen below in (19), as $\mu = 0$ in the prior distribution of $\mathbf{f}_*$. For the full proof of how to derive the mean and covariance in (18) see the work of Johnson and Wichern [2].

$$\begin{aligned} \mathbf{f}_* | X_*, X, \mathbf{f} \sim & N(K(X_*,X)K(X,X)^{-1}\mathbf{f}, \\ & K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)) \end{aligned} \tag{19}$$

This will result in a mean vector for predictions over the test data and a covariance matrix for these predictions. The diagonal of the variance portion of the distribution in (19) gives the variance at each test point. It can be seen that in the single test point scenario, $K(X_*, X_*) = \sigma_f^2$ (as this is the kernel of a test observation with itself leading to the exponential portion of (14) becoming equal to one), and the second term will be the left and right product of the covariance between the test and training data with the inverse of the within covariance of the training data. In the scenario where there is no noise in the training data (or no noise is assumed), the model will always predict the "correct" value at the training points, as seen below by the curves all passing through the points. This means that at the observations, the diagonal elements of the second term are also equal to σ_f^2, while points farther from the observations will have small values for the second term. In section 3.3, the results of adding noise to the kernel will be explored.

In figure 7, notice the shaded area, which is the mean plus/minus two times the standard deviation of $\mathbf{f}_*$ at each input value. The functions all have the same confidence bands, because they are being drawn from the same distribution. A change to any of the hyper-parameters σ_f^2, or l, however, would lead to a new model with a different confidence band (as well as a different mean vector).

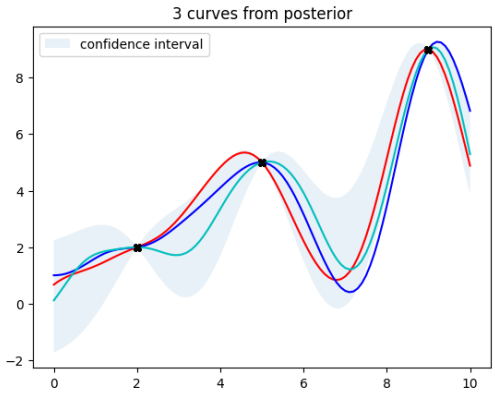

Figure 7: Another look at the second panel of figure 3. Three curves are drawn from the posterior distribution. Note that they are no longer centered around zero and more closely represent the observations.

3.3 Hyper-Parameters

Hyper-parameters are the free parameters used to modify functions and are generally set using some prior knowledge of the dataset, trial and error, or through an optimization process. In the case of Gaussian Process Models, hyper-parameters modify the kernel function. While these will vary in type and count depending on the covariance function, this paper will limit its discussion to the squared-exponential kernel, and thus its three hyper-parameters σ_f^2, σ_n^2, and l. The squared-exponential kernel with noise takes a slightly different form from what was seen in section 3.2 as seen in (20).

$$k_y(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^2\right) + \sigma_n^2 \delta_{pq} \tag{20}$$

First, let's briefly discuss each variable in (20). The length-scale, l, determines the smoothness of the response surface in the functions drawn from the posterior by adjusting how much the input needs to vary to result in significant changes in the output. The signal variance, σ_f^2, scales the exponential portion of the kernel and tends to result in a stretching of the response surface in the direction of the output. This is because σ_f^2 leads to a larger variability in the response. Lastly, σ_n^2 is the noise variance, representing the measurement error in the observed data. In the case of zero noise, $\sigma_n^2 = 0$, as was seen in section 3.2. Changing this can allow the model's functions to not perfectly pass through all training points, as well as introduce non-zero uncertainty at these points. The indicator variable $\delta_{pq} = 1$ when $x_p = x_q$, and $\delta_{pq} = 0$ otherwise. In GPMs, the noise term is generally used to represent the noise in the training data and thus is often only included in $K(X, X)$ and is left out in the other kernel evaluations.

To better understand the effects of each hyper-parameter, it is easiest to

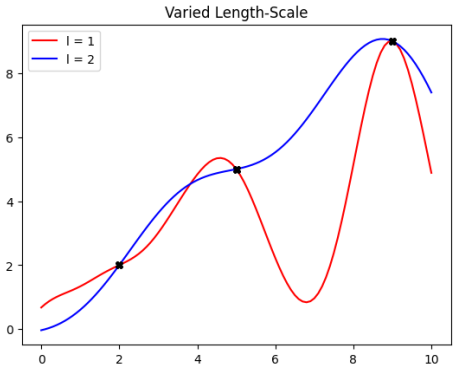

Figure 8: Observe the effects of varying the length-scale. The function in red was generated with the hyper-parameters $l = 1$, $\sigma_n^2 = 0$, and $\sigma_f^2 = 1$, while the function in blue uses $l = 2$, and all other variables are held constant. The higher length-scale results in a smoother function.

view graphical examples where each is varied in isolation, that is to say, with the other hyper-parameters held constant. Figure 8 shows an example of what may occur when l is varied. While the restriction of zero noise at times leads to unexpected functions, this is more or less what one would expect to see with these hyper-parameters, as the blue function is much smoother due to the greater length-scale. This is because increasing the length-scale essentially has the same effect as making the data points closer together, as can be seen by its presence as a scaling factor on $(x_p - x_q)^2$ in (20). Thus, all of the data points are being treated more equally, as opposed to when l is small, and the closest points have a more significant relative influence.

In figure 9, one can observe the effect of varying the signal variance. Looking at the location of σ_f^2 outside of the exponential in (20), one would expect to see the function stretched in the direction of the dependent variable,(vertically) and observing figure 9 that is the case. Notice how the peaks and valleys of the higher signal variance model are more extreme while the general shape remains similar.

