Kurosh Zamani
Contact: kurosh.zamany@gmail.com

Introduction

In parametric Gaussian Regression we already used the set of basis functions $\Phi(\boldsymbol{x})$ to map inputs into some high dimensional space and applied the linear model in this space. The linear model became than:

$$f_{\boldsymbol{X}} = \Phi_{\boldsymbol{X}}^T \boldsymbol{w} \tag{1}$$

$$y = f_{\boldsymbol{X}} + \epsilon \tag{2}$$

$$f_{\boldsymbol{X}} = \begin{bmatrix} f_X \\ f_{\boldsymbol{x}} \end{bmatrix} \tag{3}$$

Notice that training points are denoted by symbol X and the symbol $\boldsymbol{x}$ denotes test points.

$$y = f_{\boldsymbol{X}} + \epsilon \tag{4}$$

We put then a zero mean Gaussian prior with covariance Σ on $\boldsymbol{w}$'s and used the properties of Gaussian distribution to calculate the likelihood 5 and posterior 6 as follows:

$$p(\boldsymbol{y} \mid \boldsymbol{w}, \Phi_{\boldsymbol{X}}) = \mathcal{N}(\boldsymbol{y}; \Phi_{\boldsymbol{X}}^T \boldsymbol{w}, \sigma^2 I) = \mathcal{N}(\boldsymbol{y}; f_{\boldsymbol{X}}, \sigma^2 I), \tag{5}$$

$$\begin{aligned} p(\boldsymbol{w} \mid \boldsymbol{y}, \Phi_{\boldsymbol{X}}) = \mathcal{N}(\boldsymbol{w};\ & \mu + \Sigma\Phi_{\boldsymbol{X}}(\Phi_{\boldsymbol{X}}^T \Sigma \Phi_{\boldsymbol{X}} + \sigma^2 I)^{-1}(\boldsymbol{y} - \Phi_{\boldsymbol{X}}^T \mu), \\ & \Sigma - \Sigma\Phi_{\boldsymbol{X}}(\Phi_{\boldsymbol{X}}^T \Sigma \Phi_{\boldsymbol{X}} + \sigma^2 I)^{-1} \Phi_{\boldsymbol{X}}^T \Sigma \end{aligned} \tag{6}$$

Once we have this posteriors on weights, we can use the fact the function values $f_{\boldsymbol{X}}$ are just a linear map of the weights eq.1 , and that the linear map of the Gaussian distribution is still Gaussian with linear transformation of the original one, to calculate the posterior on $f_{\boldsymbol{x}}$ and get:

$$\begin{aligned} p(f_{\boldsymbol{x}} \mid \boldsymbol{y}, \Phi_X) = \mathcal{N}(f_{\boldsymbol{x}};\ & \Phi_{\boldsymbol{x}}^T \mu + \Phi_{\boldsymbol{x}}^T \Sigma \Phi_X (\Phi_X^T \Sigma \Phi_X + \sigma^2 I)^{-1}(\boldsymbol{y} - \Phi_X^T \mu), \\ & \Phi_{\boldsymbol{x}}^T \Sigma \Phi_X - \Phi_{\boldsymbol{x}}^T \Sigma \Phi_X (\Phi_X^T \Sigma \Phi_X + \sigma^2 I)^{-1} \Phi_X^T \Sigma \Phi_{\boldsymbol{x}} \end{aligned} \tag{7}$$

We then can notice the similarities between some of the terms in equation above and clean up the equation with following substitutions:

$$m_a := \Phi_a^T \mu$$

$$K_{ab} := \Phi_a^T \Sigma \Phi_b$$

which leads to:

$$\begin{aligned} p(f_{\boldsymbol{x}} \mid \boldsymbol{y}, \Phi_X) = \mathcal{N}(f_{\boldsymbol{x}};\ & m_{\boldsymbol{x}} + K_{\boldsymbol{x}X}(K_{XX} + \sigma^2 I)^{-1}(\boldsymbol{y} - m_X), \\ & K_{\boldsymbol{x}X} - K_{\boldsymbol{x}X}(K_{XX} + \sigma^2 I)^{-1} K_{X\boldsymbol{x}} \end{aligned} \tag{8}$$

One method to optimize this approach is to choose the basis functions wisely according to the data so that they can reproduce the behavior of the data as much as close as possible. Another possible approach is to ask how many of these functions we should use to optimize the model. It turns out that we can use the infinite numbers of this features which give rise to Gaussian Process Regression.

