Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification

Final Project Stochastic Process For Data Science

Name: Nthimbo Gift Tembo

Course: Data Science

Matricola: 1738522

Table of Contents

1. Gaussian Processes For Regression

In this chapter I describe the Gaussian processes for regression in a narrow way and provide the technicalities and the methodology used for regression. This constrains every detail to the data under simulation in the context. I this project work the concern is placed on supervised learning for Gaussian processes under regression and classification.

1. Introduction

Regression is one kind of supervised machine learning model where the outputs are continuous values. In general the input is denoted as X and the output as Y Hence, we have a dataset of the form $D=\{(X_i, Y_i): i=1, ...,n\}$. Given such training data, the goal is to make predictions of the targets for unseen input data X^{new} . This is apparently shown that the learning problem is inductive. Therefore, I aim to move from the finite dimension to the infinite dimension of functions that make predictions for all input data. There are different kinds of methodologies that are used to deal with supervised learning models. Some of the approaches are; first one giving a prior probability to every possible function and the second method involves restricting the class of functions to be considered. However, both these methods have their own flaws. For instance, the method does not seem to be feasible as there are so many functions to be considered which might lead to infinite. The second consideration might make our models lose some flexibility hence not giving proper prediction for out of sample data.

In line with the first method is where we introduce the possibility of Gaussian processes which is a generalization of a Gaussian probability distribution. Whereas a probability distribution describes random variables which are scalars or vectors (for multivariate distributions), ***a stochastic process*** governs the properties of functions. Gaussian processes are focused on approximating the marginal likelihood function given as;

$$p(y|X) = \int p(y|f, X)p(f|X, \theta)p(\theta)df \qquad [1.1]$$

where θ is a set containing hyper parameters for the kernels specified in the model, X is the data matrix (or design matrix). For infinite functions the marginal likelihood is not analytically tractable hence, MCMC is used to make integrations of the likelihood. In this project, I use Stan Library which

contains the hybrid version of MCMC for integrating the likelihood. The kernel function used for covariance matrix is the Radial Basis Function commonly known as the squared exponential function and is of the form;

$$k(x', x) = \sigma^2 exp(\frac{d(x', x)^2}{2l^2}) \qquad [1.2]$$

then θ in this case is θ := {σ, l}. We use θ because when the kernel, or model, becomes more complex, it becomes more tedious to write out all of these parameters. You will see this notation in much of the Bayesian Statistics literature. $d(X^i, X^j)$ is usually the euclidean distance, but we can use other distance metrics as well. X^i and X^j just refer to rows of X at different axes. This kind of kernel generates very smooth functions.

The Covariance matrix is defined for training data as below;

$$\operatorname{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \quad \text{or} \quad \operatorname{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 I, \qquad [1.3]$$

Where δ_{pq} is the Kronecker delta, which is one if and only if $p = q$ else its zero. This follows from the independence assumption about the noise. Hence in general because of the additive noise the model is given as;

$$y = f(X) + noise \qquad [1.4]$$

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right). \qquad [1.5]$$

The predictive distribution is then given as;

$$\mathbf{f}_* | X, \mathbf{y}, X_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)), \quad \text{where} \qquad [1.6]$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_* | X, \mathbf{y}, X_*] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} \mathbf{y}, \qquad [1.7]$$

$$\operatorname{cov}(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*) \qquad [1.8]$$

We leverage all four disjoint subsets of the full GP's covariance matrix above.

2. Data For regression

Data generated in the context of regression involved application of Gaussian noise for both the training and testing data. Cases were generated in which the input variable, x, was drawn from a standard Gaussian distribution, and the possible target value came from a distribution with mean of,

$$0.3 + 0.4x + 0.5\sin(2.7x) + 1.1/(1+x^2) \quad [1.3]$$

For most cases, the distribution of the target about this mean was Gaussian with standard deviation 0.1. However, with probability 0.05, a case was made an “outliers”, for which the standard deviation was 1.0 instead.

I modeled the data using Gaussian process for regression with one model having the assumption of Gaussian noise and the other model with the assumption of noise coming from a Student t distribution with 4 degrees of freedom. The model with t distribution noise assumption is not correct in this case but models the data better than the actual model with Gaussian noise assumption. Figure 1.1 shows the data generated containing both training and test samples. Figure 1.2 shows test samples in green color.

