PML 8. Gaussian Processes

Probabilistic Machine Learning Reading Group

Simón Rodríguez

February 18, 2026

ICAI - UPC

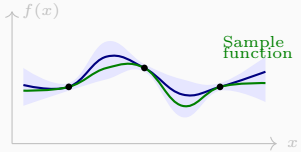
Outline

1. Introduction and Motivation
2. From Weights to Functions
(intuitive recap and beyond)
3. Defining Gaussian Processes
4. The Kernel Zoo
5. GP Regression
6. *Sparse* Gaussian Processes
7. Going Beyond

Introduction and Motivation

Probabilistic Machine Learning

- **Goal**: Learn a mapping $y = f(\mathbf{x}) + \epsilon$ from data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$.
- **Parametric Models** (e.g., Linear Regression, NNs, BNNs):
  - Assume a functional form $f(\mathbf{x}; \mathbf{w})$, learn fixed $\mathbf{w}$.
  - *Limitation*: Model complexity is bounded by fixed $|\mathbf{w}|$.
- **Non-Parametric Models** (e.g., **GPs**):
  - The number of parameters grows with N.
  - We place a prior directly on the function f.



Weight Space vs. Function Space

Weight Space View

- Inference on $\mathbf{w}$.
- Hard to manage for high-dimensions of $\mathbf{w}$.
- Posterior is often complex/multimodal.

Function Space View

- Inference on f.
- Focus on properties of f (smoothness, scale).
- For GPs, posterior is closed-form (for regression).

From Weights to Functions

(intuitive recap and beyond)

Linear Basis Function Models

→ Connection to Session 6 (Bayesian parametric models)

Model target y as a linear combination of fixed basis functions $\phi(\mathbf{x})$:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

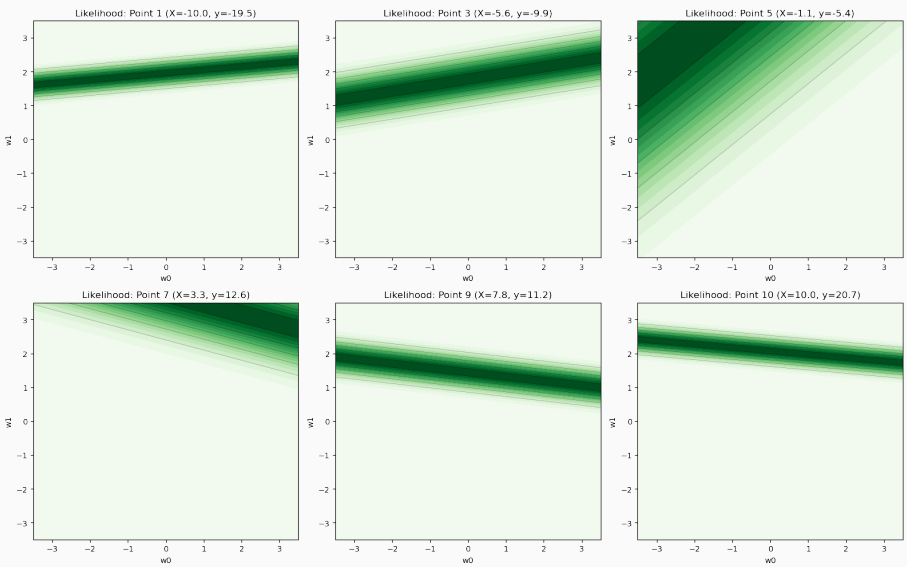
where $\boldsymbol{\phi}(\mathbf{x})$ is the **extended feature vector** (usually $\phi_0(\mathbf{x}) = 1$):

$$\boldsymbol{\phi}(\mathbf{x}) = \Big[1,\ \phi_1(\mathbf{x}),\ \ldots,\ \phi_{M-1}(\mathbf{x})\Big]^T$$

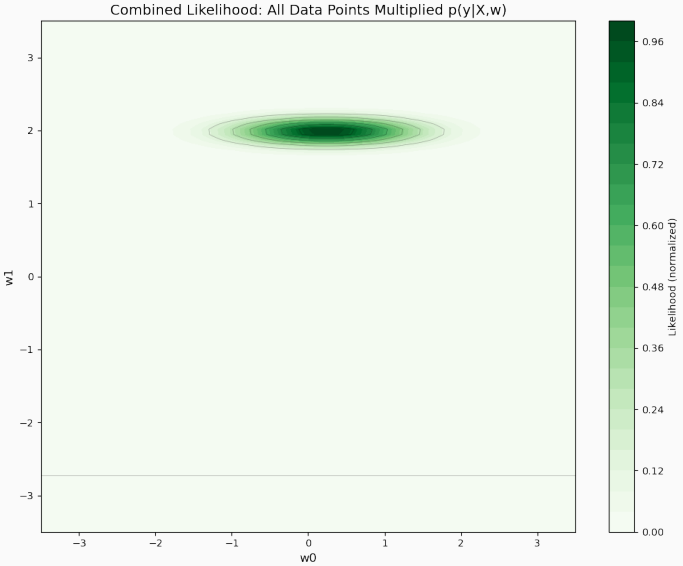
Given i.i.d data $\mathcal{D}$ and noise $\epsilon \sim \mathcal{N}(0, \sigma^2) = \mathcal{N}(0, \beta^{-1})$, likelihood is:

$$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n \mid \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1})$$

Likelihood (data-wise)



