Gaussian Processes

Advanced Statistical Inference

Simone Rossi

Where are we?

We have seen:

1. Bayesian linear regression
2. Approximate inference
    - Markov chain Monte Carlo
    - Variational inference
    - Laplace approximation
3. Bayesian logistic regression (for classification)
4. **Today**: Gaussian processes

Introduction

- Gaussian processes (GPs) are a flexible and powerful tool for regression (and classification).
- Linear models require us to specify a set of basis functions and their coefficients.
    - Polynomials? Trigonometric functions? etc?
- Can we use Bayesian inference to let the data decide?
- Gaussian processes work implicitly with an infinite-dimensional set of basis functions and learn a probabilistic combination of them.

Gaussian Processes (GPs)

Gaussian processes can be explained in two ways:

- **Weight-space view**:
  - Understand GPs as a Bayesian linear regression model with *implicit* basis functions.
  - Since the basis functions are implicit, we can even work with **infinite-dimensional** feature spaces.
- **Function-space view**:
  - Understand GPs as a distribution over functions.
  - There are no parameters in this view, only functions, so we can *implicitly* work with infinite-dimensional parameter spaces.

Function-space view of Gaussian Processes

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

Why use Gaussian?

Gaussian **distribution** have several nice properties that make them attractive:

- Sums of Gaussians are Gaussian.
- Marginals of multivariate Gaussians are Gaussian.
- Conditionals of Gaussians are Gaussian.

Conditioning a Gaussian distribution

$$\boldsymbol{f} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}\right)$$

Then

$$p(\boldsymbol{f}_2 \mid \boldsymbol{f}_1) = \mathcal{N}(\boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\boldsymbol{f}_1 - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12})$$

which simplifies if $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 = \mathbf{0}$,

$$p(\boldsymbol{f}_2 \mid \boldsymbol{f}_1) = \mathcal{N}(\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{f}_1, \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12})$$

> **Tip**
>
> This is sometimes referred to as the **Bayes' rule for Gaussian distributions**.

Gaussian Processes as a distribution over functions

- We can think of a Gaussian process as an infinite-dimensional generalization of a multivariate Gaussian distribution.
- ... or as an infinite-dimensional distribution over functions.
- All we need to do is to change indexes

Stochastic processes

A stochastic process is a collection of random variables indexed by some variable $x \in \mathcal{X}$.

$$\{f(x) : x \in \mathcal{X}\}$$

Usually $f(x) \in \mathbb{R}$ and $\mathcal{X} = \mathbb{R}^D$. So, $f : \mathbb{R}^D \to \mathbb{R}$.

. . .

For a set of inputs $\boldsymbol{X} = \{x_1, \ldots, x_N\}$, the corresponding random variables $\{f_1, \ldots, f_N\}$ with $f_i = f(x_i)$ have a joint distribution $p(f(x_1), \ldots, f(x_N))$.

. . .

A **Gaussian process** is a stochastic process such that any finite subset of the random variables has a joint Gaussian distribution.

$$(f(x_1), \ldots, f(x_N)) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

We write $f \sim \mathcal{GP}$ to denote that f is a Gaussian process.

Gaussian Processes: mean and covariance functions

To specify a Gaussian distribution, we need to define a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$.

$$\boldsymbol{a} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

To specify a Gaussian process, we need to define a mean **function** $\mu(x)$ and a covariance **function** $k(x, x')$.

$$f \sim \mathcal{GP}(\mu(\cdot), k(\cdot, \cdot))$$

- The mean function $\mu(x) = \mathbb{E}[f(x)]$.
- The covariance function $k(x, x') = \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))]$.

Take two points x_0 and x_1, their function values $f_0 = f(x_0)$ and $f_1 = f(x_1)$ are jointly Gaussian distributed with $p(f_0, f_1)$.

Mean function

We are free to choose the mean and the covariance functions however we like, subject to some constraints.

- The mean function can be used to encode prior knowledge about the mean of the function we are trying to model.
- The mean function $\mu(\boldsymbol{x})$ is often set to zero or constant.

