# CompSci Project 3
# Gaussian Process Regression on Franke Function

Ali Aslan Demir[1], Silja Borring Låstad[1], and Franziska Schöb[1]
[1] *The Njord Centre, Department of Physics, University of Oslo, P.O. Box 1048, 0316 Oslo, Norway*

Gaussian Process Regression, in the following referred to as GPR, is a probabilistic tool belonging to the class of supervised learning approaches which can be used for modelling and predicting complex functions. In this report, we investigated the usage of GPR on the Franke function, a function commonly used in regression problems. We focus on understanding the impact of a chosen prior, also referred to as kernel, on the accuracy of the resulting GPR model.

## I. INTRODUCTION

The Franke function is often used to test regression models. It contains both peaks and valleys in the rage of [0, 1]. Due to its inherent non-linearity, it is an ideal function for exploration of regression approaches and their evaluation [1]. GPR is a methodology used for probabilistic regression and its central ideas are based on Bayesian statistics. The underlying function is modelled as a so-called Gaussian process, which enables GPR to capture complex functions including an estimation of uncertainty [2].
In GPR, we use Gaussian processes to model the unknown function. This Gaussian process is defined itself by two different functions, the mean function and the covariance function, in the following referred to as kernel function. The first represents the average behavior of the unknown function, while the latter is used to describe the similarity of function values at certain points in the input space [3]. In order to predict the underlying, unknown function with GPR, the model needs to be trained as in any other machine learning approach. The model learns underlying patterns of the training data by estimating the parameters of the given kernel function to resemble the training data. The trained model can them be used to predict function values at unseen points. What differentiates GPR from other machine learning models is however, that GPR predictions are probability functions, not output values. Hereby, GPR captures the uncertainty of its own prediction and makes it very powerful when we are interested in the reliability of the resulting predictions [4].
For assessment of accuracy of GPR predictions, we focus on log-marginal likelihood, which in here represents the probability of the training data being generated from the chosen model. It is therefore also called the model evidence. We also include the R-squared measure to evaluate the error of our predictions. R-squared is defined as seen in Eq. 1

$$R^2 = (1 - \frac{u}{v}) \tag{1}$$

### Kernels

There are many different kernels which can be used in GPR. Here we introduce the ones we use in our work.

The most commonly used kernel is the Radial bias function (RBF) kernel, see Eq. 2.

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right) \tag{2}$$

$$k(x_1, x_2) = constant\_value \ \forall \ x_1, x_2 \tag{3}$$

$$k(x_i, x_j) = \sigma_0^2 + x_i \cdot x_j \tag{4}$$

$$k(x_i, x_j) = \exp\left(-\frac{2\sin^2(\pi d(x_i, x_j)/p)}{l^2}\right) \tag{5}$$

## II. METHODS

In order to model the Franke function using GPR, we generate random data points in the domain [0, 1] and evaluate the function at these points. We additionally added Gaussian noise to make the problem more realistic, as data in the real-world is never noise free. We then chose between several kernels available in the scikit package [5], including the Radial bias function (RBF), Constant kernel and Exponential-Sine-Squared kernel. We also try a combination of the first two kernels to explore the impact of a chosen kernel on the accuracy of the prediction. The GPR model is then trained with our generated data and the chosen kernel and finally used to predict the underlying function. To assess the accuracy of the obtained prediction, we use the R-squared score as well as log-marginal likelihood of the training data.

For optimization of our model we used the standard L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) algorithm, which is an approximation of the original BFGS algorithm and

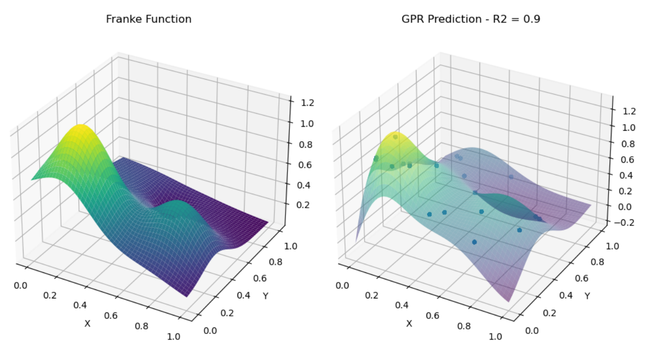

FIG. 1. Left: Franke function. Right: Surface prediction by GPR using RBF kernel only with noise level of 0.01. 20 data points were used for training. Test set of 200 used to determine R-squared score of 0.9. Blue dots represent data points used for training of the model.

