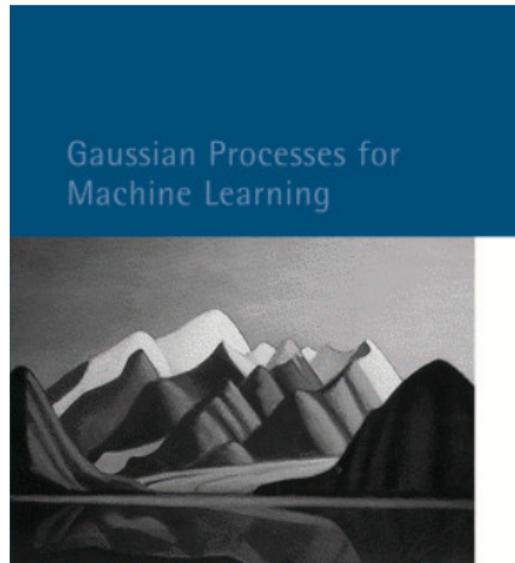


Gaussian Processes

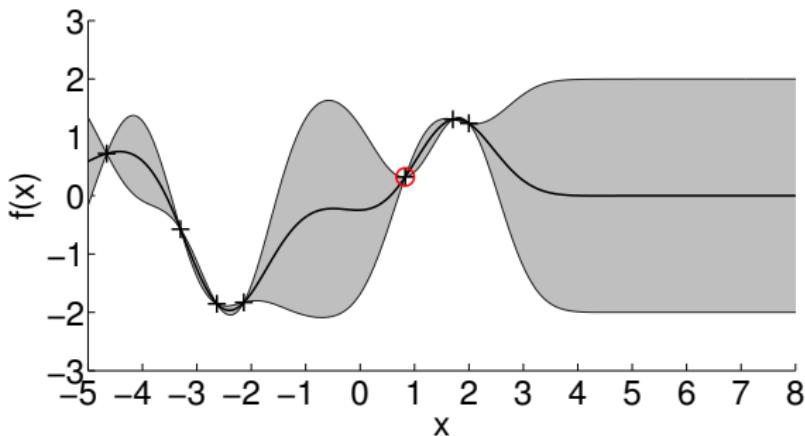
Marc Deisenroth
Centre for Artificial Intelligence
Department of Computer Science
University College London

 @mpd37
m.deisenroth@ucl.ac.uk
<https://deisenroth.cc>



Carl Edward Rasmussen and Christopher K. I. Williams

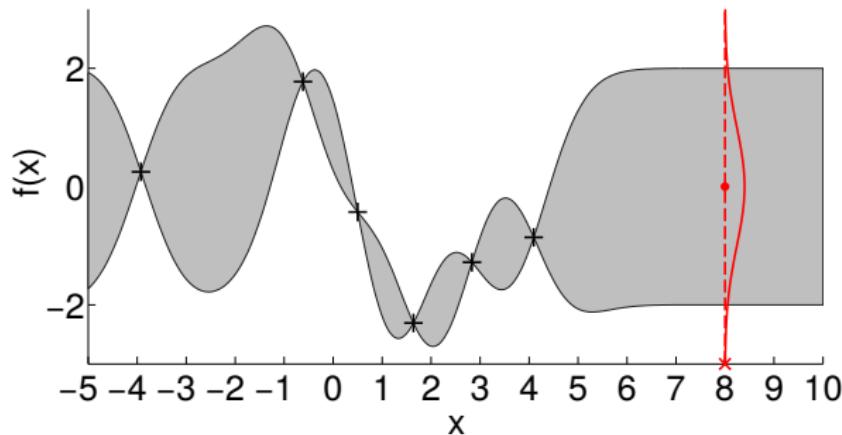
<http://www.gaussianprocess.org/>



Objective

For a set of observations $y_i = f(\mathbf{x}_i) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$, find a distribution over functions $p(f)$ that explains the data

