Gaussian Processes for Machine Learning

N. Kubiak, *Member, IEEE.*

***Abstract*— Overview of the Gaussian Processes for Machine Learning, in which we use supervised learning to train a learning machine. Using training data set and probabilistic uncertainty to train our machine to make accurate predictions of new data sets. Using Bayes' Theorem in order to mitigate uncertainty and be able to generate the *prior* and *posterior* from our known and unknown events. The big advantage is being able to include prior knowledge to our Regression model using Bayes' Theorem and generate a posterior. When using Bayes' theorem in our supervised machine learning, we can make accurate predictions based on what we learned from the training data. We will go over optimizing our parameters and finding the best fit for our Gaussian Regression, as well as the consequence of a bad fitting parameter. This paper will discuss linear and non-linear models.**

***Index Terms*—Bayes Theorem, Recursive Kernel Hilbert Spaces, Gaussian Process, Probability, Bayesian Processes, Regression**

I. Introduction

In this journal we will review supervised learning, and more specifically using Gaussian Processes. Our concern is with using our empirical data or training set, to be able to learn input/output mappings. Our goal is to be able to use these mappings from our empirical data set and be able to apply it and make an accurate prediction to a new data set and produce similar results from our supervised learning machine [1][2]. Ideally the data we would use in our learning machine would be data not seen yet in our empirical data set but would still be classifiable by the training done. The utilization of the Gaussian Process comes to fruition when we encounter a problem of uncertainty. In this case the Gaussian Processes for Machine Learning method triumphs over other methods as there is a level of flexibility when it comes to probabilistic uncertainty. There is always some uncertainty or lack of knowledge, which we cannot remove. By utilizing this uncertainty in our probability distributions, we can understand further Gaussian Processes. The approach of utilizing this uncertainty would be to give a prior probability to every possible function, where we distribute higher probability to functions, we consider more likely [1]. By using the Gaussian probability function, we can use stochastic processes to govern our functions, which we can think of as being a vector, which specify the value of f(x) at input x [1]. Then Using the Recursive Kernel Proposition, we will have the kernel matrix K as the covariance matrix of the Gaussian process estimation of f(x[n]) [5]. Since we now know our Gaussian process, the probability distribution over our functions, we can then utilize Bayes Theorem and reinforce our learning. We select a prior, which will represent our *prior* belief of our expected functions [1][2] before we see the output. Using this *prior* we can combine this with our training data to get the *posterior*. Using Bayes Rule, we can update our belief using the *prior* before we get knowledge of an event or information, which can then be used to create the *posterior* which will then be able to use that acquired information into account [1]. This Journal will cover Several topics of Gaussian Processes for Machine Learning, such as the Linear Gaussian Process, Nonlinear Gaussian Process, and Gaussian Processes in Recursive Kernel Hilbert Spaces.

II. Theory

A. Bayes Rule

In order to further understand Gaussian Processes for Machine Learning and their utilization of Bayes Rule to reinforce their learning, we will go over the concepts of Bayes' Theorem. First, we can assume a Universal set which contains all the possible events Ω, and two subsets $A, B \epsilon\ \Omega$.[5][9]

To view the probability of Event A, given that event B has occurred we would say the probability is

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{1}$$

using Bayes Rule. We can interpret this by observing that the even B has occurred and is known and observable, and effects the probability of event A, which is known to exist, but is not observable. Similarly, we can use our estimator

$$y = w^{\mathrm{T}} x + \varepsilon \tag{2}$$

and the set of parameters of the estimator w, such that we can call $p(w)$ the *prior* probability and $p(\boldsymbol{w}|y, \boldsymbol{x})$ the *posterior* probability. Using the prior and posterior we can compute the likelihood as [5][9]

$$p(y, \boldsymbol{x}|\boldsymbol{w}) = \frac{p(y, \boldsymbol{x}|\boldsymbol{W})}{p(x)} \tag{3}$$

B. Linear Regression with Gaussian Processes

We will be introducing Linear Gaussian Process for regression. We will want to keep in mind Bayes' Rule as we observe the probabilistic model for y_n, where y_n is the joint likelihood as a product of likely hoods. Then we will treat w, as a latent random variable that has a Gaussian prior, and the posterior is proportional to the prior times the likelihood.[2]

The likelihood is a joint Gaussian of the form

$$p(y|X,w) = \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\frac{(-|y-X^T x|^2)}{2\pi\sigma_n^2} \quad (4)$$

We can assume that parameter w is a linear combination of the data, and satisfies the conditions of the Central Limit theorem, such that it is a Gaussian random variable, and the mean is 0. We can next assume the prior distribution of our parameters is