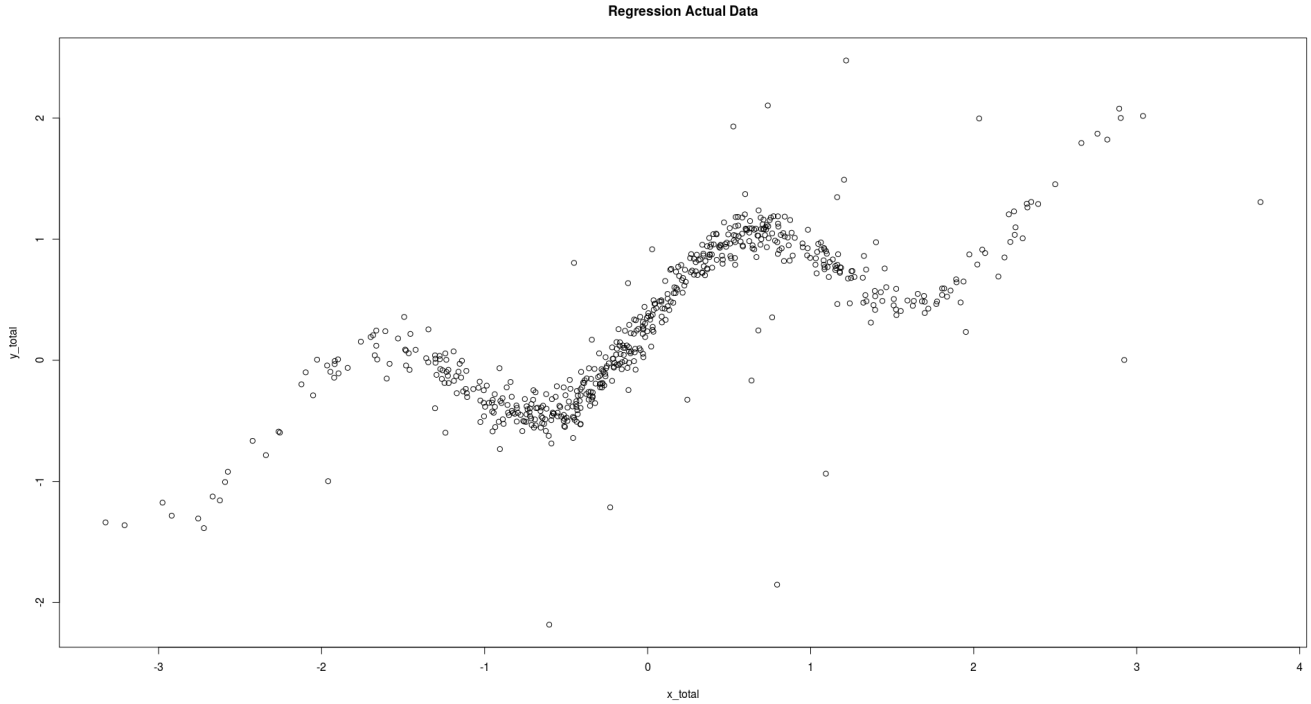

Figure 1.1: Actual Data generated for Regression

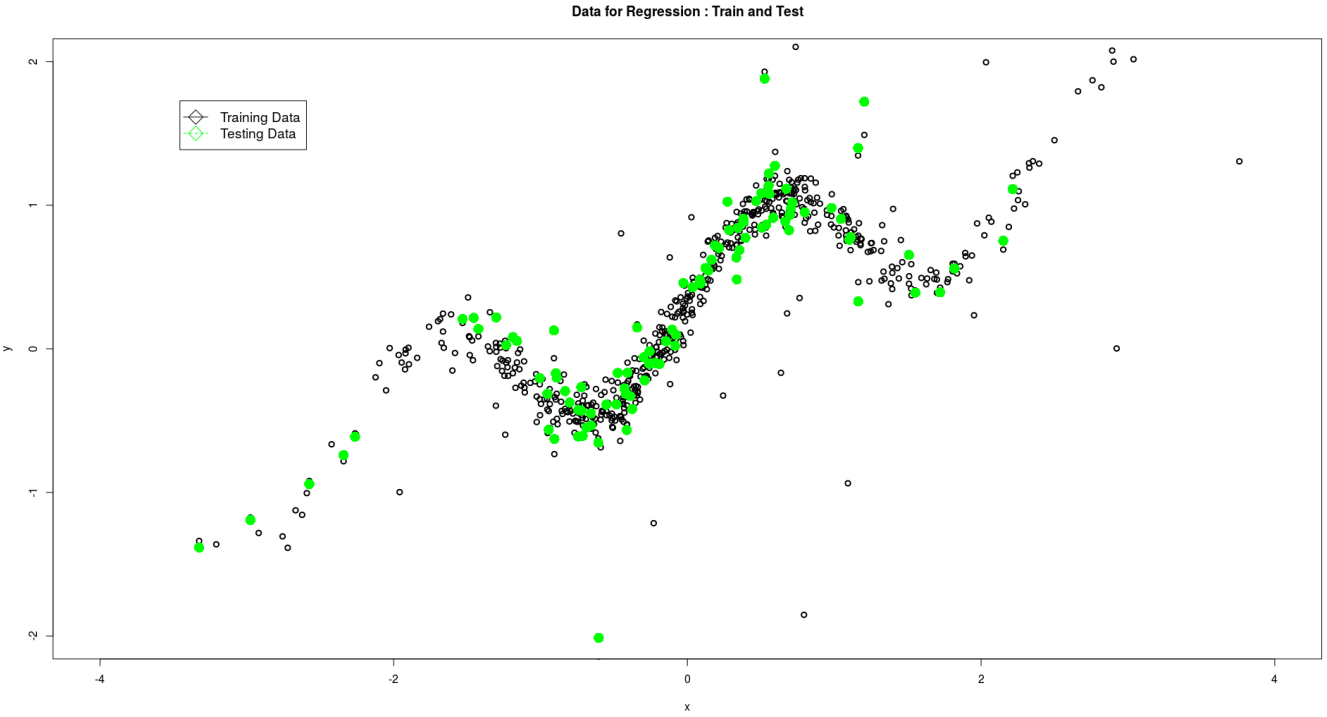

Figure 1.2 : Data for Regression divided into Training and Testing Samples.

3. Results and Generated Graphs

After running the models for the described data I plotted both the posterior realizations and the predictive realizations of the functions. The 100 training cases are shown as dots, with the input on the horizontal axis, and the target on the vertical axis. The green line gives the mean of the predictive distribution using a model in which the noise was assumed to come from a t distribution with 4 degrees of freedom. The red line gives the mean of the predictive distribution using a model in which the noise was assumed to be Gaussian as shown in the figures below.