- ▶ Probabilistic regression problem

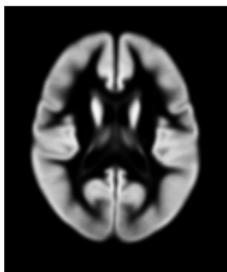
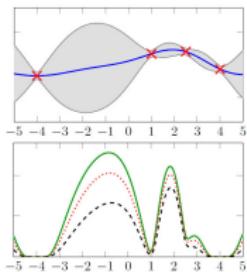
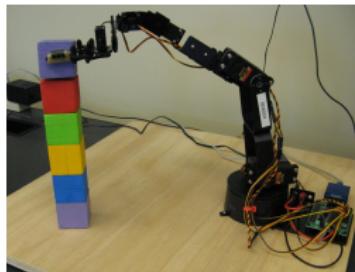


Objective

For a set of observations $y_i = f(\mathbf{x}_i) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$, find a distribution over functions $p(f)$ that explains the data

- ▶ Probabilistic regression problem

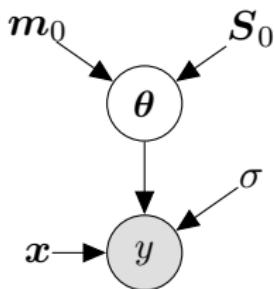
Some Application Areas



- Reinforcement learning and robotics
- Bayesian optimization (experimental design)
- Geostatistics
- Sensor networks
- Time-series modeling and forecasting
- High-energy physics
- Medical applications

$$\text{Prior} \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{S}_0)$$

$$\begin{aligned} \text{Likelihood} \quad & p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y | \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta}, \sigma_n^2) \\ & \implies y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2) \end{aligned}$$

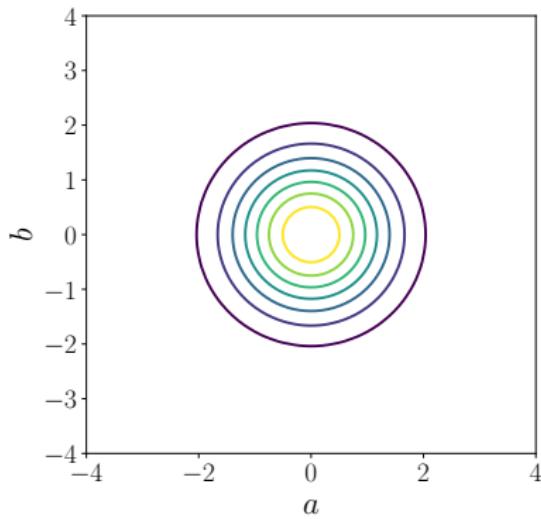


- Parameter $\boldsymbol{\theta}$ becomes a latent (random) variable
- Distribution $p(\boldsymbol{\theta})$ induces a **distribution over plausible functions**
- Choose a conjugate Gaussian prior
 - Gaussian posterior $p(\boldsymbol{\theta}|X, y) = \mathcal{N}(\boldsymbol{\theta} | \boldsymbol{m}_N, \boldsymbol{S}_N)$
 - Closed-form computations (e.g., predictions, marginal likelihood)

Distribution over Functions

Consider a linear regression setting

$$y = a + bx + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$
$$p(a, b) = \mathcal{N}(\mathbf{0}, \mathbf{I})$$



Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

$$p(a, b) = \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$f_i(x) = a_i + b_i x, \quad [a_i, b_i] \sim p(a, b)$$

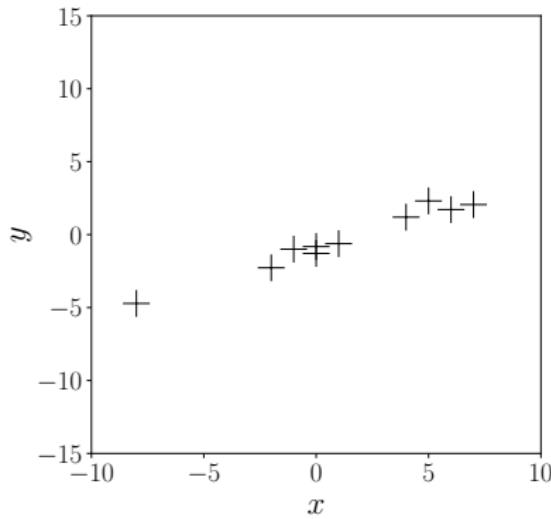
Sampling from the Posterior over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

$$p(a, b) = \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$\mathbf{X} = [x_1, \dots, x_N], \mathbf{y} = [y_1, \dots, y_N]$ Training data



