Gaussian Process Regression for Gaussian Random Fields in Cosmology

Verena Alton

2024S 250155-1 Cosmic Structures: Theory, Numerics, and Statistics

July 21, 2024

1 Introduction to Gaussian Process Regression

Gaussian Process Regression (GPR) is a non-parametric, Bayesian approach to regression that is particularly useful for modelling and predicting the behaviour of complex systems where the underlying relationships between variables are unknown or highly nonlinear. GPR is based on the concept of a Gaussian process, which is a generalisation of the Gaussian distribution to infinite-dimensional spaces. [1]

1.1 Gaussian Processes

A Gaussian process (GP) is a collection of random variables which have a joint Gaussian distribution. Formally, a Gaussian process is defined as follows.

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

where $\mathbf{x}$ represents input variables, $m(\mathbf{x})$ is the mean function, and $k(\mathbf{x}, \mathbf{x}')$ is the covariance (or kernel) function. [2, 1]

1.2 Gaussian Process Regression

In GPR, we aim to make predictions about the function f given a set of observed data points $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. The observations are assumed to be related to the latent function $f(\mathbf{x})$ via:

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ represents Gaussian noise with variance σ_n^2.

Given the observed data, we seek to determine the posterior distribution of the function values at new input points $\mathbf{x}^*$. The joint distribution of the observed outputs $\mathbf{y}$ and the function values $\mathbf{f}^*$ at the new inputs is given by:

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}^* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{m}(\mathbf{X}) \\ \mathbf{m}(\mathbf{X}^*) \end{pmatrix}, \begin{pmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I & K(\mathbf{X}, \mathbf{X}^*) \\ K(\mathbf{X}^*, \mathbf{X}) & K(\mathbf{X}^*, \mathbf{X}^*) \end{pmatrix} \right)$$

where: - $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$ is the matrix of training inputs, - $\mathbf{X}^* = [\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_m^*]^T$ is the matrix of test inputs, - $\mathbf{m}(\mathbf{X})$ and $\mathbf{m}(\mathbf{X}^*)$ are the mean vectors of the training and test inputs, - $K(\mathbf{X}, \mathbf{X})$ is the covariance matrix for the training inputs, - $K(\mathbf{X}, \mathbf{X}^*)$ is the covariance matrix between the training and test inputs, - $K(\mathbf{X}^*, \mathbf{X}) = K(\mathbf{X}, \mathbf{X}^*)^T$, - $K(\mathbf{X}^*, \mathbf{X}^*)$ is the covariance matrix for the test inputs. [2, 1]

1.3 Posterior Distribution

To make predictions, we need the conditional distribution of $\mathbf{f}^*$ given $\mathbf{y}$. Using properties of the multivariate normal distribution, this conditional distribution is given by:

$$\mathbf{f}^* | \mathbf{X}, \mathbf{y}, \mathbf{X}^* \sim \mathcal{N}(\mu^*, \mathbf{\Sigma}^*)$$

where:

$$\mu^* = \mathbf{m}(\mathbf{X}^*) + K(\mathbf{X}^*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I]^{-1}(\mathbf{y} - \mathbf{m}(\mathbf{X}))$$
$$\mathbf{\Sigma}^* = K(\mathbf{X}^*, \mathbf{X}^*) - K(\mathbf{X}^*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I]^{-1} K(\mathbf{X}, \mathbf{X}^*)$$

Here, μ^* is the mean of the predictive distribution and $\mathbf{\Sigma}^*$ is the covariance of the predictive distribution. [2, 1]

2 Application of GPR to sample a Gaussian Random Field given a certain covariance and observations

In the lecture, we already learnt how to sample a Gaussian Random Field (GRF) with correlated noise. For this task, we used a Fourier transformation, since in Fourier space the covariance between modes is already diagonal (for a stationary field) and can therefore be expressed by the power spectrum. For a discrete set of distinct wave numbers k_i, where we defined $k_j^{'} := -k_j$, we rewrote the definition of the power spectrum as the diagonal covariance matrix:

$$\tilde{C}_{ij} = \mathbb{E}[\tilde{\delta}(k_i)\tilde{\delta}^*(k_j)] = (2\pi)^3 P(k_i)\delta_{ij}.$$

With that, we created Gaussian random fields with correlated noise, where the degree of correlation depends on the parameter α of the power spectrum $P(k) = k^{-\alpha}$. See Figure 1 for an example of a correlated field with a power spectrum of $P(K) = k^{-3}$. [3]

Now imagine a scenario where the covariance of a GRF is known as well as the values of some observations. How could we sample a full GRF fulfilling the given covariance and matching the given observation values? Here we can use Gaussian process regression.

2.1 Sampling Observations

First, we have to generate some observation values. Of course in real life applications these values come from real, empiric observations. However, in this project we will generate a GRF with Fourier transformation as explained above which will serve as our (latent) function f and then use some randomly chosen observations from it to fit our new GRF with the same covariance as this original field and matching observations.

