Machine Learning and Astrostatistics School:
Applications to Massive Stars

Gaussian processes

Alejandro Veloz

ORGANIZED BY:

Universidad de Valparaíso CHILE
UNIVERSIDAD TECNICA FEDERICO SANTA MARIA
UNIVERSIDAD MAYOR
OCEANS
Funded by the European Union

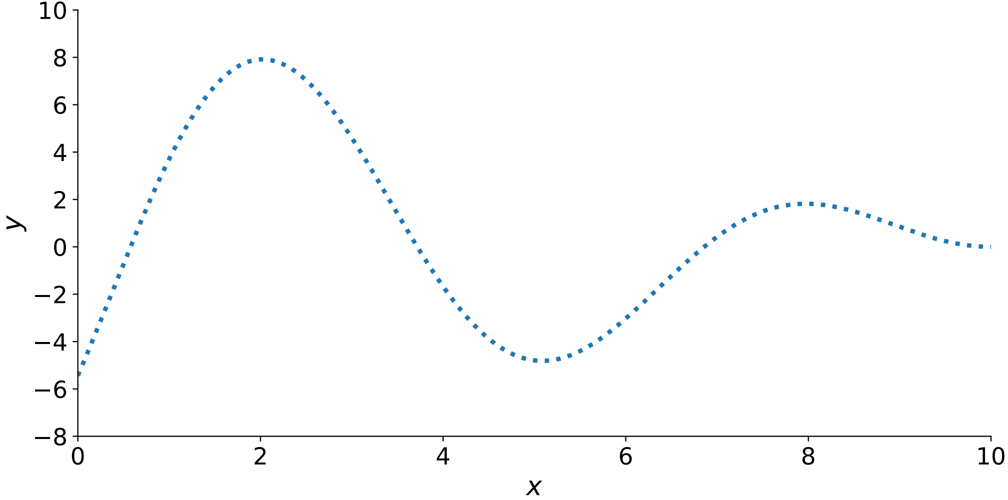
COLLABORATORS:

Instituto de Física y Astronomía
IDEUV Instituto de Estadística
UNIVERSIDAD MAYOR
CENTRO UM CMF Multidisciplinario de Física
CCTVal CENTRO CIENTÍFICO TECNOLÓGICO DE VALPARAÍSO
WOLFRAM

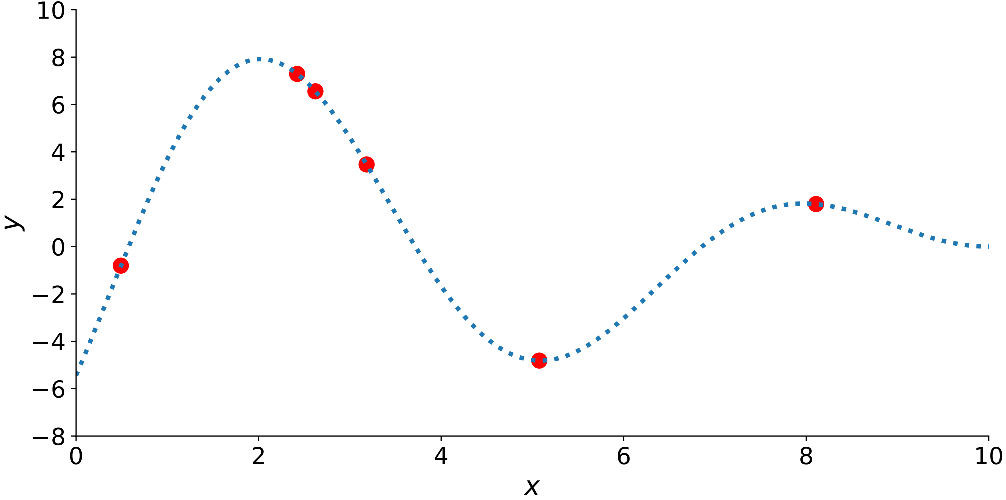
Email: alejandro.veloz@uv.cl

Slides and Labs: https://aavelozb.github.io/gp

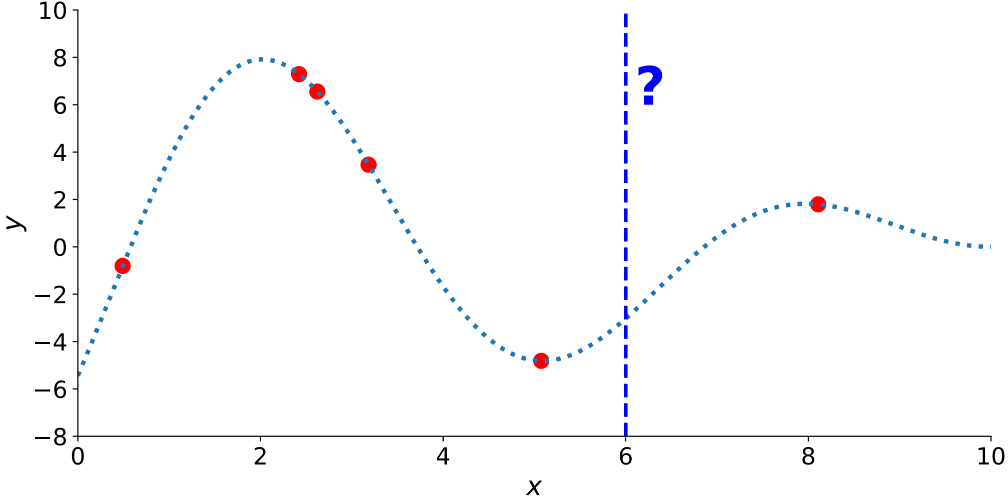
Motivation: non-linear regression



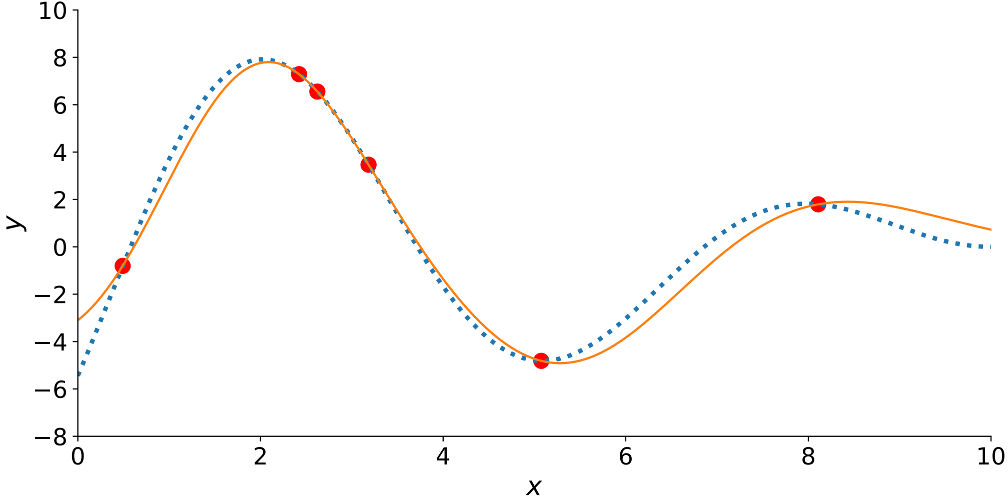
Motivation: non-linear regression



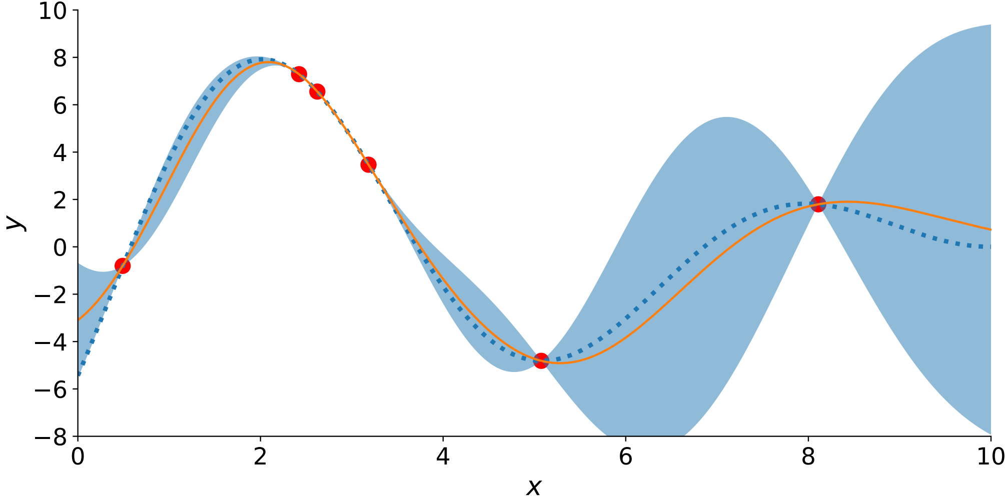
Motivation: non-linear regression



Motivation: non-linear regression



Motivation: non-linear regression



Outline

Making predictions with Gaussians

Gaussian processes for regression

Gaussian processes for classification

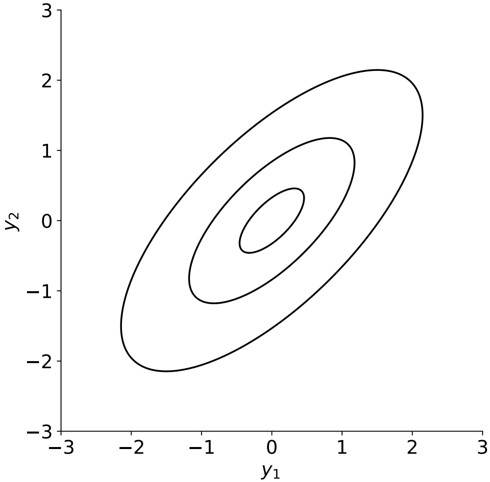
Gaussian processes and neural networks

Applications

Gaussian distribution

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right)$$

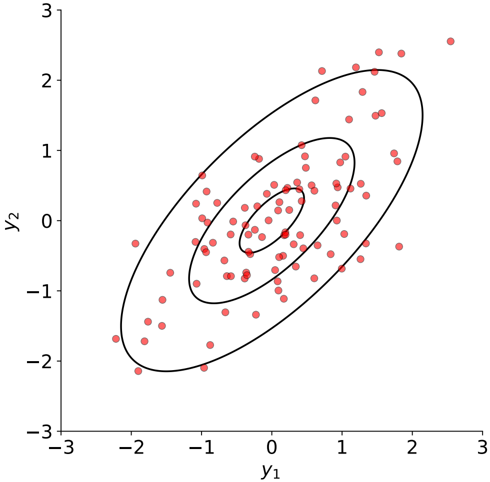
$$\Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



Gaussian distribution

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right)$$

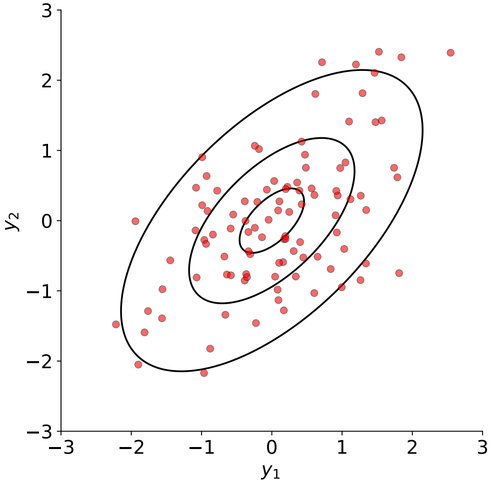
$$\Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



Gaussian distribution

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right)$$

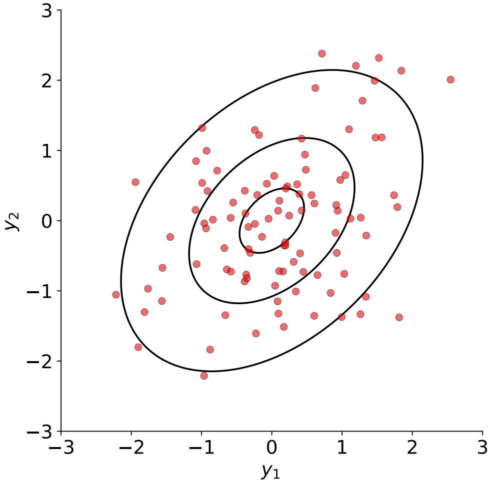
$$\Sigma = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$



Gaussian distribution

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right)$$

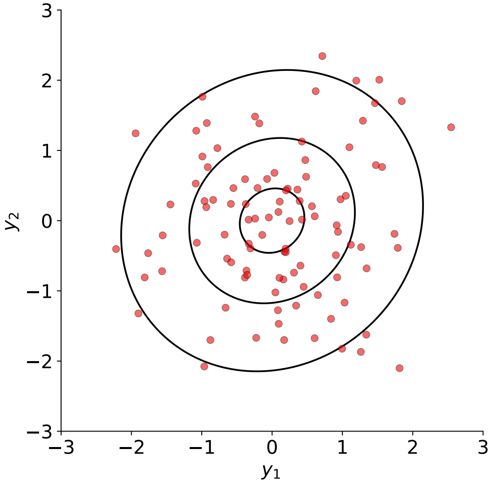
$$\Sigma = \begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$$



Gaussian distribution

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right)$$

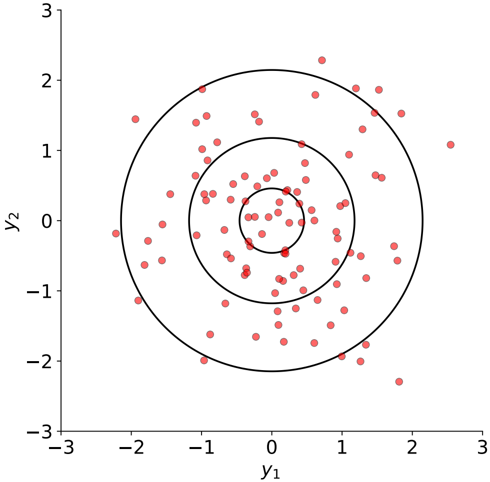
$$\Sigma = \begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix}$$



Gaussian distribution

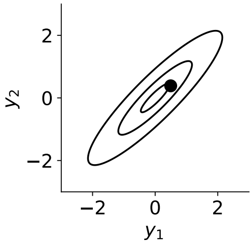
$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right)$$

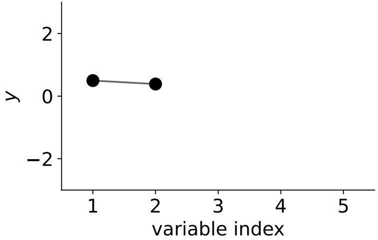
$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



New visualization

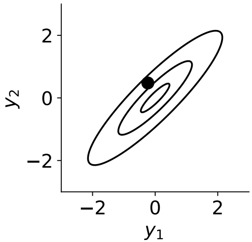
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

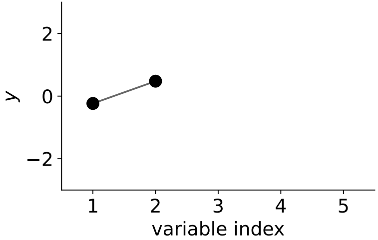




New visualization

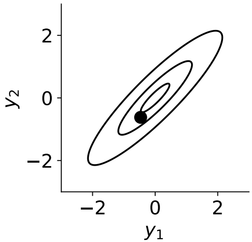
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

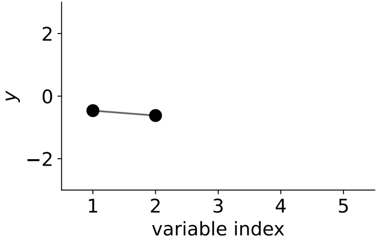




New visualization

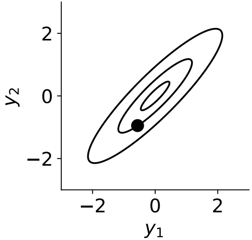
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

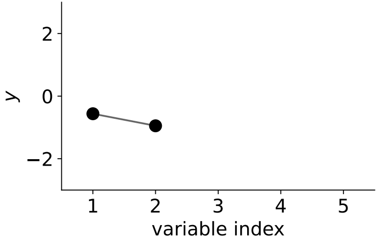




New visualization

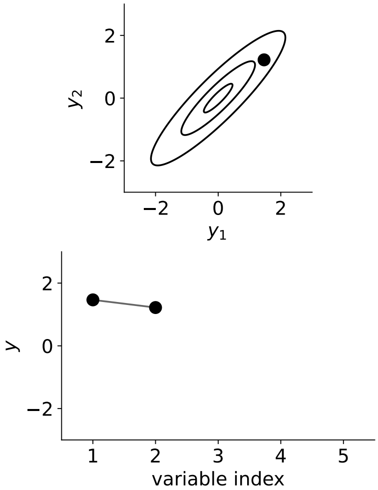
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$





