Gaussian Process Regression From First Principles

Ryan Sander
MIT CSAIL

Abstract

Gaussian Process Regression (GPR) is a remarkably powerful class of supervised machine learning algorithms that, in stark contrast to many of today's state-of-the-art regression models, relies on few parameters to make generalizable predictions of targets at test points. In this review paper, we work through the Ab initio mechanics of how GPR models make predictions over latent functions, how these models are optimized, and the covariance functions that govern how GPR models measure similarities between features.

1 Gaussian Processes

A Gaussian Process (GP) is a collection of random variables $\{\mathbf{X}_i\}_{i=1}^{n}$, such that any subset/collection of these variables is jointly Gaussian [2]:

$$\mathbf{X}_i, ..., \mathbf{X}_j \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{1}$$

Where d is the number of random variables in the subset, $\boldsymbol{\mu} \in \mathcal{R}^d$ is a vector of mean values, and $\boldsymbol{\Sigma} \in \mathcal{R}^{d \times d}$ is the covariance matrix between the random variables. Written as a joint Gaussian density:

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \det |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \mathbf{x} = \begin{bmatrix}\mathbf{x}_i & ... & \mathbf{x}_j\end{bmatrix}^T \in \mathcal{R}^d \tag{2}$$

One conceptualization of a GP is that it defines a distribution over functions: Given a Gaussian Process, specified by a mean and covariance function, we can sample a function at the point $\mathbf{x} \in \mathbb{R}^d$ from the Gaussian Process according to:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3}$$

Where $m(\cdot)$ is a mean function, and $k(\cdot, \cdot)$ is a covariance function. This is known as the function-space view of GPs [2].

2 Gaussian Process Regression

One application of Gaussian Processes is to perform regression via supervised learning, hence the name Gaussian Process Regression. This regression can be conceptualized as kernelized Bayesian linear regression, where the kernel parameterization is determined by the choice of covariance/kernel function, as well as the data used to make predictions [2]. This is known as the "weight-space view" of GPR [2].

Gaussian Process Regression can also be conceptualized in the aforementioned function-space view, in which the learner learns a distribution over functions [2] by learning mean and covariance functions of the realization of the GP at $\mathbf{x} \in \mathbb{R}^d$, denoted by $f(\mathbf{x})$:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x} - m(\mathbf{x})(f(\mathbf{x}' - m(\mathbf{x}'))]$$

The realization of a Gaussian Process $f(\mathbf{x})$ corresponds to a random variable. With these functions specified, $f(\mathbf{x})$ can be sampled from the Gaussian Process using equation 3 above:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{4}$$

This is a formalization of sampling a random variable $f(\mathbf{x})$ that depends on location $\mathbf{x} \in \mathbb{R}^d$. Estimates of the mean are produced as a linear combination of true, observed means [2]. The weighting coefficients used to produce these mean estimates are independent of the target values, placing Gaussian Process Regression models into the class of *linear smoothers* [2].

Given a **training** dataset consisting of N observations:

$$D_{\text{train}} \triangleq (\mathbf{X}, \mathbf{y}) \triangleq \{\mathbf{x}_i, y_i\}_{i=1}^{N}, \mathbf{x}_i \in \mathbb{R}^d, y \in \mathbb{R}, \tag{5}$$

As well as a **test** dataset of N' points::

$$D_{\text{test}} \triangleq \mathbf{X}_* \triangleq \{\mathbf{x}_{*,i}\}_{i=1}^{N'}, \mathbf{x}_{*,i} \in \mathbb{R}^d \tag{6}$$

GPR predicts a posterior Gaussian distribution of targets over test points $\mathbf{X}_*$ by computing the parameters of this Gaussian distribution given observed training data. We will consider both the noise-free case, and then use this to generalize to the case that models for noise in the observed target values $\mathbf{y}$.

Below, $m(\cdot)$ represents a mean function, $K(\cdot, \cdot)$ is a kernel/covariance function, $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a matrix of training inputs, $\mathbf{y} \in \mathbb{R}^n$ is a vector of training targets, $\text{Cov}(\cdot)$ is a covariance operator, and $\sigma_n^2 \in \mathbb{R}^+$ is a positive hyperparameter denoting the covariance noise of the Gaussian Process.