Adding noise to the equation has a much different effect. In this scenario, the model is accounting for the potential that the training data is not completely accurate. This most notably allows the function generated from the posterior distribution (as well as the mean of $\mathbf{f}_*$) to not pass through the training points. Of course, then there will also be non-zero variances at the training observations. This is accomplished by adding $\sigma_n^2 I$ to the training data kernel matrix, resulting in the distribution below in (21).

$$\begin{aligned}\mathbf{f}_*|X_*, X, \mathbf{f} \sim & N(K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}\mathbf{f}, \\ & K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}K(X, X_*)\end{aligned} \tag{21}$$

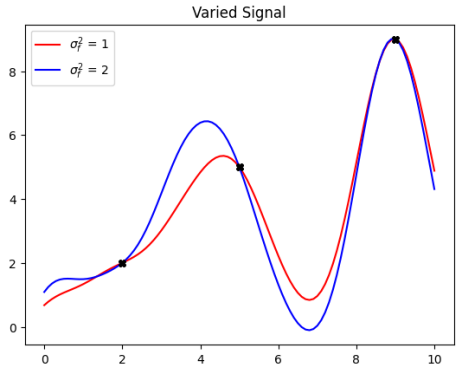

Figure 9: Observe the effects of varying the signal variance. The function in red was generated with the hyper-parameters $l = 1$, $\sigma_n^2 = 0$, and $\sigma_f^2 = 1$, while the function in blue uses $\sigma_f^2 = 2$, and all other variables are held constant. The increase in σ_f^2 leads to a function stretched in the vertical direction.

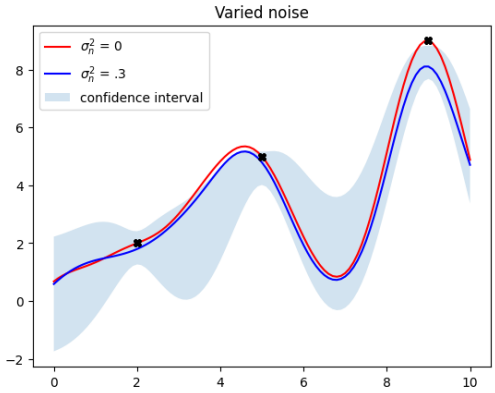

Figure 10: Observe the effects of varying the noise variance. Here, the red function remains the same as the other figures, but $\sigma_n^2 = .3$ when generating the blue function. The confidence band corresponds to the model generating the blue curve.

As can be seen in figure 10, not only does the curve not pass exactly through the points, but the confidence interval need not even be centered around the observations.

Notice that in many of the posterior plots in this section, the curves almost seem to "pull" away from the data points and towards zero. Remember that training these models involves conditioning a prior distribution with zero mean. This effect can be thought of as the model weighing the influence of the prior assumptions over the observations when farther in the domain space from the training points. This tendency becomes even more apparent when the length-

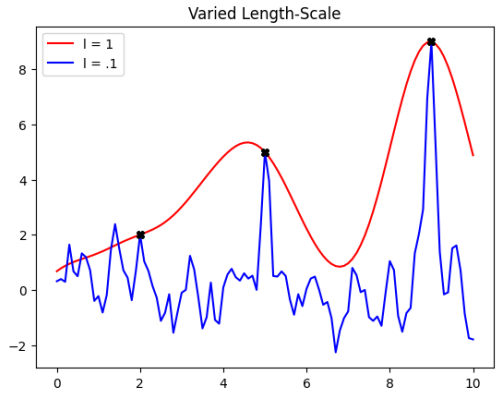

Figure 11: Observe the effects of a small length-scale relative to the distance between observations. The functions seem to default to the prior assumptions of mean zero quickly at test points far from the training points in the domain space.

scale gets very small relative to the distances between training points, as can be seen in figure 11.

3.4 Hyper-Parameter Optimization

While understanding the effects of the hyper-parameters on the resulting GPM is useful, one may still wonder how to go about choosing values for these hyper-parameters. Fortunately, a logical process for this exact purpose is both possible and easier to derive than in many other machine learning models. Recall the marginal likelihood in the weight space view, which is expressed as:

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w} \tag{22}$$

In the function-space view, the marginal likelihood takes the form:

$$p(\mathbf{f}|X) = \int p(\mathbf{f}|X, \mathbf{f}_*)p(\mathbf{f}_*|X)d\mathbf{f}_* \tag{23}$$

A likelihood, as its name suggests, provides information on the likelihood of the data given the model. Here, the likelihood of the training target values $\mathbf{f}$ is marginalized over the test target values $\mathbf{f}_*$ giving rise to the term "marginal." As this marginal likelihood is dependent on the model and, therefore, on its hyper-parameters, it makes sense to find the hyper-parameters that optimize this value. To make this easier in practice, the log marginal likelihood is instead used as the objective function for optimization, which is an equivalent problem. It is already known that $\mathbf{f}_*|X \sim N(\mathbf{0}, K)$, so by the properties of normal distributions:

$$\log\left(p(\mathbf{f}_*|X)\right) = -\frac{1}{2}\mathbf{f}_*' K^{-1}\mathbf{f}_* - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi \tag{24}$$

It is also known that $\mathbf{f}|\mathbf{f}_* \sim N(\mathbf{f}_*, \sigma_n^2 I)$. The product of the two Gaussian densities in (23) can be rewritten as another scaled Gaussian density after several algebraic steps. Integrating the resulting density and taking the log, it can be shown that:

$$\log\left(p(\mathbf{f}|X)\right) = -\frac{1}{2}\mathbf{f}'(K+\sigma_n^2 I)^{-1}\mathbf{f} - \frac{1}{2}\log|K+\sigma_n^2 I| - \frac{n}{2}\log 2\pi \tag{25}$$