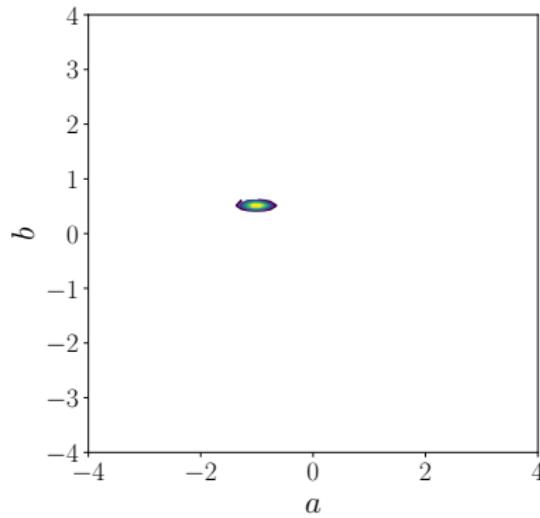
Sampling from the Posterior over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

$$p(a, b) = \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$p(a, b | \mathbf{X}, \mathbf{y}) = \mathcal{N}(\mathbf{m}_N, \mathbf{S}_N) \quad \text{Posterior}$$



Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

$$[a_i, b_i] \sim p(a, b | \mathbf{X}, \mathbf{y})$$

$$f_i = a_i + b_i x$$

- Instead of sampling parameters, which induce a distribution over functions, **sample functions directly**
 - ▶ Place a prior on functions
 - ▶ Make assumptions on the distribution of functions

- Instead of sampling parameters, which induce a distribution over functions, **sample functions directly**
 - ▶ Place a prior on functions
 - ▶ Make assumptions on the distribution of functions
- Intuition: function = infinitely long vector of function values
 - ▶ Make assumptions on the distribution of function values

- Instead of sampling parameters, which induce a distribution over functions, **sample functions directly**
 - ▶ Place a prior on functions
 - ▶ Make assumptions on the distribution of functions
- Intuition: function = infinitely long vector of function values
 - ▶ Make assumptions on the distribution of function values
 - ▶ **Gaussian process**

Overview

- 1 Gaussian Process: Definition**
- 2 Regression as Inference**
 - GP Prior
 - Likelihood
 - Marginal Likelihood
 - Posterior
 - Predictions
- 3 Model Selection**
 - GP Training
 - Hyper-Parameters
 - Inspection of the Marginal Likelihood
 - Covariance Function
- 4 Limitations and Guidelines**
- 5 Application Areas**

Gaussian Process: Definition

- We will place a distribution $p(f)$ on functions f
- Informally, a function can be considered an infinitely long vector of function values $f = [f_1, f_2, f_3, \dots]$

- We will place a distribution $p(f)$ on functions f
- Informally, a function can be considered an infinitely long vector of function values $f = [f_1, f_2, f_3, \dots]$
- A Gaussian process is a generalization of a multivariate Gaussian distribution to infinitely many variables.

- We will place a distribution $p(f)$ on functions f
- Informally, a function can be considered an infinitely long vector of function values $f = [f_1, f_2, f_3, \dots]$
- A Gaussian process is a generalization of a multivariate Gaussian distribution to infinitely many variables.

Definition (Rasmussen & Williams, 2006)

A **Gaussian process** (GP) is a collection of random variables f_1, f_2, \dots , any finite number of which is Gaussian distributed.

- We will place a distribution $p(f)$ on functions f
- Informally, a function can be considered an infinitely long vector of function values $f = [f_1, f_2, f_3, \dots]$
- A Gaussian process is a generalization of a multivariate Gaussian distribution to infinitely many variables.

Definition (Rasmussen & Williams, 2006)

A **Gaussian process** (GP) is a collection of random variables f_1, f_2, \dots , any finite number of which is Gaussian distributed.