Solution Method

From the previous discussion we know that the posterior distribution is also Gaussian. We also know that a Gaussian distribution can completely described by its mean and covariance so that we can simply choose the mean and Kernel in eq. 8 directly instead of defining basis functions. The only problem is that the Gaussian distribution is only defined for finite dimensional random vectors.The solution is to define Gaussian process as follows:

Definition: A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{9}$$

In order to implement this idea in Matlab code we are going to use some algebraic manipulations of the some terms in eq. 8 to make them computationally more efficient. One Idea would be to use the Cholesky decomposition to compute the term $A^{-1} := (K_{XX} + \sigma^2 I)^{-1}$ and solve a system of linear equations instead of directly applying the inverse in following manner:

$$A^T x = K_{\boldsymbol{x}} \tag{10a}$$

$$x = A^{-T} K_{\boldsymbol{x}} \tag{10b}$$

$$x^T = K_{\boldsymbol{x}}^T A^{-1} \tag{10c}$$

The equations above are simply mean that we can solve the eq. 10b and transpose it to get $K_{\boldsymbol{x}}^T A^{-1}$. Notice that because A is positive definite, the Cholesky decomposition always exist so that we can write:

$$A = R^T R \tag{11a}$$

$$x = R^{-1} R^{-T} K_{\boldsymbol{x}} \tag{11b}$$

The first kernel we want to try is a simple linear Kernel defined as:

$$\texttt{lin\_}\text{kernel} := \sigma_b^2 + \sigma_a^2 (x - c)(x' - c)^T \tag{12}$$

This is the most simplest kernel we can choose and the results should be simply like a Baysian linear regression. Fig. 1 shows the result of such a kernel.
Another kernel we can use is the squared exponential kernel which also known as the Gaussian kernel:

$$\texttt{se\_}\text{kernel} := \sigma^2 exp(-\frac{(x - c)(x' - c)^T}{2l^2}) \tag{13}$$

Till now we chose some parameters by try and error but we can as well optimize this parameters to get the best results. Unfortunately the posterior distribution is not linear function of parameters θ but we can still use the maximum likelihood as follows:

$$p(\boldsymbol{y} \mid \theta) = \int p(\boldsymbol{y} \mid f) p(f \mid \theta)\, df \tag{14a}$$

$$-2 \log(p(\boldsymbol{y} \mid \theta)) = -\boldsymbol{y}^T (K_{XX}(\theta) + \sigma^2 I)^{-1} \boldsymbol{y} + \log(\boldsymbol{G}), \tag{14b}$$

$$\tag{14c}$$

where

$$G =: K_{XX}(\theta) + \sigma^2 I \tag{14d}$$

with the derivative:

$$\frac{\partial - 2\log}{\partial \theta} = -\boldsymbol{y}^T \boldsymbol{G}^{-1} \frac{\partial \boldsymbol{G}}{\partial \theta} \boldsymbol{G}^{-1} \boldsymbol{y} + tr(\boldsymbol{G}^{-1} \frac{\partial \boldsymbol{G}}{\partial \theta}) \tag{15}$$

The only term in G that depends on θ is K so that we can substitute $\frac{\partial \boldsymbol{G}}{\partial \theta}$ with $\frac{\partial K}{\partial \theta}$. In order to calculate this derivative we have to notice that because we are using log of parameters in our implementation we have to compute the gradients accordingly:

$$\begin{aligned} \frac{\partial f}{\partial \log(u)} &= \frac{\partial f}{\partial u} \frac{\partial u}{\partial \log(u)} \\ &= \frac{\partial f}{\partial u} \frac{\partial e^{\log(u)}}{\partial \log(u)} \quad = \frac{\partial f}{\partial u} u \end{aligned} \tag{16}$$