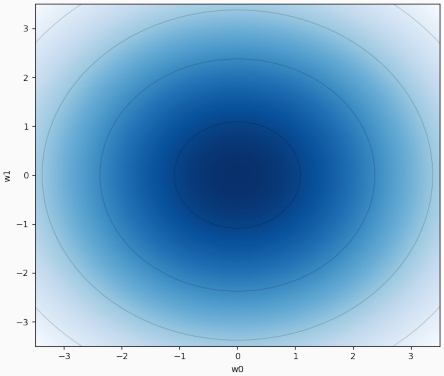
Likelihood (all data)

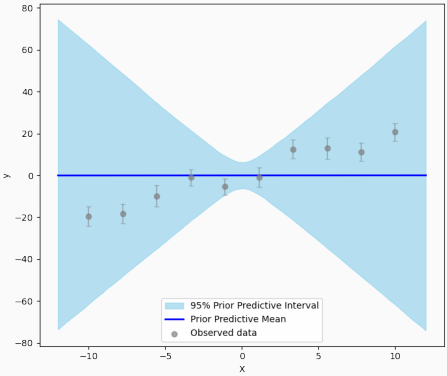


Linear Basis Function Models - Prior

We can select a **conjugate Gaussian prior** for the parameters to perform inference (*e.g.* isotropic, zero-mean):

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} \mid \mathbf{m}_0, \mathbf{S}_0) \quad \rightarrow \quad p(\mathbf{w} \mid \lambda) = \mathcal{N}(\mathbf{w} \mid \mathbf{0}, \lambda^{-1}\mathbf{I})$$





Deriving the Posterior

Conjugate model ⇒ Posterior is Gaussian (product of Gaussians):

$$p(\mathbf{w} \mid \mathbf{y}) = \mathcal{N}(\mathbf{w} \mid \mathbf{m}_N, \mathbf{S}_N)$$

Posterior Distribution Parameters

Covariance:

$$\mathbf{S}_N = (\lambda \mathbf{I} + \beta \mathbf{\Phi}^T \mathbf{\Phi})^{-1}$$

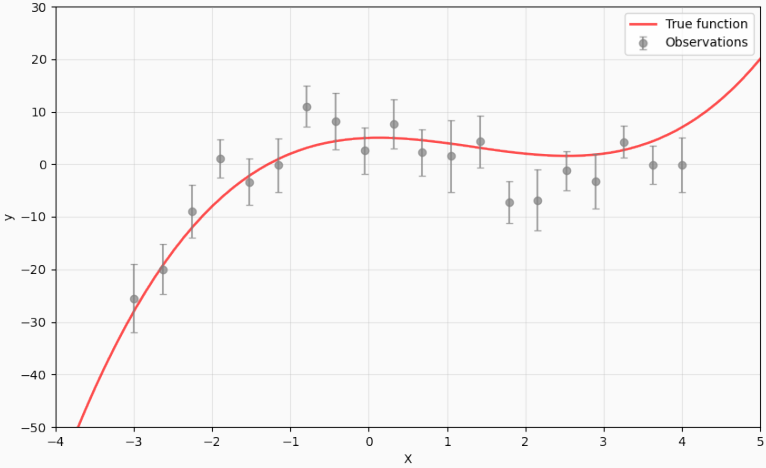
Mean:

$$\mathbf{m}_N = \beta \mathbf{S}_N \mathbf{\Phi}^T \mathbf{y}$$

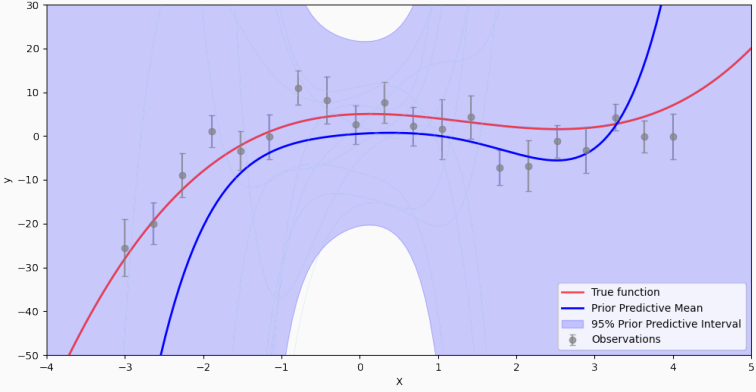
The inversion problem

We must invert a $D \times D$ matrix. Complex features (*e.g.*, high-degree polynomials, NN-like models) become computationally impossible ($D \to \infty$).

