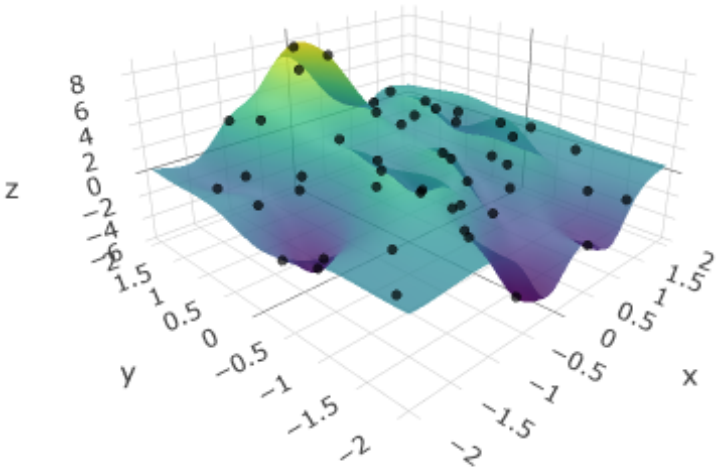
Gaussian Process Regression

Jennifer Sailor, Josh Seidman, Mark Rapala

Date: April 25, 2024



MSSC-6250 Statistical Machine Learning

Cheng-Han Yu

Introduction

For the final project, we've chosen to learn Gaussian process regression (GPR). Since Gaussian processes come up frequently in applied math, we figured that it would be a useful technique to learn and become more familiar with using.

Gaussian process regression is also known as Kriging, is a supervised machine learning method, courtesy of a mathematician named Danie G. Krige from South Africa. He wrote a master's thesis about estimating the most likely distribution of gold based on a few samples of boreholes. From there, French mathematician Georges Matheron developed the theoretical basis for GPR. [1]

Gaussian process regression is a non-parametric Bayesian approach for inference. We start with a prior distribution, then using the samples we've observed, transform it into a posterior distribution. To do this, we also need to choose a kernel function (covariance function), which will determine how the model is fitted to the sample points. For example, linear kernels will lead to a linear fit. From there, we can create a joint distribution, which follows a Gaussian distribution with a predetermined mean function (usually the zero function), and covariance matrix obtained through the kernel function.

Now we want to predict new points. Because the joint distribution of observed values and predicted values is Gaussian, we can use the observed data points to make predictions for new points. [4]

One of the pros of GPR is that because the predicted points follow a Gaussian distribution, it is relatively easy to construct confidence intervals for new predicted points, which allows for uncertainty quantification when it comes to predicting new points. GPR is also very flexible and is usually not sensitive to choice of mean function and noise.

A few cons of GPR are the computational complexity, and interpretability. It can be extremely computationally intensive when using larger datasets, and predicting new points will be just as intensive. Additionally, using more complex kernels, while it does lead to better predictions, usually results in something of a black box in terms of being able to interpret how the model is working and predicting new points.

Model/Method

Gaussian Process

To do Gaussian Process Regression (GPR), we must start with understanding the Gaussian Process (GP). The mathematical notations in this section are all from Dr. Yu's lecture on Gaussian Processes [5]. A GP is a process where all finite-dimensional distributions are multivariate Gaussian, for any choice of n and $x_1, \cdots, x_n \in \mathbb{R}^D$:

$$f(x_1), \cdots, f(x_n) \sim N_n(\mu, \Sigma).$$

Similar to how we need a mean and covariance to define a Gaussian distribution, we will need a mean and covariance function to define a Gaussian process.

$$f(\cdot) \sim GP(m(\cdot), k(\cdot, \cdot))$$

where $E(f(x)) = m(x)$ and $Cov(f(x), f(x')) = k(x, x')$. The mean function, $m(\cdot)$, has many popular choices with the most popular being having the mean of the x equal to zero, $m(x) = 0$. Other choices of the mean function are $m(x) = c$ for some constant c, or $m(x) = \beta' x$ for some vector β. Fortunately, Gaussian processes are flexible enough to be able to accommodate a variety of mean functions, and so this choice is usually not that important. We'll be using $m(x) = 0$ in our simulation study.

The more important function choice is the kernel or covariance function, $k(\cdot)$. This has the biggest influence on the GP because the kernel defines the similarity between data points. The kernel function must be a positive semi-definite function to be considered a valid covariance matrix. When using a kernel function we can often assume that it is a function that finds the distance between locations. This means that GP is a stationary process. The process must be stationary because it means that the distribution does not change over time. If it was non-stationary then the data is unpredictable and cannot be modeled. Additionally, if the distance is based on something similar to Euclidean distance we can conclude that the kernel does not depend on the direction or is isotropic. Since there are many ways to calculate the distance between locations there are many different types of kernels. Some kernels include the squared exponential kernel, exponential kernel, Brownian motion, or the Matérn kernel.