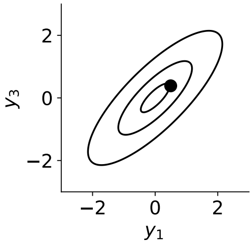
New visualization

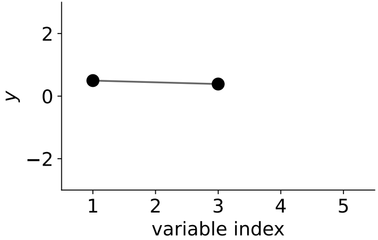
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$



New visualization

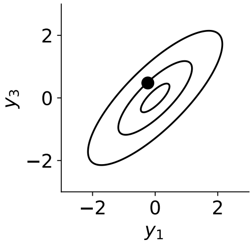
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

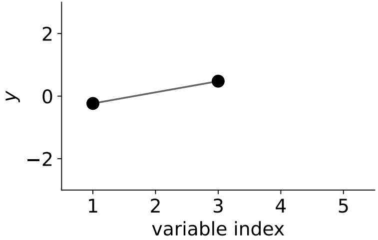




New visualization

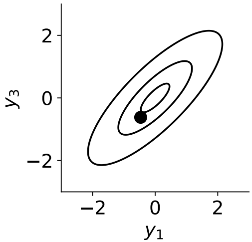
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

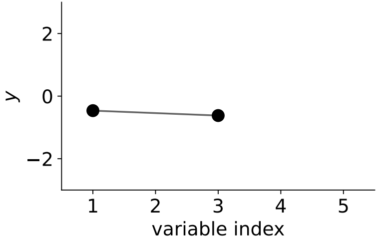




New visualization

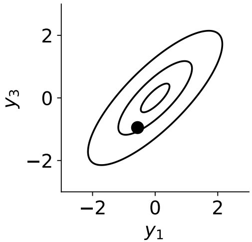
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

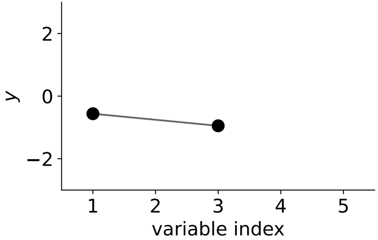




New visualization

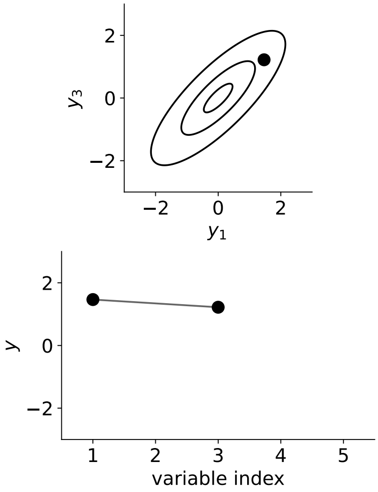
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$





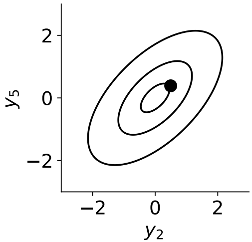
New visualization

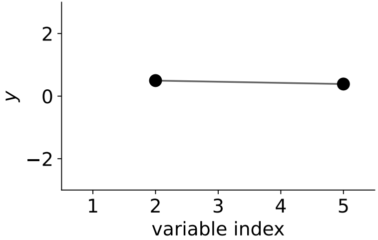
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$



New visualization

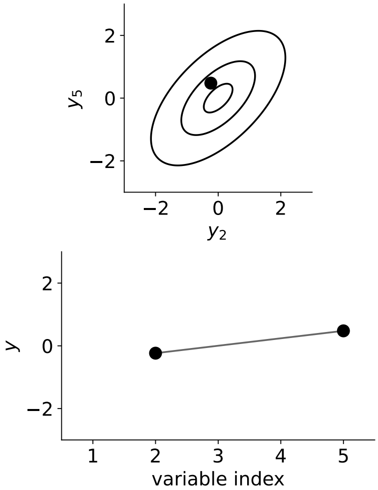
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$





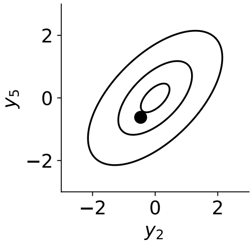
New visualization

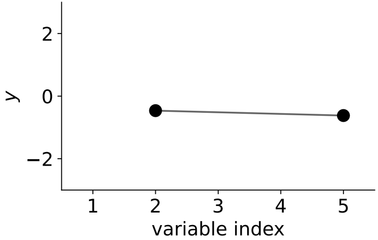
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$



New visualization

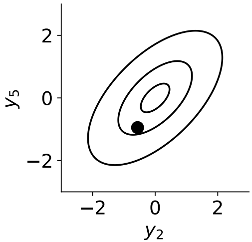
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

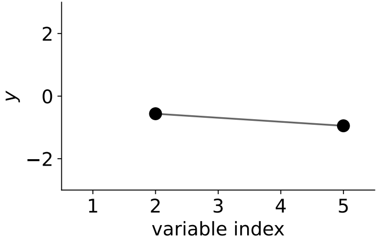




New visualization

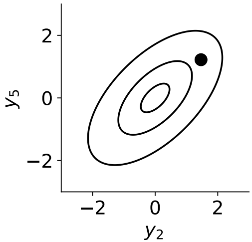
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

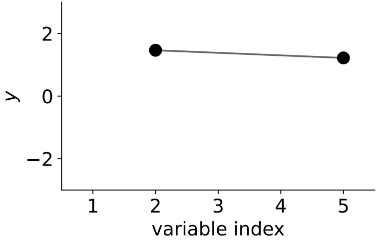




New visualization

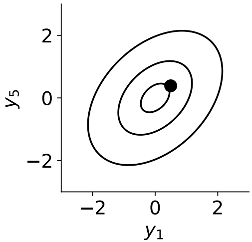
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

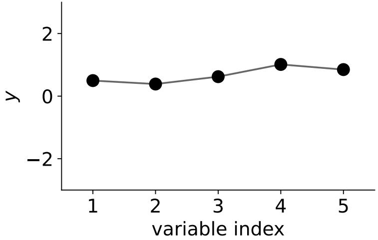




New visualization

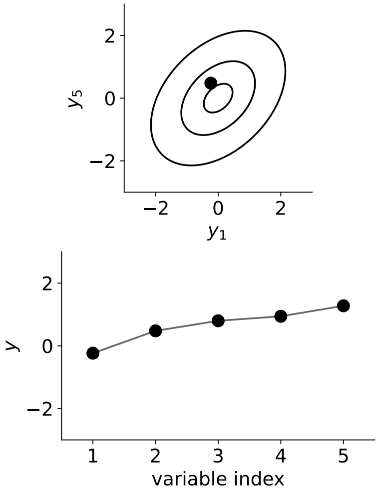
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$





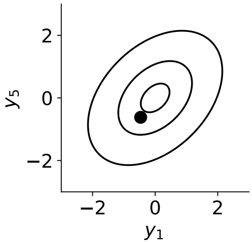
New visualization

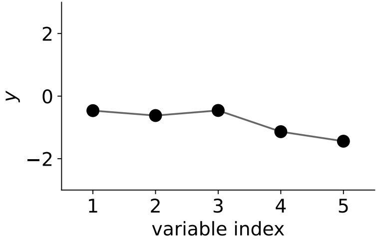
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$



New visualization

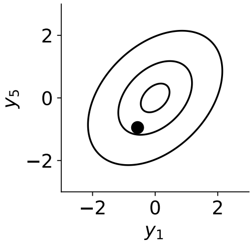
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$

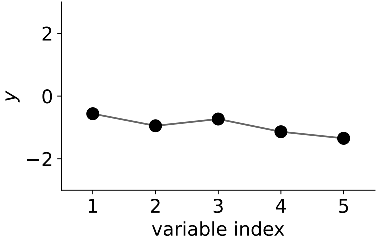




New visualization

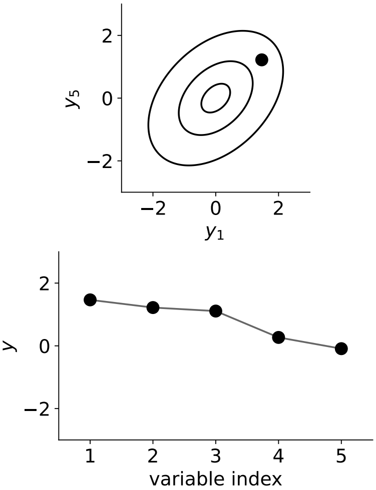
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$





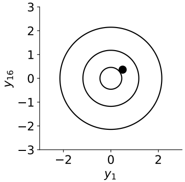
New visualization

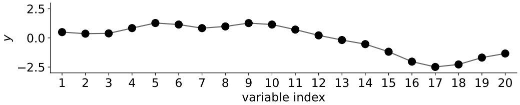
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.6 & 0.4 \\ 0.9 & 1 & 0.9 & 0.8 & 0.6 \\ 0.8 & 0.9 & 1 & 0.9 & 0.8 \\ 0.6 & 0.8 & 0.9 & 1 & 0.9 \\ 0.4 & 0.6 & 0.8 & 0.9 & 1 \end{bmatrix}$$



New visualization

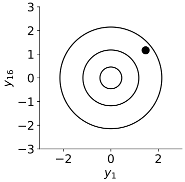
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & 0 & \dots \\ 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & \dots \\ 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & \dots \\ 0.3 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

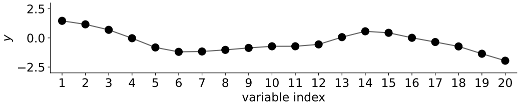




New visualization

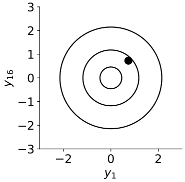
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & 0 & \cdots \\ 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & \cdots \\ 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & \cdots \\ 0.3 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

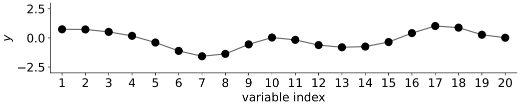




New visualization

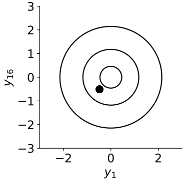
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & 0 & \cdots \\ 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & \cdots \\ 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & \cdots \\ 0.3 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

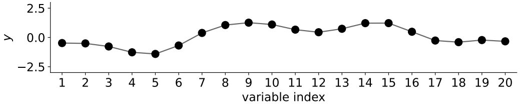




New visualization

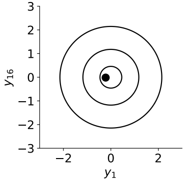
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & 0 & \dots \\ 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & \dots \\ 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & \dots \\ 0.3 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

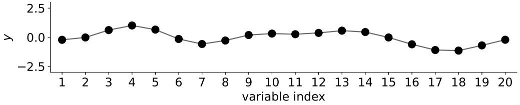




New visualization

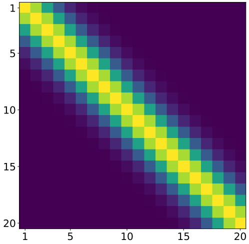
$$\Sigma = \begin{bmatrix} 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & 0 & \cdots \\ 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & 0 & \cdots \\ 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & 0 & \cdots \\ 0.3 & 0.6 & 0.9 & 1 & 0.9 & 0.6 & 0.3 & 0.1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

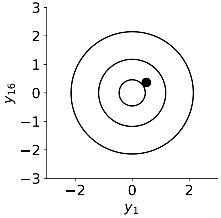


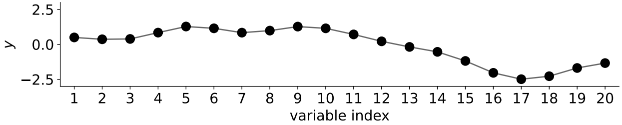


New visualization

$$\Sigma =$$

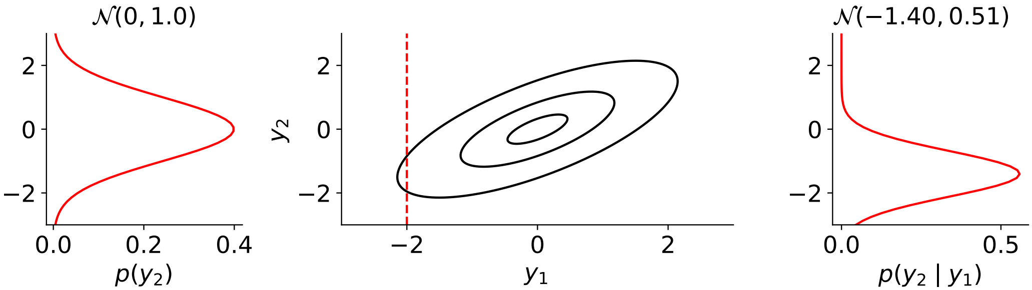






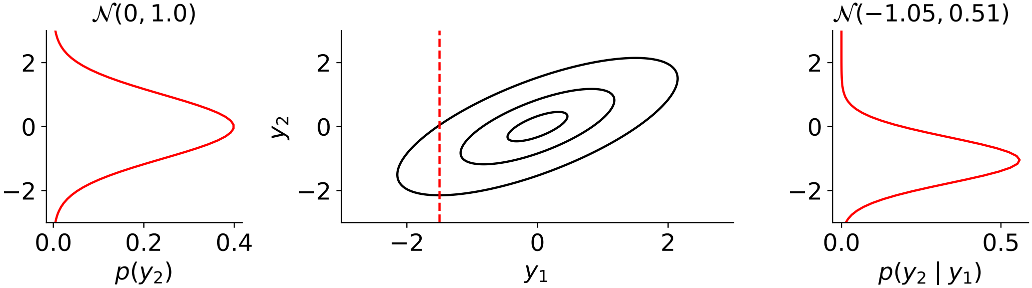
Marginalization and conditioning

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



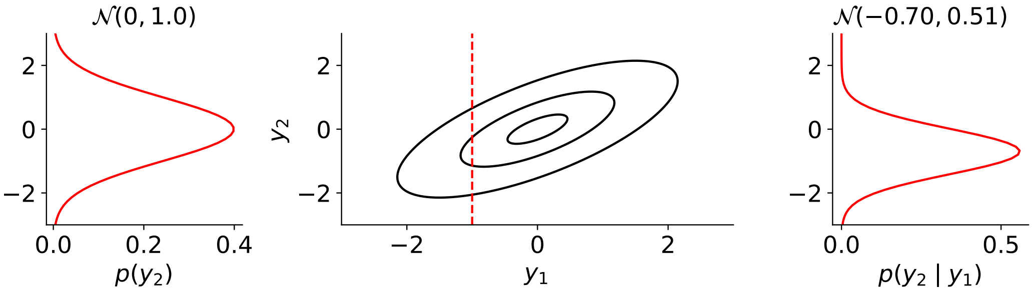
Marginalization and conditioning

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\intercal \Sigma^{-1} \mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



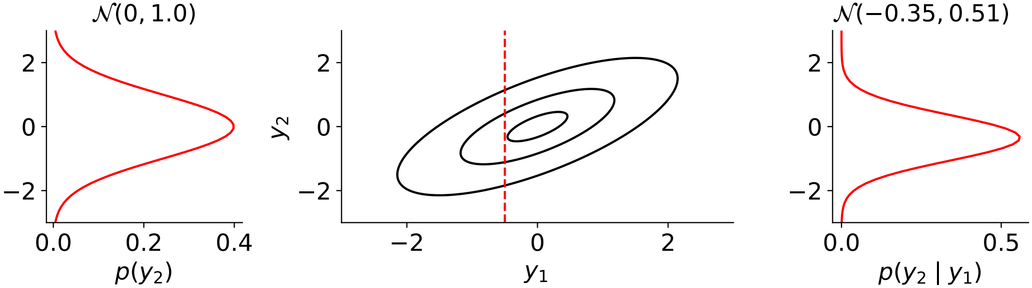
Marginalization and conditioning

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\intercal \Sigma^{-1} \mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