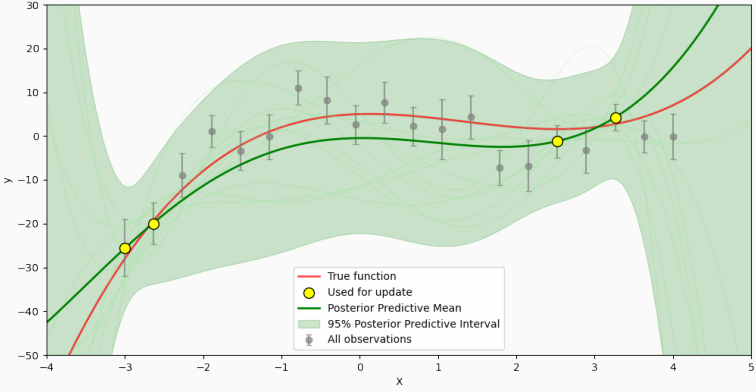
Polynomial Basis Function Models



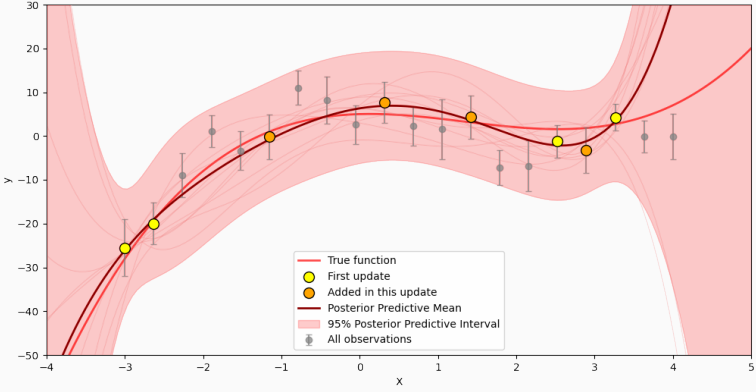
Polynomial Basis Function Models



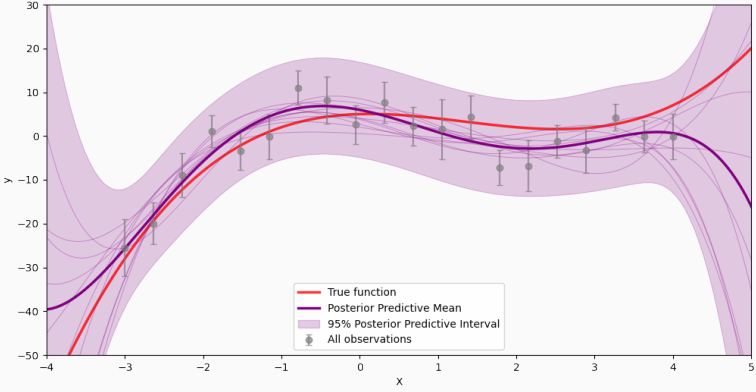
Polynomial Basis Function Models



Polynomial Basis Function Models



Polynomial Basis Function Models



The Representer Theorem

Goal: Minimize Regularized Squared Error for $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$.

Optimal params: $\nabla J(\mathbf{w}) = 0 \Rightarrow$ *structural* insight: $\mathbf{w} = \frac{1}{\lambda}\Phi^T(\underbrace{\mathbf{y} - \Phi\mathbf{w}}_{\text{Residuals}})$

The Representer Theorem

The optimal weights lie in the span of the data ($\boldsymbol{\alpha} \in \mathbb{R}^N$):

$$\mathbf{w} = \Phi^T \boldsymbol{\alpha} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) \quad \text{with} \quad \boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \Phi\mathbf{w}) = (\underbrace{\Phi\Phi^T}_{\mathbf{K}} + \lambda\mathbf{I})^{-1}\mathbf{y}$$

Why is this useful?

- We switch unknowns from $\mathbf{w}$ (size D) to $\boldsymbol{\alpha}$ (size N).
- If $D \to \infty$ (infinite features), we can still solve it using N points!

Prediction via Inner Products (The Kernel Trick)

To predict f_* for a new point $\mathbf{x}_*$, substitute $\mathbf{w} = \Phi^T \boldsymbol{\alpha}$:

$$f_* = \mathbf{w}^T \phi(\mathbf{x}_*) = (\Phi^T \boldsymbol{\alpha})^T \phi(\mathbf{x}_*) = \boldsymbol{\alpha}^T \Phi \phi(\mathbf{x}_*)$$

Using explicit form $\boldsymbol{\alpha} = (\Phi\Phi^T + \lambda \mathbf{I})^{-1}\mathbf{y}$ gives out:

$$f_* = \underbrace{\phi(\mathbf{x}_*)^T \Phi^T}_{\mathbf{k}_*^T} (\underbrace{\Phi\Phi^T}_{\mathbf{K}} + \lambda \mathbf{I})^{-1}\mathbf{y}$$

Key Insight: We only need dot products

The prediction depends **only** on inner products $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$.

- We never compute $\mathbf{w}$ (potentially a very high-dim vector).
- We never need to know the explicit map $\phi(\mathbf{x})$.

The Kernel Trick

Kernel trick

Since we only need inner products, we can replace the explicit dot product with a **Kernel Function**:

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

Arbitrary flexibility controled only via $k(\cdot, \cdot)$ (*i.e.* Gram matrix $\mathbf{K}$)

- We do not need to know $\phi(\mathbf{x})$ or compute it.
- We just need a function k that behaves like a dot product ($\mathbf{K} \succeq 0$).
- **Infinite Dimensions:** We can use a kernel (like RBF) that corresponds to $D = \infty$ without ever computing an infinite vector.

Computational Trade-off: Primal vs. Dual

	Primal (Weight Space)	**Dual (Function Space)**
Unknowns	Params. $\mathbf{w}$ $(D \times 1)$	Func. evals. $\mathbf{f}$ $(N \times 1)$
Matrix Size	$D \times D$	$N \times N$
Complexity	$O(D^3)$	$O(N^3)$
Best for	$N \gg D$	$D \gg N$

Gaussian Processes operate in the **Dual** regime, allowing us to model extremely complex functions ($D \to \infty$) as long as N is manageable.