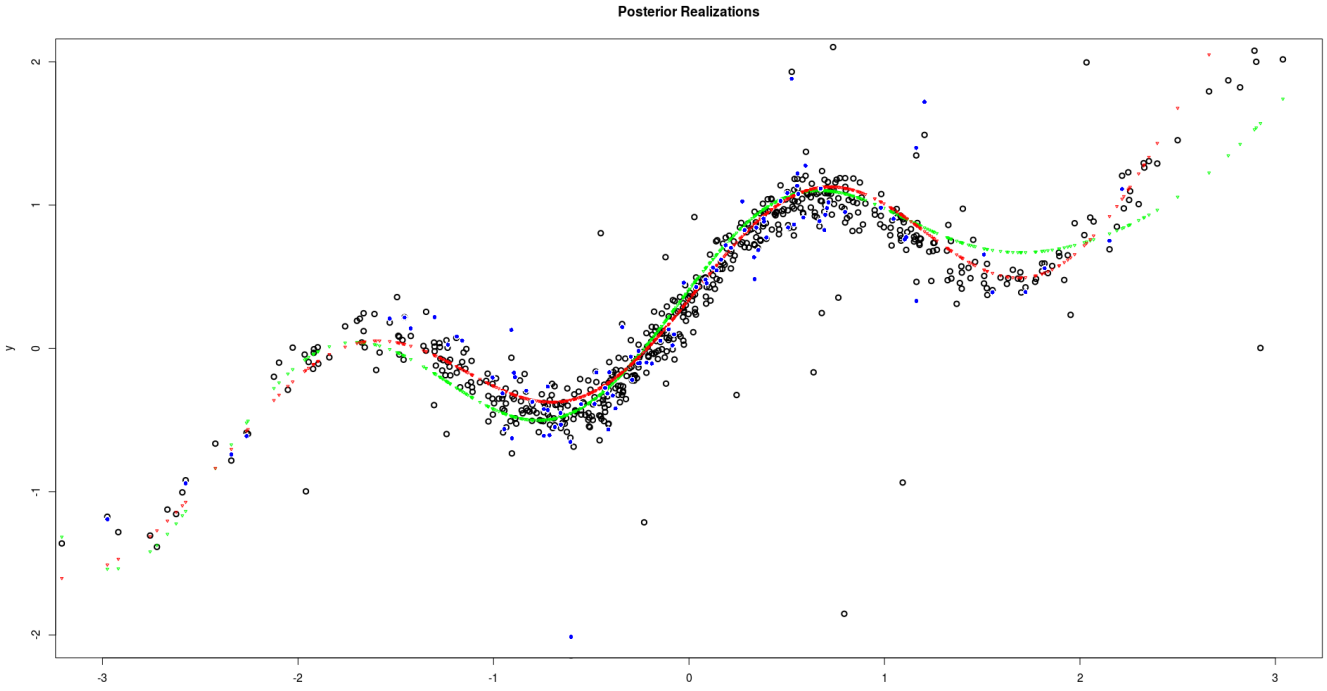

Figure 1.3 : Show of Posterior realizations. The green line shows model under Gaussian noise assumption and the red shows model under t-distribution noise assumption. The black dots are the training samples while the blue dots are the test samples.

As can be seen from the figure above the model with Student t-distribution noise models the data quite well in comparison to the model under Gaussian noise assumption. Figures 1.4 shows model Gaussian noise assumption noise and figure 1.5 shows model with Student t-distribution noise assumption in singularity.