- A Gaussian distribution is specified by a mean vector μ and a covariance matrix Σ
- A Gaussian process is specified by a **mean function** $m(\cdot)$ and a **covariance function (kernel)** $k(\cdot, \cdot)$ ► More on this later

Regression as Inference

Objective

For a set of observations $y_i = f(\mathbf{x}_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, find a (posterior) **distribution over functions** $p(f(\cdot)|\mathbf{X}, \mathbf{y})$ that explains the data. Here: \mathbf{X} training inputs, \mathbf{y} training targets

Objective

For a set of observations $y_i = f(\mathbf{x}_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, find a (posterior) distribution over functions $p(f(\cdot)|\mathbf{X}, \mathbf{y})$ that explains the data. Here: \mathbf{X} training inputs, \mathbf{y} training targets

Training data: \mathbf{X}, \mathbf{y} . Bayes' theorem yields

$$p(f(\cdot)|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y}|\mathbf{X})}$$

Objective

For a set of observations $y_i = f(\mathbf{x}_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, find a (posterior) distribution over functions $p(f(\cdot)|\mathbf{X}, \mathbf{y})$ that explains the data. Here: \mathbf{X} training inputs, \mathbf{y} training targets

Training data: \mathbf{X}, \mathbf{y} . Bayes' theorem yields

$$p(f(\cdot)|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y}|\mathbf{X})}$$

Prior: $p(f(\cdot)) = GP(m, k)$ ➤ Specify mean m function and kernel k .

Objective

For a set of observations $y_i = f(\mathbf{x}_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, find a (posterior) distribution over functions $p(f(\cdot)|\mathbf{X}, \mathbf{y})$ that explains the data. Here: \mathbf{X} training inputs, \mathbf{y} training targets

Training data: \mathbf{X}, \mathbf{y} . Bayes' theorem yields

$$p(f(\cdot)|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y}|\mathbf{X})}$$

Prior: $p(f(\cdot)) = GP(m, k)$ ➤ Specify mean m function and kernel k .

Likelihood (noise model): $p(\mathbf{y}|f(\mathbf{X})) = \mathcal{N}(f(\mathbf{X}), \sigma_n^2 \mathbf{I})$

Objective

For a set of observations $y_i = f(\mathbf{x}_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, find a (posterior) distribution over functions $p(f(\cdot)|\mathbf{X}, \mathbf{y})$ that explains the data. Here: \mathbf{X} training inputs, \mathbf{y} training targets

Training data: \mathbf{X}, \mathbf{y} . Bayes' theorem yields

$$p(f(\cdot)|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y}|\mathbf{X})}$$

Prior: $p(f(\cdot)) = GP(m, k)$ ➤ Specify mean m function and kernel k .

Likelihood (noise model): $p(\mathbf{y}|f(\mathbf{X})) = \mathcal{N}(f(\mathbf{X}), \sigma_n^2 \mathbf{I})$

Marginal likelihood (evidence): $p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|f(\cdot), \mathbf{X}) p(f(\cdot)|\mathbf{X}) df$

Objective

For a set of observations $y_i = f(\mathbf{x}_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, find a (posterior) distribution over functions $p(f(\cdot)|\mathbf{X}, \mathbf{y})$ that explains the data. Here: \mathbf{X} training inputs, \mathbf{y} training targets

Training data: \mathbf{X}, \mathbf{y} . Bayes' theorem yields

$$p(f(\cdot)|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y}|\mathbf{X})}$$

Prior: $p(f(\cdot)) = GP(m, k)$ ➤ Specify mean m function and kernel k .

Likelihood (noise model): $p(\mathbf{y}|f(\mathbf{X})) = \mathcal{N}(f(\mathbf{X}), \sigma_n^2 \mathbf{I})$

Marginal likelihood (evidence): $p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|f(\cdot), \mathbf{X}) p(f(\cdot)|\mathbf{X}) df$

Posterior: $p(f(\cdot)|\mathbf{y}, \mathbf{X}) = GP(m_{\text{post}}, k_{\text{post}})$

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Bayesian linear regression:

- Prior $p(\boldsymbol{\theta})$ on the parameters $\boldsymbol{\theta}$ allows us to encode some properties of the parameters (e.g., range, reasonable values, ...)
- Every sample $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta})$ induces a function $f_i(\cdot) := \boldsymbol{\theta}_i^\top \boldsymbol{\phi}(\cdot)$

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Bayesian linear regression:

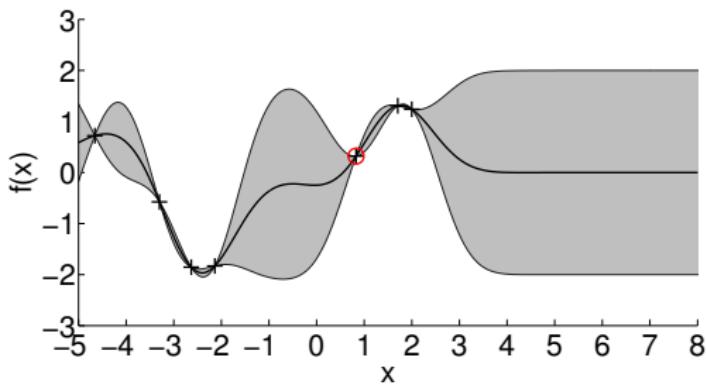
- Prior $p(\boldsymbol{\theta})$ on the parameters $\boldsymbol{\theta}$ allows us to encode some properties of the parameters (e.g., range, reasonable values, ...)
- Every sample $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta})$ induces a function $f_i(\cdot) := \boldsymbol{\theta}_i^\top \phi(\cdot)$

Gaussian process:

- GP prior: $p(f(\cdot))$
- Function plays the role of the parameters
 - ▶ Every sample $f_i(\cdot) \sim GP$ is a function

- Bayesian prior specifies assumptions on the quantity of interest
- What assumptions could we make on the underlying function?
- What characterizes the function we want to model?

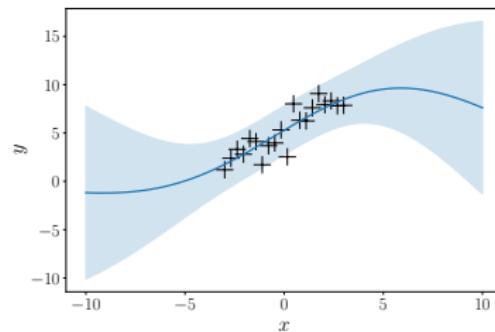
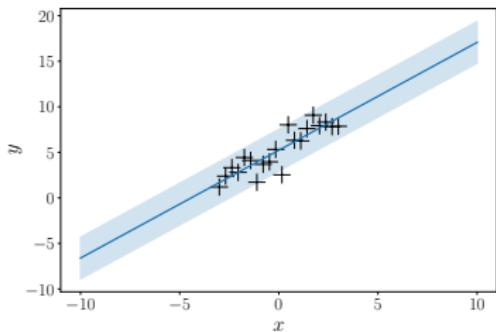
- Bayesian prior specifies assumptions on the quantity of interest
- What assumptions could we make on the underlying function?
- What characterizes the function we want to model?
 - Mean function
 - Covariance function



$$m(\mathbf{x}) = \mathbb{E}_f[f(\mathbf{x})], \quad f \sim GP$$

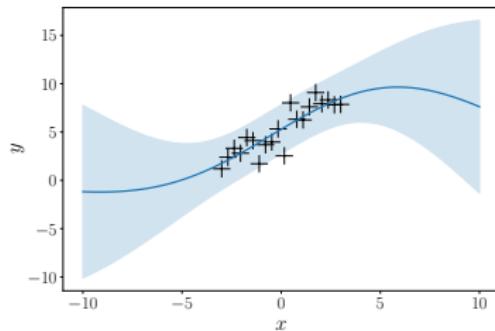
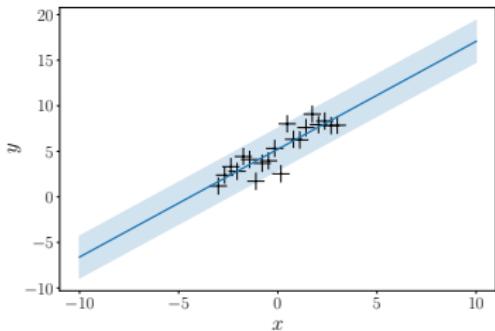
- The **average function** of the distribution over functions
- Allows us to **bias the model** (can make sense in application-specific settings)

Mean Function (2)