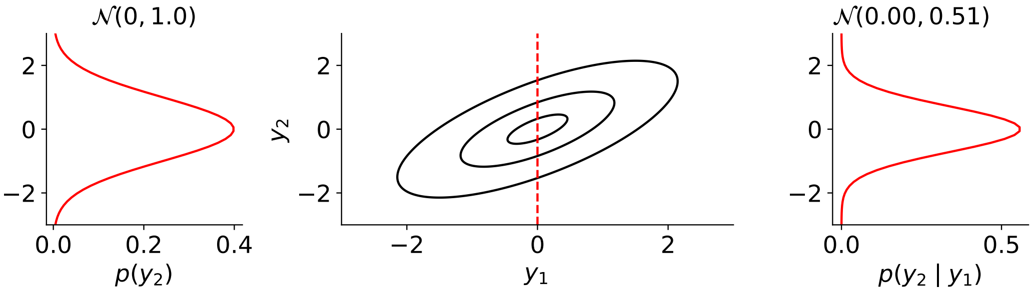
Marginalization and conditioning

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^{\intercal}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



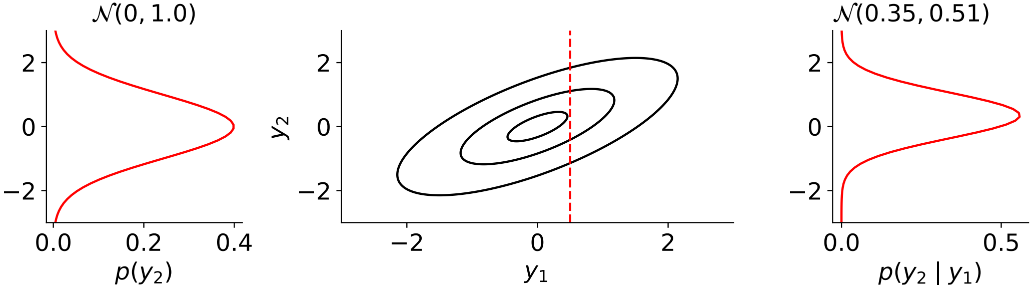
Marginalization and conditioning

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



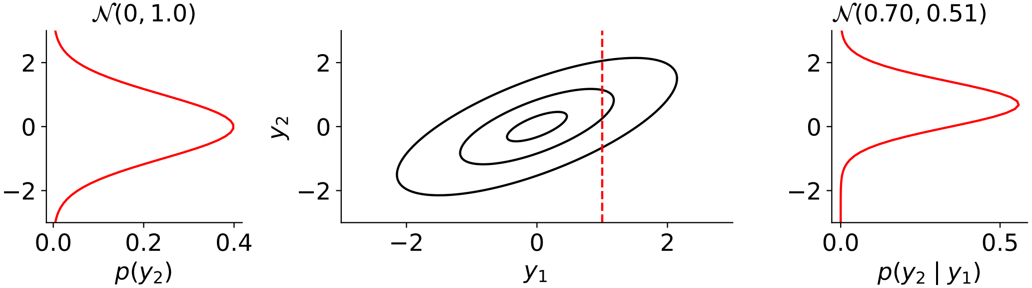
Marginalization and conditioning

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



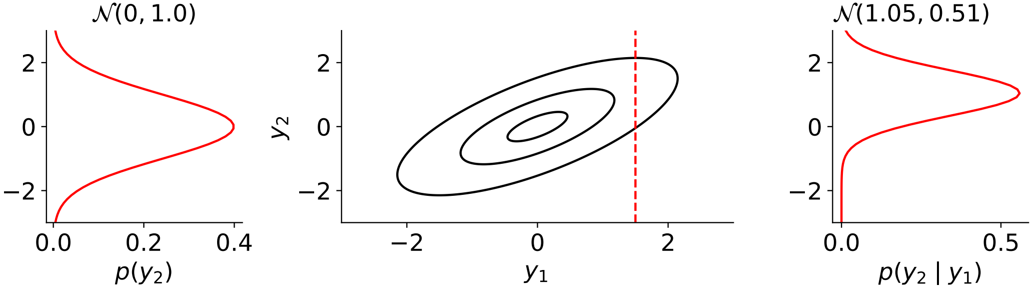
Marginalization and conditioning

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



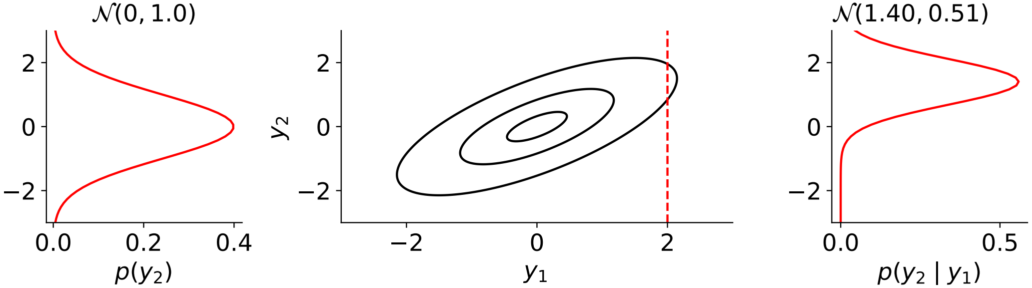
Marginalization and conditioning

$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\mathsf{T}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



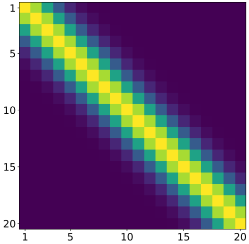
Marginalization and conditioning

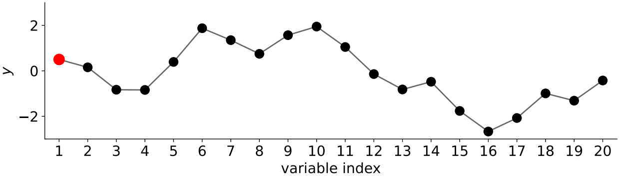
$$p(\mathbf{y} \mid \Sigma) \propto \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