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p) = \frac{1}{(2\pi|\Sigma_p|)^{(D+1)/2}} \exp\left(-\frac{1}{2}\mathbf{w}^\top \Sigma_p^{-1}\mathbf{w}\right) \quad (5)$$

$\sum_p$ is the covariance which is arbitrarily set as an identity matrix. Then we can say that the posterior with respects to X and y is

$$p(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})} \quad (6)$$

Here the numerator is the prior and likelihood, and the denominator is the marginal likelihood.[6] We can observe some key aspects of the linear Gaussian process, such that this method can also later be kernelized. By finding and using the posterior with the Total Probability Rule, [5]we can find the posterior of our prediction. Here we will show an example of a Linear Gaussian Process Regression. We use the model

$$y_n = (0.5x_n + 0.5 + w_n) \quad (7)$$

where $x_n$is a uniform random variable between 0 – 1, and w_n is Gaussian noise with a variance of σ^2. In our example, a prediction of out output x_n to x_{n-p} and past values of our output.

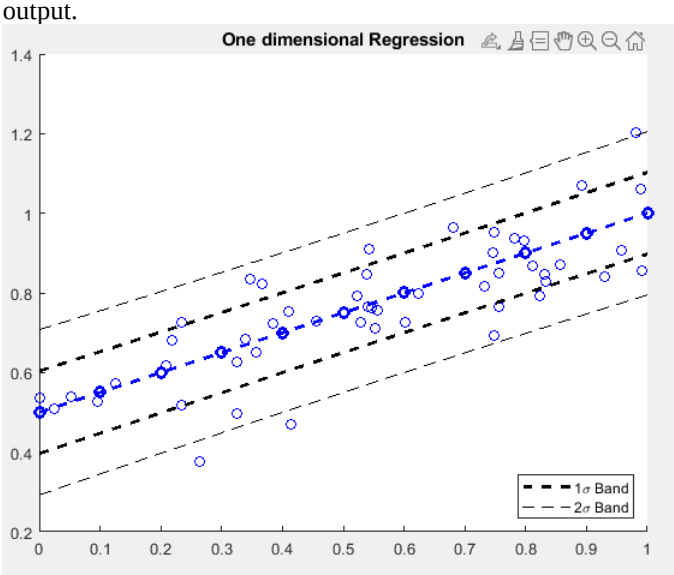

Fig 1. Linear Regression Model

TABLE I
Linear Regression Model

	Samples Band	**Theoretical Value**
σ Band	16	68%
2 σ Band	3	94%

Now we will want to see the outcomes of how the Gaussian noise σ^2 will affect the predictions made. We can increase or decrease the amount of noise which will assist us for choosing the optimal noise parameter later. It is good to observe how hyperparameters will affect our learning.[5]