Covariance function

The covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ is also known as the kernel function.

The covariance function encodes our prior beliefs about the **family of functions** we are trying to model.

. . .

Requirements for a valid covariance function:

- k must be a positive semi-definite function.

- Given locations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the covariance matrix $\boldsymbol{K}$ with entries $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ must be positive semi-definite.

We often assume k is a function of the difference between the inputs: $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x} - \boldsymbol{x}')$, which is known as a **stationary kernel**.

Squared Exponential Kernel

RBF kernel or squared exponential kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \alpha \exp\left(-\frac{1}{2\lambda^2}(\boldsymbol{x} - \boldsymbol{x}')^\top(\boldsymbol{x} - \boldsymbol{x}')\right) = \alpha \exp\left(-\frac{1}{2\lambda^2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right)$$

Defines smooth functions, where the function values are highly correlated for nearby inputs.

Hyperparameters:

- α is the amplitude
- λ is the length scale

Predictions with Gaussian Processes

Given a set of training inputs $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and outputs $\boldsymbol{f} = \{f_1, \ldots, f_N\}$, we want to make predictions for a new input $\boldsymbol{x}_\star$.

The joint distribution of the training outputs $\boldsymbol{f}$ and the test output $f_\star$ is Gaussian:

$$\begin{bmatrix} \boldsymbol{f} \\ f_\star \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{k}_\star \\ \boldsymbol{k}_\star^\top & k_{\star\star} \end{bmatrix}\right)$$

where $\boldsymbol{K} = k(\boldsymbol{X}, \boldsymbol{X})$, $\boldsymbol{k}_\star = k(\boldsymbol{X}, \boldsymbol{x}_\star)$, and $k_{\star\star} = k(\boldsymbol{x}_\star, \boldsymbol{x}_\star)$.

. . .

Because of the Gaussian conditioning property, the distribution of $f_\star$ given $\boldsymbol{f}$ is also Gaussian:

$$f_\star \mid \boldsymbol{f} \sim \mathcal{N}(\boldsymbol{k}_\star^\top \boldsymbol{K}^{-1} \boldsymbol{f}, k_{\star\star} - \boldsymbol{k}_\star^\top \boldsymbol{K}^{-1} \boldsymbol{k}_\star)$$

Predictions with Gaussian Processes

$$f_\star \mid \boldsymbol{f} \sim \mathcal{N}(\boldsymbol{k}_\star^\top \boldsymbol{K}^{-1} \boldsymbol{f}, k_{\star\star} - \boldsymbol{k}_\star^\top \boldsymbol{K}^{-1} \boldsymbol{k}_\star)$$

Assumes that the training outputs $\boldsymbol{f}$ are noise-free. *This is not the assumption we made previously.*

In general, we model the noise in the outputs as:

$$p(\boldsymbol{y} \mid \boldsymbol{f}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{f}, \sigma^2 \boldsymbol{I})$$

Predictions with Gaussian Processes with noise

The joint distribution of the training outputs $\boldsymbol{y}$ and the test output $f_\star$ is still Gaussian, but with an additional noise term:

$$\begin{bmatrix} \boldsymbol{y} \\ f_\star \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} + \sigma^2 \boldsymbol{I} & \boldsymbol{k}_\star \\ \boldsymbol{k}_\star^\top & k_{\star\star} \end{bmatrix}\right)$$

The distribution of $f_\star$ given $\boldsymbol{y}$ is:

$$f_\star \mid \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{k}_\star^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}, k_{\star\star} - \boldsymbol{k}_\star^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_\star)$$

We can also marginalize out the noise to get the distribution of $y_\star$:

$$y_\star \mid \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{k}_\star^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}, k_{\star\star} - \boldsymbol{k}_\star^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_\star + \sigma^2)$$

Weight-space view of Gaussian Processes

Review: Bayesian Linear Regression

Recall lecture on Bayesian Linear Regression:

- Modeling the data as a linear combination of the features plus noise:

$$p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{w}, \sigma^2 \boldsymbol{I})$$