2.1 Noise-Free Predictions

In the noise-free case, we have equality between the observed training targets $\mathbf{y}$ and the realized values of the Gaussian Process $\mathbf{f}$ [2]:

$$\mathbf{y} \triangleq \begin{bmatrix} y_1 & \cdots & y_N \end{bmatrix} = \mathbf{f} \triangleq \begin{bmatrix} f(\mathbf{X}_1) & \cdots & f(\mathbf{X}_N) \end{bmatrix} \tag{7}$$

Given the datasets D_{train} and D_{test} defined above, the joint distribution of Gaussian process functions over the training and testing datasets is given by [2]:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) & K(\mathbf{X}, \mathbf{X}_*) \\ K(\mathbf{X}_*, \mathbf{X}) & K(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right) \tag{8}$$

However, we are interested in the posterior distribution of the predicted GP realizations $\mathbf{f}_*$ at the test points $\mathbf{X}_*$. This posterior distribution can be computed by conditioning the predicted realizations of the Gaussian Process at the test points, given by $\mathbf{f}_*$, on the realizations of the Gaussian Process at the points defined by the training dataset, given by $\mathbf{f}$:

$$\begin{aligned} \mathbf{f}_* | \mathbf{X}_*, \mathbf{X}, \mathbf{f} \sim \mathcal{N}(&K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} \mathbf{f}, \\ &K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{X}_*)) \end{aligned} \tag{9}$$

We can therefore write the **mean** and **covariance** predictions at the test points $\mathbf{X}_*$ as:

$$\begin{aligned} \bar{f}(\mathbf{X}_*) &= K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} \mathbf{f} \\ \text{Cov}(f(\mathbf{X}_*)) &= K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{X}_*) \end{aligned} \tag{10}$$

In some applications, the mean function need not be set to zero, and can be written as a generalized $m(\mathbf{X}) \in \mathbb{R}^n$. In this case, with a non-zero mean function, the predicted **mean** and **covariance** estimates at the test points become:

$$\bar{f}(\mathbf{X}_*) = m(\mathbf{X}_*) + K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})(\mathbf{f} - m(\mathbf{X})) \tag{11}$$

$$\text{Cov}(f(\mathbf{X}_*)) = K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{X}_*) \tag{12}$$

2.2 Predicting in the Presence of Noise

In many applications of Gaussian Process Regression, it is common to model the training targets $\mathbf{y}$ to be noisy realizations of the Gaussian Process $\mathbf{f}$ [2], where noise is parameterized by a zero-mean Gaussian with positive noise covariance values given by σ_n^2:

$$y_i = f(\mathbf{x}_i) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{13}$$

Therefore, using equations 13 and 14, we can express the joint distribution over observed training targets $\mathbf{y}$ and predicted realizations of the Gaussian Process $\mathbf{f}_*$ at test points $\mathbf{X}_*$:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & K(\mathbf{X},\mathbf{X}_*) \\ K(\mathbf{X}_*,\mathbf{X}) & K(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix}\right) \tag{14}$$

Using the same conditioning process as in 15, we can express the conditional distribution of predicted Gaussian Process realizations conditioned on observed training targets $\mathbf{y}$ as:

$$\begin{aligned} \mathbf{f}_*|\mathbf{X}_*,\mathbf{X},\mathbf{y} \sim \mathcal{N}(&K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y}, \\ &K(\mathbf{X}_*,\mathbf{X}_*) - K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}K(\mathbf{X},\mathbf{X}_*)) \end{aligned} \tag{15}$$

Therefore, the predicted mean and covariance values at the test points $\mathbf{X}_*$ are given by (assuming a zero-mean function):

$$\bar{f}(\mathbf{X}_*) = K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y} \tag{16}$$

$$\text{Cov}(f(\mathbf{X}_*)) = K(\mathbf{X}_*,\mathbf{X}_*) - K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}]^{-1}K(\mathbf{X},\mathbf{X}_*) \tag{17}$$

The mean and covariance estimates from using a non-zero mean function $m(\mathbf{X})$ from the noise-free case generalize to this noisy prediction case as well.

3 Covariance Functions

Covariance functions are a crucial component of Gaussian Process Regression models, since they weight the contributions of training points to predicted test targets according to the kernel distance between observed training points $\mathbf{X}$ and test points $\mathbf{X}_*$. Recall from the previous section that one way to conceptualize Gaussian Process Regression prediction is as a linear smoothing mechanism: The mean predictions $\bar{\mathbf{f}}_*$ at test points $\mathbf{X}_*$, in fact, can be expressed as:

$$f_{*,i} \triangleq f(\mathbf{x}_{*,i}) = \sum_{j=1}^{N} \alpha_j k(\mathbf{x}_j, \mathbf{x}_{*,i}) \tag{18}$$

Where $\boldsymbol{\alpha}$ is given as:

$$\boldsymbol{\alpha} = (K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} \tag{19}$$

Therefore, the predicted means are a linear combination of observed means, with linear weights determined by the similarity between the test points and the training point whose mean contribution is being taken into account.