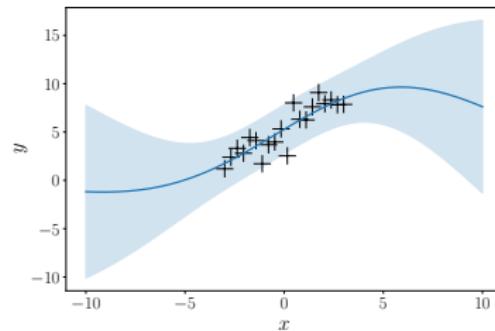
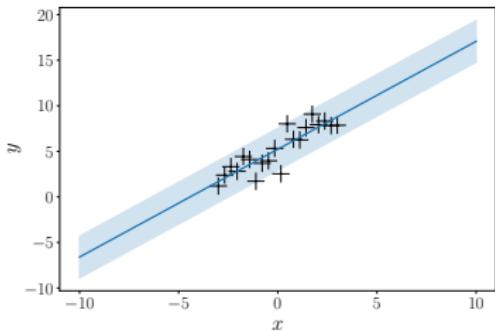
- Can be a parametrized function, e.g., linear, exponential, or neural network. Example: $m_\theta(x) = \theta^\top \phi(x)$

Mean Function (2)



- Can be a parametrized function, e.g., linear, exponential, or neural network. Example: $m_\theta(x) = \theta^\top \phi(x)$
- Prior mean function m_θ can incorporate **problem-specific prior knowledge** (e.g., in robotics, natural sciences)
- Can simplify the learning problem

Mean Function (2)



- Can be a parametrized function, e.g., linear, exponential, or neural network. Example: $m_\theta(x) = \theta^\top \phi(x)$
- Prior mean function m_θ can incorporate **problem-specific prior knowledge** (e.g., in robotics, natural sciences)
- Can simplify the learning problem
- Often: “Agnostic” mean function in the absence of data or prior knowledge: $m(\cdot) \equiv 0$ everywhere (for symmetry reasons)

- Covariance function (kernel) is symmetric and positive semi-definite

- Covariance function (kernel) is symmetric and positive semi-definite
- Compute covariances/correlations between (unknown) function values by just looking at the corresponding inputs:

$$\text{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- ▶ Kernel trick (Schölkopf & Smola, 2002)

- Covariance function (kernel) is symmetric and positive semi-definite
- Compute covariances/correlations between (unknown) function values by just looking at the corresponding inputs:

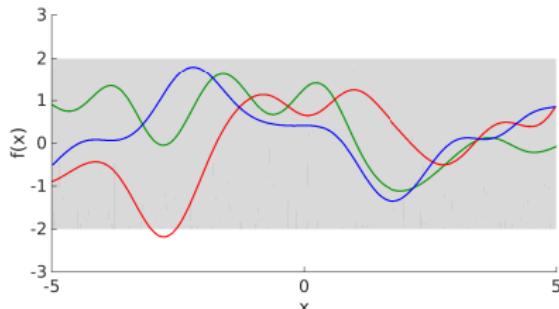
$$\text{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- ▶ Kernel trick (Schölkopf & Smola, 2002)
- Encodes high-level structural assumptions (e.g., smoothness, periodicity) of the function we want to model

Gaussian Covariance Function

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$

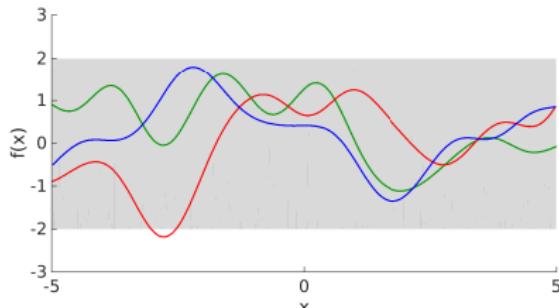
- Assumption on latent function: Smooth (∞ differentiable)



Gaussian Covariance Function

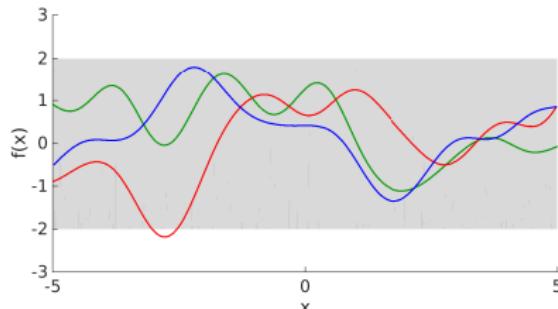
$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$