Looking more closely at (25), one can see that only the first term $-\frac{1}{2}\mathbf{f}'(K+\sigma_n^2 I)^{-1}\mathbf{f}$ includes the target values $\mathbf{f}$. This term serves the role of the data fit and changes monotonically with adjustments to each hyper-parameter. That is to say that the data fit increases when the model becomes more flexible (as the length-scale decreases or the variances increase). The second term $\frac{1}{2}log|K+\sigma_n^2 I|$ is the complexity penalty which has the opposite relation with the hyper-parameters (in terms of direction not magnitude). Finally $\frac{n}{2}log2\pi$ is simply a constant. Due to the relations between the hyper-parameters and the two primary terms of (25), as with many optimization problems, one can see that this optimization comes down to a balancing act between finding the best fit for the data and keeping the model less complex. One can imagine that this optimization can be quite difficult when the dataset becomes large in dimension and observation count due to the many factors affecting the objective function. In section 4.3, the relation between the hyper-parameters and the marginal likelihood will be further explored.

To optimize (25), as with most optimization problems, the next step is to take the derivative of this expression with respect to each hyper-parameter Θ_j. Fortunately, of the elements in (25) only K depends on Θ. Using the properties of matrix derivatives available in [3], along with the chain rule, it follows that:

$$\begin{aligned}\log\left(p(\mathbf{f}|X,\Theta)\right) =& \frac{1}{2}\mathbf{f}'K^{-1}\frac{dK}{d\Theta_j}K^{-1}\mathbf{f} - \frac{1}{2}\mathrm{tr}(K^{-1}\frac{dK}{d\Theta_j}) \\ &- \frac{1}{2}\mathrm{tr}((\alpha\alpha' - K^{-1})\frac{dK}{d\Theta_j})\end{aligned} \tag{26}$$

where K implicitly includes the noise term $\sigma_n^2 I$, and $\alpha = K^{-1}\mathbf{f}$.

While it is technically possible to find the exact solution to this optimization problem, given the time complexity of these methods, it is more practical to approximate the optimal solution through gradient ascent. Gradient ascent is an optimization technique that repeatedly finds the optimal direction to traverse, given the current parameter values, to give the maximum increase in some objection function. This is done by taking the derivatives with respect to each variable needing optimization (the hyper-parameters l,σ_n^2, and σ_f^2 in this case), and stepping in a direction with the "steepest" ascent. There are some drawbacks to this technique, in particular, the possibility of finding a local maximum rather than a global maximum. The trade-off is generally worthwhile, as the risk can be lessened with only a linear increase in runtime by performing the gradient ascent algorithm multiple times with different initial values and

choosing the solution that achieves the highest value for the objective function. This estimated optimal solution will be referred to as the maximum likelihood estimate (MLE).

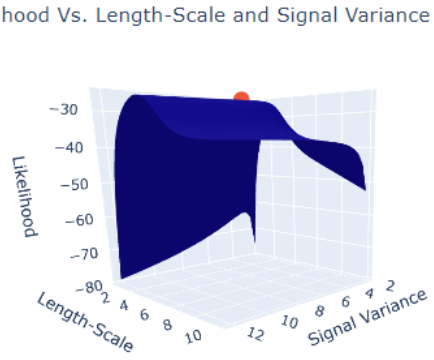

Figure 12: This plot shows the log-marginal likelihood of the GPM trained for many combinations of length-scale and signal variance on a single dataset (the dataset will be introduced in section 4.2). The x and y-axes show the length-scale and signal variance, while the z-axis shows the log-marginal likelihood. The noise variance is held constant at .3. In this case, the length-scale appears to have a larger impact on log-marginal likelihood than the signal variance. However, near the right edge, it can be seen that the two hyper-parameters are not independent in their impact on the marginal likelihood as the curvature seems to take a sharp turn downward. The red point marks the maximum likelihood estimate for the hyper-parameters.

3.5 GPM in Scikit-Learn

The Python library sci-kit-learn contains an implementation of these GP models, as well as hyper-parameter optimization. The GaussianProcessRegressor function takes in a kernel along with several other optional parameters [4]. One can define the kernel of the GPM by using any combination of kernel functions available. In this case, a constant kernel times a radial basis function kernel is used to create the squared exponential kernel without noise that has been used up to this point. Here, the constant kernel represents the signal variance, and the RBF kernel is equivalent to the main portion of the squared exponential where the only hyper-parameter is l. In this case, the range for the hyper-parameters was set to fixed, but if it is necessary to train the hyper-parameters, the range can be set to some reasonable bounds.

$$\mathrm{C}(\sigma_f^2 = 1, range = "fixed") * RBF(l = 1, range = "fixed") \quad (27)$$

Noise variance can also be handled by GaussianProcessRegressor through the alpha parameter but cannot be optimized. As a result, for the experiments

in the rest of this paper, the noise variance is set to some constant, which will be noted when necessary. Once the parameters of the function are defined, the model can be fit to the training data using the fit(x,y) function, which is essentially creating the vector $(K(X,X) + \sigma_n^2 I)^{-1}\mathbf{f}$ used to calculate the mean vector (as well as in the covariance) in the Gaussian posterior. At this point, the predict(x_*) function can be applied to get the expected value at test points. This method can also return either the covariance matrix or the standard deviations, which is simply the diagonal of the covariance matrix with the elements square-rooted. The sample_(x_*.) function draws a sample from the trained posterior model with a dimension equal to the number of values in x_*.

This library was used to check the implementation of GPM created in the early stages of this research project. This library will also be used for many of the future experiments in place of our implementation for efficiency reasons. Their implementation of the gradient ascent algorithm in particular is much more efficient and accurate than ours.