In the next subsection, we will generalize this result even further.

3.1 Mercer's Theorem

Mercer's Theorem allows us to express our kernel $k(\mathbf{x},\mathbf{x}')$ as an eigendecomposition [2]:

$$k(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i^*(\mathbf{x}') \tag{20}$$

The functions $\phi_i(\mathbf{x})$ and its conjugate $\phi_i^*(\mathbf{x}')$ are the eigenfunctions of the kernel (in our case, covariance) function $k(\mathbf{x},\mathbf{x}')$. These eigenfunctions are special because integrating them over a measure, such as a probability density $p(\mathbf{x})$ or Lebesgue measure [2], results in the eigenfunction itself, scaled by eigenvalues denoted by λ_i. We can think of this expression above as a generalized Fourier Series representation of our kernel [2]. Notice that this means we can have an infinite number of basis functions we can use to transform our inputs into features. This is similar in principle to the kernel trick. GPR transforms our standard Bayesian Regression setup into an infinite-dimensional feature space through the covariance function. Depending on the choice of covariance function, this can admit unlimited expressive power for our GPR model. This is part of what makes GPRs so powerful and versatile for a wide range of supervised machine learning tasks.

Combining this result with the linear smoothing function result above, we can rewrite our expression for the

latent predictions as a weighted sum of eigenvalue-eigenfunction combinations involving the basis functions of the kernel:

$$f_{*,i} \triangleq f(\mathbf{x}_{*,i}) = \sum_{j=1}^{N} \alpha_j k(\mathbf{x}_j, \mathbf{x}_{*,i}) \tag{21}$$

$$= \sum_{j=1}^{N} \alpha_j \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{x}_j) \phi_k^*(\mathbf{x}_{*,i}) \tag{22}$$

$$= \sum_{j=1}^{N} \left[(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \right]_j \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{x}_j) \phi_k^*(\mathbf{x}_{*,i}) \tag{23}$$

$$= \sum_{j=1}^{N} \left[\left(\begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} + \sigma_n^2 \mathbf{I} \right)^{-1} \mathbf{y} \right]_j \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{x}_j) \phi_k^*(\mathbf{x}_{*,i}) \tag{24}$$

$$= \sum_{j=1}^{N} \left[\left(\begin{bmatrix} \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{x}_1) \phi_k^*(\mathbf{x}_1) & \cdots & \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{x}_1) \phi_k^*(\mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{x}_N) \phi_k^*(\mathbf{x}_1) & \cdots & \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{x}_N) \phi_k^*(\mathbf{x}_N) \end{bmatrix} + \sigma_n^2 \mathbf{I} \right)^{-1} \mathbf{y} \right]_j \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{x}_j) \phi_k^*(\mathbf{x}_{*,i}) \tag{25}$$

$$\tag{26}$$

What is relevant about the result above? We have shown that the linear combinations of the outputs used to form the prediction are determined by the **eigenfunctions** ϕ_k and **eigenvalues** λ_k of the covariance function applied over the training and testing points. Aside from the choice of covariance function and covariance noise, predictions here are determined entirely from the data itself! This illustrates why Gaussian Process Regression is considered to be non-parametric.

Below are some example covariance functions that are frequently leveraged in Gaussian Process Regression literature [2]:

1. **Squared Exponential (SE) / Radial Basis Function (RBF) Kernel** with Automatic Relevance Determination (ARD) [3]. ARD enables learning separate lengthscales for each input dimension, and is therefore suitable for high-dimensional input spaces with variable scales and output sensitivity in each dimension. This covariance function is given by:

$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{x}')^\top \boldsymbol{\Theta}^{-2} (\mathbf{x} - \mathbf{x}') \right) \tag{27}$$

   Where $\boldsymbol{\Theta} \in \mathbb{R}^{d \times d}$ is a diagonal matrix of lengthscales. This covariance function is also used for other kernel classification and regression problems, such as kernel Support Vector Machines (SVMs) and Support Vector Regression (SVR).

2. **Rational Quadratic (RQ) Kernel** with ARD:

$$k_{\text{RQ}}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{1}{2\alpha} (\mathbf{x} - \mathbf{x}')^\top \boldsymbol{\Theta}^{-2} (\mathbf{x} - \mathbf{x}') \right)^{-\alpha} \tag{28}$$

   Where $\alpha \in \mathbb{R}^+$ is a learned hyperparameter that defines how lengthscales are weighted. The RQ kernel can be conceptualized as an infinite mixture of SE kernels, where the mixture weights are controlled by α [2].