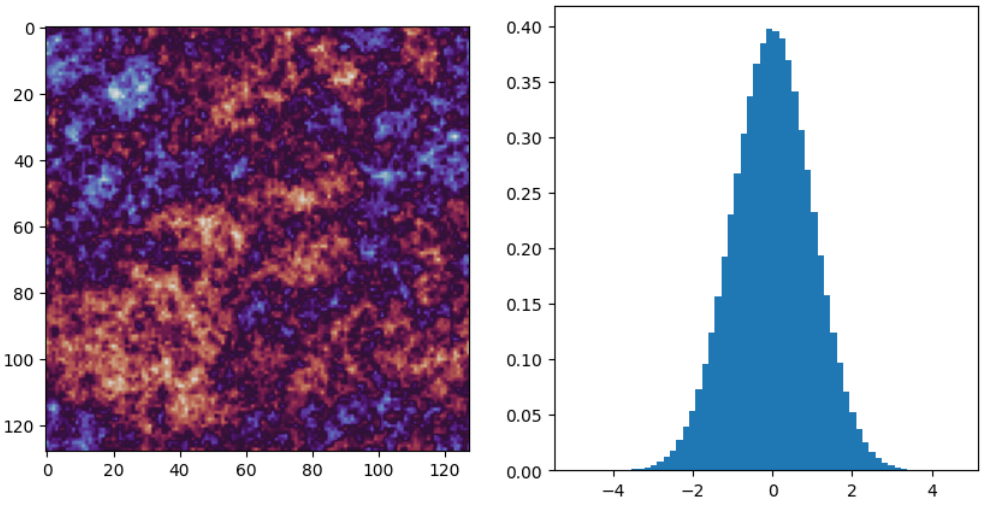

Figure 1: Gaussian random field with correlated noise, characterised by power spectrum $P(k) = k^{-\alpha}$ with $\alpha = 3$.

2.2 Expressing Power Spectrum as Covariance Kernel

Now we have to express our power spectrum as a covariance matrix which the GPR will use to generate suitable distributions. We have to convert the power spectrum, which is in Fourier space, to a covariance matrix in physical space. Therefore, we will use an inverse Fourier transformation on the power spectrum. As suggested in [4], we use that

$$\frac{1}{r^\alpha} = \frac{2\pi^{\alpha/2}}{\Gamma(\alpha/2}\int_0^\infty \lambda^{\alpha-1}e^{-\pi\lambda^2 r^2}d\lambda$$

and derive with $\mu = 1/\lambda$

$$\frac{2\pi^{\alpha/2}}{\Gamma(\alpha/2)}\int_0^\infty \lambda^{\alpha-1}e^{-\pi|k|^2/\lambda^2}d\lambda = \frac{2\pi^{\alpha/2}}{\Gamma(\alpha/2)}\int_0^\infty \mu^{(n-\alpha)-1}e^{-\pi|k|^2\mu^2}d\mu$$

$$= \frac{2\pi^{\alpha/2}}{\Gamma(\alpha/2)}\frac{\Gamma((n-\alpha)/2)}{2\pi^{n/2-\alpha/2}}\frac{1}{|k|^{n-\alpha}}$$

$$= \frac{\pi^{\alpha-n/2}\Gamma((n-\alpha)/2)}{\Gamma(\alpha/2)}\frac{1}{|k|^{n-\alpha}},$$

$$= \frac{\pi^{\alpha-n/2}\Gamma((n-\alpha)/2)}{\Gamma(\alpha/2)}|k|^{\alpha-n},$$

which holds for $0 < \alpha < n$.

We want to utilise Scikit Learn's framework for the Gaussian process regression [5], therefore we have to express the covariance as a kernel. We will implement a custom kernel which inherits from the kernel class of Scikit Learn.

```
class PowerSpectrumKernel3D(GenericKernelMixin, Kernel):
    def __init__(self, alpha=2, constant_value=1.0, nugget=1e-5):
        self.alpha = alpha
        self.constant_value = constant_value
        self.nugget = nugget

    def __call__(self, X, Y=None, eval_gradient=False):
        if Y is None:
            Y = X

        # Calculate pairwise Euclidean distances in 3D
        dists = np.sqrt(((X[:,np.newaxis]-Y[np.newaxis,:])**2).sum(axis=2))

        # Avoid zero distances to prevent division by zero or log(0)
        dists[dists == 0] = 1e-10

        # Compute the covariance matrix
        K = self.constant_value * np.abs(dists) ** (self.alpha - 3)

        # Ensure the matrix is positive semi-definite
        K = (K + K.T) / 2
        K += np.eye(K.shape[0]) * self.nugget

        if eval_gradient:
            # Gradient with respect to the hyperparameters, if needed
            K_gradient = np.zeros((X.shape[0], Y.shape[0], 2))

            # gradient with respect to constant_value
            K_gradient[..., 0] = K / self.constant_value

            # gradient with respect to alpha
            K_gradient[..., 1] = K * np.log(np.abs(dists))
            return K, K_gradient

        return K

    def diag(self, X):
        return np.full(X.shape[0], self.constant_value + self.nugget)

    def is_stationary(self):
        return True
```

The nugget enhances numerical stability. The constant_value is the $\frac{\pi^{\alpha-n/2}\Gamma((n-\alpha)/2)}{\Gamma(\alpha/2)}$ term, which is passed as an argument to the kernel for clearness reasons.

References

[1] J. Görtler, R. Kehlbeck, and O. Deussen, "A Visual Exploration of Gaussian Processes," *Distill*, vol. 4, p. e17, Apr. 2019.

[2] J. Wang, "An Intuitive Tutorial to Gaussian Process Regression," *Computing in Science & Engineering*, vol. 25, pp. 4–11, July 2023. arXiv:2009.10862 [cs, stat].

[3] O. Hahn, "Cosmic Structures - Theory, Statistics, Numerics." 2024.

[4] Chappers, "Answer to "Computing Fourier transform of power law"," Mar. 2017.

[5] "1.7. Gaussian Processes."