Gaussian processes II

Alejandro Veloz

ORGANIZED BY:

Universidad de Valparaiso CHILE

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

UNIVERSIDAD MAYOR

OCEANS

Funded by the European Union

COLLABORATORS:

Instituto de Física y Astronomía

IDEUV Instituto de Estadística

UNIVERSIDAD MAYOR

CENTRO UM CMF Multidisciplinario de Física

CCTVal CENTRO CIENTÍFICO TECNOLÓGICO DE VALPARAÍSO

WOLFRAM

Email: alejandro.veloz@uv.cl

Slides and Labs: https://aavelozb.github.io/gp/gp-2025.html

Outline

Automatic Relevance Determination (ARD)

Multi-task Gaussian processes

Gaussian processes and neural networks

Deconvolution using Gaussian processes

Automatic Relevance Determination (ARD)

- **Automatic Relevance Determination (ARD)** is a technique for determining the importance of input variables.
- Originally formulated in **neural networks** (MacKay 1994, Neal 1996).
- Extensively used in **Gaussian Process** regression.
- Optimize separate length-scale parameters for each input dimension.

ARD Kernel Function

Standard squared-exponential kernel with ARD:

$$k(x_n, x_m) = \sigma_0 \exp\left(-\frac{1}{2}\sum_{i=1}^{D}\lambda_i(x_{ni} - x_{mi})^2\right) + \sigma_3 x_n^T x_m$$

- σ_0: overall variance
- λ_i: **precision for dimension** i
- σ_3: linear term

- If λ_i is **large**, dimension i is **important**
- If λ_i is **small**, dimension i is **irrelevant**

ARD Kernel Function

Large λ_i (Relevant)

Steep kernel in dimension i
Function changes rapidly with x_i
Strong dependence on input i

Small λ_i (Irrelevant)

Flat kernel in dimension i
Function changes slowly with x_i
Weak/no dependence on input i

Example: $\lambda_1 = 1$ vs $\lambda_2 = 0.01$

- Function is sensitive to x_1.
- Function is nearly insensitive to x_2.

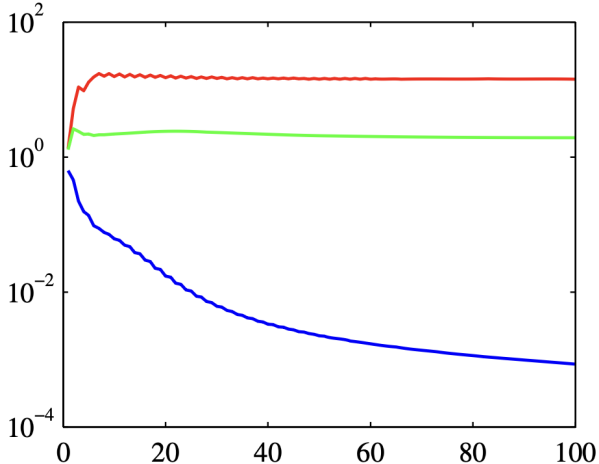
ARD

Synthetic Dataset:

- x_1: generated from Gaussian.
- **Target**: $t = \sin(2x_1) +$ Gaussian noise.
- x_2: copy of x_1 + noise (partially correlated).
- x_3: independent Gaussian (irrelevant).

Example Results

λ_1 (red), λ_2 (green), λ_3 (blue)



General Form with Linear Terms

Extended ARD kernel (for regression):

$$k(x_n, x_m) = \sigma_0 \exp\left(-\frac{1}{2}\sum_{i=1}^{D}\lambda_i(x_{ni} - x_{mi})^2\right) + \sigma_3 \sum_{i=1}^{D} x_{ni}x_{mi}$$

Two competing mechanisms:

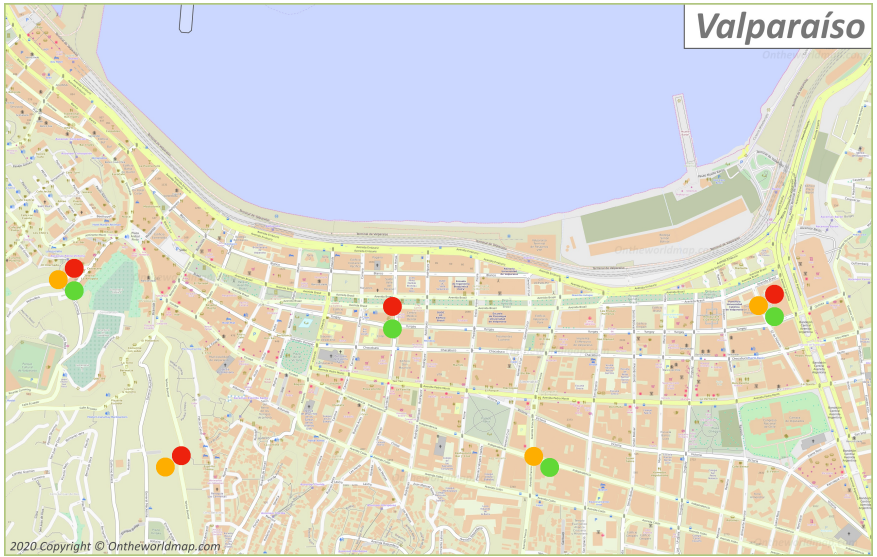
1. **Exponential term** with separate λ_i parameters
2. **Linear term** with fixed coupling

Connection to other methods

Method	Approach
Feature Selection	Discrete: keep or remove feature
Regularization (L1)	Shrink weights toward zero
ARD in Bayesian NNs	Prune weights/hyperparameters
ARD in GPs	Automatically scale input dimensions