Marginalization and conditioning

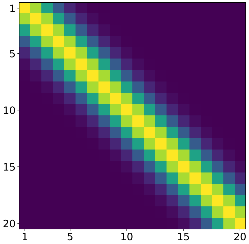
$$\Sigma =$$

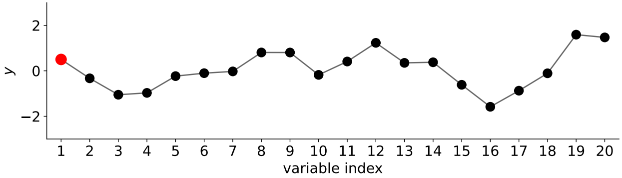




Marginalization and conditioning

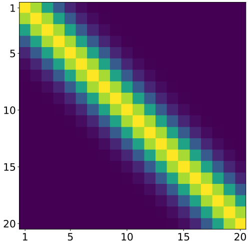
$\Sigma =$

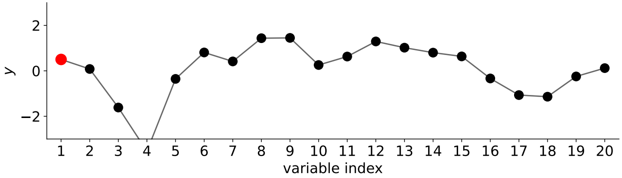




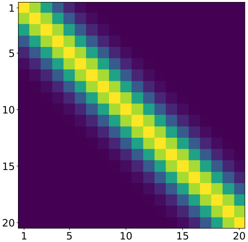
Marginalization and conditioning

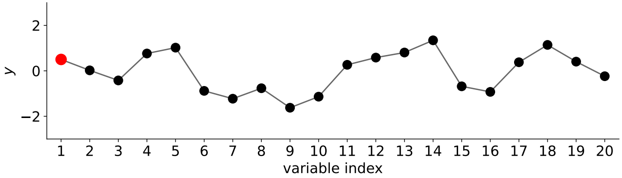
$\Sigma =$





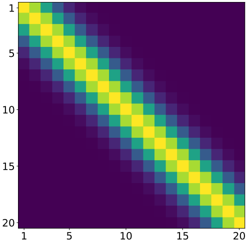
Marginalization and conditioning

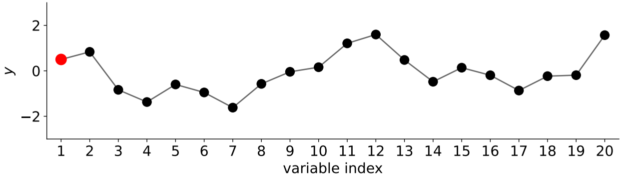
$\Sigma =$ 



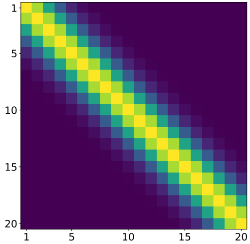
Marginalization and conditioning

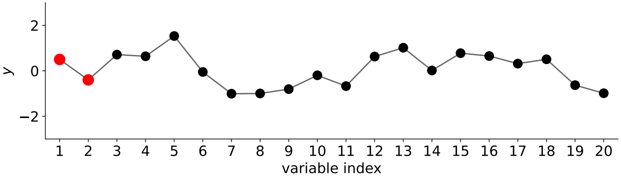
$$\Sigma =$$





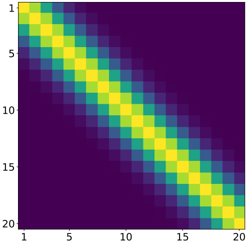
Marginalization and conditioning

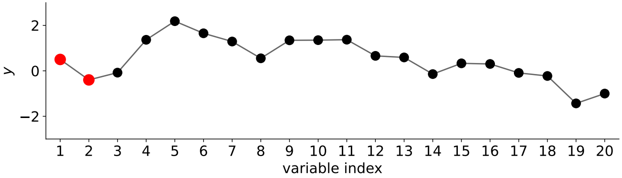
$\Sigma =$ 



Marginalization and conditioning

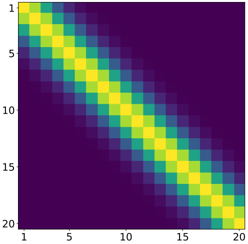
$$\Sigma =$$

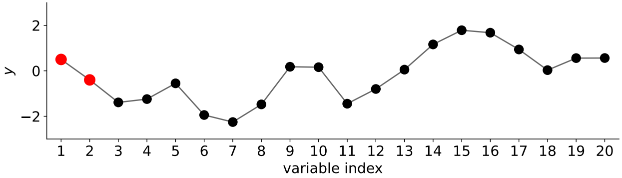




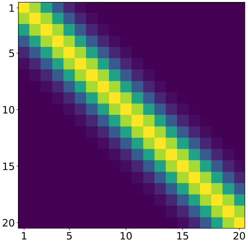
Marginalization and conditioning

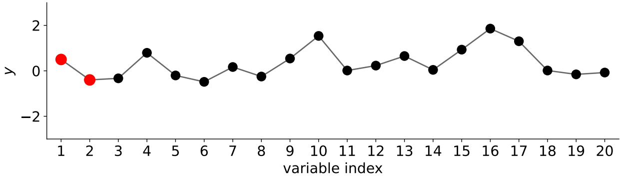
$$\Sigma =$$



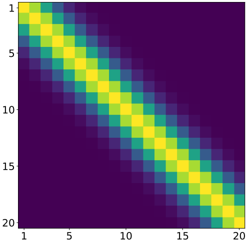


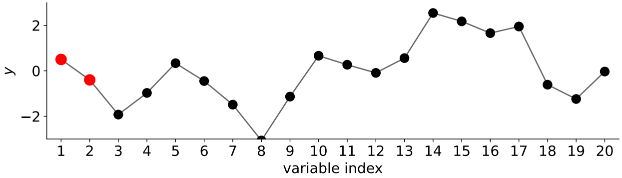
Marginalization and conditioning

$\Sigma =$ 



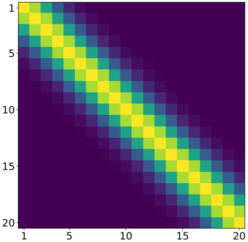
Marginalization and conditioning

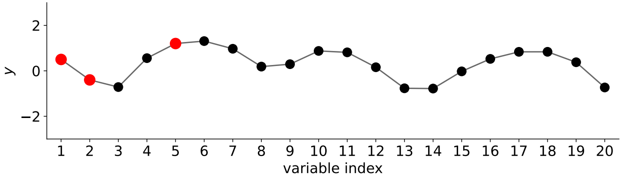
$\Sigma =$ 



Marginalization and conditioning

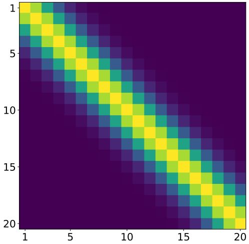
$$\Sigma =$$

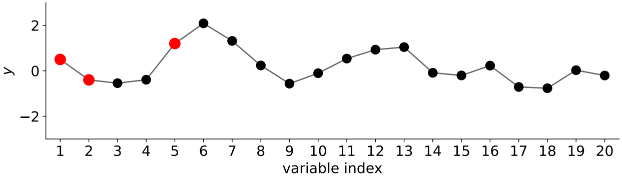




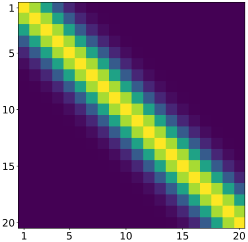
Marginalization and conditioning

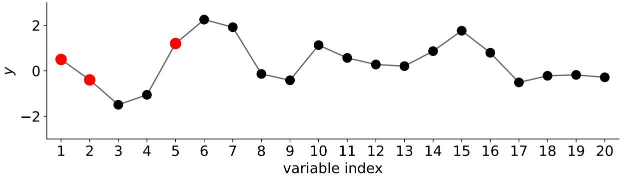
$$\Sigma =$$





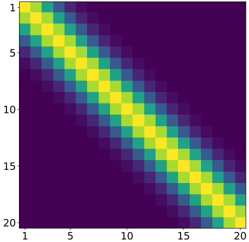
Marginalization and conditioning

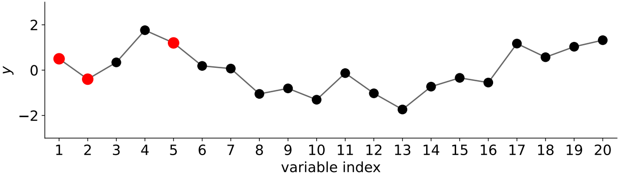
$\Sigma =$ 



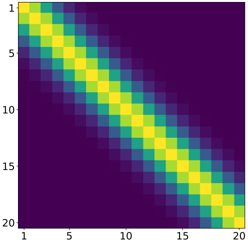
Marginalization and conditioning

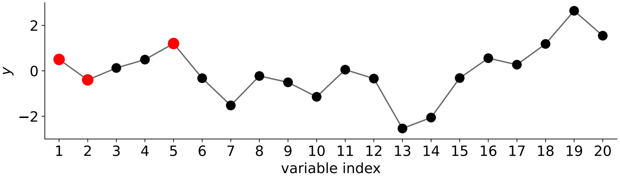
$$\Sigma =$$





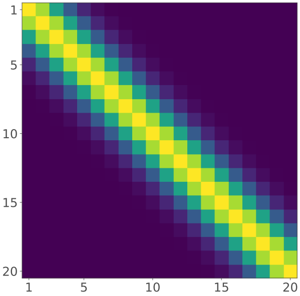
Marginalization and conditioning

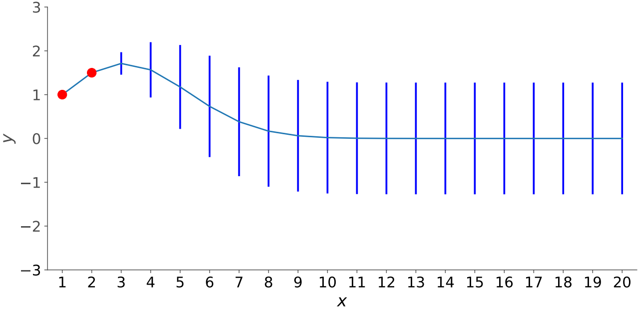
$\Sigma =$ 



Regression using Gaussians

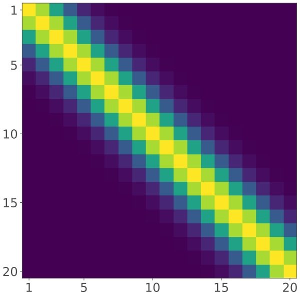
$\Sigma =$

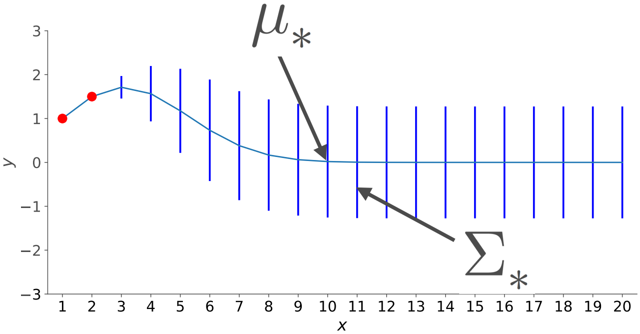




Regression using Gaussians

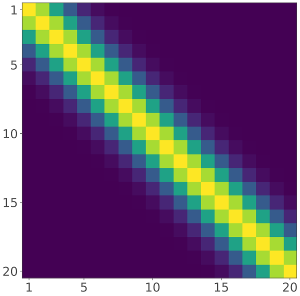
$\Sigma =$

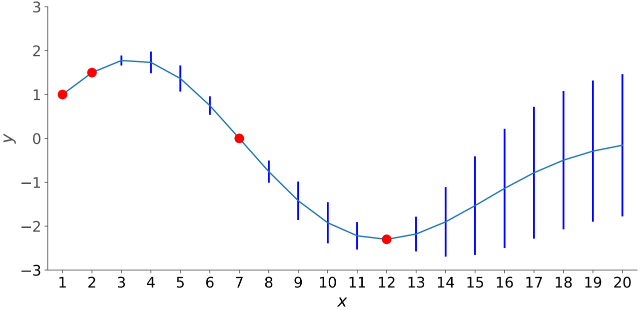




Regression using Gaussians

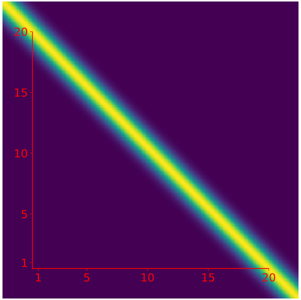
$$\Sigma =$$

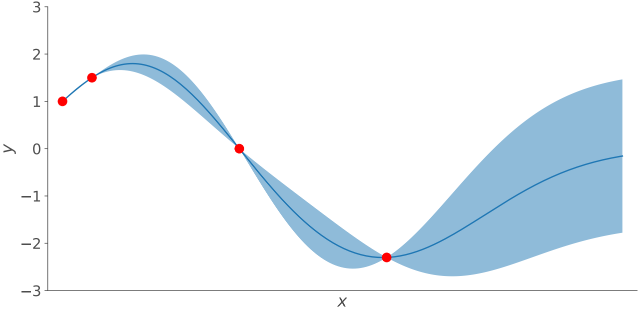




Regression using Gaussians

$\Sigma =$





Gaussian process

For any finite set of points, this process defines a joint Gaussian:

$$p(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}, \mathbf{K})$$

where $K_{ij} = K\left(\mathbf{x}_i, \mathbf{x}_j\right)$ and $\boldsymbol{\mu} = \left(m\left(\mathbf{x}_1\right), \ldots, m\left(\mathbf{x}_N\right)\right)$.

Regression using Gaussians

A Gaussian Process (GP) is denoted by:

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

where $m(\mathbf{x})$ is the **mean function** and $K(\mathbf{x}, \mathbf{x}')$ is the **kernel** or **covariance function**, i.e.,

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ \kappa(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))\,(f(\mathbf{x}') - m(\mathbf{x}'))^T\right] \end{aligned}$$

Gaussian processes in machine learning

- Gaussian processes (GPs), also known as Gaussian random fields.
- Originating in geostatistics, they were introduced by George Matheron around 1960 under the name kriging.
- Their use in machine learning has grown substantially since the 1990s.
- A Gaussian process extends the multivariate Gaussian distribution to an infinite collection of random variables indexed by input points.
- Its most common use in machine learning is probabilistic non-linear regression.
- Gaussian processes are also applied in pattern classification, dimensionality reduction, missing-data, multi-task learning, and Bayesian optimization.

Gaussian processes

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

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

Usually $f(x) \in \mathbb{R}$ and $\mathcal{X} \subset \mathbb{R}^n$

Gaussian processes

Understanding y requires only finite-dimensional distributions (FDDs):

- For any $x_1, \ldots, x_n$ and $n \in \mathbb{N}$:
  $\mathbb{P}(y(x_1) \leq c_1, \ldots, y(x_n) \leq c_n)$.
- These FDDs completely determine the law of y.

Gaussian processes

Understanding y requires only finite-dimensional distributions (FDDs):

- For any $x_1, \dots, x_n$ and $n \in \mathbb{N}$:
  $\mathbb{P}(y(x_1) \leq c_1, \dots, y(x_n) \leq c_n)$.
- These FDDs completely determine the law of y.

Gaussian process definition

A Gaussian process has Gaussian FDDs

$$(y(x_1), \dots, y(x_n)) \sim \mathcal{N}_n(\mu, \Sigma)$$

We write $y(\cdot) \sim \mathcal{GP}$ when y is a Gaussian process.

Mean and covariance function

To fully specify the law of a GP, we need to specify mean and covariance functions:

$$f(\mathbf{x}_1), \dots, f(\mathbf{x}_N) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \qquad \forall\, \mathbf{x}, \mathbf{x}' \in \mathcal{X} \subset \mathbb{R}^d$$

where

[mean function]

$$m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right]$$

[kernel covariance function]

$$\begin{aligned} k\left(\mathbf{x}, \mathbf{x}'\right) &= \text{Cov}\left[f(\mathbf{x}), f\left(\mathbf{x}'\right)\right] \\ &= \mathbb{E}\left[\left(f(\mathbf{x}) - m(\mathbf{x})\right)\left(f(\mathbf{x}') - m(\mathbf{x}')\right)\right] \end{aligned}$$

Mean function $m(\mathbf{x})$

We can use any mean function we want $m(x) = \mathbb{E}\left[f(\mathbf{x})\right]$.

Popular choices are:

- $m(\mathbf{x}) = 0$
- $m(\mathbf{x}) =$ const
- $m(x) = \phi(\mathbf{x})^\intercal \mathbf{w}$
- Neural networks.

Mean function $m(\mathbf{x})$

We can use any mean function we want $m(x) = \mathbb{E}\left[f(\mathbf{x})\right]$.

Popular choices are:

- $m(\mathbf{x}) = 0$
- $m(\mathbf{x}) = \text{const}$
- $m(x) = \phi(\mathbf{x})^\intercal \mathbf{w}$
- Neural networks.

Let $f(\mathbf{x}) = \phi(\mathbf{x})^\intercal \mathbf{w}$ with $\mathbf{w} \sim \mathcal{N}\left(\mathbf{0}, \alpha^{-1}\mathbf{I}\right)$.

Mean function $m(\mathbf{x})$

We can use any mean function we want $m(x) = \mathbb{E}\left[f(\mathbf{x})\right]$.

Popular choices are:

- $m(\mathbf{x}) = 0$
- $m(\mathbf{x}) = \text{const}$
- $m(x) = \phi(\mathbf{x})^\mathsf{T}\mathbf{w}$
- Neural networks.

Let $f(\mathbf{x}) = \phi(\mathbf{x})^\mathsf{T}\mathbf{w}$ with $\mathbf{w} \sim \mathcal{N}\left(\mathbf{0}, \alpha^{-1}\mathbf{I}\right)$.

Then $m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right]$

Mean function $m(\mathbf{x})$

We can use any mean function we want $m(x) = \mathbb{E}\left[f(\mathbf{x})\right]$.

Popular choices are:

- $m(\mathbf{x}) = 0$
- $m(\mathbf{x}) = \text{const}$
- $m(x) = \phi(\mathbf{x})^\intercal \mathbf{w}$
- Neural networks.

Let $f(\mathbf{x}) = \phi(\mathbf{x})^\intercal \mathbf{w}$ with $\mathbf{w} \sim \mathcal{N}\left(\mathbf{0}, \alpha^{-1}\mathbf{I}\right)$.

Then $m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right] = \mathbb{E}\left[\mathbf{w}^\intercal \phi(\mathbf{x})\right]$

Mean function $m(\mathbf{x})$

We can use any mean function we want $m(x) = \mathbb{E}\left[f(\mathbf{x})\right]$.

Popular choices are:

- $m(\mathbf{x}) = 0$
- $m(\mathbf{x}) =$ const
- $m(x) = \phi(\mathbf{x})^\intercal \mathbf{w}$
- Neural networks.

Let $f(\mathbf{x}) = \phi(\mathbf{x})^\intercal \mathbf{w}$ with $\mathbf{w} \sim \mathcal{N}\left(\mathbf{0}, \alpha^{-1}\mathbf{I}\right)$.

Then $m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right] = \mathbb{E}\left[\mathbf{w}^\intercal \phi(\mathbf{x})\right] = \mathbb{E}\left[\mathbf{w}^\intercal\right] \mathbb{E}\left[\phi(\mathbf{x})\right]$

Mean function $m(\mathbf{x})$

We can use any mean function we want $m(x) = \mathbb{E}\left[f(\mathbf{x})\right]$.

Popular choices are:

- $m(\mathbf{x}) = 0$
- $m(\mathbf{x}) =$ const
- $m(x) = \phi(\mathbf{x})^\intercal \mathbf{w}$
- Neural networks.

Let $f(\mathbf{x}) = \phi(\mathbf{x})^\intercal \mathbf{w}$ with $\mathbf{w} \sim \mathcal{N}\left(\mathbf{0}, \alpha^{-1}\mathbf{I}\right)$.

Then $m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right] = \mathbb{E}\left[\mathbf{w}^\intercal \phi(\mathbf{x})\right] = \mathbb{E}\left[\mathbf{w}^\intercal\right] \mathbb{E}\left[\phi(\mathbf{x})\right] = \mathbf{0}$

Kernel covariance function $k\left(\mathbf{x}, \mathbf{x}'\right)$

The covariance function determines the nature of the GP, i.e., the hypothesis space/space of functions.

We usually use a covariance function that is a function of the indexes/locations:

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \text{Cov}\left(f(\mathbf{x}), f\left(\mathbf{x}'\right)\right),$$

$k(\cdot, \cdot)$ must satisfy:

- Symmetry, i.e., $k\left(\mathbf{x}, \mathbf{x}'\right) = k\left(\mathbf{x}', \mathbf{x}\right)$.
- For any locations $\mathbf{x}_1, \ldots, \mathbf{x}_N$, the $N \times N$ Gram matrix $\mathbf{K}$ with $\mathbf{K}_{ij} = k\left(\mathbf{x}_i, \mathbf{x}_j\right)$ must be a positive semi-definite matrix.

Kernel covariance function $k\left(\mathbf{x},\mathbf{x}'\right)$

Let $f(\mathbf{x}) = \phi(\mathbf{x})^\intercal \mathbf{w}$ with $\mathbf{w} \sim \mathcal{N}\left(\mathbf{0}, \alpha^{-1}\mathbf{I}\right)$.

The kernel covariance $k(\mathbf{x},\mathbf{x}')$ is given by:

$$
\begin{aligned}
k\left(\mathbf{x},\mathbf{x}'\right) &= \mathbb{E}\left[f(\mathbf{x})f(\mathbf{x}')\right] \\
&= \phi(\mathbf{x})^\intercal \mathbb{E}\left[\mathbf{w}\mathbf{w}^\intercal\right]\phi(\mathbf{x}') \\
&= \phi(\mathbf{x})^\intercal \frac{\mathbf{I}}{\alpha}\phi(\mathbf{x}') \\
&= \frac{\phi(\mathbf{x})^\intercal\phi(\mathbf{x}')}{\alpha}
\end{aligned}
$$

Kernel covariance function $k\left(\mathbf{x}, \mathbf{x}^{'}\right)$

A widely used kernel in GPs is the squared-exponential or RBF (radial basis function) kernel is given by:

$$k\left(x, x'\right) = \sigma_f^2 \exp\left(-\frac{\left(x - x'\right)^2}{2\ell^2}\right), \qquad x \in \mathbb{R}$$

where σ_f^2 is the signal variance parameter and ℓ the length-scale parameter.

How do we draw samples from a GP?

- Given the mean function and covariance function for a GP, we can draw samples using a multivariate Gaussian distribution.
- To sample from the multivariate Gaussian distribution, we need a mean vector and a covariance matrix.
- The mean vector is obtained from the mean function.
- The covariance matrix is obtained from the covariance function.

Sampling from a GP, example

We have the set of x values:

$$\{x_1, x_2, \dots, x_N\} \subseteq \mathbb{R}$$

These are the indexes of the stochastic process.

We now compute the covariance matrix

$$\mathbf{K} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \cdots & k(x_N, x_N) \end{bmatrix}$$

We assume the mean function is constant and equal to zero, i.e., $m(x) = 0 \, \forall \, x$.

Sampling from a GP, example

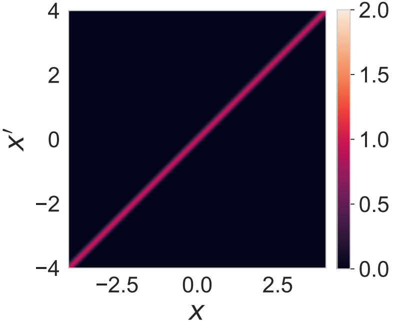
To generate functions from this GP, we will then sample from:

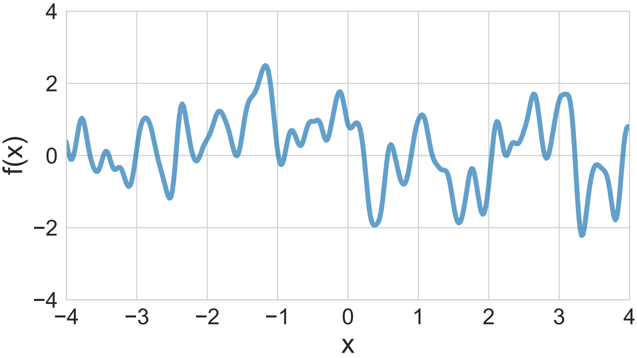
$$
f(x_1), \dots, f(x_N) \\
\sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \cdots & k(x_N, x_N) \end{bmatrix} \right)
$$

What we plot is x_i and $f(x_i)$, $\forall\, i$.