- Prior over the weights:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid \mathbf{0}, \boldsymbol{I})$$

Review: Bayesian Linear Regression

- Posterior over the weights:

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

with

$$\boldsymbol{\Sigma} = \left(\frac{1}{\sigma^2}\boldsymbol{X}^\top \boldsymbol{X} + \boldsymbol{I}\right)^{-1} \quad \text{and} \quad \boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{X}^\top \boldsymbol{y}$$

- Predictive distribution for a new input $\boldsymbol{x}_\star$:

$$p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(y_\star \mid \boldsymbol{\mu}^\top \boldsymbol{x}_\star, \boldsymbol{x}_\star^\top \boldsymbol{\Sigma} \boldsymbol{x}_\star + \sigma^2)$$

Introducing basis functions

- Transform the inputs using a set of D basis functions:

$$\boldsymbol{x} \to \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix}\phi_1(\boldsymbol{x}) & \cdots & \phi_D(\boldsymbol{x})\end{bmatrix}^\top$$

- The functions $\phi_d(\boldsymbol{x})$ are known as *basis functions*.
- Define:

$$\boldsymbol{\Phi} = \begin{bmatrix}\phi_1(x_1) & \cdots & \phi_D(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_N) & \cdots & \phi_D(x_N)\end{bmatrix} \in \mathbb{R}^{N \times D}$$

Introducing basis functions

Apply Bayesian linear regression to the transformed inputs:

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where

$$\boldsymbol{\Sigma} = \left(\frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{I}\right)^{-1} \quad \text{and} \quad \boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top\boldsymbol{y}$$

- Predictive distribution for a new input $\boldsymbol{x}_\star$:

$$p(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(y_\star \mid \boldsymbol{\mu}^\top\boldsymbol{\phi}_\star,\ \boldsymbol{\phi}_\star^\top\boldsymbol{\Sigma}\boldsymbol{\phi}_\star + \sigma^2)$$

where $\boldsymbol{\phi}_\star = \boldsymbol{\phi}(\boldsymbol{x}_\star)$.

Bayesian Linear Regression as a kernel method

- We are going to show that predictions in Bayesian linear regression can be expressed in terms of scalar products

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^\top\boldsymbol{\phi}(\boldsymbol{x}')$$

- This is known as the *kernel trick*.
- The kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ is a measure of similarity between two inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$.

For a matrix input $\boldsymbol{X}$, the kernel matrix is defined as

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \in \mathbb{R}^{N \times N}$$

Intuition of the kernel trick

Woodbury matrix identity

Recall the Woodbury matrix identity:

$$(\boldsymbol{A}+\boldsymbol{U}\boldsymbol{C}\boldsymbol{V})^{-1}=\boldsymbol{A}^{-1}-\boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1}+\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}$$

(Do not remember this formula!)

Use the Woodbury matrix identity to rewrite the posterior covariance:

$$\begin{aligned}\boldsymbol{\Sigma}&=\left(\frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}+\boldsymbol{I}\right)^{-1}\\&=\boldsymbol{I}-\boldsymbol{\Phi}^\top\left(\sigma^2\boldsymbol{I}+\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)^{-1}\boldsymbol{\Phi}\end{aligned}$$

Predictions can be expressed in terms of scalar products

$$p(y_\star\mid\boldsymbol{x}_\star,\boldsymbol{y},\boldsymbol{X})=\mathcal{N}(y_\star\mid\boldsymbol{\mu}^\top\boldsymbol{\phi}_\star,\ \boldsymbol{\phi}_\star^\top\boldsymbol{\Sigma}\boldsymbol{\phi}_\star+\sigma^2)$$

We want to express the predictive mean and variance in terms of scalar products of the basis functions.