- Assumption on latent function: Smooth (∞ differentiable)
- σ_f : Amplitude of the latent function

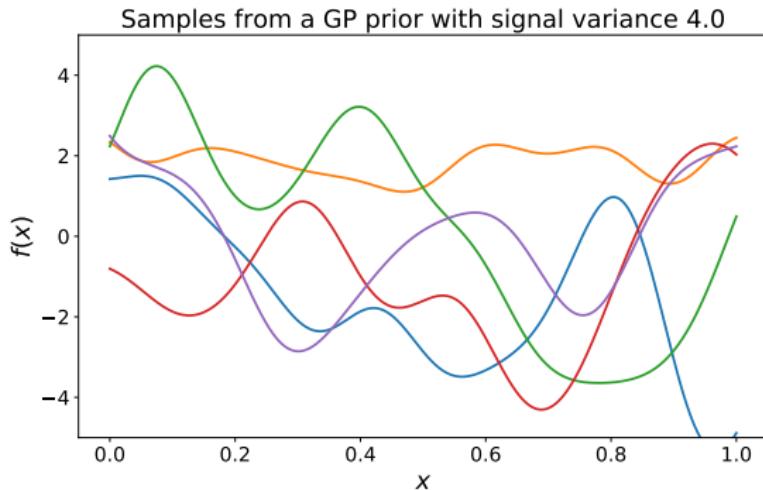


$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$

- Assumption on latent function: Smooth (∞ differentiable)
- σ_f : Amplitude of the latent function
- ℓ : Length-scale. How far do we have to move in input space before the function value changes significantly, i.e., when do function values become uncorrelated?
 - Smoothness parameter

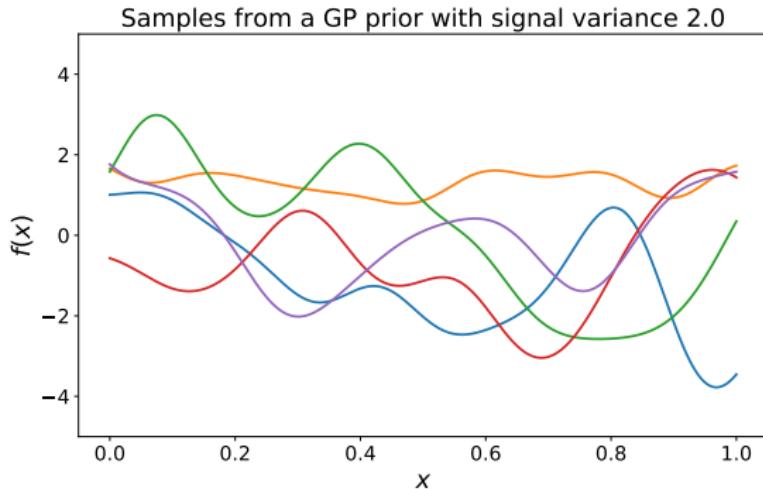


$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$



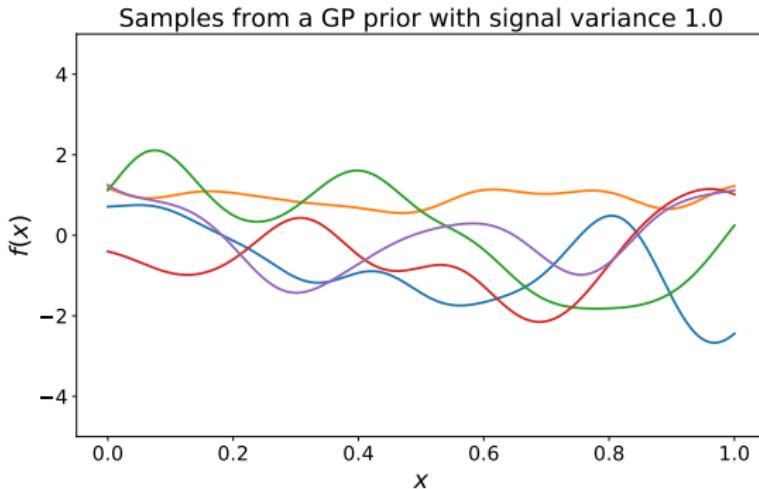
- Controls the amplitude (vertical magnitude) of the function we wish to model