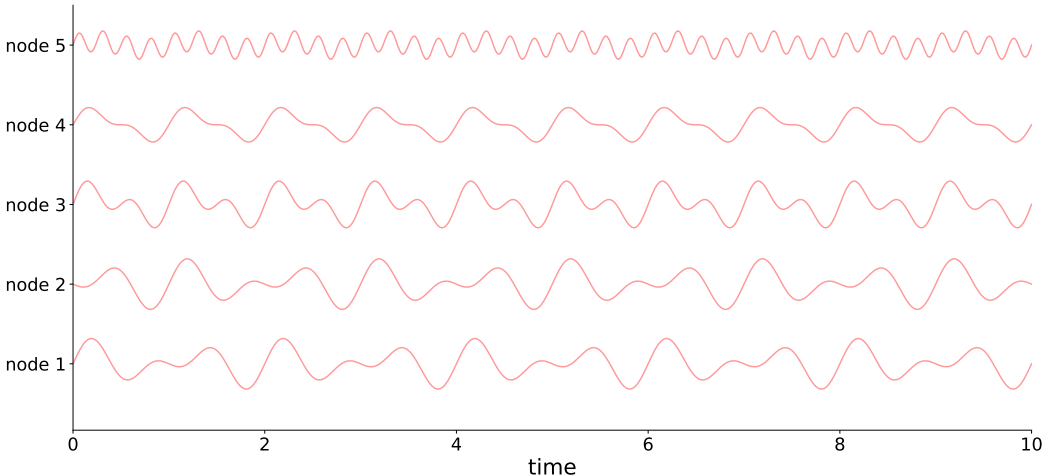
Multi-task or multi-output GPs

Multi-task GPs

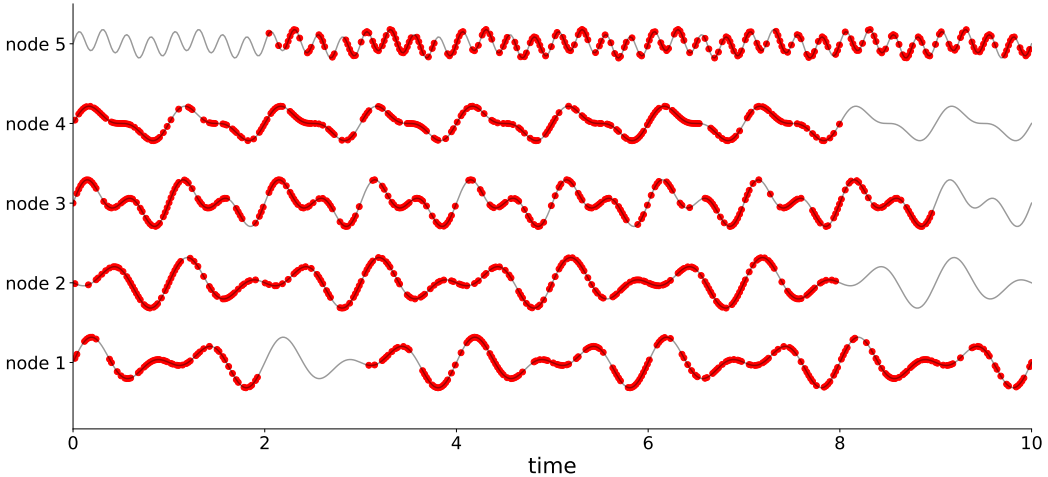


- Temperature
- Humidity
- CO2 level

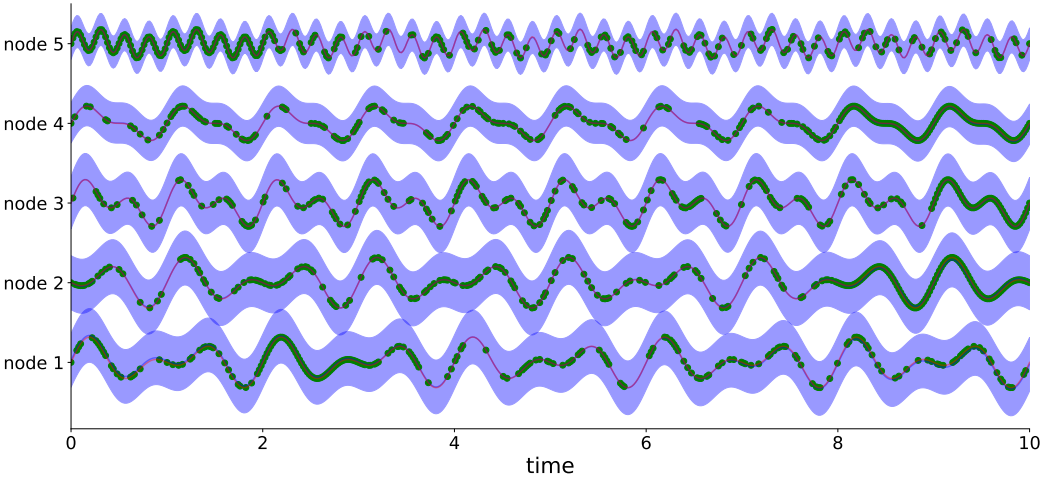
Sensors network



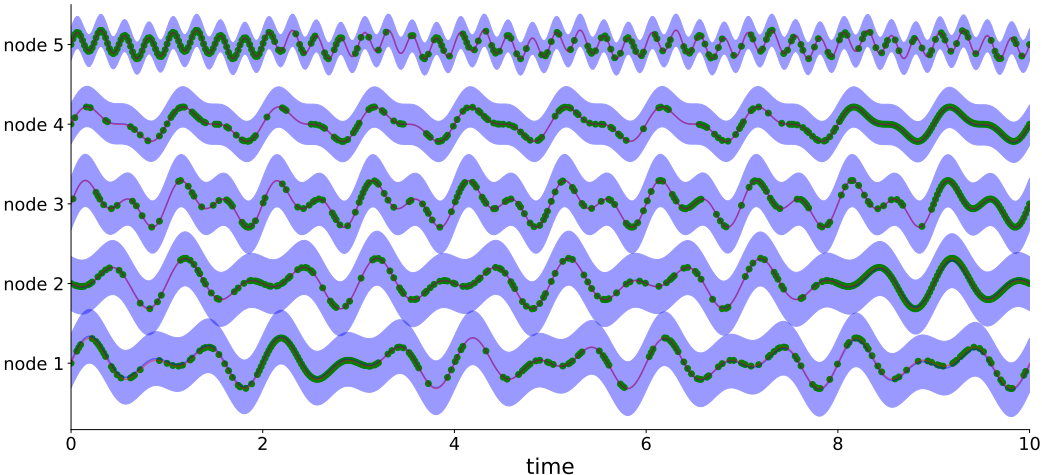
Sensors network with missing segments



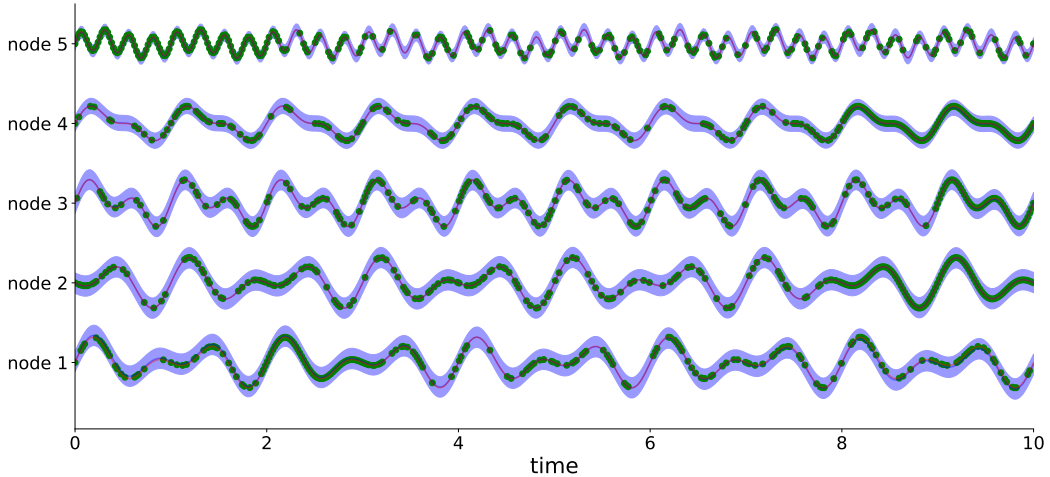
Sensors network - prediction



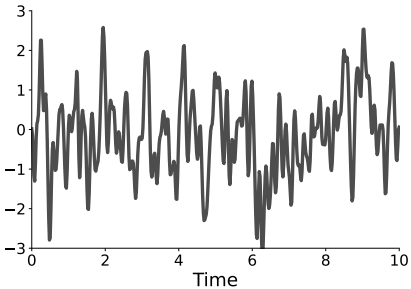
Sensors network - prediction before training