Note: Large N requires some work, but can be done at times

Defining Gaussian Processes

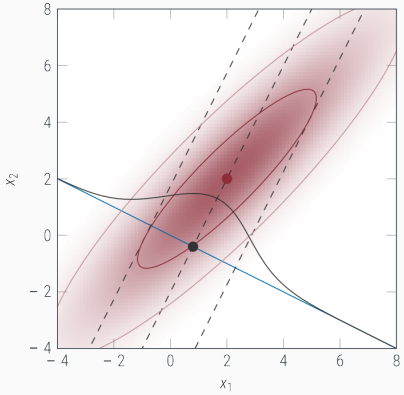
Why "Gaussian"? - *Consistency*

Linear Projection of MVNs

To project a Gaussian density, project its parameters

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$p(\mathbf{Az}) = \mathcal{N}(\mathbf{Az}; \mathbf{A}\boldsymbol{\mu}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T)$$

Marginalization and **conditioning** can be seen as particular examples of projection



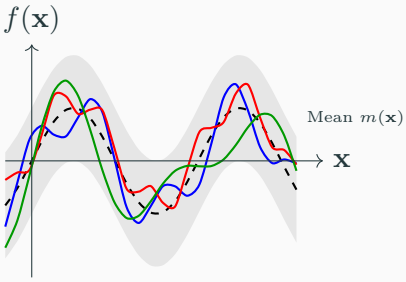
Definition

Gaussian Process (GP)

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

A GP, denoted as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, is fully specified by:

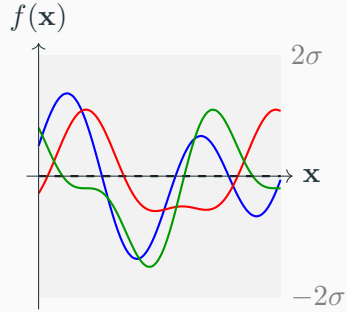
- **Mean function**:
  $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
- **Covariance function**:
  $k(\mathbf{x}, \mathbf{x}') =$
  $\mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$



Sampling from the Prior

To visualize a GP, we can sample functions from the prior.

1. Choose input points $\mathbf{X}_*$.
2. Compute covariance matrix $\mathbf{K}_{**}$ where $(\mathbf{K}_{**})_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.
3. Sample vector $\mathbf{f}_* \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{**})$.
4. Plot $\mathbf{f}_*$ vs $\mathbf{X}_*$.

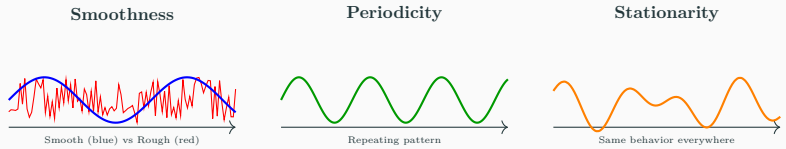


Samples from $\mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$

Covariance Function (Kernel)

The kernel $k(\mathbf{x}, \mathbf{x}')$ encodes our assumptions about the function's structure.

- **Smoothness**: How wiggly is the function? (controlled by length-scale).
- **Periodicity**: Does the function pattern repeat?
- **Stationarity**: Does the correlation depend only on the distance $\mathbf{x} - \mathbf{x}'$ (invariant to translation)?



Quick note: BNNs and GPs (Neal's Theorem)

→ Connection to Session 6 (BNNs)

Connection between BNNs and GPs

(Neal, 1996) A Bayesian Neural Network with a single hidden layer, infinite width, and i.i.d. priors on weights converges to a **Gaussian Process**.

- Let $f(\mathbf{x}) = \sum_{j=1}^{H} v_j h(\mathbf{w}_j^T \mathbf{x} + b_j)$.
- If $v_j \sim \mathcal{N}(0, \sigma_v^2/H)$, then by CLT, as $H \to \infty$, $f(\mathbf{x})$ becomes Gaussian.

The Kernel Zoo

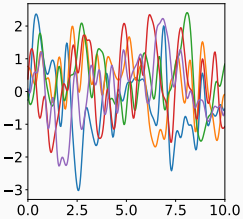
Squared Exponential (SE) Kernel

Also known as the Radial Basis Function (RBF) or Gaussian Kernel.

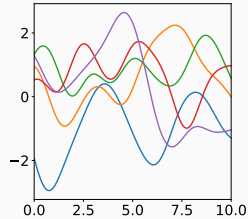
$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\ell^2}\right)$$

- ℓ: **Length-scale**. Controls "wiggliness".
- σ_f^2: **Signal variance**. Controls vertical amplitude.
- Resulting functions are infinitely differentiable (very smooth).

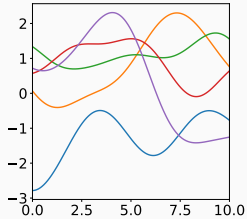
$l = 0.2$



$l = 1$



$l = 2$



Matérn Kernels

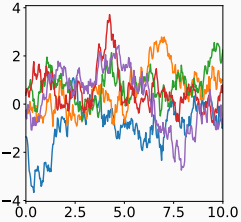
Generalization of SE that controls differentiability (roughness).

$$k_{\text{Matern}}(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{\ell} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} r}{\ell} \right)$$

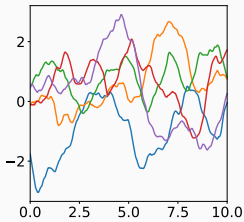
where $r = ||\mathbf{x} - \mathbf{x}'||$.

- $\nu = 1/2$: Ornstein-Uhlenbeck process (continuous, nowhere differentiable).
- $\nu = 3/2$: Once differentiable (very common in physical modeling).
- $\nu \to \infty$: Converges to SE kernel.

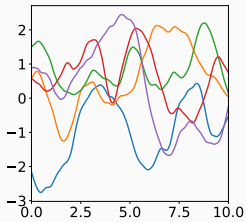
$\nu = 0.5$



$\nu = 1.5$



$\nu = 2.5$



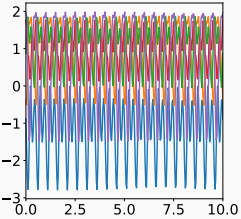
Periodic Kernel

Used for data with repeating patterns (e.g., seasonality).