3. **Matérn 5/2 Kernel** [3] with ARD:

$$k_{\text{Matérn},\nu}(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} d \right) K_\nu \left(\sqrt{2\nu} d \right)$$
$$d = (\mathbf{x} - \mathbf{x}')^\top \boldsymbol{\Theta}^{-2} (\mathbf{x} - \mathbf{x}')$$

Where $\nu = \frac{5}{2}$ is a hyperparameter that determines the degree of allowable discontinuity in the interpolation, K_ν is a 2nd order Bessel function, and $\Gamma(\cdot)$ is the Gamma function. Under certain conditions, this kernel allows for perfect interpolation [2].

4 Gaussian Process Regression Hyperparameter Optimization

Although Gaussian Process Regression models are considered to be non-parametric, their hyperparameters, such as lengthscales [2], significantly influence their predictive capabilities, and therefore should be optimized in order to maximize predictive performance in out-of-sample data. Fortunately, these hyperparameters can be optimized using gradient ascent methods [2].

The functional objective considered for optimizing the hyperparameters of a GPR model is the marginal likelihood [2]. However, since this marginal likelihood has exponential terms, generally this optimization is performed by maximizing the marginal log-likelihood [2]. Since the marginal log-likelihood function is a strict monotonic transformation of the marginal likelihood function, the set of parameters that maximizes the marginal log-likelihood will also maximize the marginal likelihood.

The marginal log-likelihood, parameterized by a set of hyperparameters $\boldsymbol{\theta}$, is given by:

$$\log p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T[K(\mathbf{X},\mathbf{X}) + \sigma_n^2]^{-1}\mathbf{y} - \frac{1}{2}\log|K(\mathbf{X},\mathbf{X}) + \sigma_n^2| - \frac{N}{2}\log 2\pi \tag{29}$$

As aforementioned, to optimize the hyperparameters of a GPR model, the derivatives of the marginal log-likelihood are computed with respect to $\boldsymbol{\theta}$:

$$\begin{aligned}
\frac{\partial}{\partial\boldsymbol{\theta}}\log p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta}) &= \frac{1}{2}\mathbf{y}^T K(\mathbf{X},\mathbf{X})^{-1}\frac{\partial K(\mathbf{X},\mathbf{X})}{\partial\boldsymbol{\theta}}K(\mathbf{X},\mathbf{X})^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}\left(K(\mathbf{X},\mathbf{X})^{-1}\frac{\partial K(\mathbf{X},\mathbf{X})}{\partial\boldsymbol{\theta}}\right) \\
&= \frac{1}{2}\mathrm{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - K(\mathbf{X},\mathbf{X})^{-1}\right)\frac{\partial K(\mathbf{X},\mathbf{X})}{\partial\boldsymbol{\theta}}\right) \\
&\text{where } \boldsymbol{\alpha} = K(\mathbf{X},\mathbf{X})^{-1}\mathbf{y}
\end{aligned}$$

These derivatives are then used to update the hyperparameters of the Gaussian Process Regression using gradient ascent methods such as Adam [1]. This results in gradient updates of the form:

$$\begin{aligned}
\boldsymbol{\theta}^{(i+1)} &= \boldsymbol{\theta}^{(i)} + \eta\nabla_{\boldsymbol{\theta}^{(i)}}\log p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta}^{(i)}) \\
&= \boldsymbol{\theta}^{(i)} + \frac{\eta}{2}\mathrm{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - K(\mathbf{X},\mathbf{X})^{-1}\right)\frac{\partial K(\mathbf{X},\mathbf{X})}{\partial\boldsymbol{\theta}^{(i)}}\right)
\end{aligned}$$

Where $\eta \in \mathbb{R}^+$ is the gradient ascent learning rate.

After the GPR's hyperparameters have been optimized on the observed training dataset $(\mathbf{X},\mathbf{y})$, the GPR is ready to perform inference on the test dataset $\mathbf{X}_*$.

5 Review and Conclusion

The above derivations describe the exact analytical framework for how GPR models make predictions, compute kernel distance between input vectors, and optimize hyperparameters.

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

[2] Rasmussen, Carl Edward. *Gaussian processes in machine learning.* Summer school on machine learning, Springer, 63–71, 2003.

[3] Bentley, Jon Louis. *Multidimensional binary search trees used for associative searching.* Commun. ACM 18, 9 (Sept. 1975), 509–517. DOI:https://doi.org/10.1145/361002.361007.