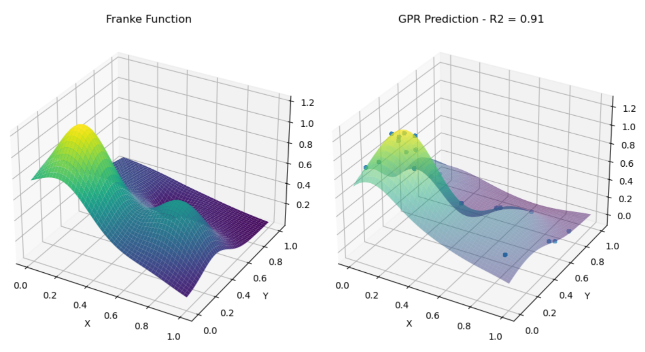

FIG. 2. Left: Franke function. Right: Prediction of function using GPR with a combination of RBF and constant kernel. Noise level is 0.01, 20 training points were used. R-squared score for prediction is 0.91. Blue dots represent data points used for training of the model.

.

commonly used in machine learning problems. This algorithm minimizes the target function $\mathbf{f}(\mathbf{x})$ by iteratively estimating the inverse Hessian matrix using gradients and previous updates, if available. This approach reduced computational complexity of the algorithm to $\mathcal{O}(n^2)$, compared to $\mathcal{O}(n^3)$ for actually calculating the Hessian matrix. Instead of storing the complete approximation of the Hessian matrix, this particular version of the algorithm just stores some vectors to represent the matrix indirectly, saving large amounts of memory.

## III. RESULTS

In the following, we visualize the prediction results of our GPR model using different kernels, different number of training data points and two different noise levels in training data. All figures are in comparison with the actual Franke function to provide an intuition of the accuracy of the predictions. Notice, that each visualisation only represents *one* sample of the posterior distribution of predicted functions. Different samples from the same predicted posterior distribution can vary strongly depending on the training data.

For the following GPR models, we have - unless otherwise stated - used 20 training data points, randomly sampled from the Franke function with added independently distributed noise of $\mathcal{N}(0, 0.1)$. We have used nine optimization restarts during training of the model unless mentioned otherwise. For testing, we used 200 unseen data points to evaluate the obtained predicted functions and calculate the R-squared score.

In figure 1, our posterior was defined by the RBF kernel only. With the above mentioned noise and training

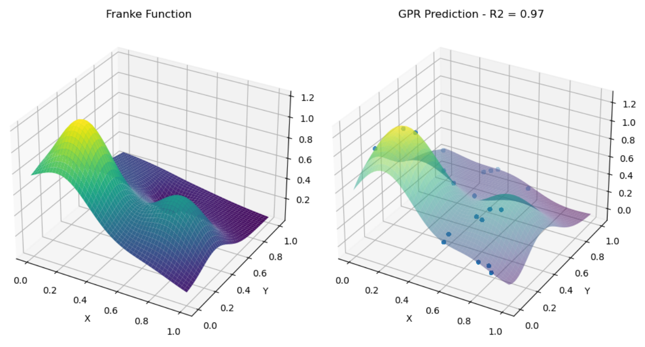

FIG. 3. Left: Franke function. Right: GPR predicted function surface using combination of RBF and constant kernel. Here, the optimizer was restarted 50 instead of the usual 9 times. R-squared reached a value of 0.97. Blue dots represent data points used for training of the model.

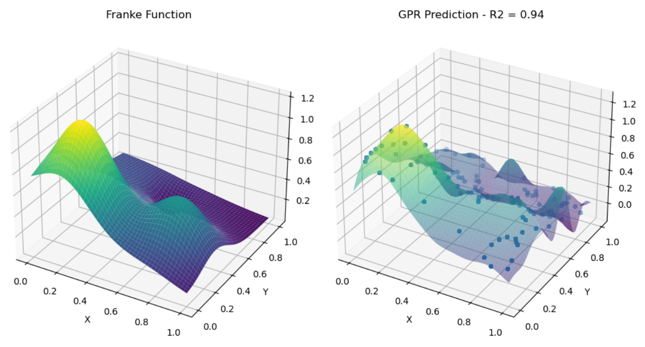

FIG. 4. Left: Franke function. Right: The GPR-based prediction with a RBF kernel and constant kernel. The number of training points was increased to 100 from the usual 20, noise level remained at 0.01. R-squared was evaluated at 0.94. Blue dots represent data points used for training of the model.

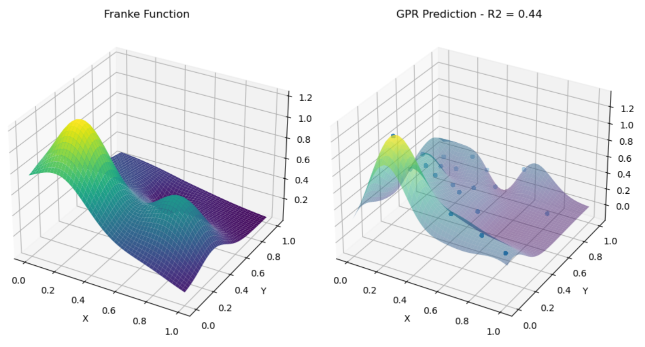

FIG. 5. Left: Franke function. Right: GPR predicted surface using both RBF and constant kernel in combination. Noise level in training data was increased to 0.1, resulting in a R-squared value of 0.44. Blue dots represent data points used for training of the model.