Variance:

$$\begin{aligned}\boldsymbol{\phi}_\star^\top\boldsymbol{\Sigma}\boldsymbol{\phi}_\star+\sigma^2&=\boldsymbol{\phi}_\star^\top\left(\boldsymbol{I}-\boldsymbol{\Phi}^\top\left(\sigma^2\boldsymbol{I}+\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)^{-1}\boldsymbol{\Phi}\right)\boldsymbol{\phi}_\star+\sigma^2\\&=\boldsymbol{\phi}_\star^\top\boldsymbol{\phi}_\star-\boldsymbol{\phi}_\star^\top\boldsymbol{\Phi}^\top\left(\sigma^2\boldsymbol{I}+\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)^{-1}\boldsymbol{\Phi}\boldsymbol{\phi}_\star+\sigma^2\\&=k_{\star\star}-\boldsymbol{k}_\star^\top(\boldsymbol{K}+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_\star+\sigma^2\end{aligned}$$

where $\boldsymbol{k}_\star=k(\boldsymbol{X},\boldsymbol{x}_\star)$ and $k_{\star\star}=k(\boldsymbol{x}_\star,\boldsymbol{x}_\star)$.

Compare this with the predictive variance in the Gaussian process we have seen before.

Predictions can be expressed in terms of scalar products

Mean:

$$\begin{aligned}
\boldsymbol{\mu}^\top \boldsymbol{\phi}_\star &= \sigma^{-2} \boldsymbol{\phi}_\star^\top \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \boldsymbol{y} \\
&= \sigma^{-2} \boldsymbol{\phi}_\star^\top \left(\boldsymbol{I} - \boldsymbol{\Phi}^\top \left(\sigma^2 \boldsymbol{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \boldsymbol{\Phi} \right) \boldsymbol{\Phi}^\top \boldsymbol{y} \\
&= \sigma^{-2} \boldsymbol{\phi}_\star^\top \boldsymbol{\Phi}^\top \left(\boldsymbol{I} - \left(\sigma^2 \boldsymbol{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right) \boldsymbol{y} \\
&= \sigma^{-2} \boldsymbol{\phi}_\star^\top \boldsymbol{\Phi}^\top \left(\boldsymbol{I} - \left(\boldsymbol{I} + \sigma^{-2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \sigma^{-2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right) \boldsymbol{y}
\end{aligned}$$

Apply again the Woodbury matrix identity with $\boldsymbol{A}, \boldsymbol{U}, \boldsymbol{V} = \boldsymbol{I}$ and $\boldsymbol{C} = \left(\sigma^{-2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1}$:

$$\boldsymbol{I} - \left(\boldsymbol{I} + \sigma^{-2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \sigma^{-2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \left(\boldsymbol{I} + \sigma^{-2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1}$$

continue...

Predictions can be expressed in terms of scalar products

$$\begin{aligned}
\boldsymbol{\mu}^\top \boldsymbol{\phi}_\star &= \sigma^{-2} \boldsymbol{\phi}_\star^\top \boldsymbol{\Phi}^\top \left(\boldsymbol{I} - \left(\boldsymbol{I} + \sigma^{-2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \sigma^{-2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right) \boldsymbol{y} \\
&= \sigma^{-2} \boldsymbol{\phi}_\star^\top \boldsymbol{\Phi}^\top \left(\boldsymbol{I} + \sigma^{-2} \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \boldsymbol{y} \\
&= \boldsymbol{\phi}_\star^\top \boldsymbol{\Phi}^\top \left(\sigma^2 \boldsymbol{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \right)^{-1} \boldsymbol{y} \\
&= \boldsymbol{k}_\star^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}
\end{aligned}$$

Again, compare this with the predictive mean in the Gaussian process we have seen before.

Intuition of the kernel trick

You can either to linear regression by working with the feature mappings $\boldsymbol{\phi}(\cdot)$ or with the kernel function $k(\cdot, \cdot)$.

1. Working with $\boldsymbol{\phi}(\cdot)$ costs $O(D^2)$ memory and $O(D^3)$ time.
2. Working with $k(\cdot, \cdot)$ costs $O(N^2)$ memory and $O(N^3)$ time.

If $D \gg N$, the kernel trick is more efficient.