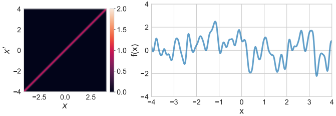
Sampling from a GP, example

$$\ell = 0.1, \sigma_f = 1$$

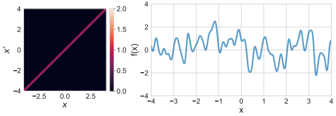




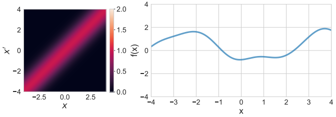
$\ell = 0.1$, $\sigma_f = 1$

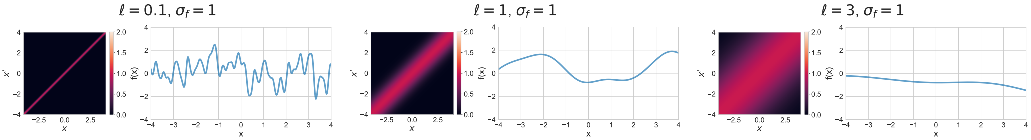


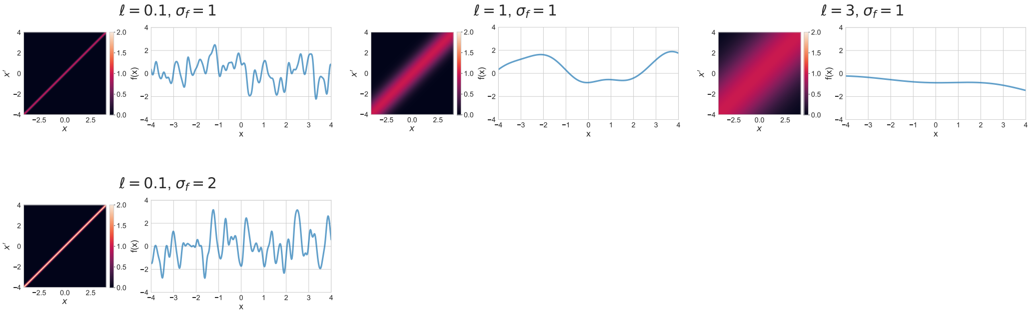
$\ell = 0.1, \sigma_f = 1$

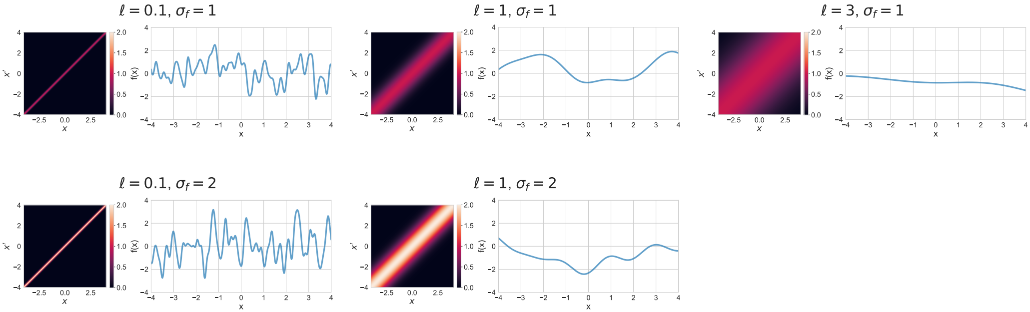


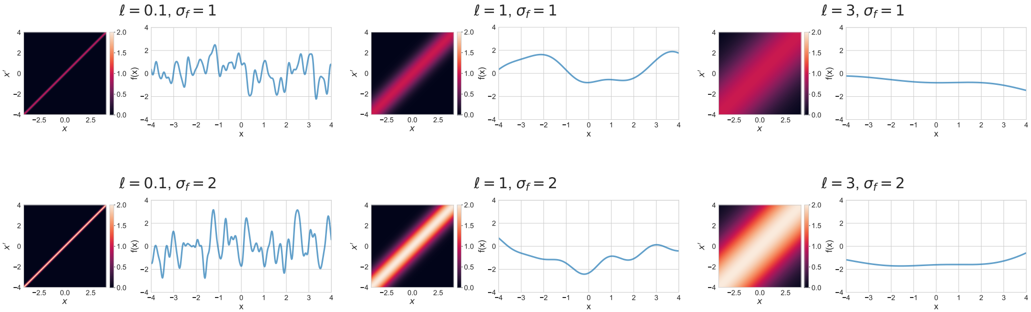
$\ell = 1, \sigma_f = 1$

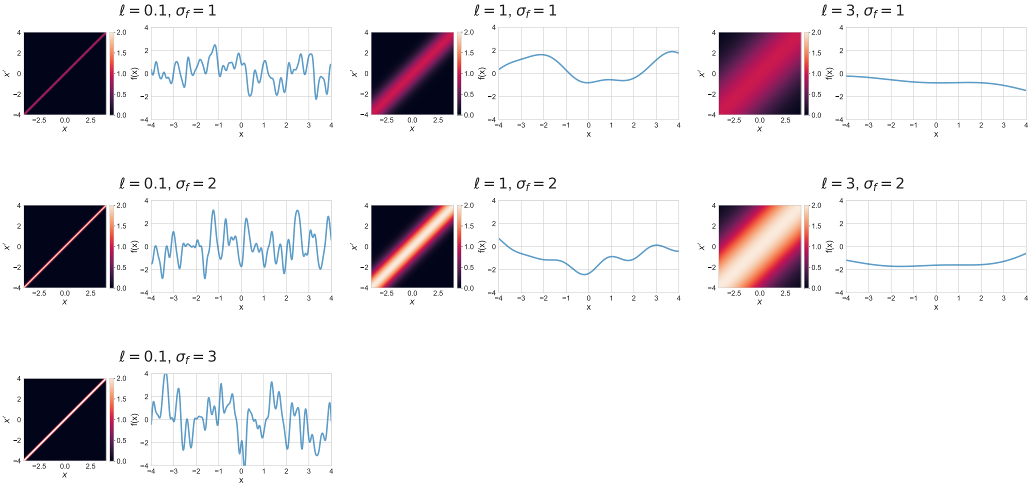



$\ell = 0.1, \sigma_f = 1$
$\ell = 1, \sigma_f = 1$
$\ell = 3, \sigma_f = 1$
x
x′
f(x)

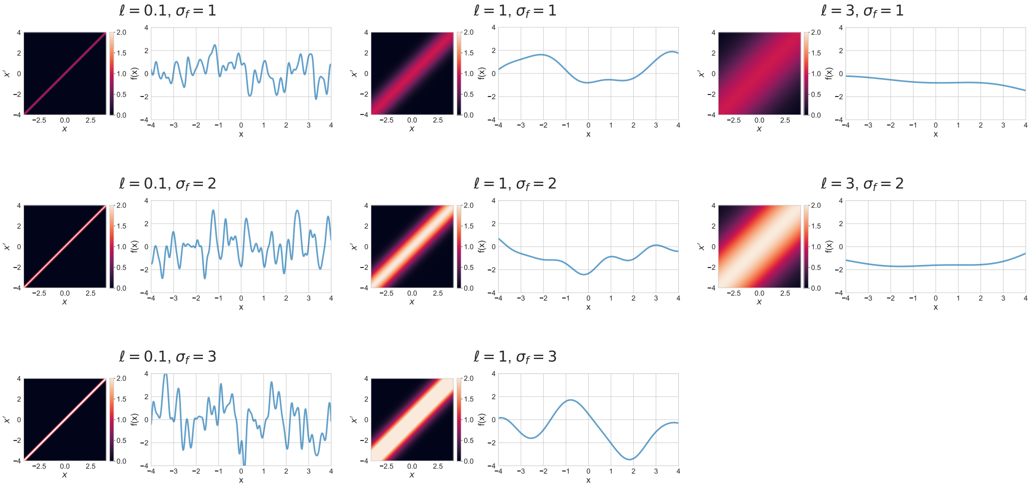

$\ell = 0.1, \sigma_f = 1$
$\ell = 1, \sigma_f = 1$
$\ell = 3, \sigma_f = 1$
$\ell = 0.1, \sigma_f = 2$
x'
x
f(x)
x

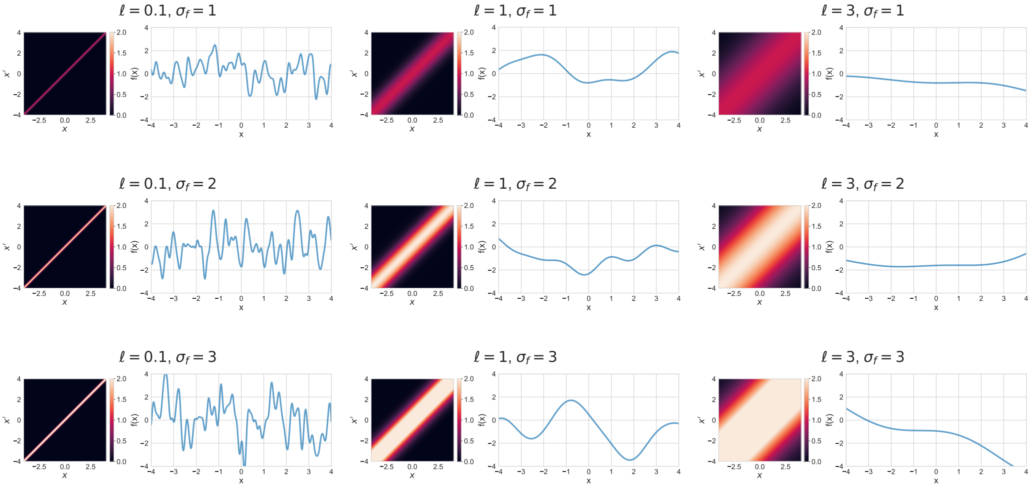

$\ell = 0.1, \sigma_f = 1$
$\ell = 1, \sigma_f = 1$
$\ell = 3, \sigma_f = 1$
$\ell = 0.1, \sigma_f = 2$
$\ell = 1, \sigma_f = 2$
x
x'
f(x)


ℓ = 0.1, σ_f = 1
ℓ = 1, σ_f = 1
ℓ = 3, σ_f = 1
ℓ = 0.1, σ_f = 2
ℓ = 1, σ_f = 2
ℓ = 3, σ_f = 2
x
x′
f(x)

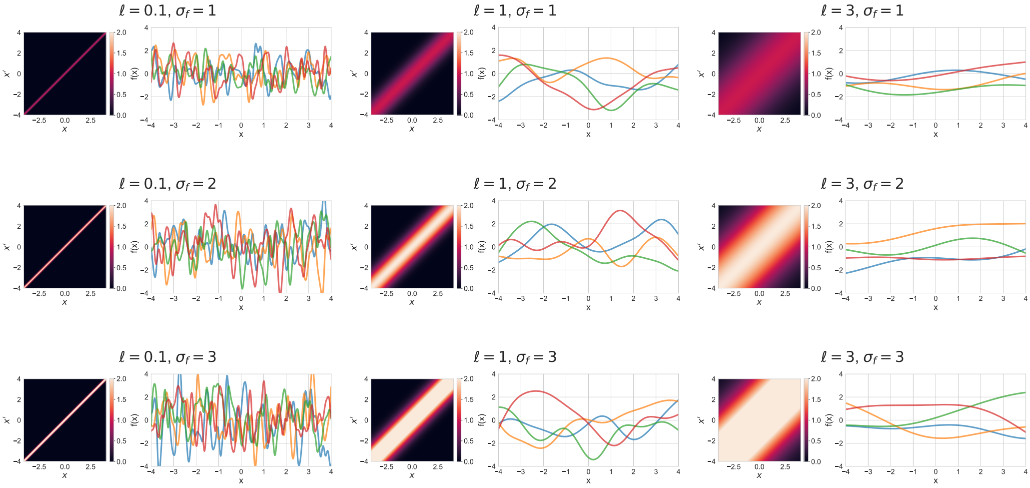


$\ell = 0.1,\ \sigma_f = 1$
$\ell = 1,\ \sigma_f = 1$
$\ell = 3,\ \sigma_f = 1$
$\ell = 0.1,\ \sigma_f = 2$
$\ell = 1,\ \sigma_f = 2$
$\ell = 3,\ \sigma_f = 2$
$\ell = 0.1,\ \sigma_f = 3$
x
x'
f(x)
x


$\ell = 0.1, \sigma_f = 1$
$\ell = 1, \sigma_f = 1$
$\ell = 3, \sigma_f = 1$
$\ell = 0.1, \sigma_f = 2$
$\ell = 1, \sigma_f = 2$
$\ell = 3, \sigma_f = 2$
$\ell = 0.1, \sigma_f = 3$
$\ell = 1, \sigma_f = 3$
x
x'
f(x)



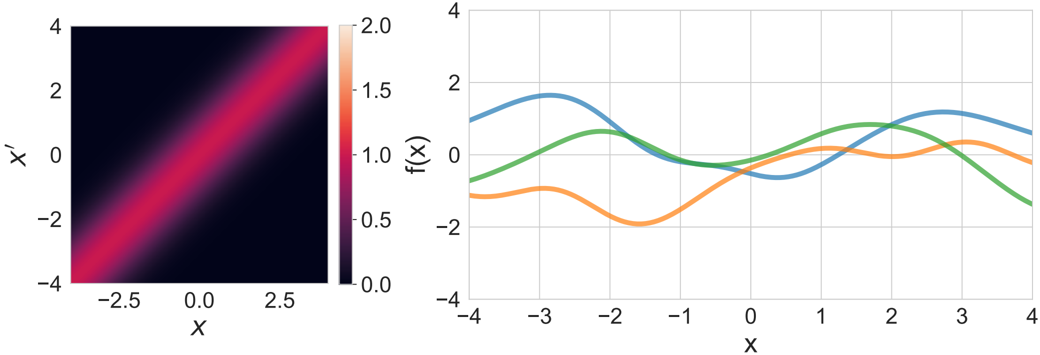
$\ell = 0.1, \sigma_f = 1$
$\ell = 1, \sigma_f = 1$
$\ell = 3, \sigma_f = 1$
$\ell = 0.1, \sigma_f = 2$
$\ell = 1, \sigma_f = 2$
$\ell = 3, \sigma_f = 2$
$\ell = 0.1, \sigma_f = 3$
$\ell = 1, \sigma_f = 3$
$\ell = 3, \sigma_f = 3$
x
x'
f(x)



$\ell = 0.1, \sigma_f = 1$
$\ell = 1, \sigma_f = 1$
$\ell = 3, \sigma_f = 1$
$\ell = 0.1, \sigma_f = 2$
$\ell = 1, \sigma_f = 2$
$\ell = 3, \sigma_f = 2$
$\ell = 0.1, \sigma_f = 3$
$\ell = 1, \sigma_f = 3$
$\ell = 3, \sigma_f = 3$

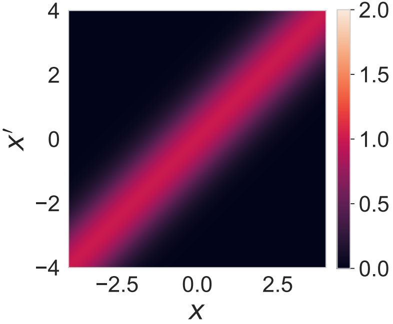
From prior to posterior

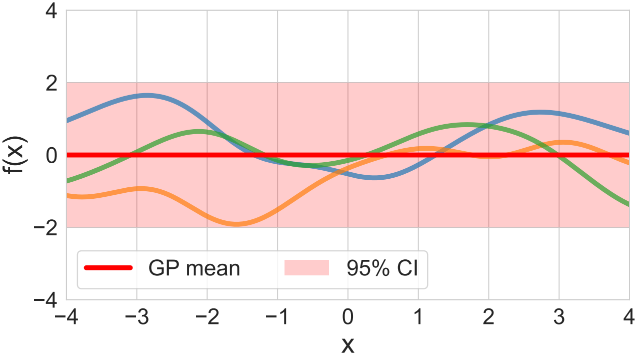
$$\ell = 1, \sigma_f = 1$$



From prior to posterior

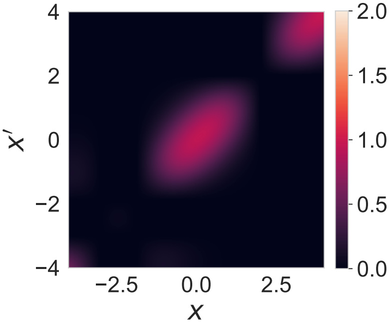
$$\ell = 1,\, \sigma_f = 1$$

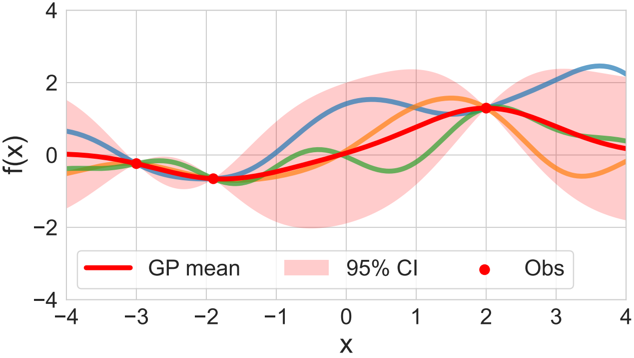




From prior to posterior

$$\ell = 1, \sigma_f = 1$$





From prior to posterior

Training set:

$$\mathcal{D} = \{(\mathbf{x}_i, f_i), i = 1 : N\}$$

Test set:

$$\{\mathbf{x}_i^*, i = 1 : N_*\}$$

The training and test sets are organized as follows:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\mathsf{T} \\ \vdots \\ \mathbf{x}_N^\mathsf{T} \end{pmatrix}$$

training inputs

We want to predict the function outputs $\mathbf{f}^*$.

From prior to posterior

Training set:

$$\mathcal{D} = \{(\mathbf{x}_i, f_i), i = 1 : N\}$$

Test set:

$$\{\mathbf{x}_i^*, i = 1 : N_*\}$$

The training and test sets are organized as follows:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\mathsf{T} \\ \vdots \\ \mathbf{x}_N^\mathsf{T} \end{pmatrix} \qquad \mathbf{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix}$$

training inputs **training targets**

We want to predict the function outputs $\mathbf{f}^*$.

From prior to posterior

Training set:

$\mathcal{D} = \{(\mathbf{x}_i, f_i), i = 1 : N\}$

Test set:

$\{\mathbf{x}_i^*, i = 1 : N_*\}$

The training and test sets are organized as follows:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\mathsf{T} \\ \vdots \\ \mathbf{x}_N^\mathsf{T} \end{pmatrix} \qquad \mathbf{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix} \qquad \mathbf{X}^* = \begin{pmatrix} \mathbf{x}_1^{*\mathsf{T}} \\ \vdots \\ \mathbf{x}_{N_*}^{*\mathsf{T}} \end{pmatrix}$$

training inputs **training targets** **test inputs**

We want to predict the function outputs $\mathbf{f}^*$.