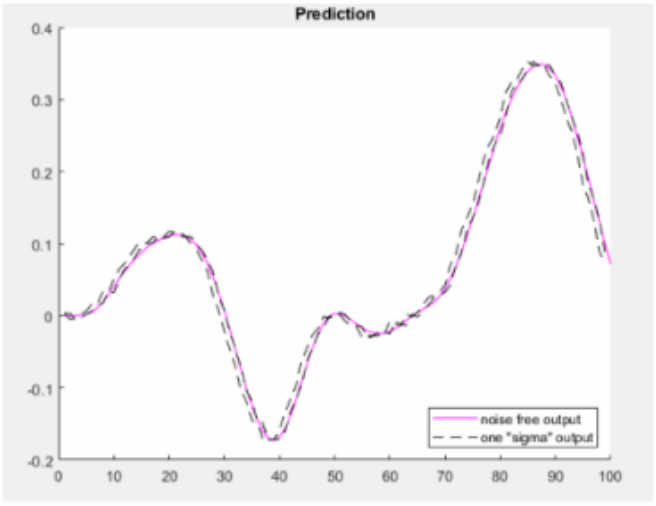

Fig 2. Sigma is 0.005

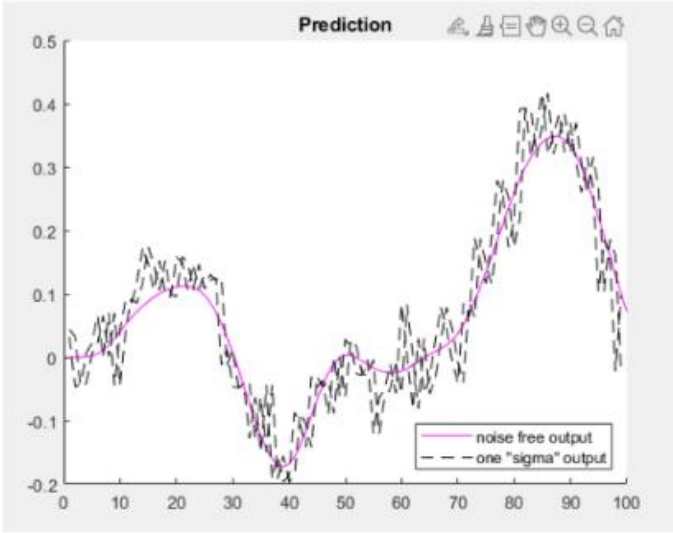

Fig 3. Sigma is 0.05

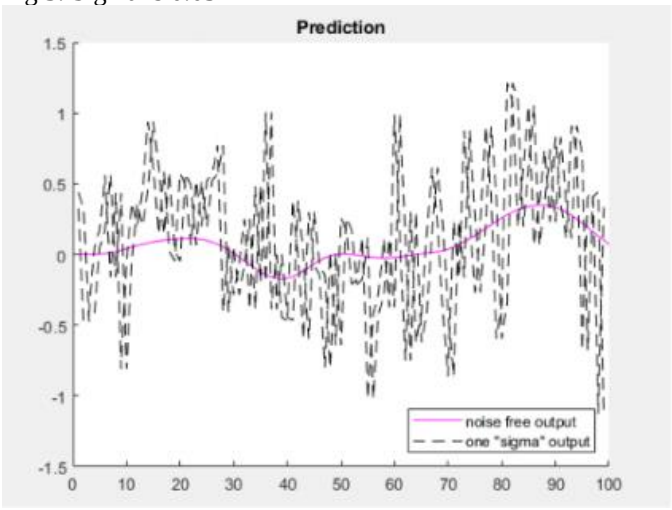

Fig 4. Sigma is 0.5

From the examples we can observe how the Gaussian noise can cause overfitting of our graph in which the noise fits too well with the noise free output with little to no error. On the contrary when sigma is large we can observe a lot of noise in which there is underfitting and the sigma output has too much error in comparison with the noise free output. The goal is then the find a hyperparameter of σ^2 which does not over fit or underfit.[5]

C. Regression in Hilbert Spaces

After going over the Linear Gaussian Processes, we'll now consider a nonlinear transformation into a Hilbert Space with the form $\varphi(x)$. We can take the expression of the predictive distribution in linear regression where

$$p(\bar{f}_*|\mathbf{x}^*, \mathbf{y}, \mathbf{X}) = \mathcal{N}\left(\bar{\mathbf{w}}^\top \mathbf{x}^*, \mathbf{x}^{*\top}\mathbf{A}^{-1}\mathbf{x}^*\right) \tag{7}$$

And

$$\mathbf{A} = \sigma_n^{-2}\mathbf{X}\mathbf{X}^\top + \boldsymbol{\Sigma}_p^{-1}$$

$$\mathbf{A} = \sigma_n^{-2}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \boldsymbol{\Sigma}_p^{-1}$$

$$\bar{\mathbf{w}} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma_n^2\boldsymbol{\Sigma}_p^{-1}\right)^{-1}\boldsymbol{\Phi}\mathbf{y}$$

Then the expressions for the inverse of A and linear parameter w, is
and $\Phi = \{\varphi(x[1] \ldots . \varphi(x[N]\}$
and $\Phi\Phi^T$ is possibly infinite.

The estimator of the function can be written as[6]

$$\bar{f}_* = \boldsymbol{\varphi}(x^*)^\top \boldsymbol{\Sigma}_p^{1/2}\boldsymbol{\Sigma}_p^{1/2}\boldsymbol{\Phi}\boldsymbol{\alpha} = \boldsymbol{\varphi}(x^*)^\top \boldsymbol{\Sigma}_p\boldsymbol{\Phi}\boldsymbol{\alpha} \tag{8}$$

Where the dot product between the test and training data is $\varphi(x^*)^T \sum_p \Phi$. Then for $\bar{w} = \sum_p \Phi\alpha$ we are able to apply the solution

$$\bar{\mathbf{w}}' = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma_n^2\boldsymbol{\Sigma}_p^{-1}\right)^{-1}\boldsymbol{\Phi}\mathbf{y} \tag{9}$$

with the following qualities.
Now we will want to find alpha, so we can multiply (9) by Φ^T which results in [6]

$$\boldsymbol{\Sigma}_p\boldsymbol{\Phi}\boldsymbol{\alpha} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma_n^2\boldsymbol{\Sigma}_p^{-1}\right)^{-1}\boldsymbol{\Phi}\mathbf{y}$$

$$\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma_n^2\boldsymbol{\Sigma}_p^{-1}\right)\boldsymbol{\Sigma}_p\boldsymbol{\Phi}\boldsymbol{\alpha} = \boldsymbol{\Phi}\mathbf{y}$$

$$\sigma_{\bar{f}_*}^2 = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^\top\left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{k}(\mathbf{x}^*)$$