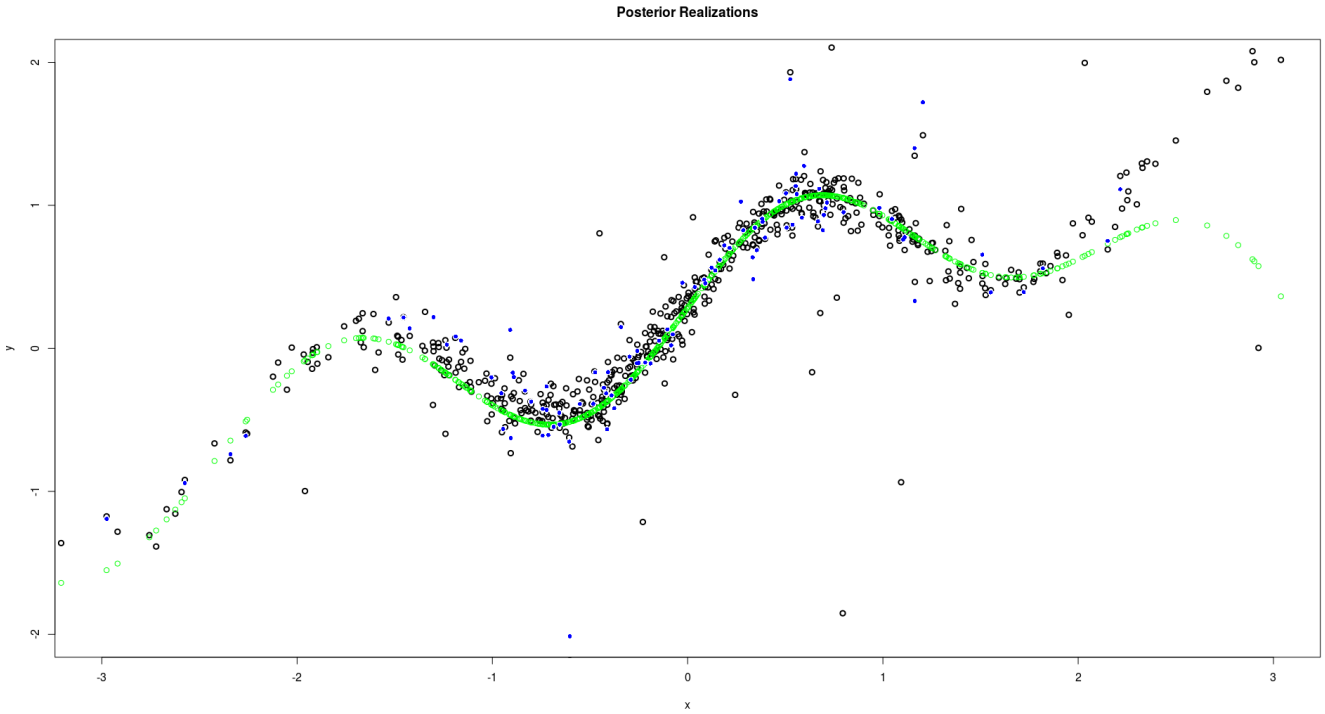

Figure 1.4 : Model with Gaussian noise.

The figure above shows single plot of model with Gaussian noise. The model tries to follow even the outliers which in this case is at a disadvantage with comparison to the model with Student t-distribution noise. This is clearly shown at the end of the line where the line does not follow actual test data point which is in blue in this case.