4 Experiments

4.1 Shifting and Scaling of Data

Some research, including that of Anirudh et al. [1] has found that non-parametric models can behave unexpectedly when transformations such as shifting and scaling are applied to the data. They observe that the resulting models do not maintain their original shape, albeit only slightly in most cases. They used this observation as the foundation of their novel uncertainty metric, which they named $\Delta - UQ$. In the initial stages of this project, there were plans to compare this uncertainty metric proposed by Anirudh et al. to the GPM uncertainty discussed in this paper. However, to pursue this goal, it was found that the use of non-shift or non-scale invariant kernels would be required, and thus, the research took another direction. Still, it can be instructive to see how data transformation impacts the GPM with a squared-exponential kernel and how the hyper-parameters can be adjusted to counteract these changes. First, shifting the input data had no impact on the shape of the function, as can be seen in figure 13. This is because the squared exponential kernel evaluation discussed in section 3.3 is exactly the same when both x_q and x_p are shifted by the same amount.

On the other hand, scaling initially leads to a shocking difference, as seen in figure 14, where the inputs and outputs are scaled by a factor of 10. However, digging deeper, this result makes a lot of sense. Here, the hyper-parameters have not been changed to account for the increase in scale. This may seem familiar to what was seen in section 3.3 with figure 11, and it is, in fact, the opposite side of the same coin. The length-scale is once again too small relative to the distance between observations in the domain, leading to the model relying more on the prior distribution of the response than on the observations. Adjusting the length-scale will lead to a more reasonable result, but the shape is still not

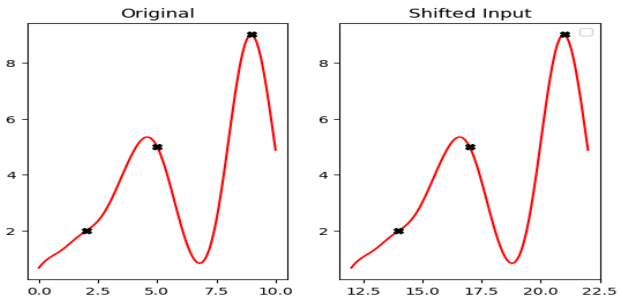

Figure 13: Notice that the two plots are identical besides the domain of X. This is due to the fact that shifting the input has no effect on the squared exponential kernel and thus on the shape of the GPM.

quite the same, as can be seen in figure 15.

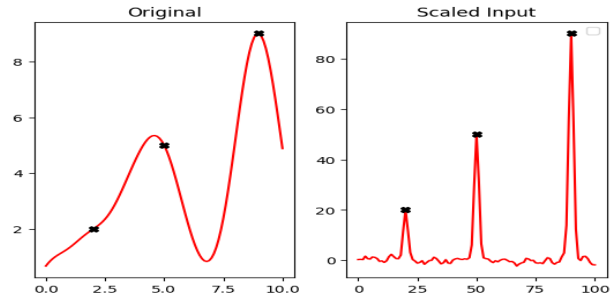

Figure 14: Unlike in figure 13, it can be seen here that scaling the data has a huge impact on the shape of the GPM. This is due to the fact that scaling the domain does have an impact on the squared-exponential kernel if l is held fixed.

This difference is due to the signal variance not being scaled to account for the scaling of the output. After adjusting σ_f^2 by a factor of ten, the same shape is indeed achieved, as seen in figure 16. The math behind why this scaling of σ_f^2 will be further discussed in section 4.2.

Finally, a shift of the output variable was attempted, and it was found that despite adjusting the hyper-parameters appropriately, the models do not align. This is due to the prior distribution on the response remaining the same with a $\mathbf{0}$ mean. With the data now lying farther from the prior assumptions, this original mean "pulls" the response towards it. This makes the choice of an appropriate length-scale all the more important the larger the scale of the response variable in the data, and may even warrant a data transformation before the training of the GPM for some datasets.

Seeing how the scale of the domain and length-scale interact may raise an

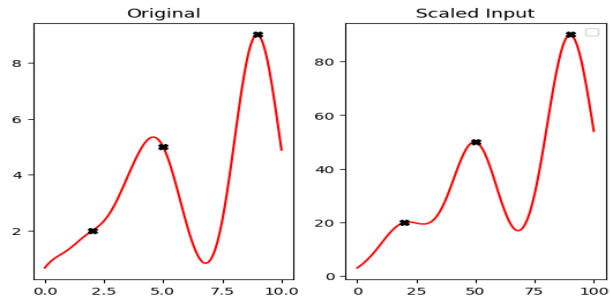

Figure 15: After scaling l, the panels look much more similar but are still not quite the same. This is because of the presence of $\mathbf{f}$ (the observed response) in (19), which is now scaled, leading to a different evaluation for the mean portion in the posterior model.

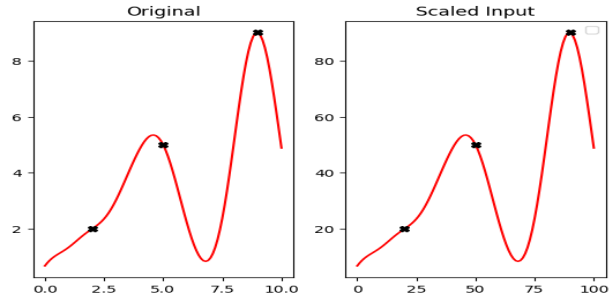

Figure 16: After properly adjusting both l and σ_n^2 see that the shape of the GPM can be maintained through scaling of both the domain and range of the data.

important question for the reader. What if the data has many variables, and the scale of these variables is drastically different? While so far (and for the rest of this paper) the examples have all been in two dimensions for ease of visualization, GPMs can be used with any number of variables. In the first panel figure 18, it can be seen that the wrong choice of length-scale can lead to poor predictions due to the extreme tendency for the response to be pulled toward zero. However, increasing length-scale to counteract this may lead to mistreatment of the variable with a smaller scale. In this case, it is important to consider re-scaling the data to the same range or using different length-scales for each variable.