Then, we will have found our value for alpha

$$\boldsymbol{\alpha} = \left(\boldsymbol{\Phi}^\top\boldsymbol{\Sigma}_p\boldsymbol{\Phi} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} = \left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}$$

Now for the expression $\alpha = (K + \sigma_n^2 I)^{-1} y$ and $K = \Phi^T \sum_p \Phi$ We can assume K is a matrix for which the entries are

$$k(\mathbf{x}, \mathbf{z}) = \boldsymbol{\varphi}(\mathbf{x})^\top\boldsymbol{\Sigma}_p\boldsymbol{\varphi}(\mathbf{z}) = \boldsymbol{\varphi}(\mathbf{x})^\top\boldsymbol{\Sigma}_p^{1/2}\boldsymbol{\Sigma}_p^{1/2}\boldsymbol{\varphi}(\mathbf{z}) \tag{10}$$

This matrix happens to be a kernel dot product of vectors in a Hilbert space which have been transformed linearly. The choice of this matrix is implicit in the choice of the kernel. The matrix K defines the prior on the parameters as well. Which proves that the kernel dot product between the data is then.[6]

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}^\top(\mathbf{x}_i)\boldsymbol{\Sigma}_p\boldsymbol{\varphi}(\mathbf{x}_i) \tag{11}$$

D. Regression with Gaussian Process Networks

Now we will want to be able to identify the mean and variance of the predictive posterior in the Kernel Gaussian Process. From the previous section we know that alpha is the solution of the dual parameters, and the matrix K which defines our

$$\bar{f}_* = \boldsymbol{\varphi}(x^*)^\top\bar{\mathbf{w}}' \tag{12}$$

kernel that is a dot product of vectors linearly transformed by the matrix $\sum_p^{1/2}$. Our choosing of the Matrix is implicit. Then when we use the definition of a kernel, we can rewrite our regression

Then, $k^T(x^*) = \{k(x^*, x[1]) \ldots k(x^*, x[N])\}$ which is equal to

$$\bar{f}_* = \mathbf{k}^\top(\mathbf{x}^*)\left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} \tag{13}$$

$$\bar{f}_* = \mathbf{k}^\top(\mathbf{x}^*)\left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}$$

the column vector of dot products between the test data and training data of the Hilbert Space.[6] Such that
Is the expectation of the Gaussian process at sample x^* which will be conditional to X, and y. Then we can compute the variance by using the definition of the kernel with $A = \sigma_n^{-2}\Phi\Phi^T + \sum_p^{-1}$, with equation (13), then compute the inverse of a using the matrix inversion lemma, from which

$$\mathbf{A}^{-1} = \boldsymbol{\Sigma}_p - \boldsymbol{\Sigma}_p\boldsymbol{\Phi}\left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\Sigma}_p \tag{14}$$

And from expression [5]

$$\sigma_{\bar{f}_*}^2 = \boldsymbol{\varphi}(\mathbf{x})^{*\top}\mathbf{A}^{-1}\boldsymbol{\varphi}(\mathbf{x}^*) \tag{15}$$

We can obtain

$$\boldsymbol{\varphi}(\mathbf{x})^{*\top}\boldsymbol{\Sigma}_p\boldsymbol{\varphi}(\mathbf{x}^*) - \boldsymbol{\varphi}(\mathbf{x})^{*\top}\boldsymbol{\Sigma}_p\boldsymbol{\Phi}\left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\Sigma}_p\boldsymbol{\varphi}(\mathbf{x}^*)$$

And use the definition of the kernel such that the variance is
The Kernel k(x,x) evaluated at x^* is

$$\mathbf{k}(\mathbf{x}^* = \{k(\mathbf{x}^*, \mathbf{x}[1] \cdots k(\mathbf{x}^*, \mathbf{x}[N])\}^\top. \tag{16}$$