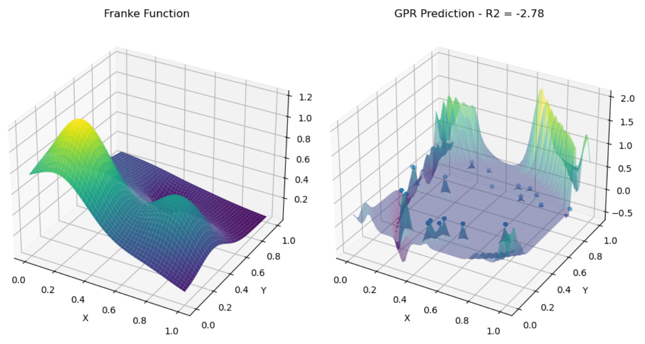

FIG. 6. Left: Franke function. Right: Exp-Sine-Squared kernel leads to this prediction by GPR with a noise level of 0.01 and 20 training data points. R-squared equals -2.78. Blue dots represent data points used for training of the model.

data points, we obtained an R-squared value of 0.9. the perdicted function captures the main peak as well as the valleys, despite the noisy data.

In figures 2, 3, and 4 we have explored a combination of the RBF and constant kernel for the prediction of the posterior distribution. This combination results in r-squared scores of at least 0.91 for our standard parameters and up to 0.97 when using 50 optimizer restarts instead of the usual nine in figure 3. All models are able to capture the main peaks and valleys of the Franke function as seen in the visualisations.

We further explored this kernel combination with higher numbers of training points and can see in figures 4 that neither 100 instead of 20 training points do not necessarily increase the accuracy of the model. We therefore continued using 20 training points as the standard.

Moreover, we have increased the noise level in our training data using the same model in figure 5. It is clearly visible in both visualisation and r-squared score of 0.44, that an increased noise level leads to poor performance of our model. the model starts to overfit on the noisy data, leading to wrong predictions of the underlying function.

Additionally, we have utilized the Exponential-Sine-Squared kernel in our model to test its performance, see figure 6. As expected, as the kernel favors periodic functions, the resulting prediction only achieves a r-squared score of -2.78, showing the very poor performance of this particular model on our problem.

Approaches using the Dot kernel in combination with the RBF kernel, which are not visualised here, have led to promising predictions with r-squared values of above 0.9 when using low noise levels and only 20 training points.

## IV. CONCLUSION

Based on our results from using GPR on the Franke function, we conclude that GPR is a useful method to model complex functions such as the Franke function. Many of our models have resulted in good predictions with r-squared measures of above 0.9. While all noise level, number of training points and optimizer restarts influenced our prediction results, the choice of kernel had the most significant impact on the accuracy of the GPR prediction. We achieved the best results with a combination of the RBF and Constant kernel, favoring smoothness in the predicted functions. This highlights the importance of choosing a suitable prior when doing GPR, as results can differ drastically based on the kernel assumptions.

Surprisingly, an increased number of training data points did not directly correlate with an increase in accuracy in our models. Our model instead seemed to overfit on the given, noisy, training data, resulting in less smooth predictions. We have not explored higher number of training points than 500, as we did not see an improved performance. Further, GPR is getting more and more computationally expensive the more data points are given. This results in GPR needing to use approximations when given more than 10.000 training data points. We are far from this number of points, however it is important to keep in mind in larger, real-world applications of GPR. Recent approaches, which try to distribute and partition kernel matrix multiplications by combining gradients linearly, have demonstrated that it is possible to do exact GPR on a million data points [6].

[1] R. Franke, Technology Report NPS-53-79-003 (1979), 10.21236/ADA081688.

[2] E. Schulz, M. Speekenbrink, and A. Krause, Journal of Mathematical Psychology **85**, 1 (2018).

[3] C. K. I. Williams and C. E. Rasmussen, Mathematical Social Sciences **28**, 224 (1994).

[4] C. E. Rasmussen & C. K. I. Williams, *MIT Press* (2006).

[5] V. Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, P. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, , A. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, and E. Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, Journal of Machine Learning Research **12**, 2825 (2011).

[6] K. A. Wang, G. Pleiss, J. R. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson, Advances in Neural Information Processing Systems **32**, 1 (2019), arXiv:1903.08114.