Gaussian Process Regression

Gaussian Process Regression takes the results of Gaussian Process and uses it as the prior distribution. In this setting, the function prior is defined with the mean function set to 0, implying no bias towards any specific function behavior, and a covariance function (or kernel function) that changed the smoothness and correlations between the function values at different inputs.

$$f(\cdot) \sim GP(0, k(\cdot, \cdot))$$

This prior applies to all the observed data points $x_1, \cdots, x_n$ as well as any new predicted points x^*. These predictions are treated as random variables following a joint Gaussian distribution along with the observed function values created by the covariance function.

$$f(x_1), \cdots, f(x_n), f(x^*) \sim N(0, \mathbf{\Sigma})$$

It is important to note that the covariance matrix $\mathbf{\Sigma}$ is structured such that it consists of the covariance between observed points K, covariance between observed points and new points K_*, and the covariance between new points K_{**}.

$$\mathbf{\Sigma} = \left(\begin{array}{ccc|c} k(x_1, x_1) & \cdots & k(x_1, x_n) & k(x_1, x^*) \\ \vdots & & \vdots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) & k(x_n, x^*) \\ \hline k(x^*, x_1) & \cdots & k(x^*, x_n) & k(x^*, x^*) \end{array}\right) = \left(\begin{array}{c|c} K & K_* \\ \hline K_*^T & K_{**} \end{array}\right)$$

This structure makes Gaussian Process Regression very efficient when it comes to the computation of predictions and uncertainty estimates.

Now shifting to looking at the posterior distribution. The posterior distribution includes updated information about the regression function after incorporating observed data points. Given the observed information $f(x_1), \cdots, f(x_n)$, the posterior distribution of the function values at the new points x^* is represented as a Gaussian distribution.

$$f(x^*)|f(x_1), \cdots, f(x_n) \sim N(\boldsymbol{\mu^*}, \mathbf{\Sigma^*})$$

Where the mean and covariance matrix of the posterior distributions are computed using the observed data and covariance structure of the Gaussian Process:

$$\boldsymbol{\mu^*} = K_*^T K^{-1}\mathbf{f} \qquad \boldsymbol{\Sigma^*} = K_{**} - K_*^T K^{-1} K_*$$

where $\mathbf{f} = (f(x_1), \cdots, f(x_n))^T$. The covariance matrix of the posterior distribution, $\boldsymbol{\Sigma^*}$, accounts for uncertainty in the predictions. This posterior update process enables Gaussian Process Regression to provide not only the point estimates but also quantify uncertainty.

One additional step that can be done with Gaussian Process Regression is to add a Nugget term. This is used when we can't observe the true $f(\cdot)$ (usually due to noise) and we would like to estimate it. In this setup, we model the observed data y_i as noisy results of the true function $f(x_i)$, incorporating some Gaussian noise with variance σ^2. Therefore our setup changes to:

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim N\left(0, \sigma^2\right)$$
$$f(\cdot) \sim GP(0, k(\cdot, \cdot; \theta))$$

$$y_1, \ldots, y_n, f(x^*) \sim N(0, \boldsymbol{\Sigma}), \quad \boldsymbol{\Sigma} = \left(\begin{array}{c|c} K + \sigma^2 I & K_* \\ \hline K_*^T & K_{**} \end{array}\right)$$

$$f(x^*) \mid y_1, \ldots, y_n \sim N\left(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*\right), \quad \boldsymbol{\mu}^* = K_*^T \left(K + \sigma^2 I\right)^{-1} \mathbf{y}, \quad \boldsymbol{\Sigma}^* = K_{**} - K_*^T \left(K + \sigma^2 I\right)^{-1} K_*$$

with $\mathbf{y} = (y_1, \ldots, y_n)^T$. These formulas are essentially the same as the formulas with no nugget, but using the observed data instead of the true function values and substituting $K + \sigma^2 I$ for K.

Model Properties

A Gaussian process yields probabilistic predictions in the form of a Gaussian distribution. To show the result of a model, or a fitted surface, this is usually visualized by taking the mean of the GP. However, we are able to compute confidence intervals for our predicted values in the same way that we might compute confidence intervals for any Gaussian distribution.