From prior to posterior

Training set:

$\mathcal{D} = \{(\mathbf{x}_i, f_i), i = 1 : N\}$

Test set:

$\{\mathbf{x}_i^*, i = 1 : N_*\}$

The training and test sets are organized as follows:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\mathsf{T} \\ \vdots \\ \mathbf{x}_N^\mathsf{T} \end{pmatrix} \quad \mathbf{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix} \quad \mathbf{X}^* = \begin{pmatrix} \mathbf{x}_1^{*\mathsf{T}} \\ \vdots \\ \mathbf{x}_{N_*}^{*\mathsf{T}} \end{pmatrix} \quad \mathbf{f}^* = \begin{pmatrix} f_1^* \\ \vdots \\ f_{N_*}^* \end{pmatrix}$$

training inputs **training targets** **test inputs** **predictions**

We want to predict the function outputs $\mathbf{f}^*$.

From prior to posterior

The prior GP joint distribution of $\mathbf{f}$ and $\mathbf{f}^*$ is multivariate normal, i.e.:

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\mathsf{T} & \mathbf{K}_{**} \end{pmatrix} \right)$$

where:

$\mathbf{K}$ is of size $N \times N$, with $[\mathbf{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

$\mathbf{K}_*$ is of size $N \times N_*$, with $[\mathbf{K}_*]_{ij} = k(\mathbf{x}_i, \mathbf{x}_{*j})$.

$\mathbf{K}_{**}$ is of size $N_* \times N_*$, with $[\mathbf{K}_{**}]_{ij} = k(\mathbf{x}_{*i}, \mathbf{x}_{*j})$.

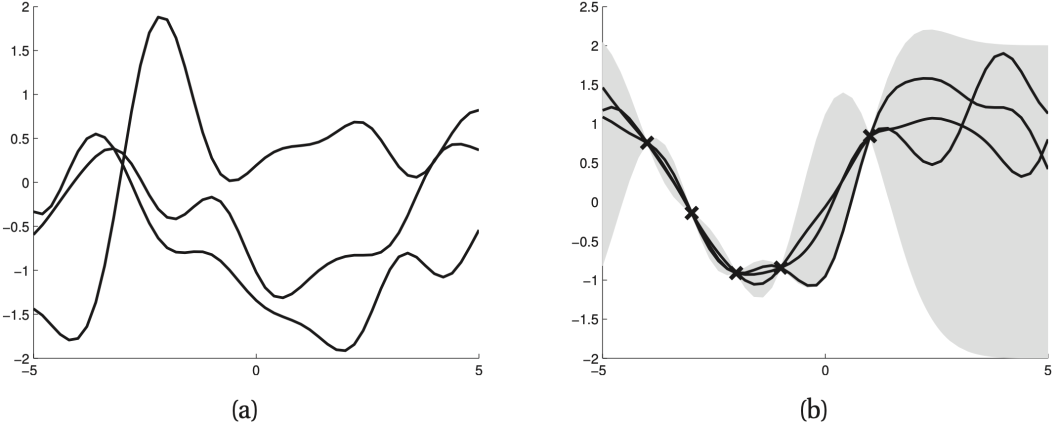
From prior to posterior

Conditioning on the training targets $\mathbf{f}$ gives the posterior GP, i.e.:

$$\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}_*, \Sigma_*)$$

The posterior has the following form:

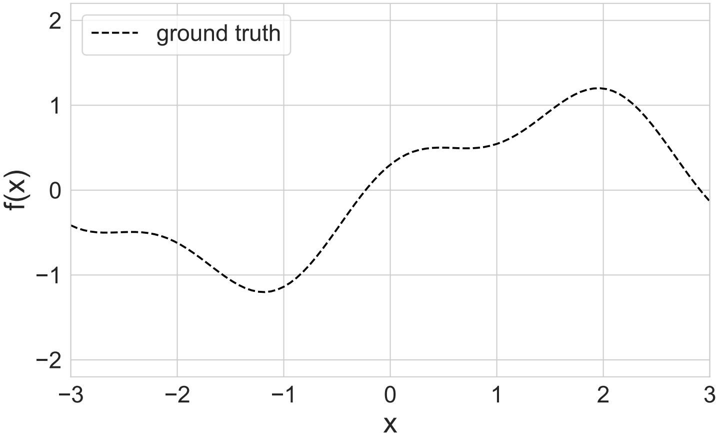
$$\boldsymbol{\mu}_* = \boldsymbol{\mu}(\mathbf{X}_*) + \mathbf{K}_*^\intercal \mathbf{K}^{-1}(\mathbf{f} - \boldsymbol{\mu})$$
$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^\intercal \mathbf{K}^{-1} \mathbf{K}_*$$



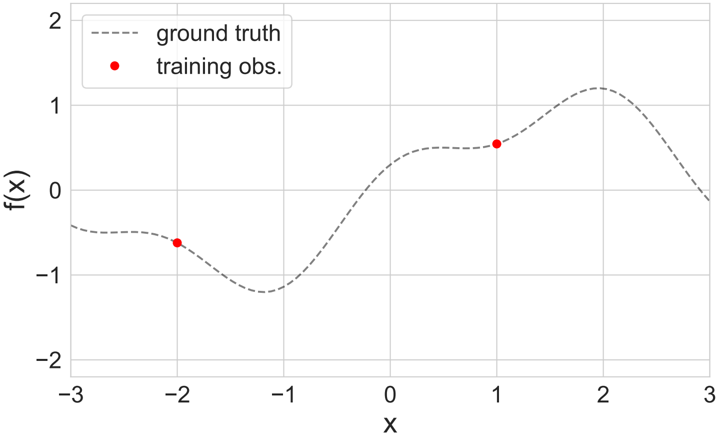
(a) Samples from the prior, $p(\mathbf{f} \mid \mathbf{X})$, using a squared exponential kernel.

(b) Samples from a GP posterior, $p\left(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{f}\right)$.

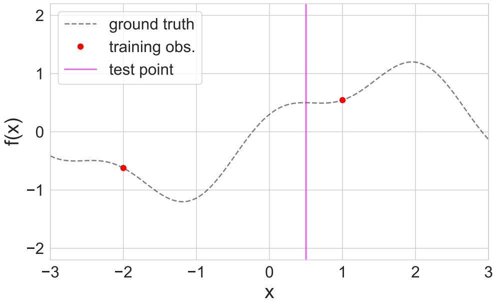
Numerical example



Numerical example



Numerical example



Training samples:

$$(x_1, f(x_1)) = (-2, -0.62)$$
$$(x_2, f(x_2)) = (1, 0.54)$$

$$x^* = 0.5$$
$$f^* = ?$$

Numerical example

Suppose f is a Gaussian process, then

$$\underbrace{f(x_1), f(x_2)}_{\text{training samples}}, \underbrace{f(x^*)}_{\text{test sample}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$$

where:

$$\boldsymbol{\Sigma} = \left(\begin{array}{cc|c} k(x_1, x_1) & k(x_1, x_2) & k(x_1, x^*) \\ k(x_2, x_1) & k(x_2, x_2) & k(x_2, x^*) \\ \hline k(x^*, x_1) & k(x^*, x_2) & k(x^*, x^*) \end{array}\right) = \left(\begin{array}{cc|c} 1 & 0.01 & 0.04 \\ 0.01 & 1 & 0.88 \\ \hline 0.04 & 0.88 & 1 \end{array}\right)$$

The value of the function f^* at the testing point $x^* = 0.5$ is computed from $p(\mathbf{f}_* \mid \mathbf{f}) = \mathcal{N}(\boldsymbol{\mu}_*, {}_*)$ with:

$$\begin{aligned} \boldsymbol{\mu}_* &= \mathbf{K}_*^\intercal \mathbf{K}^{-1}(\mathbf{f} - \boldsymbol{\mu}) \\ &= \begin{pmatrix} k(x^*, x_1) & k(x^*, x_2) \end{pmatrix} \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) \\ k(x_2, x_1) & k(x_2, x_2) \end{pmatrix}^{-1} \begin{pmatrix} f(x_1) \\ f(x_2) \end{pmatrix} \\ &= 0.46 \end{aligned}$$

Numerical example

Suppose f is a Gaussian process, then

$$\underbrace{f(x_1), f(x_2)}_{\text{training samples}}, \underbrace{f(x^*)}_{\text{test sample}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$$

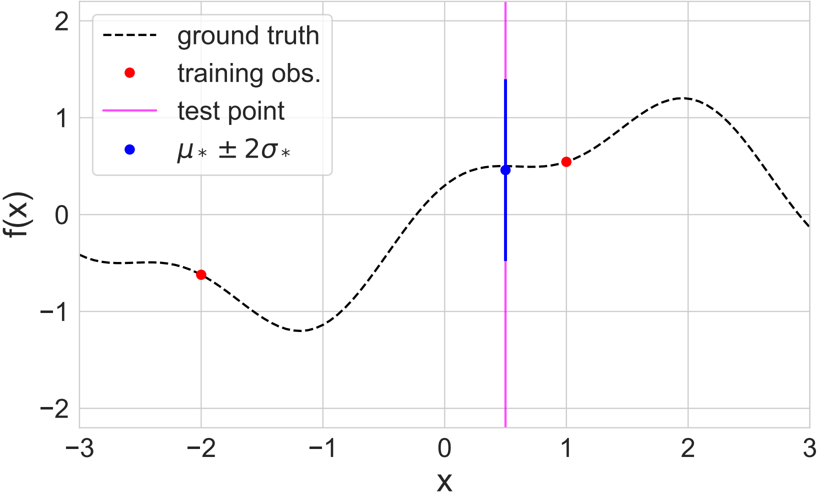
where:

$$\boldsymbol{\Sigma} = \left(\begin{array}{cc|c} k(x_1, x_1) & k(x_1, x_2) & k(x_1, x^*) \\ k(x_2, x_1) & k(x_2, x_2) & k(x_2, x^*) \\ \hline k(x^*, x_1) & k(x^*, x_2) & k(x^*, x^*) \end{array}\right) = \left(\begin{array}{cc|c} 1 & 0.01 & 0.04 \\ 0.01 & 1 & 0.88 \\ \hline 0.04 & 0.88 & 1 \end{array}\right)$$

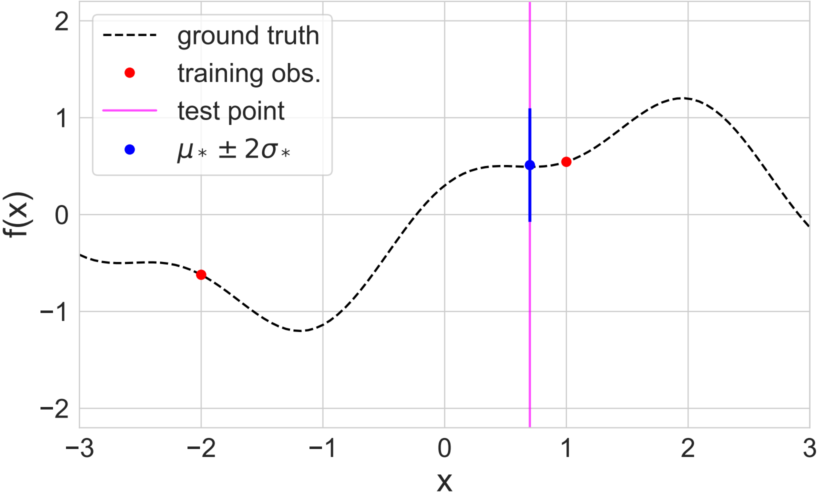
The value of the function f^* at the testing point $x^* = 0.5$ is computed from $p(\mathbf{f}_* \mid \mathbf{f}) = \mathcal{N}(\boldsymbol{\mu}_*, {}_*)$ with:

$$\begin{aligned} \boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_*^\intercal \mathbf{K}^{-1} \mathbf{K}_* \\ &= k(x^*, x^*) - \begin{pmatrix} k(x^*, x_1) & k(x^*, x_2) \end{pmatrix} \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) \\ k(x_2, x_1) & k(x_2, x_2) \end{pmatrix}^{-1} \begin{pmatrix} k(x^*, x_1) \\ k(x^*, x_2) \end{pmatrix} \\ &= 0.22 \end{aligned}$$

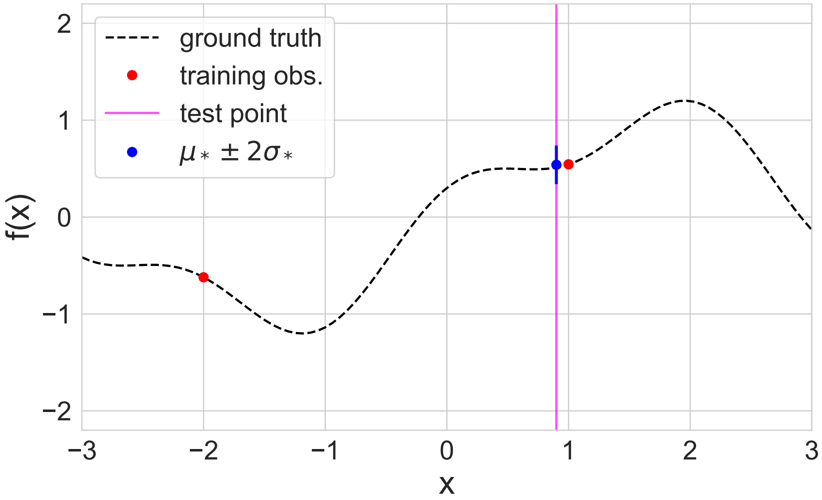
Numerical example



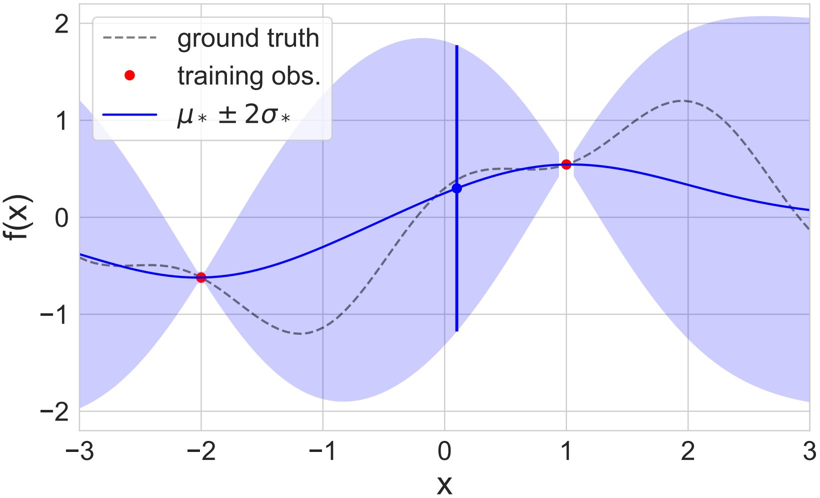
Numerical example



Numerical example



Numerical example



Predictions using noisy observations

Noisy version of the underlying function:

$$y = f(\mathbf{x}) + \epsilon, \qquad \epsilon \sim \mathcal{N}\left(0, \sigma_y^2\right)$$

The covariance of the observed noisy responses is:

$$\text{cov}\left[y_p, y_q\right] = K\left(\mathbf{x}_p, \mathbf{x}_q\right) + \sigma_y^2 \mathbb{I}(p = q)$$

$$\text{cov}[\mathbf{y} \mid \mathbf{X}] = \mathbf{K} + \sigma_y^2 \mathbf{I}_N \triangleq \mathbf{K}_y$$

Predictions using noisy observations

The joint density of training and test samples is given by:

$$y_1, y_2, \dots, y_N, f_* \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_y & \mathbf{k}_* \\ \mathbf{k}_*^\intercal & k_{**} \end{pmatrix}\right)$$

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_y & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right)$$

Predictions using noisy observations

The posterior predictive density is:

$$
\begin{aligned}
p\left(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}\right) = \mathcal{N}\left(\mathbf{f}_* \mid \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*\right) \\
\boldsymbol{\mu}_* &= \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{y} \\
\boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}_y^{-1} \mathbf{K}_*
\end{aligned}
$$

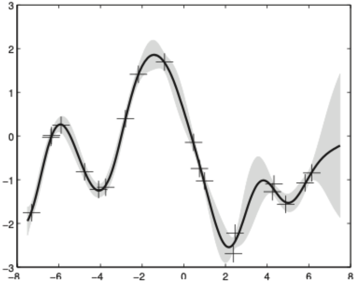
In the case of a single test input, this simplifies as follows:

$$p\left(f_* \mid \mathbf{x}_*, \mathbf{X}, \mathbf{y}\right) = \mathcal{N}\left(f_* \mid \mathbf{k}_*^T \mathbf{K}_y^{-1} \mathbf{y}, k_{**} - \mathbf{k}_*^T \mathbf{K}_y^{-1} \mathbf{k}_*\right)$$

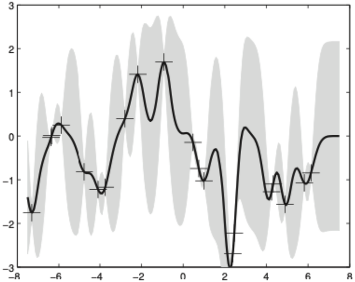
where $\mathbf{k}_* = \left[\kappa\left(\mathbf{x}_*, \mathbf{x}_1\right), \ldots, \kappa\left(\mathbf{x}_*, \mathbf{x}_N\right)\right]$ and $k_{**} = \kappa\left(\mathbf{x}_*, \mathbf{x}_*\right)$. Another way to write the posterior mean is as follows:

$$\bar{f}_* = \mathbf{k}_*^T \mathbf{K}_y^{-1} \mathbf{y} = \sum_{i=1}^{N} \alpha_i \kappa\left(\mathbf{x}_i, \mathbf{x}_*\right)$$

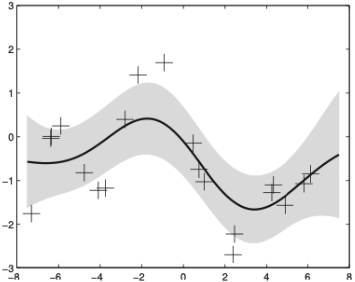
where $\boldsymbol{\alpha} = \mathbf{K}_y^{-1} \mathbf{y}$.