Sensors network - prediction after training

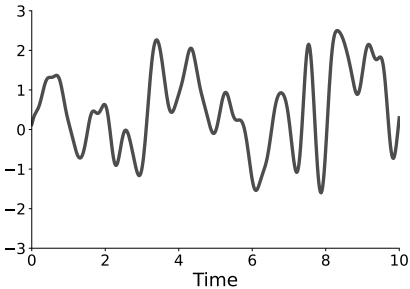


Single-output Gaussian process



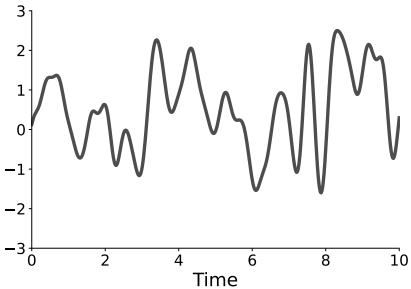
$$f(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

Single-output Gaussian process



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

Single-output Gaussian process

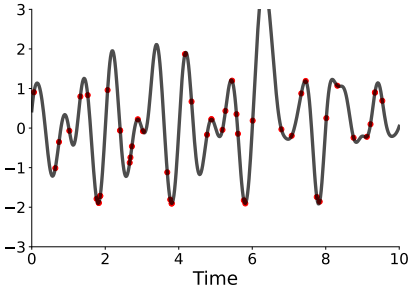


$$f(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

Training dataset:

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) \mid i = 1, \ldots, N\}$$

Single-output Gaussian process



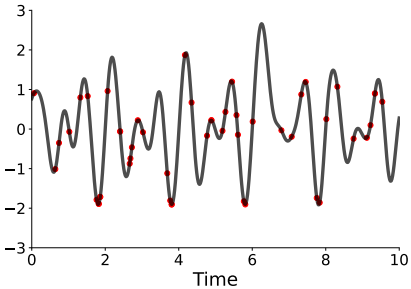
$$f(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

Training dataset:

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) \mid i = 1, \ldots, N\}$$

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}\right)$$

Single-output Gaussian process



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

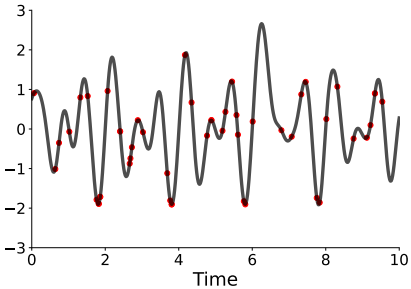
Training dataset:

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) \mid i = 1, \ldots, N\}$$

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}\right)$$

In matrix form: $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$

Single-output Gaussian process



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

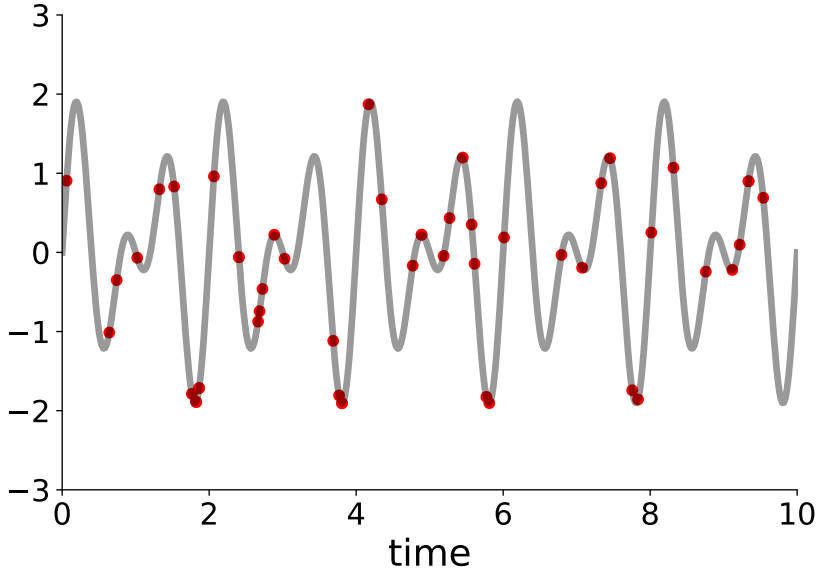
Training dataset:

$$\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)) \mid i = 1, \ldots, N\}$$

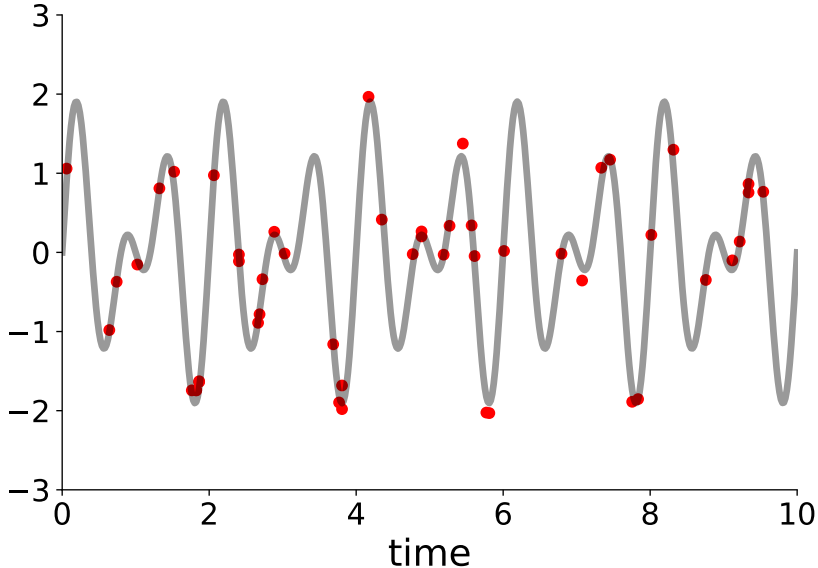
$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}\right)$$

In matrix form: $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ **For prediction:** $p(f(\mathbf{x}_*) \mid \mathbf{f})$

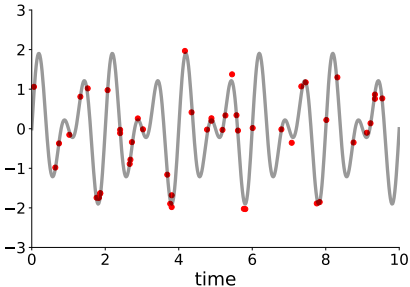
Noisy observations



Noisy observations

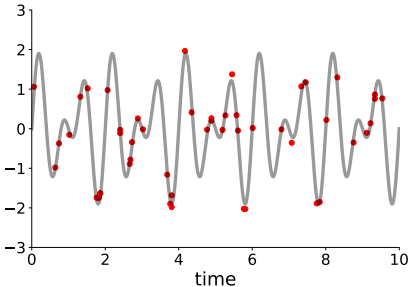


GP with noisy observations



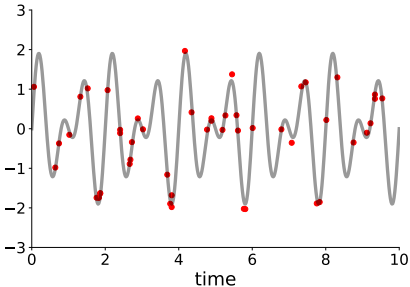
$$f(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

GP with noisy observations



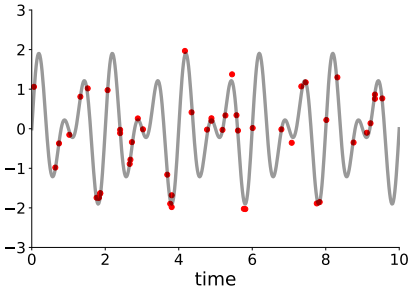
$$f(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$
$$y\left(\mathbf{x}_i\right) = f\left(\mathbf{x}_i\right) + \epsilon_i$$