And

So, we can propose that the kernel matrix **K** is the covariance matrix of the Gaussian process estimation. We assume the process has 0 mean, so that the covariance matrix is [5]

$$\begin{aligned}\mathbb{E}[f(\mathbf{x}[n])f(\mathbf{x}[m])] &= \mathbb{E}[\boldsymbol{\varphi}(\mathbf{x}[n])^{\top}\mathbf{w}'\mathbf{w}'^{\top}\boldsymbol{\varphi}(\mathbf{x}[n])] \\ &= \boldsymbol{\varphi}^{\top}(\mathbf{x}[n])\mathbb{E}[\mathbf{w}'\mathbf{w}'^{\top}]\boldsymbol{\varphi}(\mathbf{x}[m]) \\ &= \boldsymbol{\varphi}^{\top}(\mathbf{x}[n])\boldsymbol{\Sigma}_p\boldsymbol{\varphi}(\mathbf{x}[m]) = k(\mathbf{x}[n],\mathbf{x}[m])\end{aligned}$$

Then the covariance matrix is

$$\mathbf{C}_{yy} = \mathbf{K} + \sigma_n^2\mathbf{I}$$

Which we have now proven that the kernel matrix is equal to the covariance matrix of the training regressors.[6]

E. Gaussian Regression Models

We saw in previous section how the Gaussian process used kernels or covariance function. We also observed how we have the free parameter, and that a good selection of our parameter can optimize the performance of the estimator, while a bad selection can give us poor performance. We can observe different types of regression models with different noise covariances.

III. Gaussian Modeling

A. Noise as a Parameter

To better understand optimizing the parameters and performance of our learning machine we will observe what will happen to our prediction as we change aspects of the covariance. First, we will want to generate some Gaussian data.

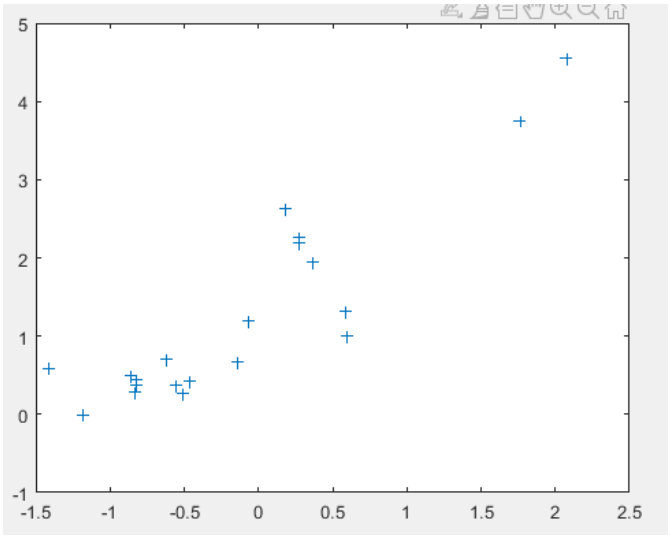

Fig 5. Gaussian Data Generation

Next, we will instruct our learning machine to compute the marginal likelihood and to generalize from the training data [1].

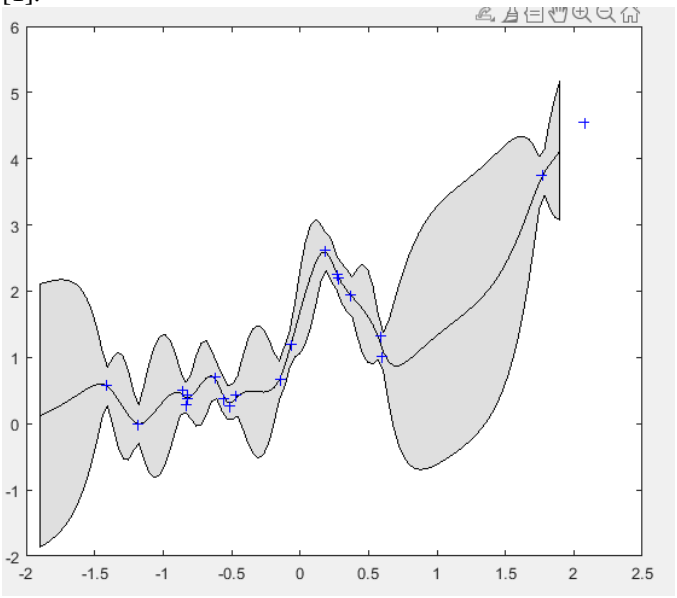

Fig 6. Gaussian Prediction

Typically, we would not allow the *priori* to have knowledge of the hyperparameters or the functions for mean, covariance, or likelihood, so we can adjust for this in our experiment. We will use the square exponential covariance function, which takes two hyperparameters. The hyperparameters are both non-negative and we assume the mean function is zero and can be disregarded. Then we will optimize the hyperparameters by minimizing the log marginal likelihood with respect to the hyperparameters [1].