(a)



(b)



(c)

Which kernel functions are appropriate for a given problem?

How does the choice of different hyperparameter values affect the resulting model?

Kernels in Gaussian Processes

A **kernel** (or covariance function) $k(\mathrm{x}, \mathrm{x}')$ is the foundation of a Gaussian process. It defines the covariance structure between function values at any two input points, completely specifying the GP's prior distribution and determining:

- **Smoothness** of sample functions
- **Length-scale** (characteristic distance over which correlations decay)
- **Amplitude** (overall vertical scale of variations)
- **Periodicity** (if applicable)
- **Non-stationarity** (if applicable)

Kernels must be **positive semi-definite** to ensure valid covariance matrices.

Kernel Effects

1. **Signal Variance (σ_f^2)**

- Controls the **amplitude** or overall vertical scale of function variations.
- Larger values: function can vary over a wider range.
- Smaller values: function values stay closer to the mean.

2. **Length-Scale (ℓ)**

- Controls how quickly correlations **decay with distance.**
- **Large** ℓ: Points remain strongly correlated over longer distances. Sample functions are smooth and slowly varying.
- **Small** ℓ: Correlations decay rapidly, function can vary more rapidly and rougher behavior.

Two common kernels

RBF (squared exponential) kernel:

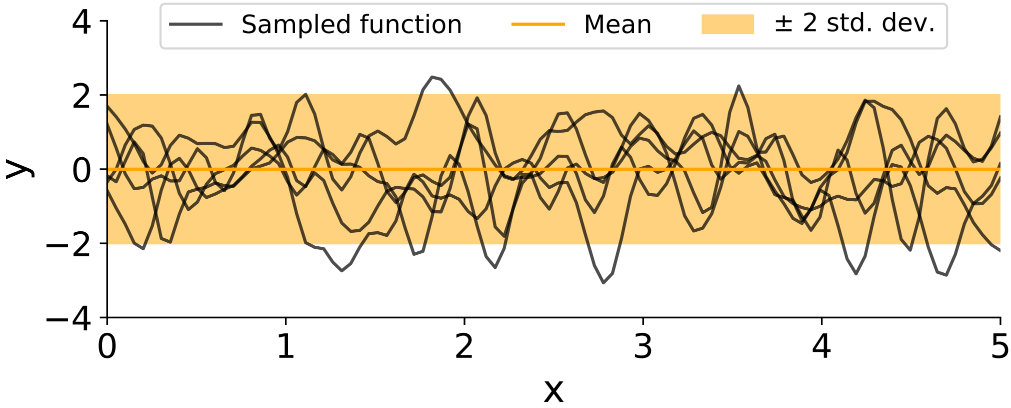
$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \, \text{exp}\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\,\ell^2}\right).$$

Matérn kernel (general $\nu > 0$) can be written as:

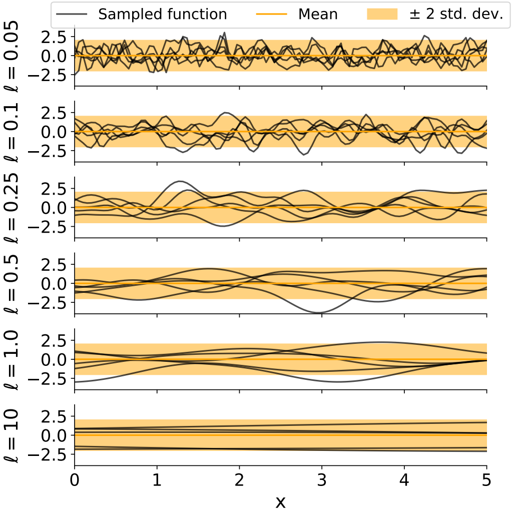
$$k_{\text{Matérn}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \frac{1}{\Gamma(\nu)\,2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{\ell}\|\mathbf{x} - \mathbf{x}'\|\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{\ell}\|\mathbf{x} - \mathbf{x}'\|\right),$$

where K_ν is the modified Bessel function of the second kind, ν controls smoothness, ℓ is the length scale, and σ_f^2 is the signal variance.