For example, working with $k(\cdot, \cdot)$ allows us to implicitly use infinite-dimensional feature spaces.

RBF kernel has infinite basis functions

One of such examples is the RBF (or squared exponential) kernel we have seen before:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \alpha \exp\left(-\frac{1}{2\lambda^2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right)$$

We can show that the RBF kernel corresponds to an infinite-dimensional feature space.

RBF kernel has infinite basis functions

Suppose $\lambda = 1$ and $\alpha = 1$, and $\boldsymbol{x} \in \mathbb{R}$.

$$k(x, x') = \exp\left(-\frac{1}{2}(x - x')^2\right) = \exp\left(-\frac{1}{2}x^2\right)\exp\left(-\frac{1}{2}x'^2\right)\exp(xx')$$

Use the Taylor expansion of the exponential function:

$$\exp(xx') = 1 + xx' + \frac{1}{2!}(xx')^2 + \frac{1}{3!}(xx')^3 + \cdots$$

Define the infinite-dimensional feature mappings:

$$x \to \phi(x) = \exp\left(-\frac{1}{2}x^2\right)\begin{bmatrix}1 & x & \frac{1}{\sqrt{2!}}x^2 & \frac{1}{\sqrt{3!}}x^3 & \cdots\end{bmatrix}^\top$$

Model selection in Gaussian Processes

Model selection in Gaussian Processes

- In practice, we need to choose the kernel function and its hyperparameters.
  - For example, in the RBF kernel, we need to choose the amplitude α and the length scale λ.
- The choice of the kernel function and its hyperparameters is known as *model selection*.
- We already saw how to do model selection in Bayesian linear regression using the marginal likelihood.
- In Gaussian processes, we can also use the marginal likelihood to perform model selection.

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f}) \, \mathrm{d}\boldsymbol{f}$$

Marginal likelihood in Gaussian Processes

Let's be explicit about the hyperparameters $\boldsymbol{\theta}$ of the kernel function $k(\cdot, \cdot)$:

$$p(\boldsymbol{y} \mid \boldsymbol{\theta}) = \int p(\boldsymbol{y} \mid \boldsymbol{f}, \boldsymbol{\theta}) p(\boldsymbol{f} \mid \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{f} = \mathcal{N}(\boldsymbol{y} \mid \mathbf{0}, \boldsymbol{K} + \sigma^2 \boldsymbol{I})$$

Maximize the marginal (log-)likelihood with respect to the hyperparameters $\boldsymbol{\theta}$:

$$\arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} -\frac{1}{2} \boldsymbol{y}^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y} - \frac{1}{2} \log \det \Big(\boldsymbol{K} + \sigma^2 \boldsymbol{I} \Big) - \frac{N}{2} \log(2\pi)$$

Question: How to optimize the marginal likelihood?

Optimization of the marginal likelihood

- The marginal likelihood is a non-convex function of the hyperparameters $\boldsymbol{\theta}$.
- No closed-form solution to the optimization problem, but we can use gradient-based optimization methods.
- We can use automatic differentiation to compute the gradients of the marginal likelihood with respect to the hyperparameters.

$$\frac{\partial}{\partial \boldsymbol{\theta}_i} \log p(\boldsymbol{y} \mid \boldsymbol{\theta}) = \frac{1}{2} \boldsymbol{y}^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\theta}_i} (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y} - \frac{1}{2} \mathrm{tr} \left((\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\theta}_i} \right)$$

Remember to guarantee that the hyperparameters are valid (e.g., positive for the length scale).

Why the marginal likelihood works

$$\underbrace{-\frac{1}{2} \boldsymbol{y}^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}}_{\text{Model fit}} \underbrace{-\frac{1}{2} \log \det \Big(\boldsymbol{K} + \sigma^2 \boldsymbol{I} \Big)}_{\text{Model complexity}} - \frac{N}{2} \log(2\pi)$$

The marginal likelihood is a trade-off between the fit to the data and the complexity of the model.