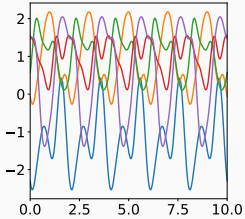
$$k_{Per}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{2\sin^2(\pi||\mathbf{x} - \mathbf{x}'||/p)}{\ell^2}\right)$$

- p: Period of the repetition.
- ℓ: Length-scale (how much it varies within a period).

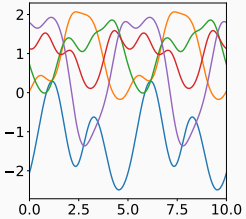
$p = 0.5$



$p = 2$



$p = 5$



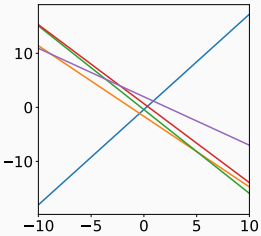
Linear Kernel

Corresponds to Bayesian Linear Regression.

$$k_{Lin}(\mathbf{x}, \mathbf{x}') = \sigma_b^2 + \sigma_v^2(\mathbf{x} - c)^T(\mathbf{x}' - c)$$

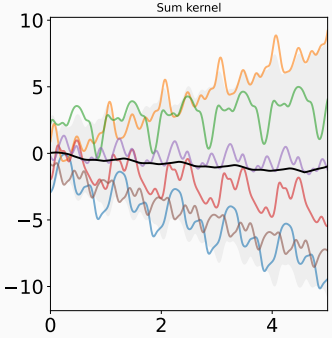
- Not stationary (depends on absolute position, not just distance).
- Samples are straight lines (or planes).

$\sigma = 1$

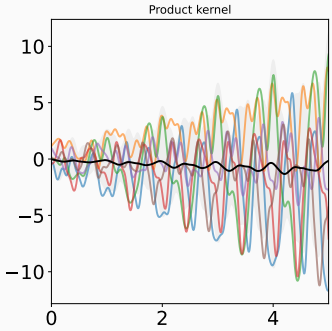


Combining Kernels

The sum, product (and more operations) of valid kernels are valid kernels.



$k = k_1 + k_2$



$k = k_1 \times k_2$

GP Regression

Regression with GPs

Assume $y = f(\mathbf{x}) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. We have set

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Joint distr. of train targets $\mathbf{y}$ and the test function value f_* is jointly Gaussian:

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & \mathbf{k}_* \\ \mathbf{k}_*^T & k_{**} \end{bmatrix}\right)$$

where $\mathbf{K} = K(\mathbf{X}, \mathbf{X})$ (shape $N \times N$), $\mathbf{k}_* = K(\mathbf{X}, \mathbf{x}_*)$ (shape $N \times 1$) and $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$ (scalar)

Predictive Posterior Equations

Applying the conditioning theorem to our GP joint:

GP Predictive distribution

Mean:

$$\bar{f}_* = \mathbf{k}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}$$

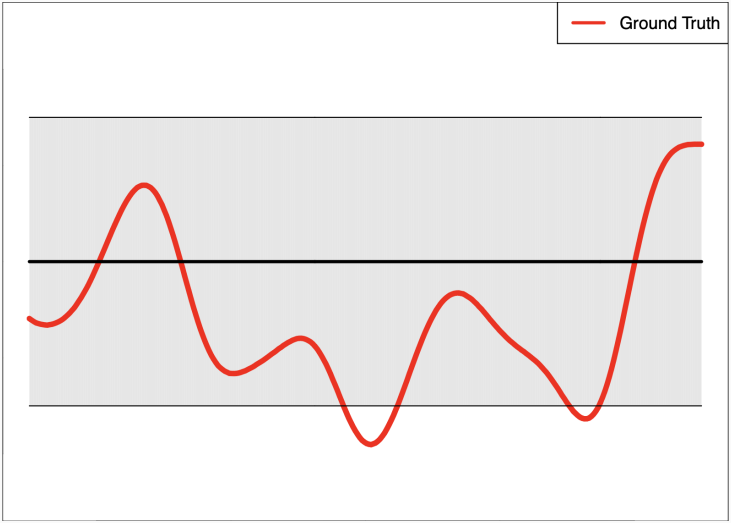
Variance:

$$\mathbb{V}[f_*] = k_{**} - \mathbf{k}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_*$$

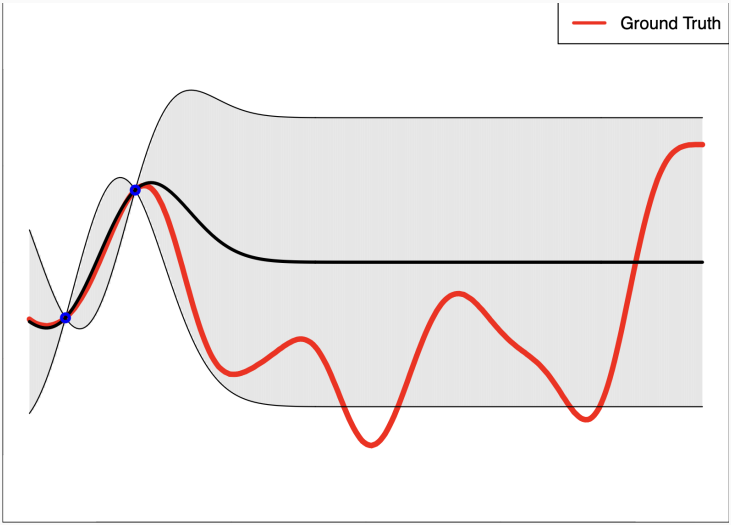
And also, since $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}$, then:

$$\bar{f}(\mathbf{x}_*) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$$

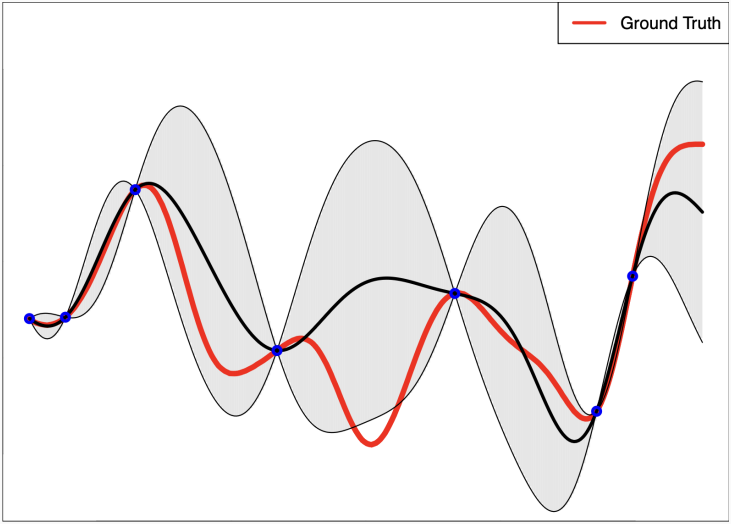
GP Regression - Fit



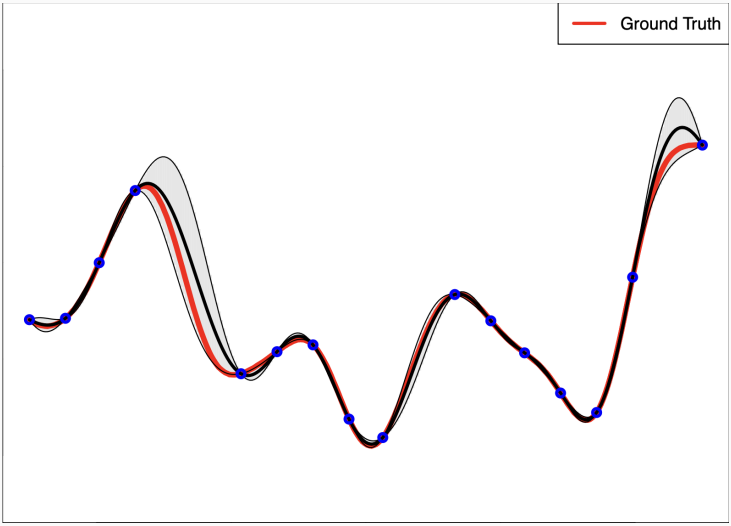
GP Regression - Fit



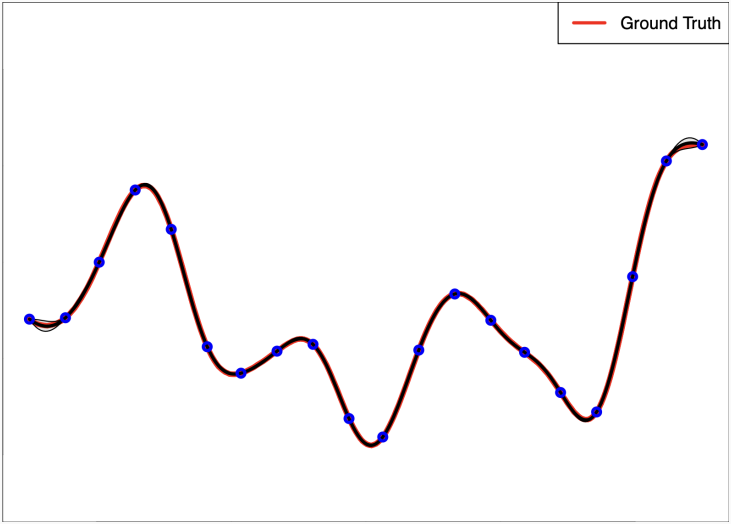
GP Regression - Fit



GP Regression - Fit



GP Regression - Fit



Computational Complexity

- Bottleneck: Inverting $(\mathbf{K} + \sigma_n^2 \mathbf{I})$ (or Cholesky).
- $\mathbf{K}$ matrix is $N \times N$.
- **Training cost**: $\mathcal{O}(N^3)$.
- **Prediction cost**: $\mathcal{O}(N^2)$ per test point (for variance).
- **Memory**: $\mathcal{O}(N^2)$.
- This usually limits standard GPs to $N \sim \mathcal{O}(10^4)$.

$\rightarrow$ *Too expensive!*

Computational Complexity

- Bottleneck: Inverting $(\mathbf{K} + \sigma_n^2 \mathbf{I})$ (or Cholesky).
- $\mathbf{K}$ matrix is $N \times N$.
- **Training cost**: $\mathcal{O}(N^3)$.
- **Prediction cost**: $\mathcal{O}(N^2)$ per test point (for variance).
- **Memory**: $\mathcal{O}(N^2)$.
- This usually limits standard GPs to $N \sim \mathcal{O}(10^4)$.

$\rightarrow$ *Too expensive!*

Solution: Approximate the full GP using a set of $M \ll N$ variables.

Sparse Gaussian Processes

Inducing Points

We introduce a set of M **Inducing Inputs** $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$ and corresponding **Inducing Variables** $\mathbf{u} = f(\mathbf{Z})$.