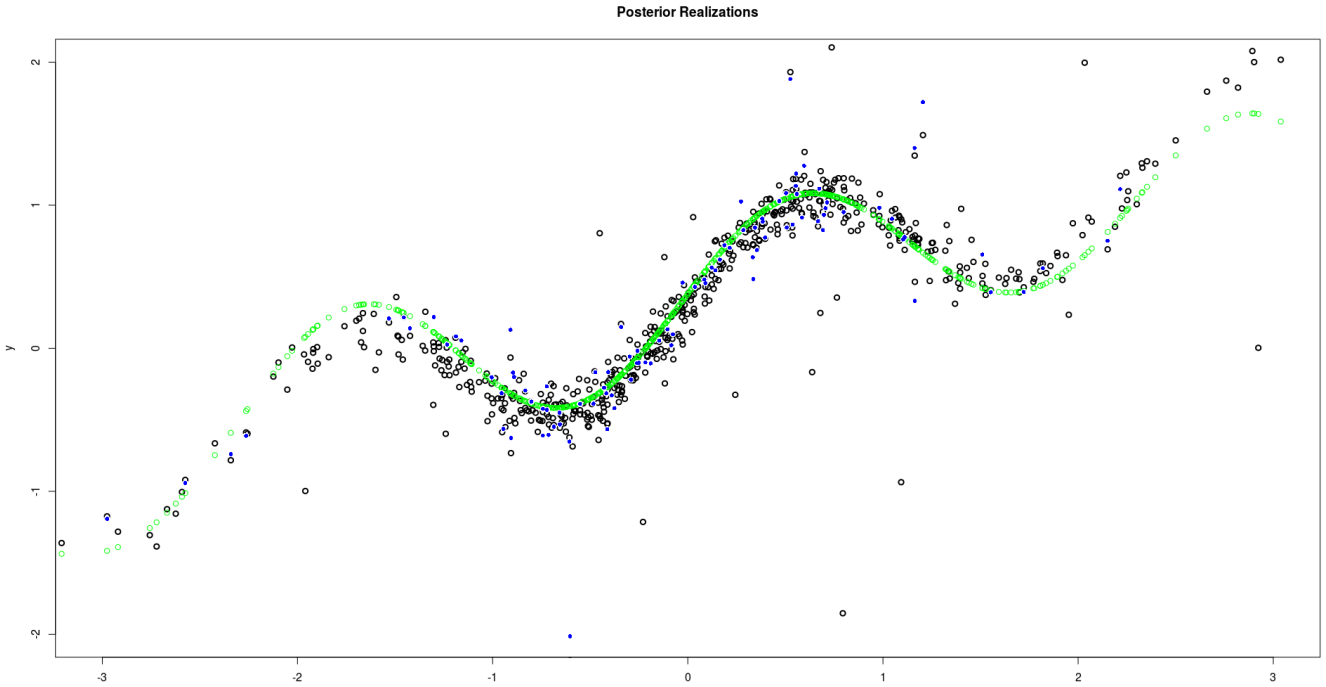

Figure 1.5 : Model with Student-t distribution noise Assumption

Though the difference in modeling the data between the two models is subtle, the model under Student-t distribution noise assumption models the data in question quite well in comparison to the model with Gaussian noise. The advantage is apparent in the way the model under Student-t distribution noise models the data in last part of the graphs. This advantage is also shown in the graphs for posterior predictive results in figure 1.6.

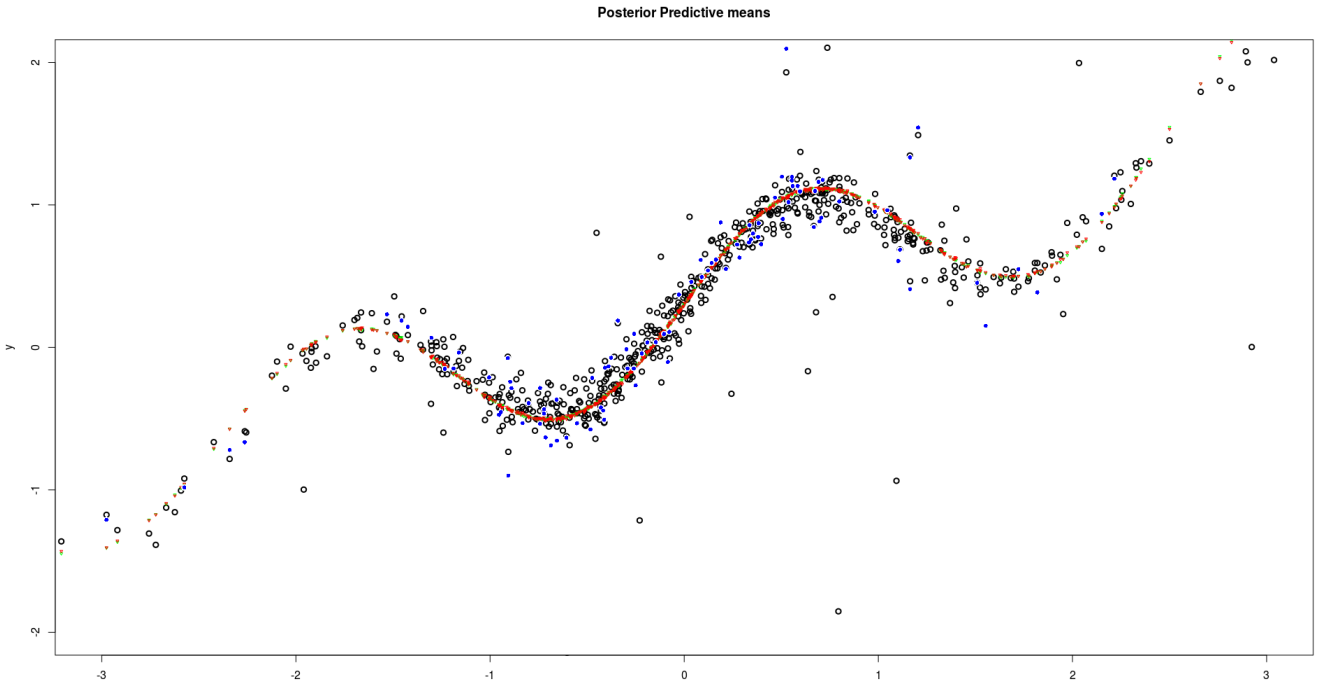

Figure 1.6 : Posterior Predictive means for the two models. The green dots are for Gaussian noise model and the red dots for the Student-t distribution model.