GP with noisy observations



$$f(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$
$$y\left(\mathbf{x}_i\right) = f\left(\mathbf{x}_i\right) + \epsilon_i$$
$$\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$$

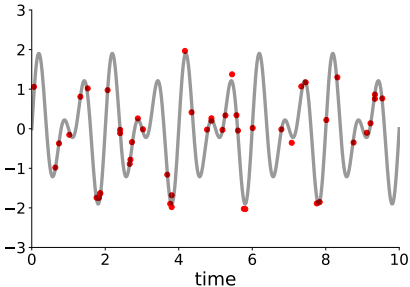
GP with noisy observations



$$
\begin{aligned}
f(\mathbf{x}) &\sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right) \\
y\left(\mathbf{x}_i\right) &= f\left(\mathbf{x}_i\right) + \epsilon_i \\
\epsilon_i &\sim \mathcal{N}\left(0, \sigma^2\right) \\
\mathcal{D} &= \left\{\left(\mathbf{x}_i, y\left(\mathbf{x}_i\right)\right) \mid i = 1, \ldots, N\right\}
\end{aligned}
$$

$$
\begin{bmatrix} y\left(\mathbf{x}_1\right) \\ \vdots \\ y\left(\mathbf{x}_N\right) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k\left(\mathbf{x}_1, \mathbf{x}_1\right) & \cdots & k\left(\mathbf{x}_1, \mathbf{x}_N\right) \\ \vdots & \ddots & \vdots \\ k\left(\mathbf{x}_N, \mathbf{x}_1\right) & \cdots & k\left(\mathbf{x}_N, \mathbf{x}_N\right) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \right)
$$

GP with noisy observations

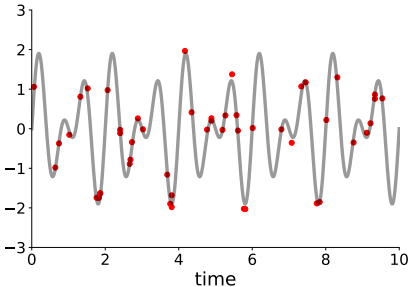


$$f(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$
$$y\left(\mathbf{x}_i\right) = f\left(\mathbf{x}_i\right) + \epsilon_i$$
$$\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$$
$$\mathcal{D} = \left\{\left(\mathbf{x}_i, y\left(\mathbf{x}_i\right)\right) \mid i = 1, \ldots, N\right\}$$

$$\begin{bmatrix} y\left(\mathbf{x}_1\right) \\ \vdots \\ y\left(\mathbf{x}_N\right) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k\left(\mathbf{x}_1, \mathbf{x}_1\right) & \cdots & k\left(\mathbf{x}_1, \mathbf{x}_N\right) \\ \vdots & \ddots & \vdots \\ k\left(\mathbf{x}_N, \mathbf{x}_1\right) & \cdots & k\left(\mathbf{x}_N, \mathbf{x}_N\right) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}\right)$$

In matrix form: $\mathbf{f} \sim \mathcal{N}\left(\mathbf{0},\, \mathbf{K} + \sigma\mathbf{I}\right)$

GP with noisy observations

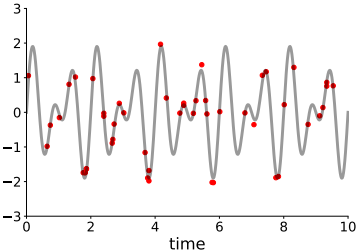


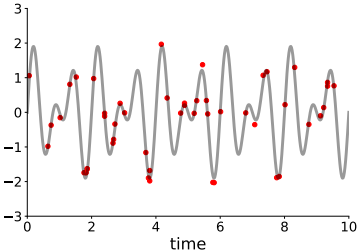
$$f(\mathbf{x}) \sim \mathcal{GP}\left(0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$
$$y\left(\mathbf{x}_i\right) = f\left(\mathbf{x}_i\right) + \epsilon_i$$
$$\epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$$
$$\mathcal{D} = \left\{\left(\mathbf{x}_i, y\left(\mathbf{x}_i\right)\right) \mid i = 1, \ldots, N\right\}$$

$$\begin{bmatrix} y\left(\mathbf{x}_1\right) \\ \vdots \\ y\left(\mathbf{x}_N\right) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k\left(\mathbf{x}_1, \mathbf{x}_1\right) & \cdots & k\left(\mathbf{x}_1, \mathbf{x}_N\right) \\ \vdots & \ddots & \vdots \\ k\left(\mathbf{x}_N, \mathbf{x}_1\right) & \cdots & k\left(\mathbf{x}_N, \mathbf{x}_N\right) \end{bmatrix} + \sigma^2 \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}\right)$$

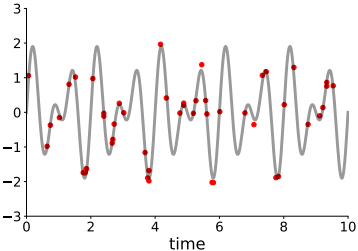
In matrix form: $\mathbf{f} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K} + \sigma\mathbf{I}\right)$ **For prediction:** $p\left(f\left(\mathbf{x}_*\right) \mid \mathbf{y}\right)$

Multiple-output Gaussian process

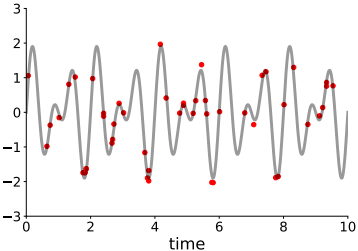




Multiple-output Gaussian process

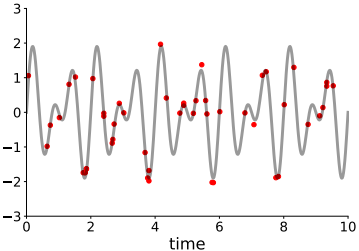


$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$



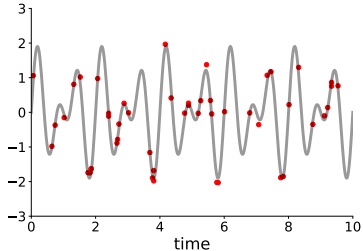
$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

Multiple-output Gaussian process



$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

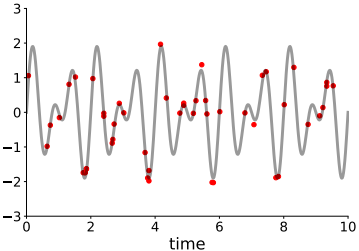
$$\mathcal{D}_1 = \left\{\left(\mathbf{x}_{i,1}, f_1\left(\mathbf{x}_{i,1}\right)\right) \mid i = 1, \ldots, N_1\right\}$$



$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

$$\mathcal{D}_2 = \left\{\left(\mathbf{x}_{i,2}, f_2\left(\mathbf{x}_{i,2}\right)\right) \mid i = 1, \ldots, N_2\right\}$$