In the situation that we know we have no noise in our data, GPR will interpolate the data points perfectly. However, if we have reason to believe that our observed values have noise,

we can incorporate this into the model. The resulting GP will not fit our observed data points perfectly, but it might lead to better predictions on new data.

Finally, the hyperparameters vary depending on the kernel function used. In the case of the squared exponential kernel

$$k(x, x'|\tau, h) = \tau^2 \exp\left(-\frac{\|x - x'\|^2}{2h^2}\right)$$

We can see that there are two hyperparameters, τ, and h. For the most popular covariance function in Gaussian processes, the Matérn covariance function, we choose only one hyperparameter, a positive integer p. In the limit as p approaches infinity, the Matérn function converges to the squared exponential function [2]. Historically, the Matérn 5/2 covariance function is used in the Gaussian process due to its ability to not overfit but also have a decent level of smoothness. The equation of the function is as follows

$$C_{5/2}(d) = \sigma^2 \left(1 + \frac{\sqrt{5}d}{\rho} + \frac{5d^2}{3\rho^2}\right) \exp\left(-\frac{\sqrt{5}d}{\rho}\right)$$

where $\nu = \frac{5}{2}$, σ, and ρ are positive parameters of the covariance function.

Simulation Study

Here, we've used a simulated data set by taking 49 points inside the region $[-2, 2] \times [-2, 2] \in \mathbb{R}^2$. We get our response variable by plugging in these 2D points into MATLAB's peaks function [3] shown below.

$$f(x, y) = 3(1-x)^2 \exp(-x^2-(y+1)^2) - 10(x/5 - x^3 - y^5) \exp(-x^2-y^2) - 1/3 \exp(-(x+1)^2 - y^2)$$

Finally, we add a bit of random noise to the response variable. The data points along with the true surface is shown below.

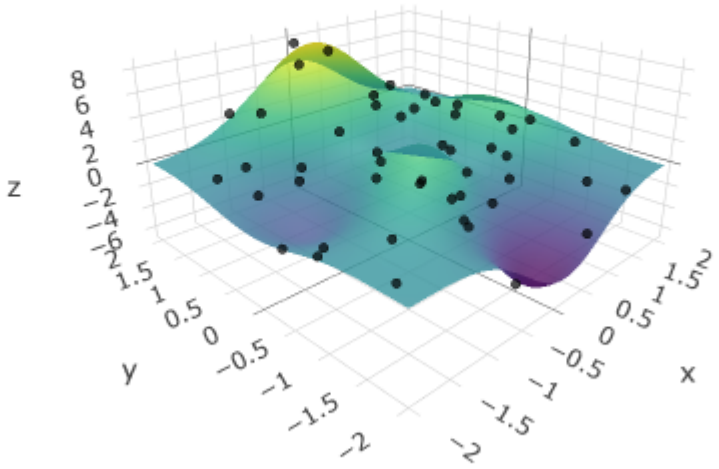


Figure 1: Data with true surface

Gaussian Process Regression

We perform Gaussian process regression on the data points with the Matérn 5/2 kernel and the zero mean function. Here is the resulting surface using the fitted GPR model with no nugget.

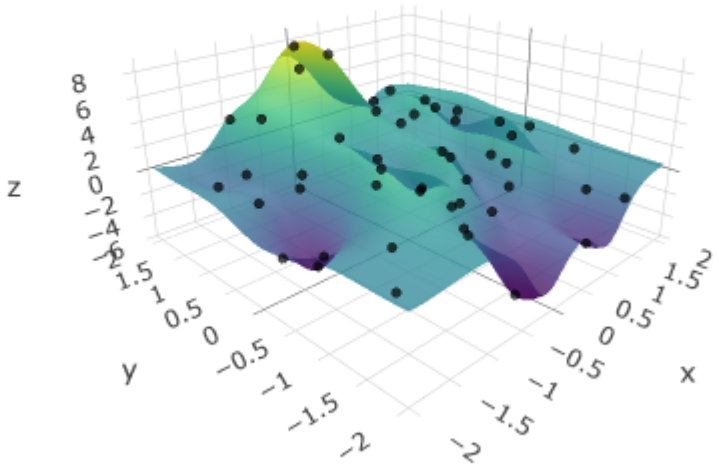


Figure 2: Data with surface from GPR

This seems similar to the true surface. The mean squared error of this model is approximately 0.8417621.

However, this model has no nugget, which means we aren't accounting for the random noise in the data. When using Matérn 5/2 kernel with a nugget of around 0.003 (as chosen by the `gpkm` function), we obtain a different result.