Applications of Gaussian Processes

Time Series Foundation Models

Gaussian processes are a natural choice for modeling time series data, as they can capture complex temporal dependencies and provide uncertainty estimates.

Amazon uses GPs to generate synthetic time series data for pre-training a foundation model for time series forecasting

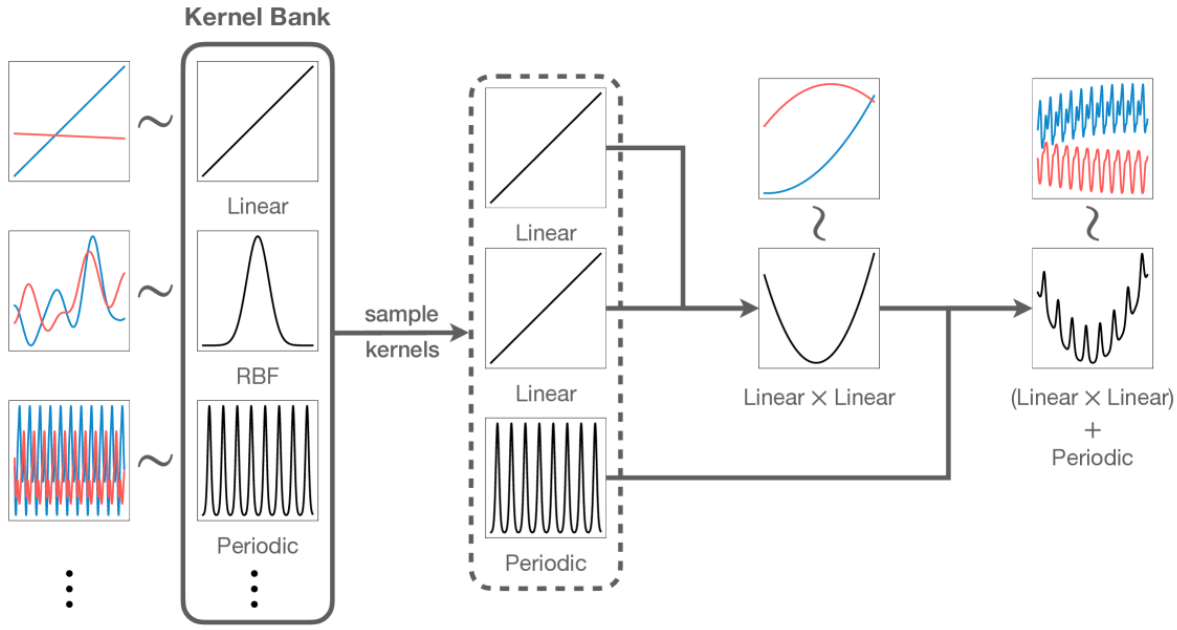
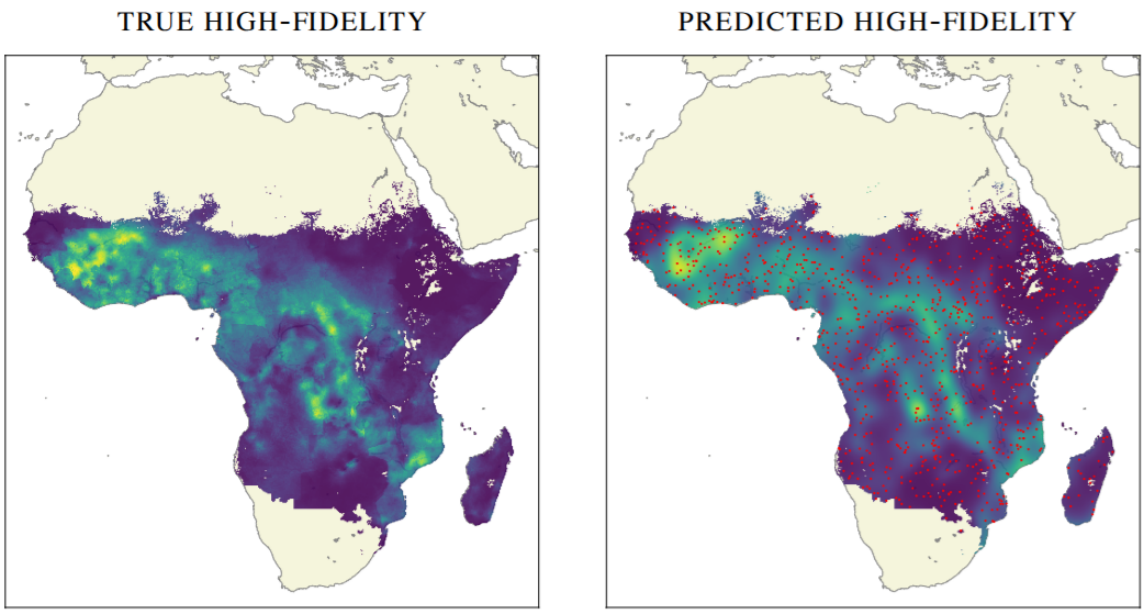