The posterior predictive means for the two models shows a very minimal difference. In this case both models have a good perhaps a similar way of modeling the data, though with intuition one would think that the model with Student-t noise would not model the data quite well since the data was generated using the Gaussian noise.

2. Gaussian Processes For Classification

1. Introduction

In both contexts, Regression and Classification processes for Gaussian Processes the predictions are viewed as function approximation problems **[01]**. The intention is to predict an infinite set of functions that model the data to the expectations while varying the hyper-parameters of the kernel function. Aside from the Gaussian regression processes, the Gaussian Classification problems presents a rather demanding challenge for the predictions of these functions. This is because in classification problems the assumption of the Gaussian Likelihood does not hold. For classification problems, I considered using either the discriminative or the generative approach. However, the discriminative approach presents a more daunting challenge when calculating the posterior predictive distributions as its not direct approach. The two approaches are described below. The main intention is to find the joint distribution of the target given the data points. Hence, we need $P(Y, X)$. Using the well known Bayesian formula this can be gotten in two different ways either by $P(Y, X) = P(Y|X)P(X)$ or $P(Y, X) = P(X|Y)P(Y)$. Hence, overall we need to choose which approach is better to our modeling of the problem presented.

1. Discriminative approach.
   The discriminative approach models $P(y|x)$

2. Generative approach.
   The generative approach models the class condition distributions given the data. That is we need to find $P(X|y)$ where y represents class labels. We need to find how much probabilities each of the defined classes presents in terms of the given data point. Therefore, we find the aggregate class probabilities for that given data point with the aim of choosing a class that has contributes more to the aggregation. As can be seen that the generative approach is more direct as compared to the discriminative approach which would otherwise make us go round in cycles just to get the probabilities.

2. Data For Classification

The generation of data for three way classification problem involved drawing data for pairs (X^i, Y^i) . This data was generated by first randomly drawing quantities X_1^i, X_2^i, X_3^i and X_4^i independently from a uniform distribution over the interval (0, 1). The classes of each item t^i was encoded as 0, 1, or 2 was selected as follows;

If the two-dimensional Euclidean distance of (X_1^i, X_2^i) from the point (0.4, 0.5) was less than 0.35, the class was set to 0; otherwise, if $0.8 * X_1^i + 1.8 * X_2^i$ was less than 0.6, the class was set to 1; and if neither of these conditions held, the class was set to 2. Hence, in this context the remaining two features for each case have no impact on the generated data for the targets. I generated 1000 cases in this way, of which 400 were used for training the model, and 600 for testing the resulting predictive performance. The 400 training case are shown in Figure 2.1 below.