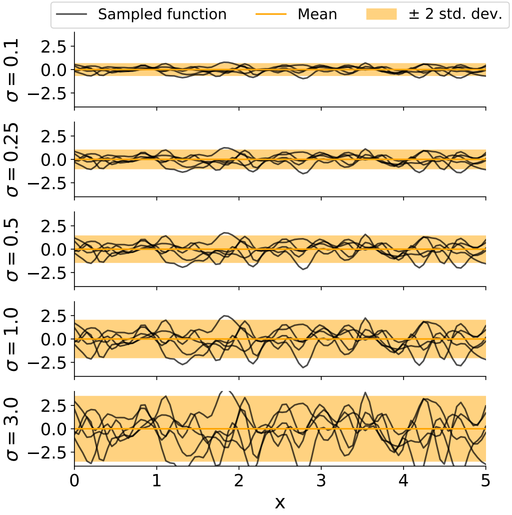
RBF kernel - samples from prior



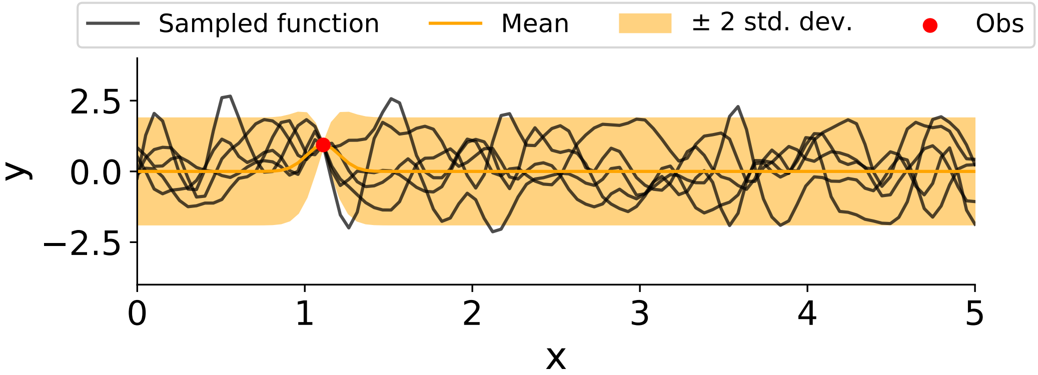
RBF kernel - samples from prior



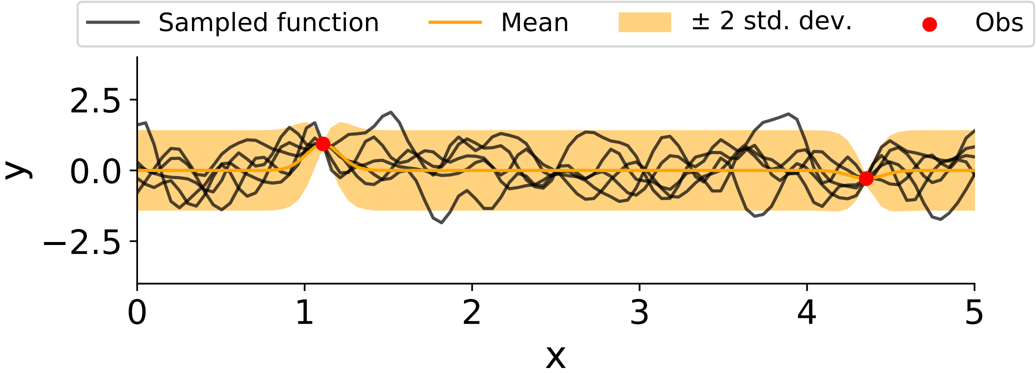
RBF kernel - samples from prior



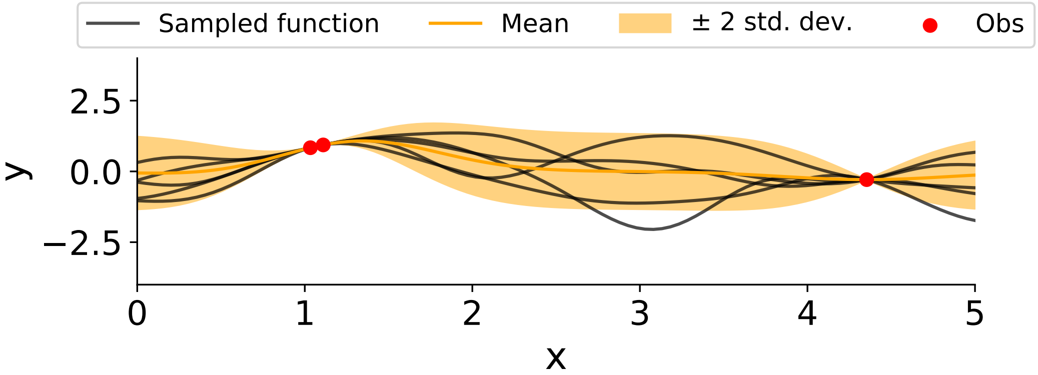
RBF kernel - samples from posterior



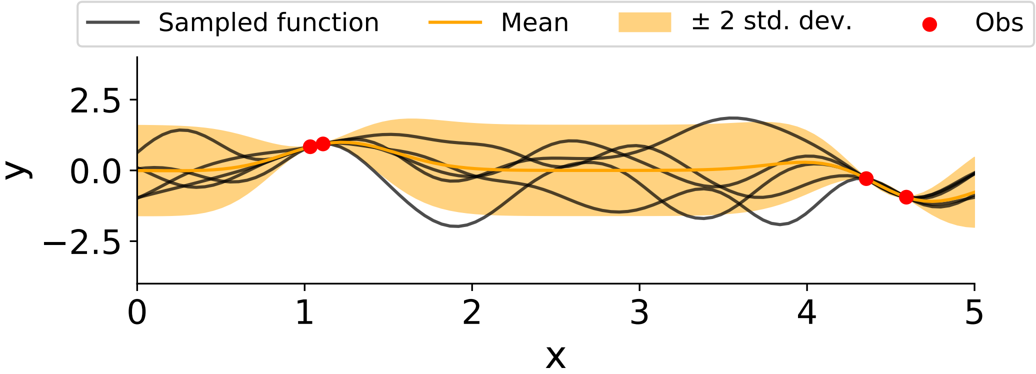
RBF kernel - samples from posterior



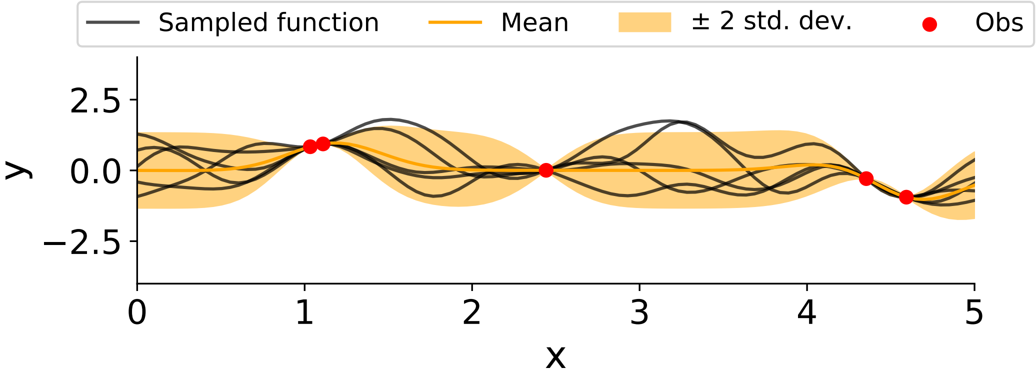
RBF kernel - samples from posterior



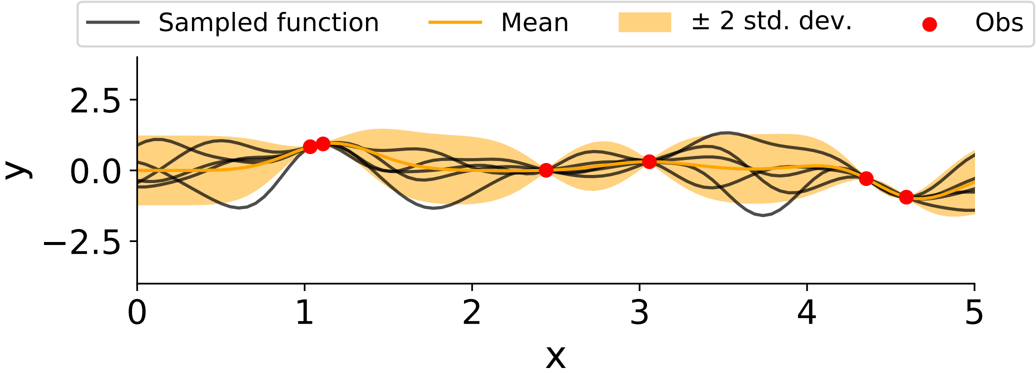
RBF kernel - samples from posterior



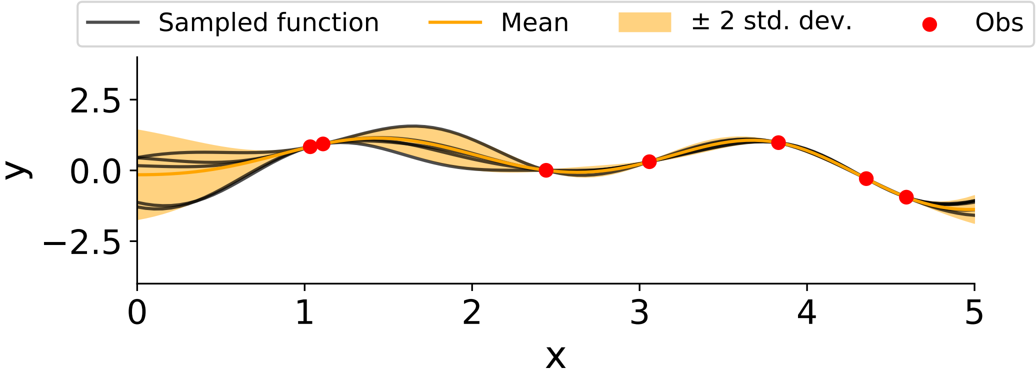
RBF kernel - samples from posterior



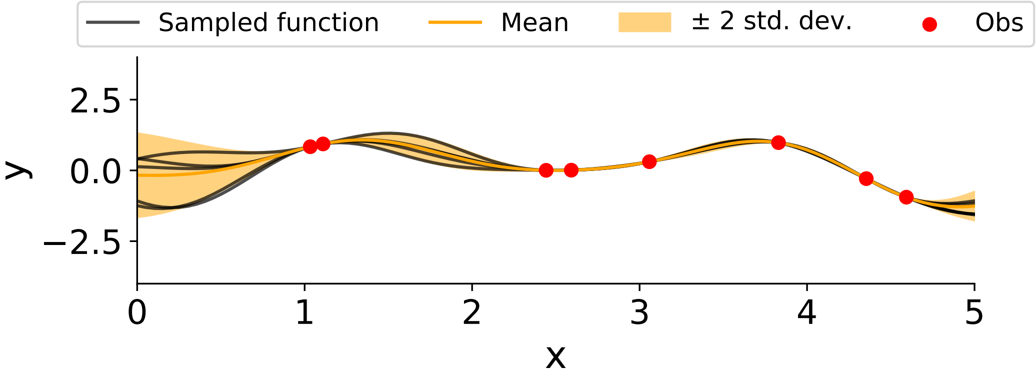
RBF kernel - samples from posterior



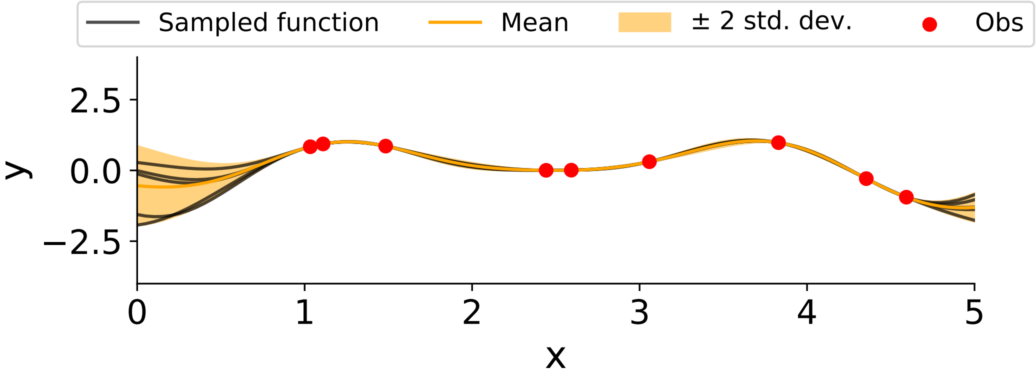
RBF kernel - samples from posterior



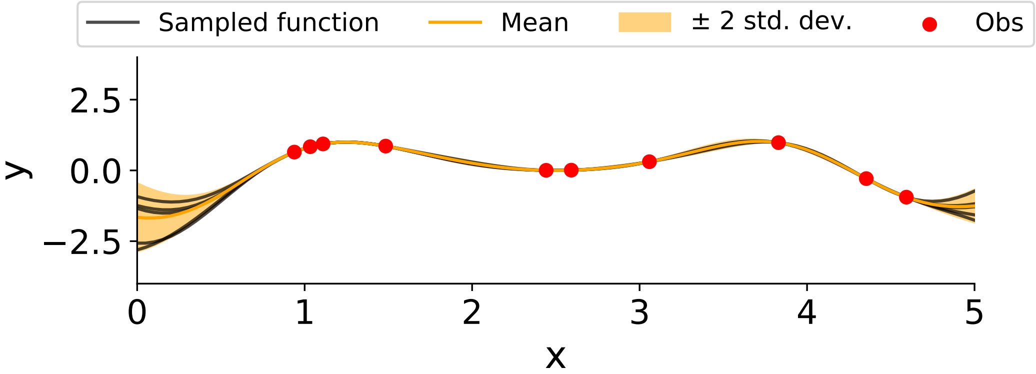
RBF kernel - samples from posterior



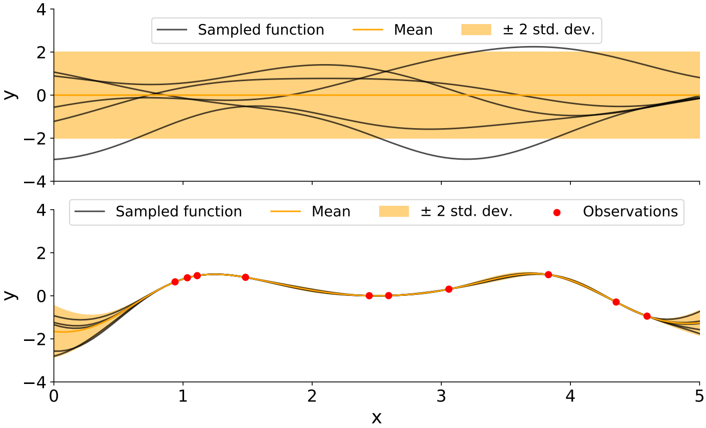
RBF kernel - samples from posterior



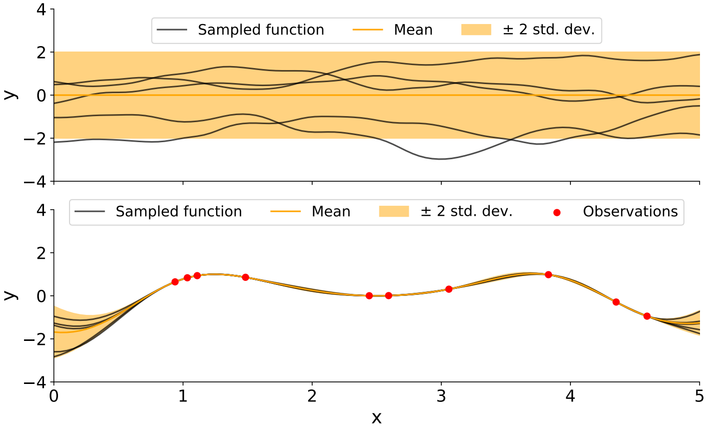
RBF kernel - samples from posterior



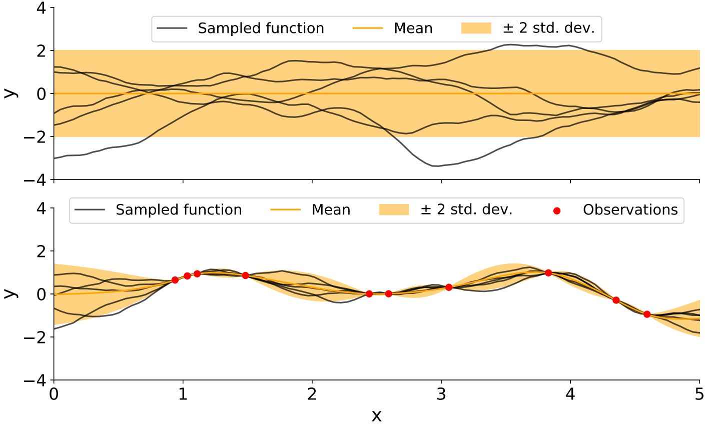
RBF kernel - prior / posterior



Rational-quadratic kernel - prior / posterior



Matern kernel - prior / posterior



3. **Stationarity**

- **Stationary kernel**: Covariance depends only on the *difference* $\mathbf{x} - \mathbf{x}'$, invariant under translation
- **Non-stationary kernel**: Covariance depends on absolute locations, allowing position-dependent behavior

4. **Isotropy**

- **Isotropic kernel**: Covariance depends only on the *Euclidean distance* $\|\mathbf{x} - \mathbf{x}'\|$
- **Anisotropic kernel**: Different characteristic length-scales and couplings across dimensions

Stationary Kernels

A stationary covariance function satisfies:

$$\text{Cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x} - \mathbf{x}') = \psi(\mathbf{x} - \mathbf{x}')$$

Translation invariant: $k(\mathbf{x} + \mathbf{a}, \mathbf{x}' + \mathbf{a}) = k(\mathbf{x}, \mathbf{x}')$ for all shifts $\mathbf{a}$.

Example: RBF kernel

Isotropic Kernels

An isotropic kernel depends only on distance:

$$k(\mathbf{x}, \mathbf{x}') = \phi(\|\mathbf{x} - \mathbf{x}'\|)$$

where $\phi : [0, \infty) \to \mathbb{R}$ is a scalar function.

Property: Every isotropic kernel is stationary, but not all stationary kernels are isotropic.

Example: RBF kernel is isotropic when $\Lambda = \ell^2 I$ (identity matrix)

Anisotropic Kernels

A stationary *but not isotropic* kernel depends on the full lag vector $\mathbf{h} = \mathbf{x} - \mathbf{x}'$, including directional information:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\top \Lambda^{-1} (\mathbf{x} - \mathbf{x}')\right)$$

where Λ is a positive definite matrix defining different characteristic length-scales and couplings across dimensions.

Use case: When different input dimensions have different "importance" or correlation decay rates.

Non-Stationary Kernels

Covariance depends on absolute locations, not just their difference. Allow position-dependent behavior and varying smoothness.

Examples: Polynomial kernel, neural network kernel

Kernel: Squared-Exponential (RBF)

1D Form:

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

Multi-dimensional isotropic form:

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

Characteristics: Infinitely differentiable (very smooth sample functions), **stationary and isotropic**, rapid decay with distance, widely used due to simplicity and smoothness.

Parameters: σ_f^2, signal variance (amplitude); ℓ, length-scale.

Kernel: Matérn

General form:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)^{\nu} K_{\nu} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)$$

where K_ν is a modified Bessel function, $\nu > 0$ controls smoothness, and ℓ is the length-scale.

Common versions: $\nu = 1/2$, exponential kernel; $\nu = 3/2$; $\nu \to \infty$, approaches RBF kernel.

Characteristics: More flexible smoothness control than RBF, includes exponential kernel as special case, stationary and isotropic.

Parameters: σ_f^2, signal variance; ℓ, length-scale; ν, smoothness parameter.

Kernel: Rational Quadratic (RQ)

Form:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\alpha\ell^2}\right)^{-\alpha}$$

Characteristics: Mixture of RBF kernels with different length-scales; can be viewed as weighted superposition of exponential kernels at different scales, **stationary and isotropic.**

Parameters: σ_f^2, signal variance, ℓ, length-scale; α, mixing parameter (larger α, closer to RBF).

Kernel: Periodic

Form:

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

Characteristics: Captures periodic patterns in data; essential for time-series with seasonal patterns.

Parameters: σ_f^2: signal variance; ℓ: length-scale; p: period (wavelength).

Kernel: Polynomial

Form:

$$k(\mathbf{x}, \mathbf{x}') = \left(\sigma_f^2 \mathbf{x} \cdot \mathbf{x}' + c\right)^d$$

or

$$k(\mathbf{x}, \mathbf{x}') = \left(\sigma_f^2 \mathbf{x} \cdot \mathbf{x}' + c\right)^d + \sigma_n^2$$

Characteristics: Often used with small d (typically 1, 2, or 3), **non-stationary kernel.**

Parameters: σ_f^2, signal variance, c, offset, d, degree (model complexity), σ_n^2, noise variance (if included).

Kernel: White Noise

Form:

$$k(x, x') = \sigma_n^2 \delta(x - x')$$

where $\delta(\cdot)$ is the Dirac delta function.

In practice (discrete inputs with finite precision):

$$k_{\text{white noise}}(\mathbf{x}, \mathbf{x}') = \begin{cases} \sigma_n^2 & \text{if } \mathbf{x} = \mathbf{x}' \\ 0 & \text{otherwise} \end{cases}$$

Kernel: White Noise

- **Pure noise**: No correlation between any two distinct points.
- **Variance only on diagonal**: Contributes σ_n^2 to the diagonal of the covariance matrix.
- **Numerical stability**: Often added to smooth kernels to stabilize matrix inversion.
- **Independent observations**: Assumes each observation contains independent measurement noise.

Kernel: White Noise

1. **Likelihood noise term**: Observations $y_i = f(\mathbf{x}_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$
2. **Jitter for numerical stability**: Added to other kernels to improve matrix conditioning:

$$k_{\text{total}} = k_{\text{smooth}}(\mathbf{x}, \mathbf{x}') + \sigma_n^2 \delta(\mathbf{x} - \mathbf{x}')$$

3. **Model white noise in data**: When you expect uncorrelated noise in observations

Kernel combinations

Kernels are closed under addition and multiplication:

- **Sum (additive)**: $k_{\text{total}} = k_1 + k_2$ (combines features)
  - Example: RBF + Periodic (smooth + periodic patterns)
  - Example: RBF + White Noise (smooth signal + independent noise)
- **Product (multiplicative)**: $k_{\text{total}} = k_1 \times k_2$ (modulates one kernel by another)
  - Example: (RBF) $\times$ (Periodic) (smooth periodic trends)

Estimating the kernel parameters

$$K_y(x_p, x_q) = \sigma_f^{\,2} \operatorname{exp}\left(-\frac{1}{2\ell^2}(x_p - x_q)^2\right) + \sigma_y^{\,2}\delta_{pq}$$

We can maximize the marginal likelihood:

$$p(\mathbf{y} \mid \mathbf{X}) = \int p(\mathbf{y} \mid \mathbf{f}, \mathbf{X})\, p(\mathbf{f} \mid \mathbf{X})\, d\mathbf{f}$$

Estimating the kernel parameters

We can maximize the marginal likelihood:

$$p(\mathbf{y} \mid \mathbf{X}) = \int p(\mathbf{y} \mid \mathbf{f}, \mathbf{X}) p(\mathbf{f} \mid \mathbf{X}) d\mathbf{f}$$

where:

$$p(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}(\mathbf{f} \mid \mathbf{0}, \mathbf{K})$$

$$p(\mathbf{y} \mid \mathbf{f}) = \prod_i \mathcal{N}\left(y_i \mid f_i, \sigma_y^2\right)$$

Estimating the kernel parameters

The marginal likelihood is given by:

$$\begin{aligned}\log p(\mathbf{y} \mid \mathbf{X}) &= \log \mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_y\right) \\ &= \underbrace{-\frac{1}{2}\mathbf{y}\mathbf{K}_y^{-1}\mathbf{y}}_{\text{data fit}} \underbrace{-\frac{1}{2}\log\left|\mathbf{K}_y\right|}_{\text{model complexity}} \underbrace{-\frac{N}{2}\log(2\pi)}_{\text{constant}}\end{aligned}$$

Estimating the kernel parameters

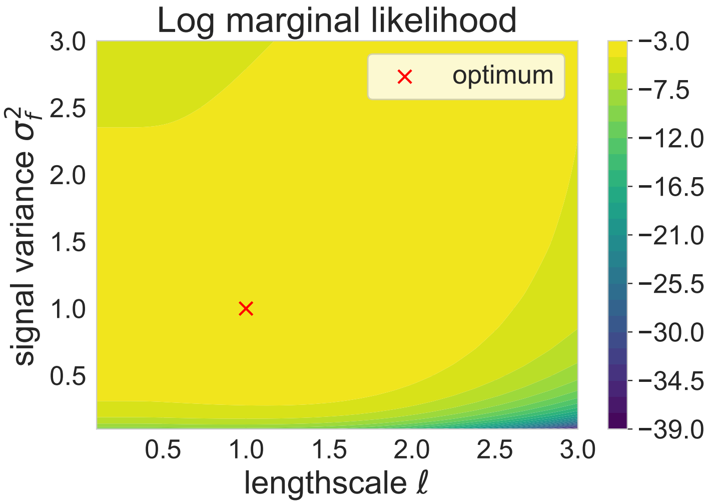
Let the kernel parameters (also called hyper-parameters) be denoted by $\boldsymbol{\theta}$.

One can show that:

$$\begin{aligned}\frac{\partial}{\partial \theta_j} \log p(\mathbf{y} \mid \mathbf{X}) &= \frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_j} \mathbf{K}_y^{-1} \mathbf{y} - \frac{1}{2} \text{tr} \left(\mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_j} \right) \\ &= \frac{1}{2} \text{tr} \left(\left(\boldsymbol{\alpha} \boldsymbol{\alpha}^T - \mathbf{K}_y^{-1} \right) \frac{\partial \mathbf{K}_y}{\partial \theta_j} \right)\end{aligned}$$

where $\boldsymbol{\alpha} = \mathbf{K}_y^{-1} \mathbf{y}$.

Log marginal likelihood



Computational cost

- One difficulty with GPs is the computational cost of training them: $O\left(n^3\right)$ (and $O\left(n^2\right)$ memory).
- They work our of the box for n in the order of a few thousands.
- There are many ways to side-step this cost: inducing inputs, efficient matrix-vector multiplications, random features, etc.
- These days we can use GPs for n in the order of tens of millions.