Figure 1: KernelSynth: kernels are sampled from a bank, randomly composed ($\times$ or $+$), and used in a GP prior to generate diverse synthetic time series for pretraining. Ansari et al. (2024).

Spatial Modeling & Science



Geostatistics

- Interpolate geospatial fields: rainfall, mineral deposits, temperature, infection spread, etc.
- Closed-form uncertainty → principled gap-filling

Protein Stability Prediction

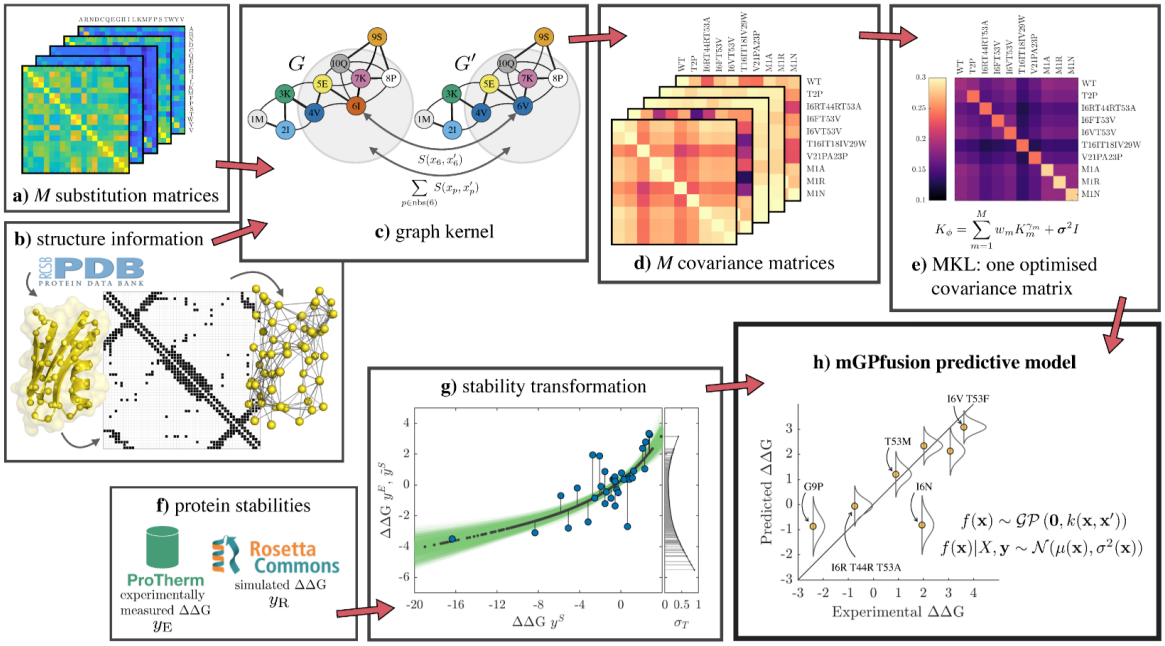


Figure 2: GP surrogate for predicting protein stability changes ($\Delta\Delta G$) upon mutation.