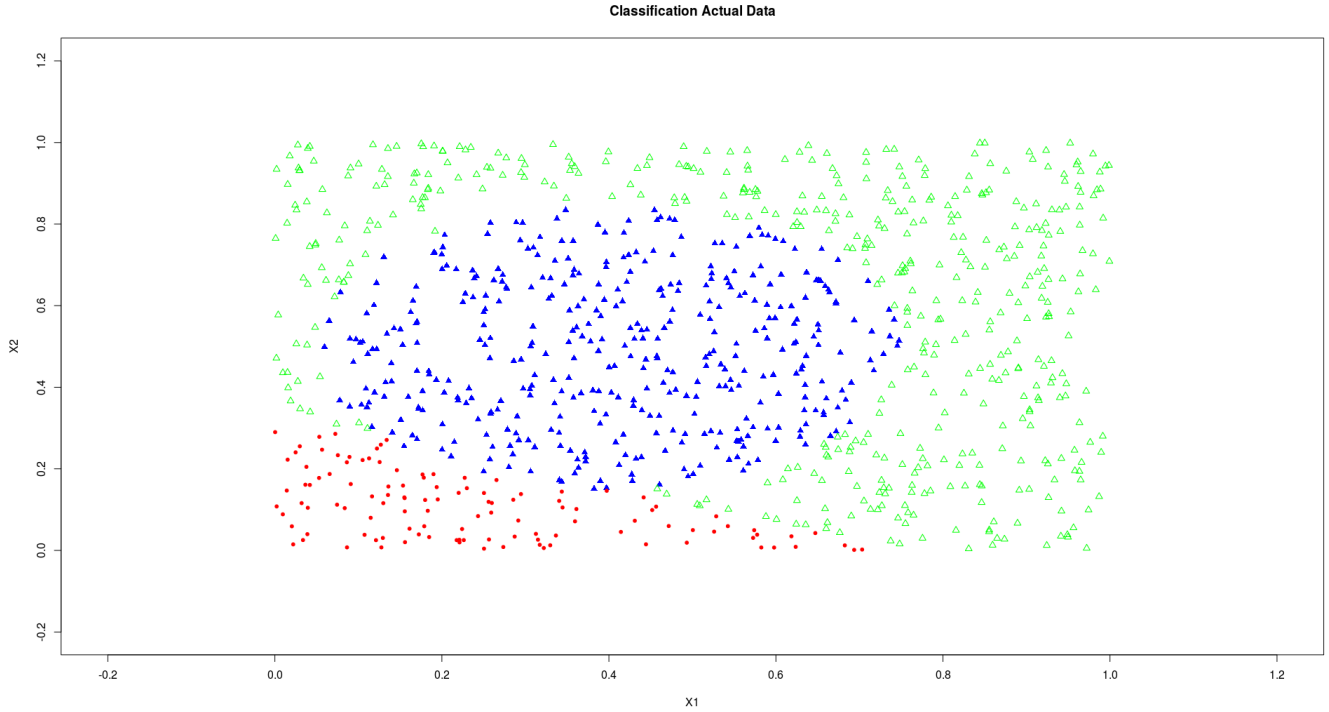

Figure 2.1 : Actual training data generated for classification.

3. Results and Generated Graphs

The data for Classification was modeled using Gaussian Processes for latent values y^i , whose co-variance function consisted of three terms — a constant part (fixed at 10), an exponential part in which the magnitude, η, and the scales for the four inputs, p_u , were variable hyper-parameters, and a jitter part, fixed at J = 10.

To make predictions for test cases, we can average together the predictive probabilities based on iterations after equilibrium was apparently reached. The data used in the model consist of 400 training cases and 600 test cases. The final predictive probabilities were found by averaging the predictions found in this way for each of the iterations used. The figure 2.2 below shows the changes in the values of p_u .

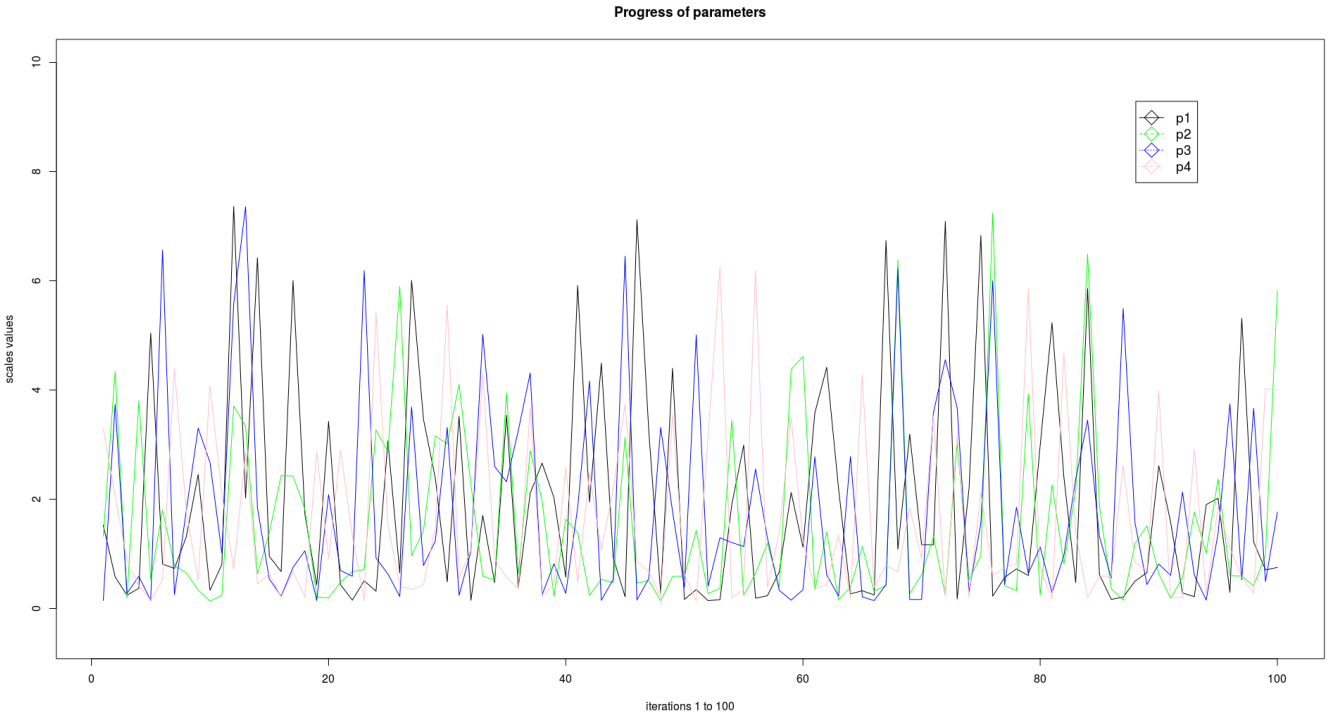

Figure 2.2 : Progress of the four hyper-parameters for 100 iterations.