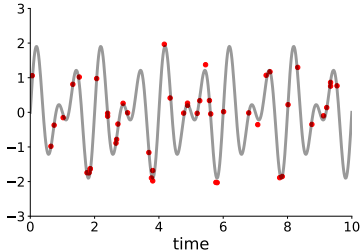
Multiple-output Gaussian process



$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

$$\mathcal{D}_1 = \left\{\left(\mathbf{x}_{i,1},\, f_1\left(\mathbf{x}_{i,1}\right)\right) \mid i = 1, \ldots, N_1\right\}$$

$$\mathbf{f}_1 \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_1\right)$$

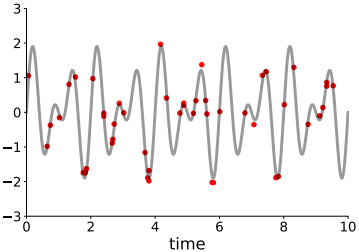


$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

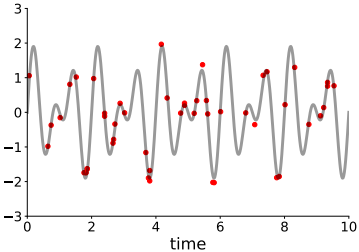
$$\mathcal{D}_2 = \left\{\left(\mathbf{x}_{i,2},\, f_2\left(\mathbf{x}_{i,2}\right)\right) \mid i = 1, \ldots, N_2\right\}$$

$$\mathbf{f}_2 \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_2\right)$$

Multiple-output Gaussian process



$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$



$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

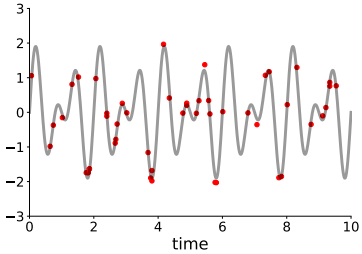
$$\mathcal{D}_1 = \left\{\left(\mathbf{x}_{i,1},\, f_1\left(\mathbf{x}_{i,1}\right)\right) \mid i = 1, \ldots, N_1\right\}$$

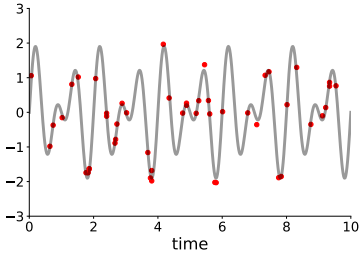
$$\mathcal{D}_2 = \left\{\left(\mathbf{x}_{i,2},\, f_2\left(\mathbf{x}_{i,2}\right)\right) \mid i = 1, \ldots, N_2\right\}$$

$$\mathbf{f}_1 \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_1\right) \qquad \mathbf{f}_2 \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_2\right)$$

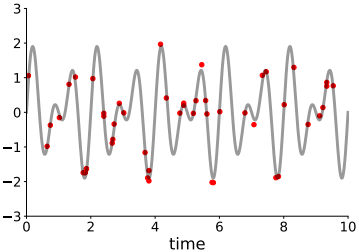
$$\underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}}_{\mathbf{f}} \sim \mathcal{N}\left(\underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{0}}, \underbrace{\begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{bmatrix}}_{\mathbf{K}_{\mathbf{f},\mathbf{f}}}\right)$$

Multiple-output Gaussian process

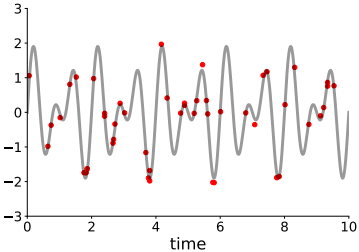




Multiple-output Gaussian process

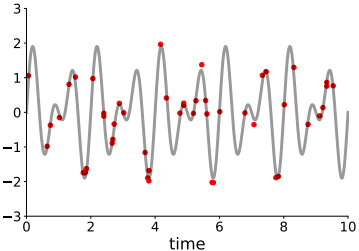


$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$



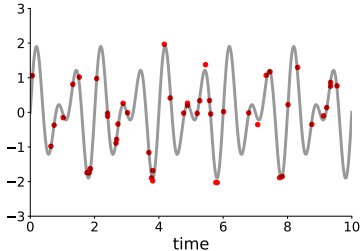
$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

Multiple-output Gaussian process



$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

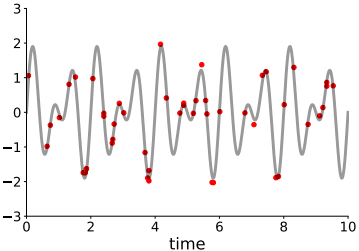
$$\mathcal{D}_1 = \left\{\left(\mathbf{x}_{i,1}, y_1\left(\mathbf{x}_{i,1}\right)\right) \mid i = 1, \ldots, N_1\right\}$$



$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

$$\mathcal{D}_2 = \left\{\left(\mathbf{x}_{i,2}, y_2\left(\mathbf{x}_{i,2}\right)\right) \mid i = 1, \ldots, N_2\right\}$$

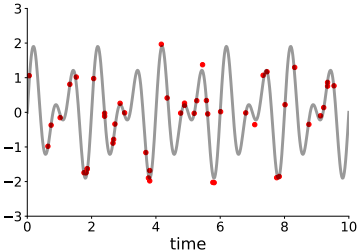
Multiple-output Gaussian process



$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

$$\mathcal{D}_1 = \left\{\left(\mathbf{x}_{i,1},\, y_1\left(\mathbf{x}_{i,1}\right)\right) \mid i = 1, \ldots, N_1\right\}$$

$$\mathbf{y}_1 \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_1 + \sigma_1^2 \mathbf{I}\right)$$

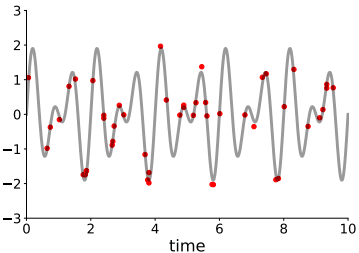


$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

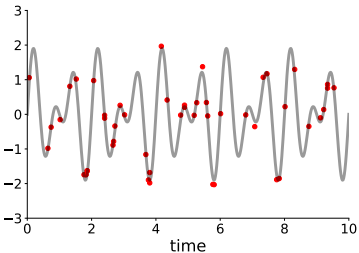
$$\mathcal{D}_2 = \left\{\left(\mathbf{x}_{i,2},\, y_2\left(\mathbf{x}_{i,2}\right)\right) \mid i = 1, \ldots, N_2\right\}$$

$$\mathbf{f}_2 \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_2 + \sigma_2^2 \mathbf{I}\right)$$

Multiple-output Gaussian process



$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$



$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

$$\mathcal{D}_1 = \left\{\left(\mathbf{x}_{i,1},\, y_1\left(\mathbf{x}_{i,1}\right)\right) \mid i = 1, \ldots, N_1\right\}$$

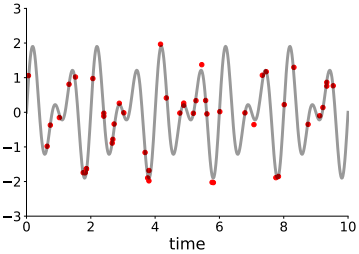
$$\mathcal{D}_2 = \left\{\left(\mathbf{x}_{i,2},\, y_2\left(\mathbf{x}_{i,2}\right)\right) \mid i = 1, \ldots, N_2\right\}$$

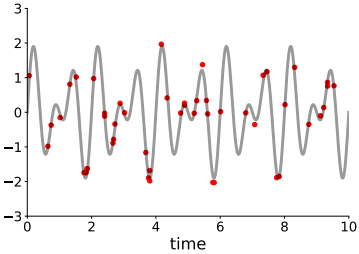
$$\mathbf{y}_1 \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_1 + \sigma_1^2 \mathbf{I}\right)$$

$$\mathbf{f}_2 \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_2 + \sigma_2^2 \mathbf{I}\right)$$