The second panel of figure 18 shows a function drawn from a GPM where a different length-scale was set for each variable. To account for this change, the squared exponential kernel requires a slight adjustment. The squared exponential kernel takes the form seen in (28) in the multivariate case.

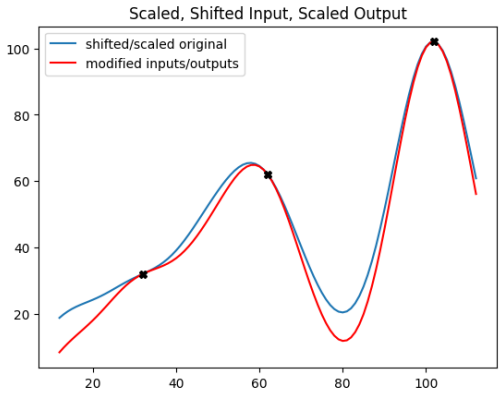

Figure 17: Shifting of the range of the data is not so easy to adjust for. In this plot, the blue curve, which tends to sit higher than the red curve, is a transformation of the output of the GPM trained on the original data. The other curve shows the results of the GPM trained on the transformed data. The model tends to pull the response towards zero here due to the prior model's assumption of zero mean.

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)' M (\mathbf{x}_p - \mathbf{x}_q)\right) + \sigma_n^2 \delta_{pq} \tag{28}$$

In (28) M is a matrix representing the length-scales of the kernel. In the first panel of figure 18, $M = l^{-2}\mathbf{I}$, to reflect the fact that all length-scales are the same. Alternately, in panel 2, $M = \text{diag}(\boldsymbol{l})^{-2}$, where $\boldsymbol{l}$ is a vector of positive values which need not be the same. This gives the kernel the freedom for l to vary between variables. A GPM was trained using this slightly modified kernel. This kernel includes a very small length-scale in the x_2 direction and a larger one in the x_1 direction. This alternate posterior generates functions that are very smooth throughout the domain of the data, as seen in panel 2.

In both of these cases, M is a diagonal matrix. As a result, the entries of M impact each variable independently. Adding some non-diagonal matrix Λ can allow the specification of length-scales that handle the interaction of variables.

4.2 Effect of Hyper-Parameters on Uncertainty

It is now clear that hyper-parameters play a significant role in determining the shape of the predictive surface of the GPM posterior, as well as the shape of the functions drawn from this posterior. It should come as no surprise then that the hyper-parameters also influence the uncertainty of the model, as this uncertainty is also dependent on the kernel function as seen in the distribution of the posterior (29). In fact, unlike some other models (in particular parametric models), the hyper-parameters, kernel, and independent variables are the only contributors to uncertainty in GPMs as seen by the lack of $\mathbf{f}$'s presence in the

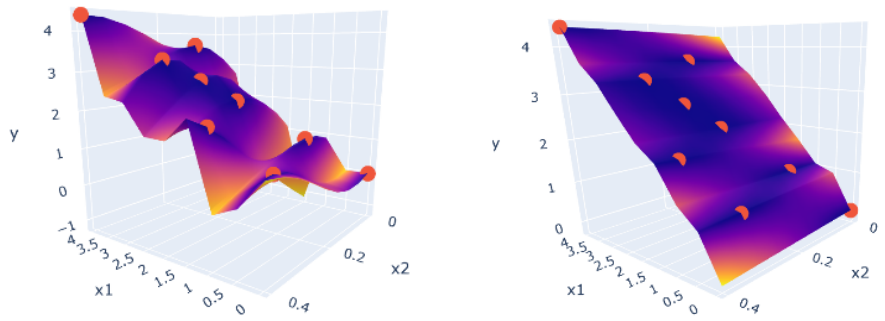

Figure 18: The first panel shows what may occur if a length-scale small in scale relative to the scale of one of the variables is chosen. Notice that in the x_2 direction, the curvature of the surface is quite smooth, while in the x_1 direction, the surface is very jagged. In this case $M = \begin{bmatrix} .2 & 0 \\ 0 & .2 \end{bmatrix}$ The second panel shows the results of assigning different length-scales to each variable, leading to a smooth surface throughout. Here $M = \begin{bmatrix} 2 & 0 \\ 0 & .2 \end{bmatrix}$

.

covariance matrix in (29).

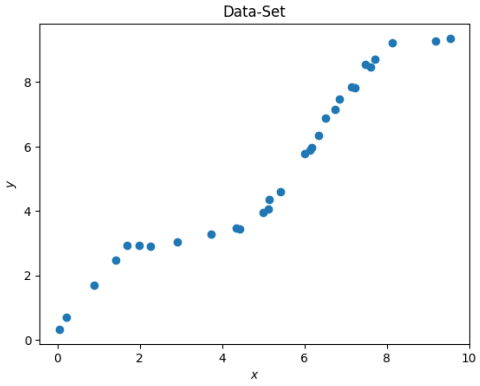

Figure 19: Data generated from the equation $y = X + sin(X) + \epsilon$, where ϵ is generated uniformly over a small interval to add randomness.

$$\begin{aligned} \mathbf{f}_*|X_*, X, \mathbf{f} \sim & N(K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}\mathbf{f}, \\ & K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1}K(X, X_*) \end{aligned} \tag{29}$$