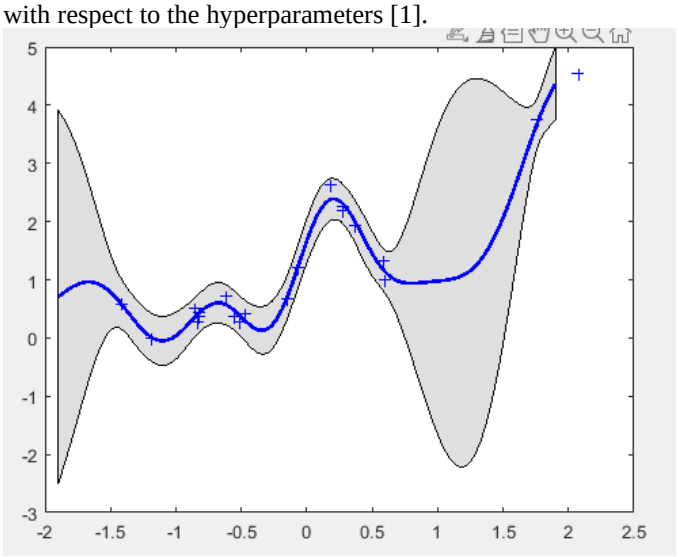

Fig 7. Optimal Gaussian Prediction of Square Exponential

We can observe what happens to our prediction as we change the optimal values of the hyperparameter. For example, we have standard deviation noise which we can increase by a factor of 5. This will over-predict. The noise is 5* the optimal, the estimation is too smooth and the so the estimation error can be high, but the confidence interval is reliable.

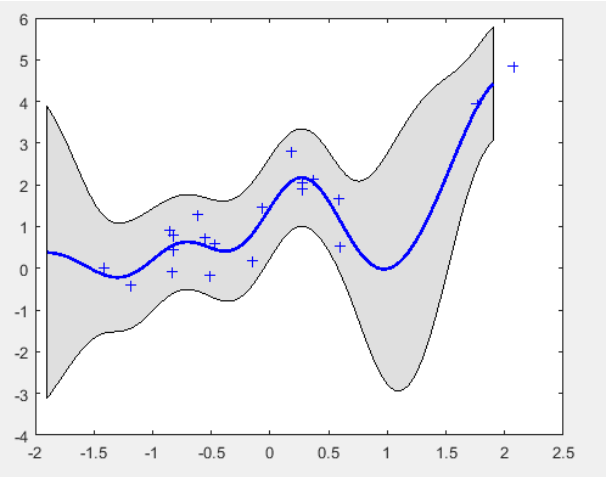

Fig 8. High Noise Covariance Regression

And if we decrease our noise by a factor of 10, we will have the effect of having an estimation that could be too optimistic and overfitting. It seems to be better but it may not be an accurate or realistic prediction.

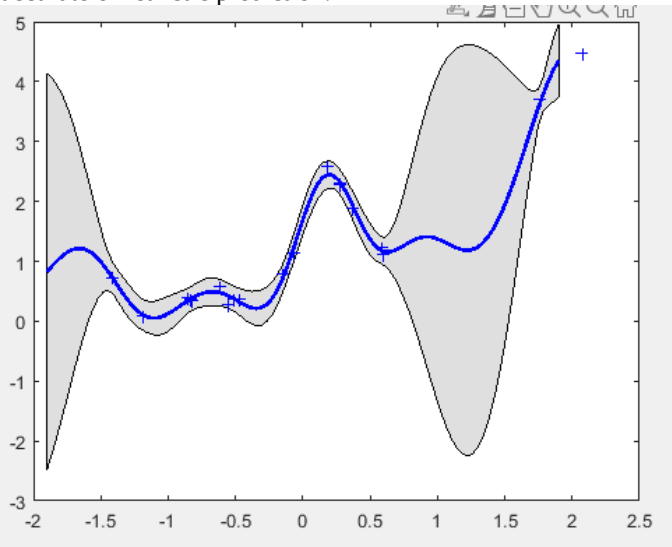

Fig 9. Low Noise Covariance Regression

As we observed for these Regression Models, we were able to define the hyperparameters of a gaussian process. We were then able to differentiate the noise parameter and change the value to see the outcome of our noise being increased or decreased. It is important to always seek optimal parameters such that they give you an accurate prediction which does not over or under predict and classify data.[1][6]