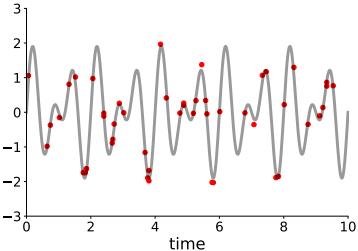
$$\underbrace{\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}}_{\mathbf{y}} \sim \mathcal{N}\left(\underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{0}}, \underbrace{\begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{bmatrix}}_{\mathbf{K}_{\mathbf{f},\mathbf{f}}} + \underbrace{\begin{bmatrix} \sigma_1^2 \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \sigma_2^2 \mathbf{I} \end{bmatrix}}_{\Sigma}\right)$$

Multiple-output Gaussian process

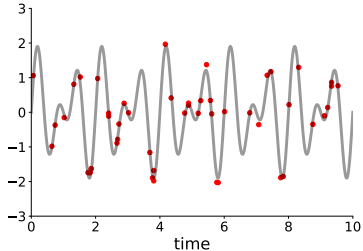




Multiple-output Gaussian process

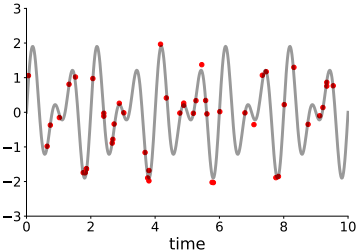


$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$



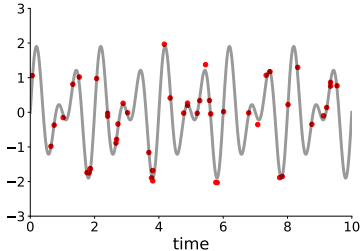
$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

Multiple-output Gaussian process



$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

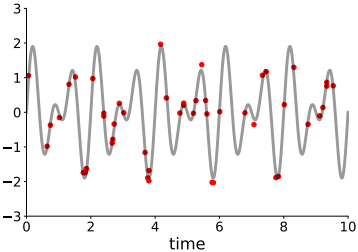
$$\mathcal{D}_1 = \left\{\left(\mathbf{x}_{i,1},\, f_1\left(\mathbf{x}_{i,1}\right)\right) \mid i = 1, \ldots, N_1\right\}$$



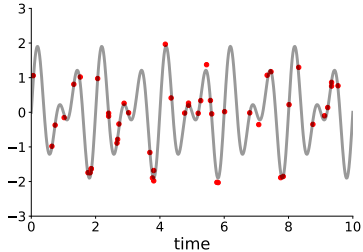
$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

$$\mathcal{D}_2 = \left\{\left(\mathbf{x}_{i,2},\, f_2\left(\mathbf{x}_{i,2}\right)\right) \mid i = 1, \ldots, N_2\right\}$$

Multiple-output Gaussian process



$$f_1(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_1\left(\mathbf{x}, \mathbf{x}'\right)\right)$$



$$f_2(\mathbf{x}) \sim \mathcal{GP}\left(0,\, k_2\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

$$\mathcal{D}_1 = \left\{\left(\mathbf{x}_{i,1},\, f_1\left(\mathbf{x}_{i,1}\right)\right) \mid i = 1, \ldots, N_1\right\}$$

$$\mathcal{D}_2 = \left\{\left(\mathbf{x}_{i,2},\, f_2\left(\mathbf{x}_{i,2}\right)\right) \mid i = 1, \ldots, N_2\right\}$$

$$\mathbf{K}_{\mathbf{f},\mathbf{f}} = \begin{bmatrix} \mathbf{K}_1 & ? \\ ? & \mathbf{K}_2 \end{bmatrix}$$

Build a cross-covariance function $\mathrm{cov}[f_1(\mathbf{x}),\, f_2(\mathbf{x}')]$ such that $\mathbf{K}_{\mathbf{f},\mathbf{f}}$ is positive semi-definite.

Why model task correlations?

Many applications: multi-output sensors, multi-channel time series, multi-label prediction.

If tasks are correlated, sharing structure can:

- Improve predictions for low-data tasks.
- Regularize hyperparameters and reduce overfitting.
- Provide coherent multi-output uncertainty.

Multi-task GPs

Multi-task Gaussian Processes (GPs) model multiple correlated outputs jointly.

Models:

- Intrinsic Coregionalization Model (ICM).
- Linear Model of Coregionalization (LMC).

The idea is to exploit task correlations to improve prediction and uncertainty quantification.

Vector-valued GP

Multi-task GP: $f : \mathcal{X} \to \mathbb{R}^D$, $f(x) = (f_1(x), \dots, f_D(x))^\intercal$.

Mean function:

$$m : \mathcal{X} \to \mathbb{R}^D.$$

Matrix-valued kernel:

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{D \times D},$$

encoding both input similarity and inter-task covariance.

Independent multi-task baseline

Simplest approach: one independent GP per task $i = 1, \dots, D$:

$$f_i(X) \sim \mathcal{N}\big(m_i(X), K_i(X, X)\big).$$

Joint covariance (stacking all tasks) is block-diagonal:

$$K(X, X) = \mathsf{diag}\big(K_1(X, X), \dots, K_D(X, X)\big).$$

No information sharing across tasks; correlations are ignored.

Intrinsic Coregionalization Model

Consider D outputs collected in $f(x) = (f_1(x), \dots, f_D(x))^\intercal$.

ICM assumes:

$$f(x) \sim \mathcal{GP}(0,\ K((x,i),(x',j))),$$

with separable covariance

$$\text{Cov}\left(f_i(x), f_j(x')\right) = B_{ij}\, k(x,x').$$

where $k(x,x')$ is a scalar positive definite kernel, and $B \in \mathbb{R}^{D\times D}$ is symmetric positive semidefinite.

ICM kernel as Kronecker product

Let $X = (x_1, \dots, x_N)$ and $K_X \in \mathbb{R}^{N \times N}$ with

$$(K_X)_{nm} = k(x_n, x_m).$$

Stack all outputs as $\text{vec}(F) \in \mathbb{R}^{ND}$, where $F_{n,i} = f_i(x_n)$.
Under ICM:

$$\text{vec}(F) \sim \mathcal{N}(0,\, K_{\text{ICM}}), \quad K_{\text{ICM}} = B \otimes K_X.$$

This shows a separable structure: task covariance $\times$ input covariance.

ICM: positive semidefiniteness of B

B must be positive semidefinite: $z^\mathsf{T} B z \geq 0$ for all $z \in \mathbb{R}^D$.

A common parameterization:

$$B = WW^\mathsf{T} + \mathsf{diag}(v),$$

with $W \in \mathbb{R}^{D \times R}$ (low rank) and $v \in \mathbb{R}^D_{\geq 0}$.

This guarantees $B \succeq 0$ and controls rank of inter-task correlations via R and marginal variances via v.

ICM: correlation coefficients

Covariance between tasks i, j at a fixed input:

$$\text{Cov}(f_i(x), f_j(x)) = B_{ij}\, k(x, x).$$

If $k(x, x) = \sigma_k^2$, the task covariance (up to σ_k^2) is B_{ij}.