Now consider again the squared exponential kernel shown in (30) below. Taking a closer look at this equation, one can, at the very least, see the effect

each hyper-parameter will have on the kernel. It is very clear to see that as σ_n^2 and σ_f^2 increase, so too does the magnitude of the kernel. The same holds true for the length-scale, as an increase in l leads to a decrease in the magnitude of the exponential. Since the exponential is negative, this leads to larger values for the kernel. This, however, is only a piece of the puzzle when evaluating the effects of these hyper-parameters on the uncertainty, as the second term in the variance of (29) is more difficult to understand.

$$k_y(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^2\right) + \sigma_n^2 \delta_{pq} \tag{30}$$

Before trying to break down (29), it makes sense to look at the relationship graphically. For this purpose, a two-dimensional dataset of thirty points has been created. The dependent variable X was generated randomly between zero and ten, and the dependent variable $y = X + sin(X) + \epsilon$, where ϵ is generated uniformly over a small interval to add randomness. By training many GPMs while varying only one hyper-parameter at a time, one can see the effects of each hyper-parameter upon the uncertainty. Here,"total uncertainty" is used as an estimate for the uncertainty and is calculated as the sum of the diagonal entries of the covariance of the response in the posterior model. The diagonal is used for this purpose as these elements of the covariance matrix correspond to the variance at each test point used in the model. In contrast, the off-diagonal elements represent the pair-wise covariance of each test point with each other and are thus not incorporated into this metric.

The plots in figure 20 suggest that there is a monotonic relation between each hyper-parameter and the uncertainty, with that relationship being negative for l, and positive for σ_f^2 and σ_n^2. One may be surprised by this relation, thinking that choices of hyper-parameters with high marginal likelihood would lead to models with lower uncertainty. As seen in section 3.4, there is not a monotone relationship between the hyper-parameters and the marginal likelihood, as that is the objective function for optimizing these values. If this were the case, the marginal likelihood would be a poor choice for optimization as there would never be a bounded solution. The covariance function of the GPM (which is what is used for uncertainty), however, as mentioned at the start of this section, is completely independent of $\mathbf{f}$. So if the covariance will be the same no matter the values of $\mathbf{f}$, so long as X and X_* remained unchanged, the uncertainty in GPMs is simply a measure of confidence in prediction under the assumption that the selected kernel and hyper-parameters are the underlying truth for the dataset. Figure 21 shows this lack of relation between total uncertainty and marginal likelihood.

When trying to understand the mathematical reasoning behind these monotonic relationships, it is easiest to start with the first term of the covariance matrix in (29), which is rewritten on its own below in (31). This term is simply the kernel evaluated for all test points. Therefore, the diagonal elements are distances calculated for each test point with itself, meaning the exponential term will become one regardless of the length-scale's value, leaving a diagonal with all elements equal to σ_f^2. This means that the first term increases linearly with σ_f^2

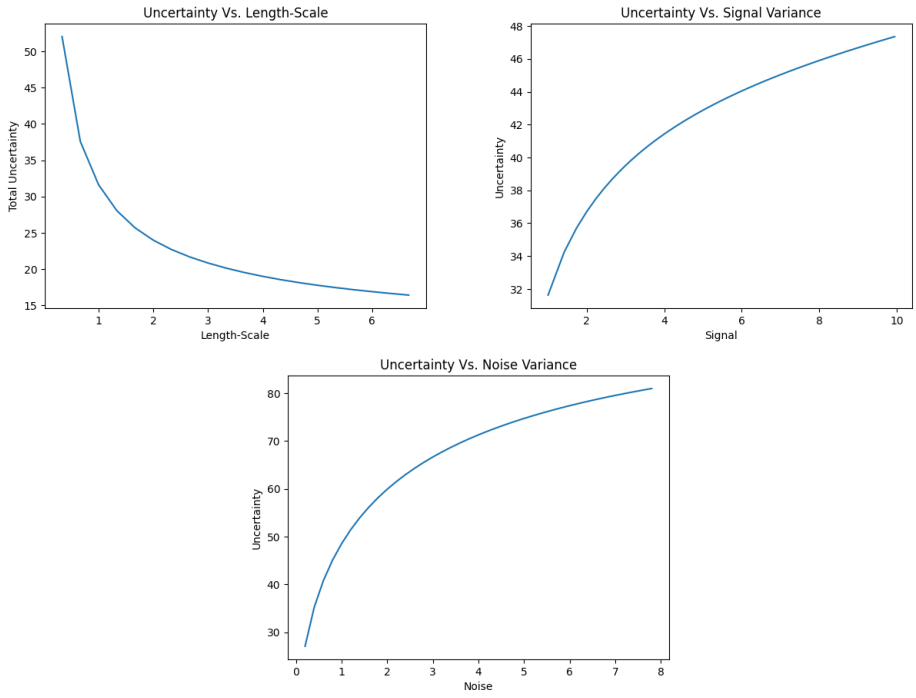

Figure 20: The plots show the relationship between a GPM's uncertainty and the three hyper-parameters of the squared exponential kernel. When varying each hyper-parameter the others were held constant. In these cases their values are $l = 1$, $\sigma_f^2 = 1$, and $\sigma_n^2 = .3$.

and is unaffected by length-scale. Noise variance is also irrelevant to this term as it is only used when evaluating the covariance of the training observations.