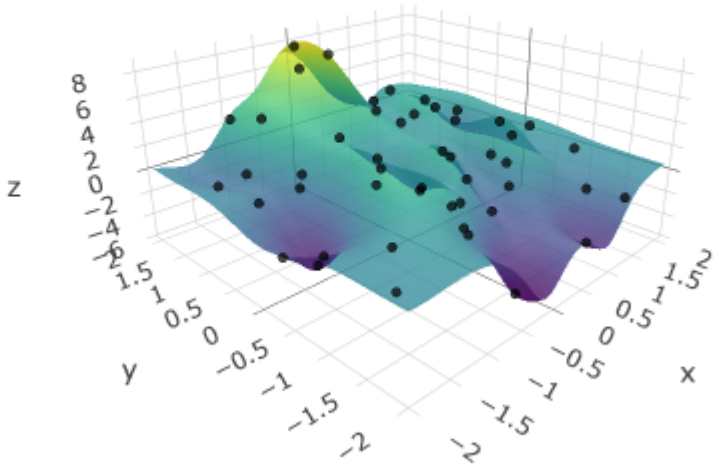


Figure 3: Data with surface from GPR (with nugget)

This doesn't seem very different from the previous model, but our mean squared error has indeed gone down to approximately 0.820613.

Finally, we can perform GPR with a different kernel. Experts do not recommend using the Gaussian kernel (radial basis kernel) due to its infinite differentiability, we can see its results below.

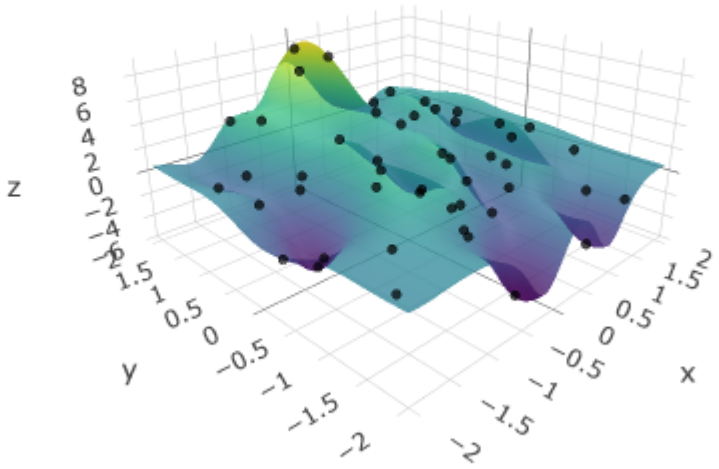


Figure 4: Data with surface from GPR (with Gaussian kernel)

The mean squared error is 0.8141003, which is actually lower than the previous two GPRs we performed. `gpkm` estimated the best σ to be approximately 0.004898843.

K Nearest Neighbors Regression

Now we can compare to KNN regression with $k = 3$ (chosen by cross validation). The model result is shown below.

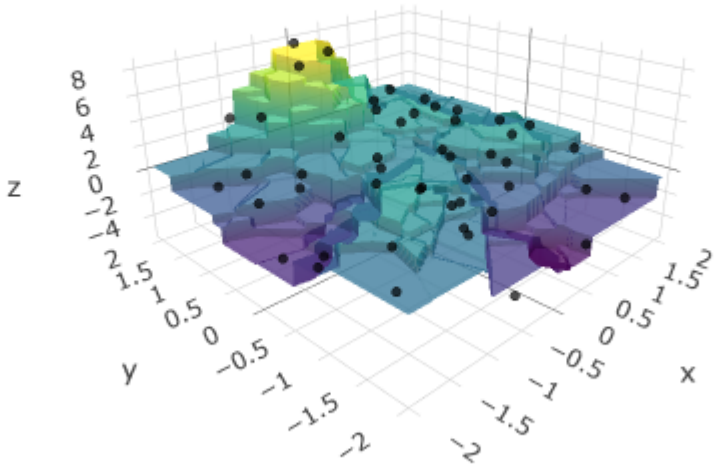


Figure 5: Data with surface from KNN

While it captures the general shape, it doesn't appear to fit as well as GPR. The mean squared error ends up being 1.238872, which is significantly larger than both of GPR's results. One thing to note was that another result of KNN regression was that if you choose a high value of K the surface will have a smaller range of values.