Pearson correlation coefficient:

$$\rho_{ij} = \frac{B_{ij}}{\sqrt{B_{ii} B_{jj}}}.$$

ICM: latent-factor construction (rank-1)

Single latent GP:

$$u(x) \sim \mathcal{GP}(0, k(x, x')).$$

Linear mixing to outputs:

$$f_i(x) = a_i \, u(x), \quad i = 1, \dots, D.$$

Then

$$\mathrm{Cov}(f_i(x), f_j(x')) = a_i a_j \, k(x, x'),$$

Hence $B = aa^{\mathsf{T}}$, which is rank-1 and positive semidefinite.

ICM: low-rank latent-factor view

Generalize to R latent GPs $u_r(x) \sim \mathcal{GP}(0, k(x, x'))$, independent over r.

Mixing:

$$f_i(x) = \sum_{r=1}^{R} w_{ir} u_r(x),$$

with $W = (w_{ir}) \in \mathbb{R}^{D \times R}$.

Then

$$\text{Cov}(f_i(x), f_j(x')) = \sum_{r=1}^{R} w_{ir} w_{jr} k(x, x') = B_{ij} k(x, x'),$$

with $B = WW^{\mathsf{T}}$.

LMC: generalization

LMC allows multiple input kernels with latent GPs
$u_q(x) \sim \mathcal{GP}(0, k_q(x, x'))$, independent over $q = 1, \dots, Q$.

Mixing:

$$f_i(x) = \sum_{q=1}^{Q} a_{iq}\, u_q(x).$$

Then

$$\text{Cov}(f_i(x), f_j(x')) = \sum_{q=1}^{Q} a_{iq} a_{jq}\, k_q(x, x') = \sum_{q=1}^{Q} B_{ij}^{(q)} k_q(x, x'),$$

with $B^{(q)} = a_{:q} a_{:q}^{\mathsf{T}}$.

ICM as special case of LMC

If all latent processes share the same kernel k, i.e. $k_q = k$ for all q:

$$\text{Cov}(f_i(x), f_j(x')) = \left(\sum_{q=1}^{Q} B_{ij}^{(q)} \right) k(x, x') = B_{ij} k(x, x'),$$

where $B = \sum_{q=1}^{Q} B^{(q)}$.

- ICM typically refers to a single shared kernel and one coregionalization matrix, while LMC uses sums of such components.
- LMC can capture richer patterns when tasks behave differently across inputs.

Matrix form and structure

Under ICM, for N inputs and D tasks:

$$\text{vec}(F) \sim \mathcal{N}(0,\, B \otimes K_X).$$

Outputs are coupled via B and share smoothness through k.

Under LMC:

$$K_{\text{LMC}} = \sum_{q=1}^{Q} B^{(q)} \otimes K_X^{(q)},$$

with $(K_X^{(q)})_{nm} = k_q(x_n, x_m)$.

Noise modeling

Observations:

$$y(x) = f(x) + \varepsilon, \quad \varepsilon \sim \mathcal{N}_D(0, \Sigma),$$

with $\Sigma \in \mathbb{R}^{D \times D}$ positive semidefinite.

Choices: - Diagonal: $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_D^2)$. - Full: $\Sigma = LL^\intercal$ with $L \in \mathbb{R}^{D \times r}$.

Correlated noise models shared disturbances across tasks.

Joint covariance with noise

Let K be the noise-free covariance from ICM/LMC over all tasks and inputs.

For N inputs and D tasks (stacked), a common form is:

$$K_n = K + I_N \otimes \Sigma,$$

where I_N is the $N \times N$ identity matrix.

The Kronecker structure is preserved when Σ is diagonal; with full Σ structure is more complex but still interpretable.

Predictions and uncertainty

At test inputs X_*:

$$p(f(X_*) \mid Y) = \mathcal{N}(\mu_*, \Sigma_*).$$

For each task t:

- Predictive mean $\mu_{*,t}(x)$.
- Variance $\mathrm{Var}(f_t(x))$ for credible bands.

Joint Σ_* contains cross-task predictive covariances.

Learned B and Σ

Coregionalization matrix:

$$B = WW^{\mathsf{T}} + \mathrm{diag}(v),$$

learned from data.

Interpretation:

- B_{ii} is the marginal variance of task i.
- B_{ij} is the learned covariance between tasks i and j.

Noise covariance $\Sigma = LL^{\mathsf{T}}$ can also be inspected (e.g. heatmaps) to understand shared noise.

Independent GPs vs ICM

Independent GPs:

- Separate mean, kernel, and likelihood per task.
- No cross-task covariance; hyperparameters estimated from each task alone.

ICM:

- Shared input kernel hyperparameters.
- Cross-task structure encoded in B and possibly Σ.

With few observations per task, ICM can regularize and improve generalization.

When to use ICM vs LMC

ICM (single shared kernel):

- Tasks have similar smoothness and variation across input space.
- Fewer hyperparameters, simpler optimization.

LMC (multiple kernels):

- Tasks differ in smoothness, periodicity, or characteristic scales.
- More flexible but higher computational and optimization cost.

Practical strategy: Start with low-rank ICM; increase rank or move to LMC if cross-task structure appears underfitted.

Infinite Neural Network = Gaussian Process

Single Hidden Layer Architecture

One-Hidden-Layer Network

Network structure:

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Hidden layer: J units producing $\mathbf{a}^{(1)} \in \mathbb{R}^J$
- Output layer: single linear unit producing $y_k \in \mathbb{R}$

Forward pass:

$$y_k = f(\mathbf{x}; \mathbf{w}) = \sum_{j=1}^{J} w_{kj}\, h\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{k0}$$

where:

- $h(\cdot)$ is the activation function (e.g., ReLU, tanh).
- w_{ji}, w_{j0}: weights and bias of hidden unit j.
- w_{kj}, w_{k0}: weights and bias of the single output neuron.

Data Flow: Inputs and Outputs

Component	Input(s)	Output
Single neuron	$\mathbf{x}$	scalar activation a
Single hidden layer	$\mathbf{x}$	vector $\mathbf{a}^{(1)} \in \mathbb{R}^J$
Whole finite NN	$\mathbf{x}$	scalar prediction $y_k = f(\mathbf{x}; \mathbf{w})$

Training

Weights $\mathbf{w}, \mathbf{w}_0$ are **parameters** adjusted by backpropagation and gradient descent

From deterministic to Bayesian

Bayesian view (Neal 1994)

Weights become **random variables** with a prior distribution:

Output layer weights:

$$w_{1j} \sim \mathcal{N}\left(0, \frac{\alpha}{J}\right), \quad w_{10} \sim \mathcal{N}(0, \sigma^2)$$

Hidden layer weights:

$$w_{ji} \sim \mathcal{N}(0, \sigma_w^2), \quad w_{j0} \sim \mathcal{N}(0, \sigma_b^2)$$

From deterministic to Bayesian

What Changes?

In the Bayesian view, the **network output becomes a random variable**:

$$f(\mathbf{x}) \sim ?$$

- The **input** is a fixed point $\mathbf{x}$
- **Randomness** is added by random weights
- The **output** is a random scalar $f(\mathbf{x})$

Central Limit Theorem

For any fixed finite set of inputs $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$:

The vector of outputs:

$$\left(f(\mathbf{x}^{(1)}), \dots, f(\mathbf{x}^{(N)})\right)$$

can be written as a sum of J i.i.d. random vectors (one contribution per hidden unit).

By the multivariate Central Limit Theorem:

$$\lim_{J \to \infty} \left(f(\mathbf{x}^{(1)}), \dots, f(\mathbf{x}^{(N)})\right) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma}_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$.