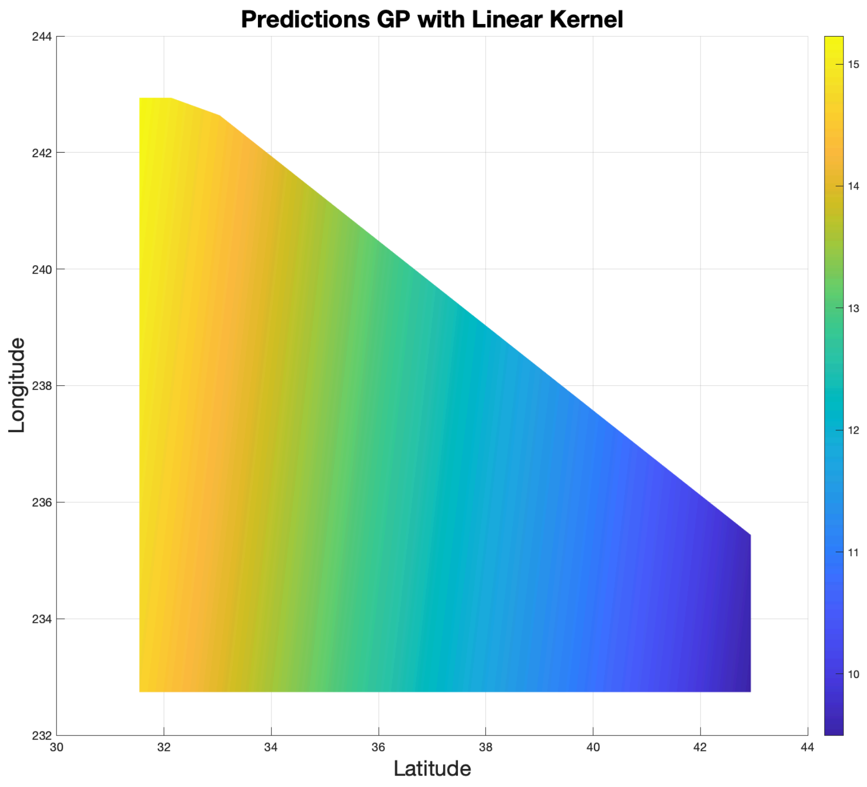

Figure 1 Gaussian Process Regression Predictions - Linear Kernel

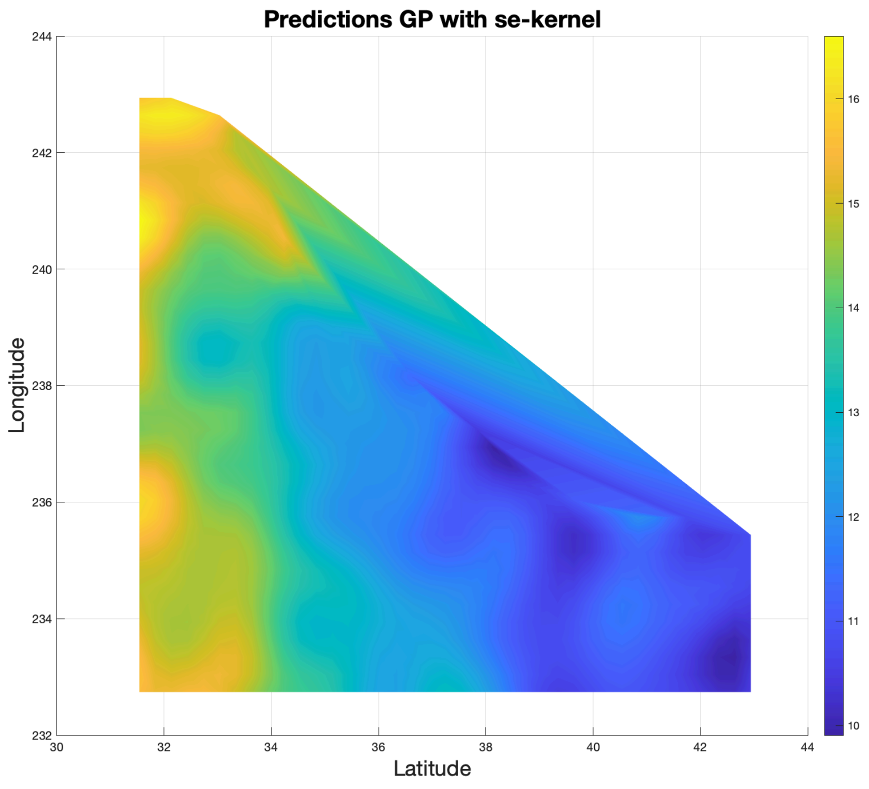

Figure 2 Gaussian Process Regression Predictions - se-kernel

That means that we just need to simply multiply the derivatives with its corresponding parameter. The implementation of this part can be found in se-kernel function in our Code. The prediction with the optimized parameters is shown in Fig. 3

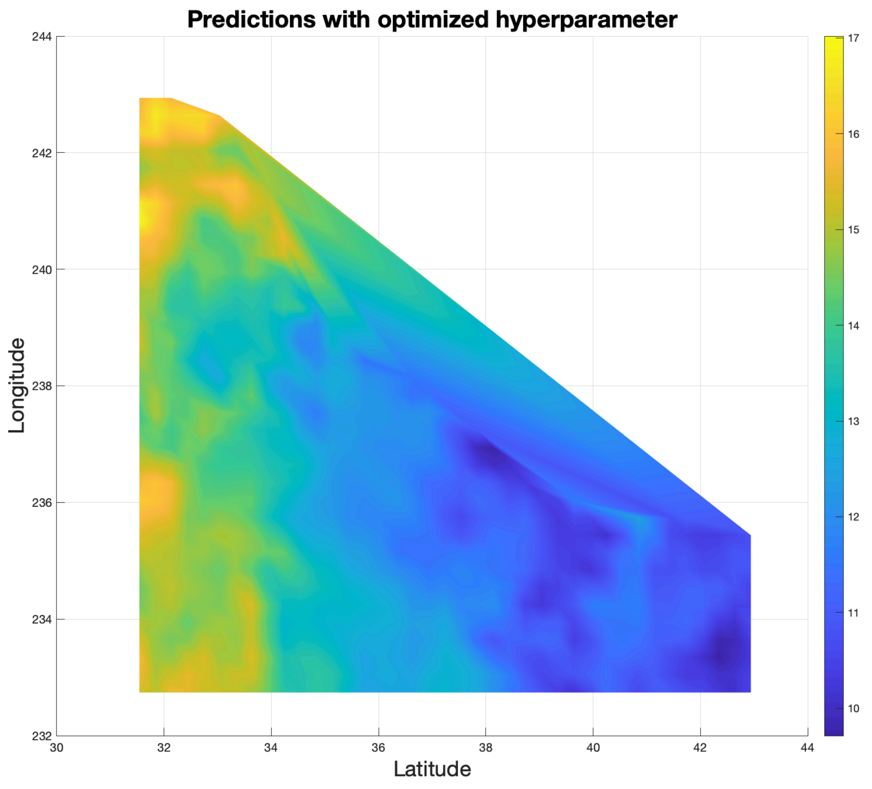

Figure 3 Predictions with optimized Hyperparameters- se-Kernel

Results and Conclusion

Our first simple model worked as expected much like a linear Baysian Inference. It captured the gist of the data for sure but it lacks severely in details. Conversely the second model with a nonlinear properties (with se-kernel) added the lacking details and made the model much better representative of the data. The third model with optimized hyperparameters yet refined our previously good model and managed to make the square error of predictions down to just nearly 25 as apposed to 75 for our non-optimized model. This fact reflect itself also in prediction plots Fig. 3 vs. 2. Although they both very similar but the results in 3 are much more refined.