Polynomial Regression

Lastly, we will compare the results to polynomial regression with $d = 4$. We tested degrees 2 through 6 and chose degree 4 due to its lowest mean squared error. We included a cross interaction term as well

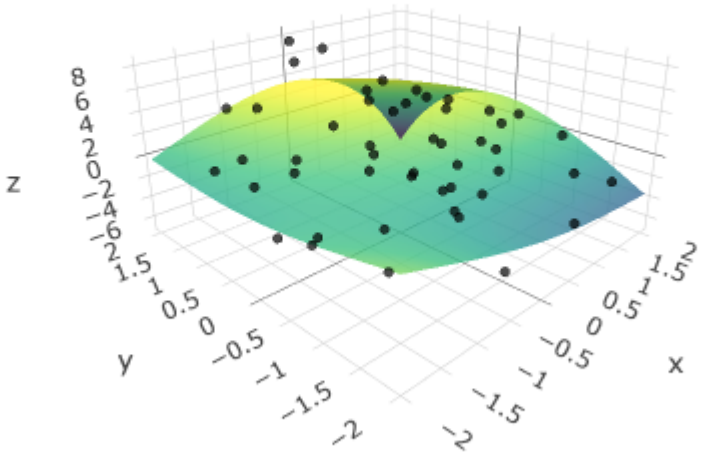


Figure 6: Data with surface from polynomial regression ($d = 4$)

Polynomial regression performed the poorest of all the methods with a mean squared error of 3.071027. This is more than double the mean squared error of our other methods. When looking at the plot above you see that it attempted to capture the overall trend but it failed to capture any of the peaks or valleys. This makes sense, because our points do not follow any type of polynomial trend, making it extremely difficult to use a polynomial regression approach for this dataset.

Discussion

Our project involved implementing Gaussian Process Regression on a simulated dataset and comparing the predictive accuracy between GPR models having different kernels as well as comparing the accuracy of GPR models in general to KNN and polynomial regression models. We found the GPR model with the Gaussian kernel and an estimated nugget to perform the best of the three GPR models we trained, and we found all three GPR models to perform significantly better than the KNN and polynomial regressions we employed, as we can see in the table below.

Model	Kernel	MSE
GPR ($\sigma = 0$)	Matérn 5/2 kernel	0.8417621
GPR ($\sigma \approx 0.003$)	Matérn 5/2 kernel	0.820613
GPR ($\sigma \approx 0.005$)	Gaussian kernel	0.8141003
KNN regression ($K = 3$)	Linear	1.238872
Polynomial regression ($d = 4$)	-	3.071027

We learned from the data by using our two predictor variables (x and y) and running them through the various regression models to predict the value of z, our response variable. To assess the accuracy, we then compared the predicted value with the actual value from our simulated set and calculated the MSE, which we used as the performance metric most pertinent to our analysis. Since we used a simulated dataset, the main contribution of our project lies in the methodology and results, which demonstrate the effectiveness of GPR over other machine learning models, particularly when sufficient flexibility is needed, such as in handwriting recognition or image detection.

Some limitations of GPR include the sensitivity of kernel choice, computational complexity, and lack of interpretability. We encountered this when considering which kernels to use, where we opted to implement the Matérn 5/2 kernel since it historically performs well, and we chose the Gaussian kernel due to its smoothness and differentiability. Our choices were relatively intuitive and did not involve extensive analysis of the many alternate kernel choices available, which may be necessary in other cases. Also, since our dataset was relatively simple, consisting of one response and two predictor variables, as well as 49 observations, we did not run into the issue of computational complexity; however, such small datasets can be unlikely in practice. The lack of interpretability of our results is also evident, in that there is no inherent meaning behind our findings since our data is simulated, and the method by which our predicted values are obtained is complex and lacks a definitive inferential structure; therefore, we concluded that Gaussian Process Regression is mostly useful for prediction and is limited in its inferential ability.

We believe we could improve our model for better prediction results by taking a closer look at alternate kernels and applying appropriate nuggets; however, we are content with the results for the scope of our project, particularly since our findings corroborated our hypothesis that the Gaussian kernel would perform the best among the GPR models, and the Gaussian Process, KNN, and polynomial regression models would perform successively worse.

References

[1] *Gaussian Process*. https://en.wikipedia.org/wiki/Gaussian_process.

[2] *Matérn Covariance Function*. https://en.wikipedia.org/wiki/Matern_covariance_function.

[3] *Peaks Function*. https://www.mathworks.com/help/matlab/ref/peaks.html.

[4] Kaixin Wang. *Introduction to Gaussian Process Regression, Part 1: The Basics*. https://medium.com/data-science-at-microsoft/introduction-to-gaussian-process-regression-part-1-the-basics-3cb79d9f155f.

[5] Cheng-Han Yu. *Gaussian Processes: MSSC 6250 Statistical Machine Learning*. https://mssc6250-s24.github.io/website/slides/12-gp.html#/title-slide.