B. The Constant Kernel as a Parameter.

After viewing the effects of the parameters on our Gaussian Regression Model, we will want to remember that we will need to add a constant to out model such that it is constructed to be

$$y = v^T \varphi(x) + b$$

v is a set of linear parameters. We can rewrite this as

$$y = \mathbf{w}^\top \begin{pmatrix} \varphi(\mathbf{x}) \\ \sqrt{k} \end{pmatrix}$$

where $w = \begin{pmatrix} v \\ b\sqrt{k} \end{pmatrix}$ is our set on linear parameters. Then our non-linear transformation will be $\begin{pmatrix} \varphi(x) \\ \sqrt{k} \end{pmatrix}$, then we will take the dot product of these two equations which end up being equal to

$$\varphi^T(x)\varphi(z) + k = k_f(x,z) + k \tag{17}$$

So, our kernel will end up having three elements, where k is a parameter. However, in most situations k will become irrelevant when changed. [6]We can compute the eigen analysis of

$$\mathbf{K_f} + \sigma_n^2 \mathbf{I} + k\mathbf{1}_{N,N} \tag{18}$$

Where alpha is

$$\begin{aligned} \boldsymbol{\alpha} &= (\mathbf{K_f} + \sigma_n^2 \mathbf{I} + k\mathbf{1}_{N,N})^{-1}\mathbf{y} \\ &= \left(\mathbf{Q}\left(\boldsymbol{\Lambda} + \sigma_n^2 \mathbf{I}\right)\mathbf{Q}^\top + kN\mathbf{u}\mathbf{u}^\top\right)^{-1}\mathbf{y} \end{aligned}$$

Then we can use lemma as twice the matrix inversion which we will find to be

$$f(\mathbf{x}^*) = \boldsymbol{\alpha}^\top \mathbf{k}(\mathbf{x}^*) - \frac{1}{N}\mathbf{1}^\top \mathbf{y} \tag{19}$$

This will remove the mean of regressor, and again k will be negligible. If the kernel is constructed using the product of kernels, then this will not be true for k, and k will become an adjustable parameter

C. Adjusting Parameters

We have now seen different ways that parameters can influence our Gaussian Regression. The most common of the parameter estimations consists of validation.[6] We first want to make sure that our data set isn't too small, or that we have a high number of hyper parameters because this could become unpractical. In cases like those, we can investigate other criteria which involve a probabilistic method. The expression of marginal likelihood of y as a function of the estimator f is

$$p(y|X,\theta) = \int_f p(y|X,f,\theta)p(f|X,\theta)df \tag{20}$$

Where f is constructed with parameter θ, and is also a Gaussian process with a covariance matric that has been constructed with kernel functions[6]. The parameter of this kernel functions is

$$p(f|X) = \mathcal{N}(0,K) \tag{21}$$

And the likelihood of y is a Gaussian with mean f and covariance of $\sigma_n^2 I$, such that we can rewrite it as

$$p(y|f) = \mathcal{N}(f, \sigma_n^2 I)$$

Then we can put these distributions into our integral (20) and find the result. This result happens to be another Gaussian

distribution, in which we are interested in the exponent, so we will have to compute the log of the function we get

$$logp(y|X,\theta) = -\frac{1}{2}y^T K_y^{-1} y - \frac{1}{2}\log|K_y| - \frac{n}{2}log2\pi \quad (22)$$

where

$$K_y = K_f + \sigma_n^2 I$$

And

$$K_y = K_f(\theta)$$

Now we can look at the terms of the log-likelihood. We can note that there is a maximum of the likelihood where the optimal parameters are. The criterion for the hyperparameter optimization comes from this maximum likelihood.

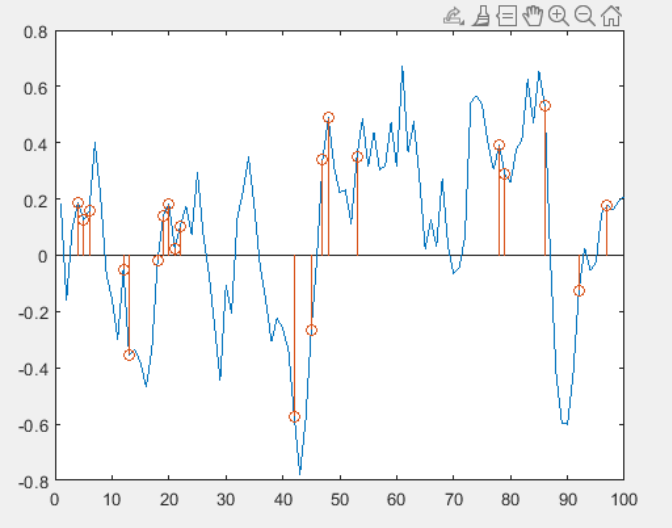

Fig 10. Optimization of Parameter y

The optimization can be performed using a gradient ascent procedure[6]

$$\theta^{(k+1)} = \theta^{(k)} + \mu\Delta_\theta logp(y|X,\theta) \quad (23)$$

Although we are trying to optimize and find ideal hyperparameters this will not guarantee to have a single minimum, so we will have to expand this to have different initializations. We can use the *Leave On Out* procedure. We compute the log likelihood of a sample in our training set. So for each of our samples y_i, we will construct a data set where we will take out sample y_i and x_i as our validation set. Next, we will want to compute the likelihood of this, as well at the derivative. We will average this for all the samples with the log likelihood which is [6]

$$logp(\, y_i|X_{-i}, y_{-i}, \theta) = -\frac{1}{2}log{\sigma^2}_i - \frac{y_i - \mu_i^2}{2{\sigma^2}_i} - \frac{1}{2}log2\pi \quad (24)$$