- Measuring protein stability is **slow and costly** — only a few hundred labeled examples
- GPs thrive in the **small-data regime** with principled uncertainty

Challenges in modern Gaussian Processes

Challenges in modern Gaussian Processes

- **Scalability**: Gaussian processes have $\mathcal{O}(N^3)$ complexity in the number of data points N.
  - Solutions: subsets of data, kernel approximations, inducing points, random Fourier features, etc.
- **Non-Gaussian likelihoods**: Exact Gaussian processes assume Gaussian likelihoods, but many real-world datasets have non-Gaussian likelihoods.
  - Solutions: approximate inference

. . .

Modern Gaussian processes introduce **two** very different types of approximations:

- **Model approximation**: for scalability
- **Inference approximation**: for non-Gaussian likelihoods

Scalable Gaussian Processes

Motivation:

- Gaussian processes have $\mathcal{O}(N^3)$ complexity in the number of data points N.
- Above $N \approx 10,000$ data points, exact Gaussian processes become impractical.

Solution: Find approximations to scale Gaussian processes to large datasets.

Two main approaches:

- Sparse Gaussian processes with inducing points (inspired by the function-space view)
- Kernel approximations using random Fourier features (inspired by the weight-space view)

Inducing points

Idea: Augment the model with a set of $M \ll N$ inducing points $\boldsymbol{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M\}$ with corresponding function values $\boldsymbol{u} = \{u_1, \ldots, u_M\}$.

Problem: Somehow, solve the inference problem in this augmented model (which is still Gaussian) on a lower computational budget than the original model.

Is this possible? Is it still possible to get exact inference in this augmented model? At what cost?

. . .

Yes, the predictive distribution for a new input $\boldsymbol{x}_\star$ is:

$$p(y_\star \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{Z}) = \mathcal{N}\left(y_\star \mid \boldsymbol{k}_{\star\boldsymbol{Z}}^\top \boldsymbol{Q}_{\boldsymbol{Z}\boldsymbol{Z}}^{-1} \boldsymbol{K}_{\boldsymbol{Z}\boldsymbol{X}} (\boldsymbol{\Lambda} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y},\ k_{\star\star} - \boldsymbol{k}_{\star\boldsymbol{Z}}^\top \left(\boldsymbol{K}_{\boldsymbol{Z}\boldsymbol{Z}}^{-1} - \boldsymbol{Q}_{\boldsymbol{Z}\boldsymbol{Z}}^{-1}\right) \boldsymbol{k}_{\star\boldsymbol{Z}} + \sigma^2\right)$$

where $\boldsymbol{Q}_{\boldsymbol{Z}\boldsymbol{Z}} = \boldsymbol{K}_{\boldsymbol{Z}\boldsymbol{Z}} + \boldsymbol{K}_{\boldsymbol{Z}\boldsymbol{X}} (\boldsymbol{\Lambda} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}}$.

Yes, the computational cost is $O(NM^2)$, instead of $O(N^3)$.

Gaussian Processes classification

- Gaussian processes can be used for classification problems.
- Model the latent function $f(\boldsymbol{x})$ as a Gaussian process and apply sigmoid function to get the probability of the class label:

$$p(y = 1 \mid \boldsymbol{x}) = \sigma(f(\boldsymbol{x}))$$

- Equivalent to assume the Bernoulli likelihood...

$$p(y \mid f) = \text{Bern}(y \mid \sigma(f))$$

- ... and factorization

$$p(\boldsymbol{y} \mid \boldsymbol{f}) = \prod_{n=1}^{N} p(y_n \mid f_n)$$

- The posterior predictive distribution is not Gaussian anymore and intractable.

Approximate inference for Gaussian Processes classification

As we did for Bayesian logistic regression, we need to use approximate inference methods.

- Laplace approximation:
  1. Find the mode of the posterior distribution using optimization methods.
  2. Compute the Hessian at the mode.
  3. Approximate the posterior distribution as a Gaussian with the mode and the Hessian.
- Variational inferece