- $\mathbf{Z}$ live in the same space as $\mathbf{X}$ (but don't have to be subset of data).
- We assume $\mathbf{u}$ summarizes the global structure of the function f.

The joint prior is:

$$p(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})p(\mathbf{u})$$

where $p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{MM})$.

Key idea

Use $\mathbf{Z}$ as **sufficient statistics** for the data so we can use $f_{\mathbf{Z}}$ to predict f_* instead of $f_{\mathbf{X}}$, *i.e.* $f_* \perp f_{\mathbf{X}} \mid f_{\mathbf{Z}}$

FITC Approximation (Snelson & Ghahramani, 2006)

Fully Independent Training Conditional (FITC)

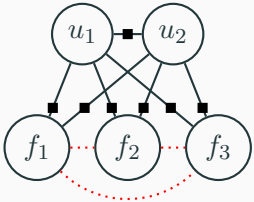
Assumption: Training points are independent given inducing vars. $\mathbf{u}$.

$$p(\mathbf{f}|\mathbf{u}) \approx \prod_{i=1}^{N} p(f_i|\mathbf{u})$$

This approximates the covariance $\mathbf{K}_{NN}$ as **Low Rank + Diagonal**:

$$\mathbf{K}_{FITC} = \mathbf{Q}_{NN} + \text{diag}(\mathbf{K}_{NN} - \mathbf{Q}_{NN})$$

- $\mathbf{Q}_{NN} = \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}$
- Preserves exact diagonal variance.
- **Issue**: Optimizing $\mathbf{Z}$ can overfit + forced independencies (bias)



Variational Free Energy (VFE)

VFE Approximation

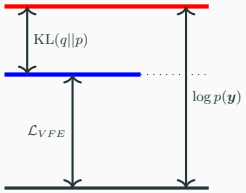
Use variational $q(\mathbf{f}, \mathbf{u})$ to minimize KL divergence to the true posterior $p(\mathbf{f}, \mathbf{u}|\mathbf{y})$:

$$q(\boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u})$$

We maximize the **Evidence Lower Bound (ELBO)**:

$$\mathcal{L}_{VFE} = \sum_{i=1}^{N} \mathbb{E}_q[\log p(y_i|f_i)] - \mathrm{KL}(q(\mathbf{u})||p(\mathbf{u}))$$

$$\log p(\mathbf{y}) \geq \mathcal{L}_{VFE}$$



Variational Sparse GPs

To see where it comes from, consider the full (augmented) model:

Augmented Model

$$p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}) = \underbrace{p(\boldsymbol{y}|\boldsymbol{f})}_{\text{Likelihood}} \times \underbrace{p(\boldsymbol{f}|\boldsymbol{u})}_{\text{Conditional}} \times \underbrace{p(\boldsymbol{u})}_{\text{Prior at } Z}$$

The marginal likelihood is: $p(\boldsymbol{y}|\boldsymbol{X}) = \int_{\boldsymbol{f},\boldsymbol{u}} p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|\boldsymbol{X}, \boldsymbol{Z})\, d\boldsymbol{f}\, d\boldsymbol{u}$

where $p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|\boldsymbol{X}, \boldsymbol{Z}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{X}, \boldsymbol{Z})p(\boldsymbol{u}|\boldsymbol{Z})$

Augmented Marginal Likelihood

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int_{\boldsymbol{f},\boldsymbol{u}} p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{X}, \boldsymbol{Z})p(\boldsymbol{u}|\boldsymbol{Z})\, d\boldsymbol{f}\, d\boldsymbol{u}$$

Variational Sparse GPs

i. Introduce **variational distribution** $q(\boldsymbol{f}, \boldsymbol{u}) \approx p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y})$.

ii. Optimize ELBO:

$$\log p(\boldsymbol{y}|\boldsymbol{X}) \geq ELBO(q) = \int_{\boldsymbol{f},\boldsymbol{u}} q(\boldsymbol{f}, \boldsymbol{u}) \log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f}, \boldsymbol{u})} \, d\boldsymbol{f} d\boldsymbol{u}$$

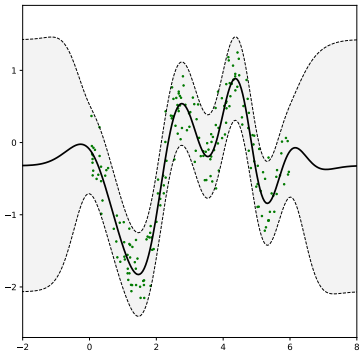
iii. Use factorization to simplify $q(\boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u})$:

$$\begin{aligned} \text{ELBO} &= \mathbb{E}_q \left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f}) \cancel{p(\boldsymbol{f}|\boldsymbol{u})} p(\boldsymbol{u})}{\cancel{p(\boldsymbol{f}|\boldsymbol{u})} q(\boldsymbol{u})} \right] \\ &= \mathbb{E}_{q(\boldsymbol{f})}[\log p(\boldsymbol{y}|\boldsymbol{f})] - \text{KL}(q(\boldsymbol{u})||p(\boldsymbol{u})) \end{aligned}$$

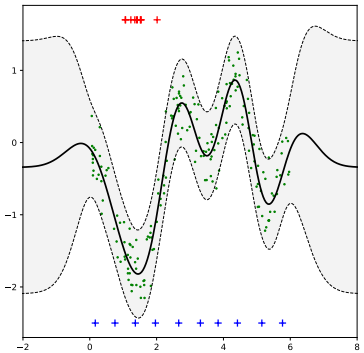
→ Final expression reduces complexity to $\mathcal{O}(\mathbf{NM^2})$, enabling to optimize the model and perform predictions on large datasets.