Infinite Neural Network = Gaussian Process

Theorem (Neal 1994)

When a one-hidden-layer Bayesian neural network has $J \to \infty$ hidden units with i.i.d. zero-mean weight priors (properly scaled), the random function converges in distribution to a Gaussian Process:

$$f(\cdot) \sim \mathcal{GP}\left(0, k(\mathbf{x}, \mathbf{x}') = \alpha\, \mathbb{E}\left[h\left(\sum_i w_{ji} x_i + w_{j0}\right) h\left(\sum_i w_{ji} x'_i + w_{j0}\right)\right] + \sigma^2\right)$$

Mean Function

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}\left[\sum_{j=1}^{J} w_{1j}\, h\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{10}\right] \\
&= \sum_{j=1}^{J} \mathbb{E}[w_{1j}] \cdot \mathbb{E}\left[h\left(\sum_i w_{ji} x_i + w_{j0}\right)\right] + \mathbb{E}[w_{10}] \\
&= \sum_{j=1}^{J} 0 \cdot \mathbb{E}\left[h\left(\sum_i w_{ji} x_i + w_{j0}\right)\right] + 0 \\
&= 0
\end{aligned}
$$

where we used that all weights have **zero mean**.

Covariance Function

For any two distinct inputs $\mathbf{x}, \mathbf{x}'$:

$$k(\mathbf{x}, \mathbf{x}') = \text{Cov}\left[f(\mathbf{x}), f(\mathbf{x}')\right] = \mathbb{E}\left[f(\mathbf{x}) \cdot f(\mathbf{x}')\right]$$

(since $\mathbb{E}[f(\mathbf{x})] = 0$)

Covariance Function Derivation

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[\left(\sum_{j=1}^{J} w_{1j} h\left(\sum_i w_{ji} x_i + w_{j0}\right) + w_{10}\right)\left(\sum_{j=1}^{J} w_{1j} h\left(\sum_i w_{ji} x'_i + w_{j0}\right) + w_{10}\right)\right] \\
&= \mathbb{E}\left[\sum_{j=1}^{J} w_{1j}^2\, h\left(\sum_i w_{ji} x_i + w_{j0}\right) h\left(\sum_i w_{ji} x'_i + w_{j0}\right)\right] + \mathbb{E}[w_{10}^2] \\
&= \sum_{j=1}^{J} \mathbb{E}[w_{1j}^2]\mathbb{E}\left[h\left(\sum_i w_{ji} x_i + w_{j0}\right) h\left(\sum_i w_{ji} x'_i + w_{j0}\right)\right] + \mathsf{Var}(w_{10}) \\
&= \sum_{j=1}^{J} \frac{\alpha}{J}\, \mathbb{E}\left[h\left(\sum_i w_{ji} x_i + w_{j0}\right) h\left(\sum_i w_{ji} x'_i + w_{j0}\right)\right] + \sigma^2 \\
&= \alpha\, \mathbb{E}\left[h\left(\sum_i w_{ji} x_i + w_{j0}\right) h\left(\sum_i w_{ji} x'_i + w_{j0}\right)\right] + \sigma^2
\end{aligned}
$$

Extension to Deep Networks

Published as a conference paper at ICLR 2018

DEEP NEURAL NETWORKS AS GAUSSIAN PROCESSES

Jaehoon Lee[*†], **Yasaman Bahri**[*†], **Roman Novak**, **Samuel S. Schoenholz**, **Jeffrey Pennington**, **Jascha Sohl-Dickstein**

Google Brain
{jaehlee, yasamanb, romann, schsam, jpennin, jaschasd}@google.com

Extension to Deep Networks

Single hidden layer to multiple layers

The Central Limit Theorem argument extends by **induction** over layers. If the input to layer ℓ is governed by a GP, then:

- The output of layer ℓ is a **sum of independent terms** (one per neuron).
- By CLT, it is also a GP.
- This applies recursively to all layers.

Result: An **infinitely wide deep network** (all layers with $N_\ell \to \infty$) is a Gaussian Process.

Recursive kernel computation

Building kernels layer by layer

For a deep network with L hidden layers, denote:

- $z^{(\ell)}(x)$: output of layer ℓ (pre-activation of next layer)
- $x^{(\ell)}(x)$: activations of layer ℓ (post-nonlinearity)

Layer ℓ output:

$$z_i^{(\ell)}(x) = b_i^{(\ell)} + \sum_{j=1}^{N_\ell} W_{ij}^{(\ell)} x_j^{(\ell-1)}(x)$$

where $x_j^{(\ell-1)}(x) = \varphi(z_j^{(\ell-1)}(x))$ (activation function φ).

Recursive kernel formula

Base case (input layer):

$$K^{(0)}(x, x') = \mathbb{E}[z^{(0)}(x)z^{(0)}(x')]$$

For raw inputs:

$$K^{(0)}(x, x') = \sigma_b^2 + \sigma_w^2 \frac{\mathbf{x} \cdot \mathbf{x}'}{d_{in}}$$

where σ_w^2, σ_b^2 are weight and bias variances, d_{in} is input dimension.

Recursive kernel: inductive step

Assume layer $\ell - 1$ computes a GP with kernel $K^{(\ell-1)}$.

For layer ℓ, the output is:

$$z_i^{(\ell)}(x) = b_i^{(\ell)} + \sum_{j=1}^{N_\ell} W_{ij}^{(\ell)} \varphi(z_j^{(\ell-1)}(x))$$

As $N_\ell \to \infty$, by CLT:

$$z^{(\ell)}(x) \sim \mathcal{GP}(0, K^{(\ell)})$$

with kernel:

$$K^{(\ell)}(x, x') = \sigma_b^2 + \sigma_w^2 \mathbb{E}_{z^{(\ell-1)} \sim \mathcal{GP}(0, K^{(\ell-1)})} \left[\varphi(z^{(\ell-1)}(x)) \, \varphi(z^{(\ell-1)}(x')) \right]$$

Gaussian Process Deconvolution

Gaussian Process Deconvolution

Application	Domain
De-reverberation	Acoustic signal processing
Super-resolution	Image restoration, low-resolution to high-resolution
Perfusion imaging	fMRI, medical imaging (hemodynamic response)
Seismic imaging	Geophysics, signal propagation
Astronomy	Point-spread function deconvolution

The Deconvolution Problem

We observe noisy, possibly incomplete measurements y of a convolved signal:

$$y(t) = \int h(\tau)x(t-\tau)d\tau + \epsilon(t)$$

where:

- $x(t)$ is the latent source signal.
- $h(\tau)$ is the convolution filter (known or unknown).
- $\epsilon(t)$ is the measurement noise.
- $y(t)$ is the observed signal

Challenges

Traditional deconvolution methods (Wiener, Inverse FT) often suffer from:

- **Numerical instability** when filter has low spectral power.
- **No uncertainty quantification** (only point estimates).
- **Assumptions on signal structure** that may be unrealistic.

The GP Solution

Place a **Gaussian process prior** on the latent source $x(t)$.

The GP Hierarchical Model:

Given GP prior: $x(t) \sim \mathcal{GP}(m(t), k(t, t'))$

The posterior deconvolution is:

$$\hat{x}|y \sim \mathcal{GP}(\hat{m}(t), \hat{k}(t, t'))$$

where covariances are derived through convolution operators applied to the kernel k.

Example result