If the observation that increased length-scale leads to a decrease in uncertainty holds for all datasets, then the diagonal elements of the second term of (31) must increase, as the first term's diagonal is unaffected. While the length-scale can not be factored out nicely due to the exponential term, one can see that all elements of $K(X_*, X)$ and $K(X, X_*)$ will increase as the length-scale increases, as explained previously. The off-diagonal elements of $(K(X, X)+\sigma_n^2 I)$ will also increase, while the diagonal will remain unaffected. Unfortunately, this covariance function also needs to be inverted, making confirming this relation quite tricky. In the future, it may be worth exploring this relation mathematically, but in all experiments, we found that it held.

$$K(X_*, X_*) - K(X_*, X)(K(X, X) + \sigma_n^2 I)^{-1} K(X, X_*) \tag{31}$$

It is easy to see why there is a linear relation between uncertainty and signal variance in the no-noise case. Without the noise term, the signal variance can be

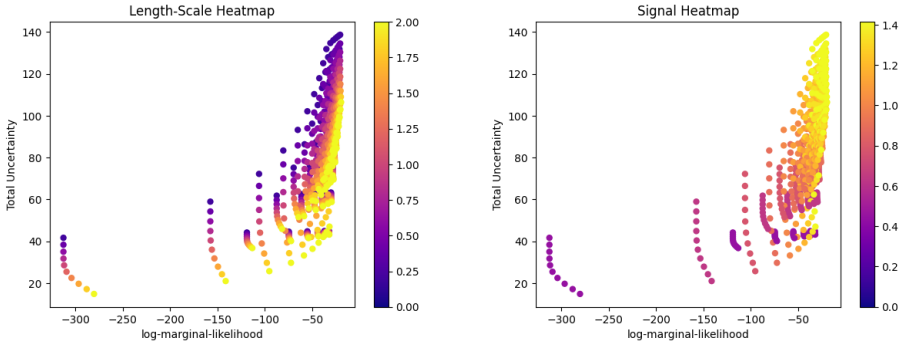

Figure 21: Both plots show the lack of relation between uncertainty and log-marginal likelihood in GPMs trained for many combinations of hyper-parameters. In the first panel, the color of the points represents the value of l, while in the second panel, it represents σ_f^2. Notice that there seem to be "strands" of data points and that these "strands" vary in color in the left graph but not the right. This seems to imply that proportionally similar changes to l have a smaller impact on the marginal likelihood than changes to σ_f^2. It is also clear to see that not all of these "strands" take the same shape, implying that for different values of the σ_f^2 and σ_n^2, l has a different impact on the marginal likelihood, which will be discussed in section 4.3 Most importantly these plots once again confirm the lack of relation between marginal likelihood and total uncertainty.

factored out of each covariance matrix. The inverse in the second term leads to a canceling of two of these factored signal variances, leaving σ_f^2 leading both terms. In the noisy case, an issue similar to the one encountered when analyzing length-scale arises. Figure 22 shows that for this dataset, the monotonic relation of the hyper-parameters and uncertainty holds for all used combinations of hyper-parameters.

One may notice that figure 20 does not reflect this linear relation between σ_n^2 and total uncertainty as $\sigma_n^2 \neq 0$. This is the case because attempting to train GPM priors with $\sigma_n^2 \neq 0$ often leads to errors in computation. This is likely the result of observations that are too close together in the domain, resulting in kernel matrices that are not well-suited for inversion.

4.3 Hyper-Parameter Re-estimation

In section 3.4, the optimization of hyper-parameters using the log-marginal likelihood as the objective function was discussed. Recall that this optimization is generally performed as an estimation through gradient accent. Gradient accent is most known to have issues when there are many local maximums or very "flat" surfaces in the objective function. Figure 23 shows an example of this log-marginal likelihood surface with varying l, and σ_f^2. Even over this small range of hyper-parameters with a relatively simple dataset, one can see that this surface is quite complicated. While l is small, even minuscule increases

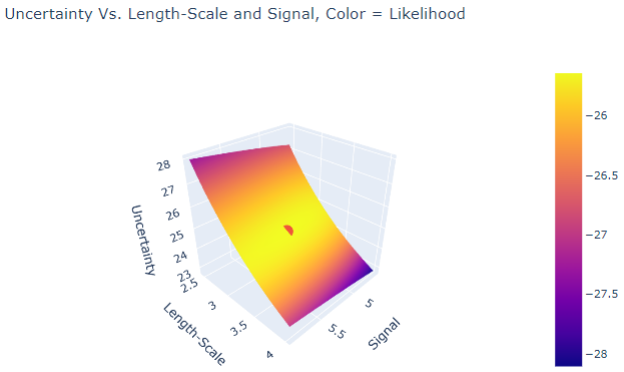

Figure 22: This plot shows the total uncertainty for many combinations of length-scale and signal variance. The x and y-axes show the length-scale and signal variance, while the z-axis shows the total uncertainty. The noise variance is held constant at .3. In this case, the length-scale appears to have a larger impact on uncertainty than the signal variance. The red point marks the maximum likelihood estimate for the hyper-parameters, and the color scale shows the log-marginal likelihood. These factors will be discussed more in the following section.

to l lead to significant improvements in terms of marginal likelihood. On the other hand, for the most part, over these values of the hyper-parameters σ_f^2 seems to have little impact on the marginal likelihood. One can imagine that when adding more dimensions to the data or more complex relations between the variables, these objective surfaces will only get more complicated.

The complexity of these surfaces raises questions as to how stable this optimization process is and, more specifically, how accurately it can estimate the hyper-parameters of the true distribution. With this in mind, an experiment was conducted to test the relationship between hyper-parameter re-estimation, the marginal likelihood, and the shape of the marginal likelihood surface seen in figure 23. A GPM was trained for several combinations of hyper-parameters in the range seen in figure 23. Each model was used to generate 10 sample datasets of 100 data points, each using the same values for X_* as in the original model. Call the j-th sample generated from the i-th original model (trained on the original dataset with some unique set of hyper-parameters) y_{ij}. Each new dataset $[X_* y_{ij}]$ was then used to retrain a new GPM m_{ij} and estimate the optimal hyper-parameters. Once all 10 of the sample datasets generated from the same initial GPM were used to re-estimate the hyper-parameters, that is, we have m_{ij} for $j = 1, ..., 10$, the standard deviation of these new hyper-parameters was stored as σ_i. The results of this experiment on the dataset used for section 4.2 and displayed in figure 19 can be seen in figure 24. This figure shows the same surface as figure 23 but colored by the value of σ_i found in the experiment using the corresponding hyper-parameters.