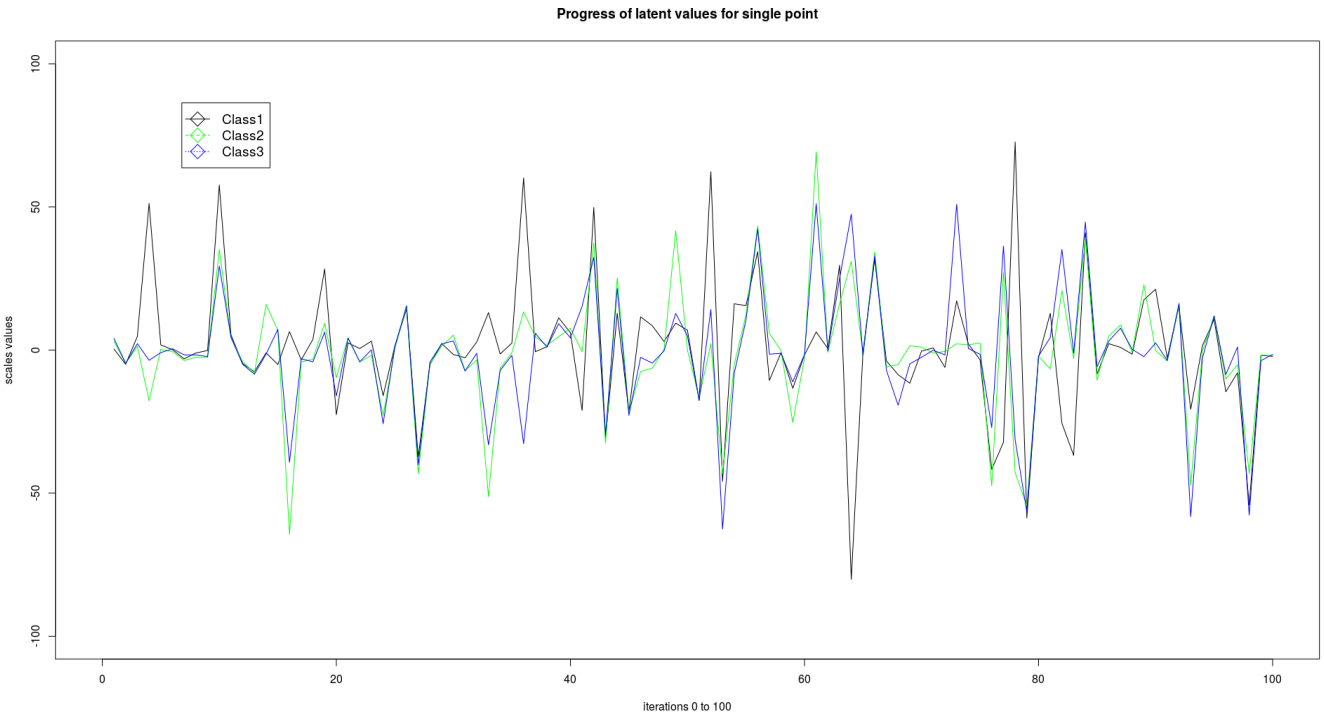

Figure 2.3 : The progress of latent values for each class in the 100 iterations.

3. Conclusions

The covariance function used contains hyper-parameters. It is noted that the errors produced when I vary the hyper-parameters are of different magnitudes. If for instance the scale is varied while keeping the other parameters constant then the posterior realizations will be different as opposed to varying all the hyper-parameters. Hence, in conclusion I can say that Gaussian processes are highly influenced by the hyper-parameters of the covariance function used. Gaussian processes provide a flexible means of non-parametrically modeling regression behavior, but that flexibility also hinders the performance of point estimates derived from maximum marginal likelihood and even regularized maximum marginal likelihood, especially when the observed data are sparse. In order to take best advantage of this flexibility we need to infer Gaussian process hyper-parameters with Bayesian methods. However, inferring the hyper-parameters is not quite easy.

4. References

1. C. E. Rasmussen & C. K. I. Williams, “Gaussian Processes for Machine Learning”, The MIT Press, 2006.
2. Andre Zapico, “Applied Gaussian Processes in Stan”, A Case Study, 2016.
3. Radford M. Neal, “Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification”, University of Toronto, Technical Report No. 9702, January 20, 1997.