Sparse GP fit

Model learns to locate inducing inputs to its benefit during training!



Full GP fit ($N = 200$)



Sparse fit ($M = 10$)

Reproduction of Snelson and Gharamani (2009) with adverse $\mathbf{z}$ initialization

Computational Summary

By using Sparse GPs (VFE, FITC or others):

1. We invert $\mathbf{K}_{MM}$ $(M \times M)$ instead of $\mathbf{K}_{NN}$.
2. **Training Complexity**: $\mathcal{O}(NM^2)$.
3. **Prediction Complexity**: $\mathcal{O}(M^2)$ per test point.
4. **Storage**: $\mathcal{O}(NM)$.

If $M \approx 100 - 500$ and $N = 1,000,000$:

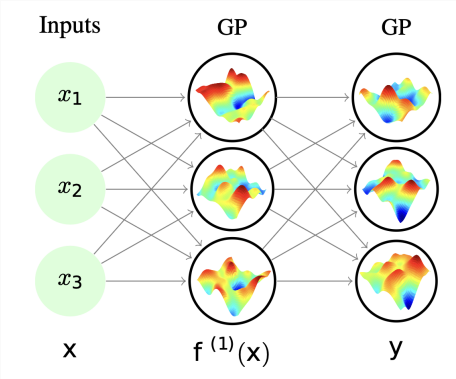
$$NM^2 \ll N^3$$

This enables GPs ("Big Data GPs") to scale to millions of points.

Going Beyond

Beyond Gaussianity – Deep Gaussian Processes

Composition of GP mappings into a deep-net-like structure



(Garrido-Merchan et al., 2021)

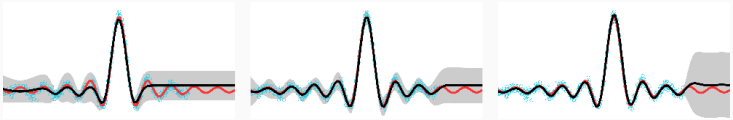
Decoupled Sparse GPs

The Decoupled Approach (Cheng & Boots, 2017)

Core Idea: Separate the representations in the Reproducing Kernel Hilbert Space (RKHS). This allows for different M to learn the mean and covariances, breaking the $\mathcal{O}(M^3)$ limit.

$$\tilde{\mu} = \Psi_\alpha \mathbf{a}, \qquad \tilde{\Sigma} = \left(I + \Psi_\beta^T B \Psi_\beta\right)^{-1}$$

where usually $M_\beta < M_\alpha$.



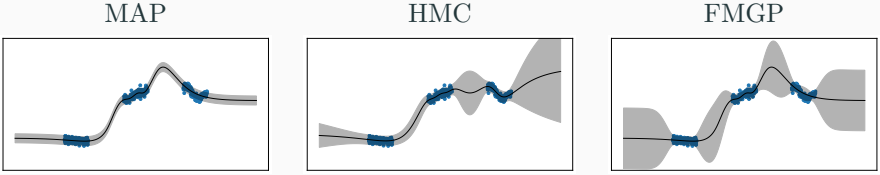
(a) M = 10 (b) M_α = 100, M_β = 10 (c) M = 100

Other Applications

Implicit Processes (Ma et al., 2019): Generalization of GPs, BNNs and more under the same framework (*not necessarily Gaussian*)

$$\mathbf{z} \sim p(\mathbf{z}), \quad f(\mathbf{x}_i) = g_\theta(\mathbf{x}_i, \mathbf{z}), \quad \forall \mathbf{x}_i \in \mathcal{X}$$

Post-hoc NN Bayesian approximates (Ortega et al., 2025): **Fixed-Mean GPs** used to obtain uncertainty estimates in pre-trained NN systems (even in large structures like ResNet)



Overview on GPs - Pros and Cons

Advantages $\oplus$

- **Probabilistic**: Principled uncertainty quantification
- **Flexible**: Non-parametric, complexity directly from data
- **Analytic**: Exact, closed-form inference (regression)
- **Data Efficient**: Good performance in low-data regimes.

Limitations $\ominus$

- **Scalability**: $\mathcal{O}(N^3)$ computational bottleneck (**fix**: Sparse GPs)
- **Inference**: Intractable for non-Gaussian likelihoods (e.g., Classification)
- **Kernel Choice**: Performance heavily relies on picking the right kernel

$\rightarrow$ Still a very active research area!

References

1. Murphy, K. P. *Machine Learning: A Probabilistic Perspective*, Chapters 15 & 17.
2. Rasmussen, C. E. & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*, MIT Press.
3. Neal, R. M. (1996). *Bayesian Learning for Neural Networks*.
4. Snelson, E., & Ghahramani, Z. (2006). *Sparse Gaussian processes using pseudo-inputs* (FITC). NeurIPS.
5. Titsias, M. K. (2009). *Variational learning of inducing variables in sparse Gaussian processes* (VFE). AISTATS.
6. Ma, C., Li, Y., & Hernández-Lobato, J. M. (2019). *Variational Implicit Processes*. ICML.
7. E.C. Garrido-Merchan. (2021) *Advanced Methods for Bayesian Optimization in Complex Scenarios* (PhD Thesis), UAM.
8. Ortega, L. A. et al. (2024). *Fixed-Mean Gaussian Processes for Post-hoc Bayesian Deep Learning*, arXiv:2412.04177.

Questions?

Thank you!