Studying these plots, there does seem to be some relation between the

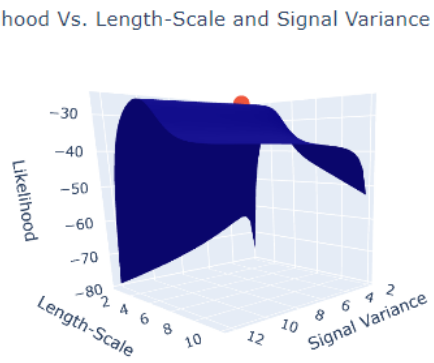

Figure 23: This plot shows the log-marginal likelihood of the GPM trained for many combinations of length-scale and signal variance on the same dataset. The x and y-axes show the length-scale and signal variance, while the z-axis shows the log-marginal likelihood. The noise variance is held constant at .3. The red point marks the maximum likelihood estimate for the hyper-parameters.

marginal likelihood seen by the darkest blue regions laying along the "ridge" that the optimal combination of hyper-parameters sits on. This is made more interesting by the fact that datasets generated along this ridge are not the datasets with the lowest uncertainty, as seen in section 4.2. Despite these datasets having more variance than some others (particularly where l is larger), the GPM optimization is more accurately able to re-estimate the hyper-parameters.

When conducting this experiment, we believed that the more likely relation would be that the hyper-parameter estimation error would be more correlated with the slope of the surface than the marginal likelihood itself. Looking at the ridge of the graph mentioned before, it can be seen that for the original dataset, the marginal likelihood does not drastically favor any particular value for the signal variance. One might then think that it may be harder to re-estimate signal variance in this scenario, but that did not seem to be the case. Instead, both hyper-parameters appeared to follow a similar pattern for all choices of the original hyper-parameters used in the experiment.

This experiment was repeated five more times on different datasets. It was somewhat difficult to execute in mass, as the runtime is quite slow due to the hundreds of models trained, each requiring hyper-parameter optimization. In addition, the graphical portion of the experiment makes it easiest to interpret the results of each iteration of the experiment individually. The results of the other iterations of the experiment did not strongly contradict the observations above but did not perfectly align with them either. σ_i was found to be quite low at the MLE of the hyper-parameters in all cases, but in some cases, more extreme values of the hyper-parameters seemed to result in lower σ_i.

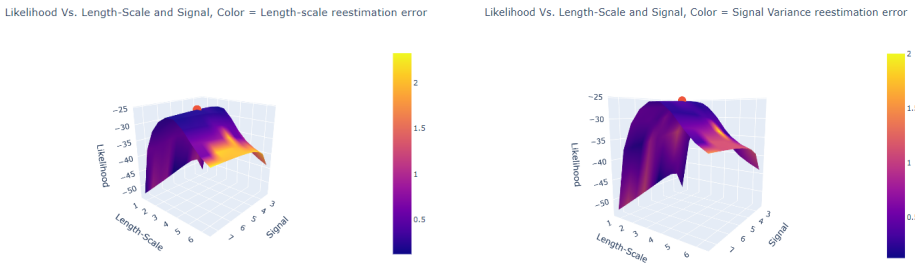

Figure 24: These plots are the same surface seen in figure 23, where the color depends on the hyper-parameter re-estimation for data generated from GPMs trained on the same dataset with different hyper-parameters. The first panel shows the re-estimation error of l, while the second panel shows the re-estimation error of σ_f^2.

5 Conclusion

In conclusion, this paper has provided a detailed examination of Gaussian Process Models (GPMs) and their relevance in regression tasks. It began by highlighting the limitations of traditional parametric regression models, which often struggle to capture the complex relationships present in real-world data. Non-parametric models, such as GPMs, offer a flexible alternative by allowing the underlying function to adapt to the data without imposing rigid assumptions.

A deep dive was conducted into the fundamental concepts of GPMs, including their mathematical formulation, key properties, and practical considerations, such as the potential need for data transformations or modifications to the properties of hyper-parameters used. It was discussed how GPMs define a distribution over functions, enabling uncertainty quantification and robust predictions, even in scenarios with noise or limited data. The role of the hyper-parameters of the squared-exponential kernel and their effects on the model predictions and covariance was explored in detail. Furthermore, the role of uncertainty estimation and marginal likelihood in enhancing the performance of GPMs was investigated through experiments conducted on simulated datasets. These simulations made evident just how complicated GPM optimization can be while discussing the potential of a relationship between the marginal likelihood and hyper-parameter re-estimation.

The overarching goal of this paper has been to provide a more intuitive introduction to GPMs, answering many questions likely to be posed by anyone new to these models while attempting to make some useful observations on the relationship between key components of Gaussian Process modeling, such as uncertainty, marginal likelihood, and hyper-parameters. While the research did not conclusively uncover any hidden secrets of GPMs, we hope that by contributing to the understanding of non-parametric methods and their relevance in data-driven research and applications, further developments or intuition may come easier.

Bibliography

[1] R. Anirudh et al. *Single Model Uncertainty Estimation via Stochastic Data Centering*. Tech. rep. Lawrence Livermore National Laboratory, 2022.

[2] R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis*. 6th ed. Result 4.6. Pearson, 2007, p. 160.

[3] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[4] F Varoquaux et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.