This is obtained using the predictive likelihood of our test data. Next, we must be able to compute the mean μ_i and the variance σ_i^2 from the predictive posterior equation. This can be reduced as[6]

$$\mu_i = y_i - [K^{-1}y]_i/[K^{-1}]_{ii} \quad (25)$$

And

$$\sigma_i^2 = 1/[K^{-1}]_{ii}$$

With these equations we can then find the expression of the sum of derivatives of the predictive log likelihood with respects to the LOO samples.

$$\frac{\partial L_{LOO}}{\partial \theta_j} = \sum_{i=1}^{N}\left(\alpha_i[\mathbf{Z}_j\boldsymbol{\alpha}]_i - \frac{1}{2}\left(1 + |\frac{\alpha_i^2}{[\mathbf{K}^{-1}]_{ii}}\right)[\mathbf{Z}_j\mathbf{K}_{ii}^{-1}]\right)/[\mathbf{K}^{-1}]_{ii} \quad (26)$$

IV. Conclusion

In this paper we have covered a range of topics, ranging from how to construct and parameterize Linear and Non-Linear Gaussian Regressions. On top of that, we have found how to optimize the very same parameters that we use for those Regression's. Overall, the Gaussian Regression are very well tailored to classification of functions with probabilistic uncertainties. Through our paper we have seen how useful and diverse the applications for Gaussian Processes for Machine Learning are, although popular, the method isn't as used due to computational limitations. Often the computations for Gaussian models can be hit with constraints and limitations when it comes to large data sets. [1][2] The Gaussian Processes for Machine Learning however are still incredibly useful and informative when it comes to data that is scalable with computation. A big advantage of the Gaussian Process is being able to include prior knowledge to your model and generate a posterior. Using Bayes' rule, we can expand on what we are able to classify using probabilistic methods. By using Bayes' theorem in our supervised machine learning, we are able to make accurate predictions based on what we learned from the training data. The main idea being that we have some prior knowledge about an event that we want to measure. This means our machine can make a prediction or a hypothesis of what it thinks will happen in the event given. The Gaussian process can be applied to linear and nonlinear regressions. It was interesting to see how increasing/decreasing the parameters can change the accuracy of our machine, and that it why it's important to find the ideal parameter. I think there is a lot to be learned through this method, especially through the theory. I hope there is more utilization of this tool as our computational capacity continues to increase and get faster.

V. Acknowledgements

I would like to give a big thank you to Dr. Manel Martinez-Ramon for teaching me Machine Learning and having patience with me as one of his students. I admire his worth ethic and proficiency in his field and hope to continue learning and understanding the topic. Although the concepts seem difficult, Dr. Martinez- Ramon always provided explanation and gave us his time.

VI. Sources

[1] C. E. Rasmussen and W. C. K. I., Gaussian processes for machine learning. Cambridge, Mass: MIT Press, 2008.

[2] M. Krasser, "Gaussian processes," Gaussian processes - Martin Krasser's Blog. [Online]. Available:

http://krasserm.github.io/2018/03/19/gaussian-processes/. [Accessed: 18-Dec-2021].

[4] z_ai, “Probability learning II: How Bayes' theorem is applied in machine learning,” Medium, 14-Nov-2020. [Online]. Available: https://towardsdatascience.com/probability-learning-ii-how-bayes-theorem-is-applied-in-machine-learning-bd747a960962. [Accessed: 18-Dec-2021].

[5] M. Martinez-Ramon, “Gaussian processes in a feature (kernel) Hilbert space” in ECE 517, 2021

[6] M. Martinez-Ramon, “Gaussian processes in a feature (kernel) Hilbert space” in ECE 517, 2021

[7] M. Martinez-Ramon “lntroduction to linear Gaussian processes for machine learning” in ECE 517, 2021

[8] Berger, J. O. (1985). Statistical Decision Theory and Bayesian Analysis. Springer, New York. Second edition

[9] Stone, James. (2013). Bayes' Rule: A Tutorial Introduction to Bayesian Analysis. 10.13140/2.1.1371.6801.