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$



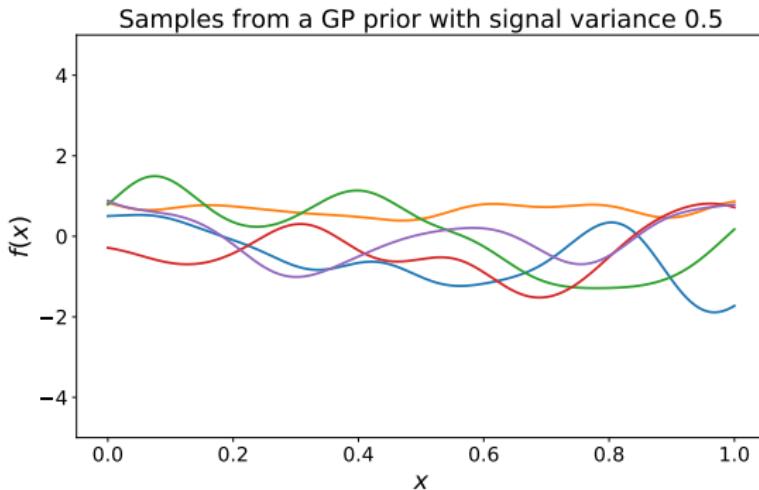
- Controls the amplitude (vertical magnitude) of the function we wish to model

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$



- Controls the amplitude (vertical magnitude) of the function we wish to model

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$

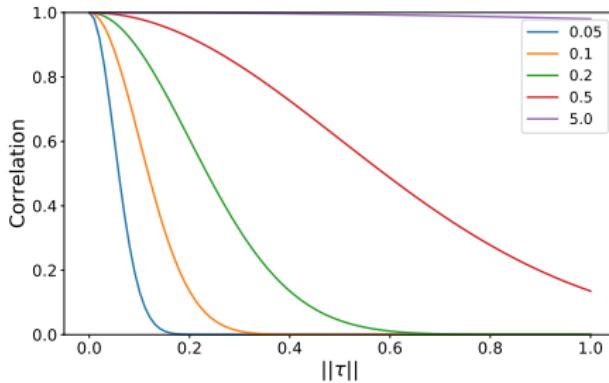


- Controls the amplitude (vertical magnitude) of the function we wish to model

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$

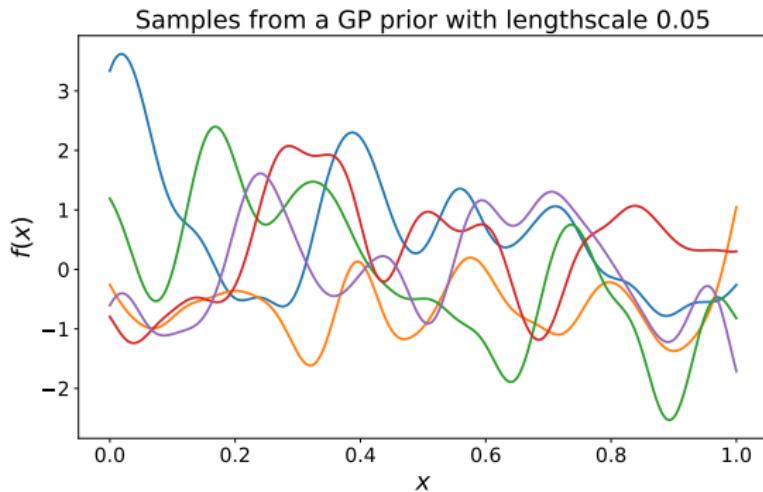
- How “wiggly” is the function?
- How much information we can transfer to other function values?
 - ▶ Correlation between function values
- How far do we have to move in input space from \mathbf{x} to \mathbf{x}' to make $f(\mathbf{x})$ and $f(\mathbf{x}')$ uncorrelated?

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$



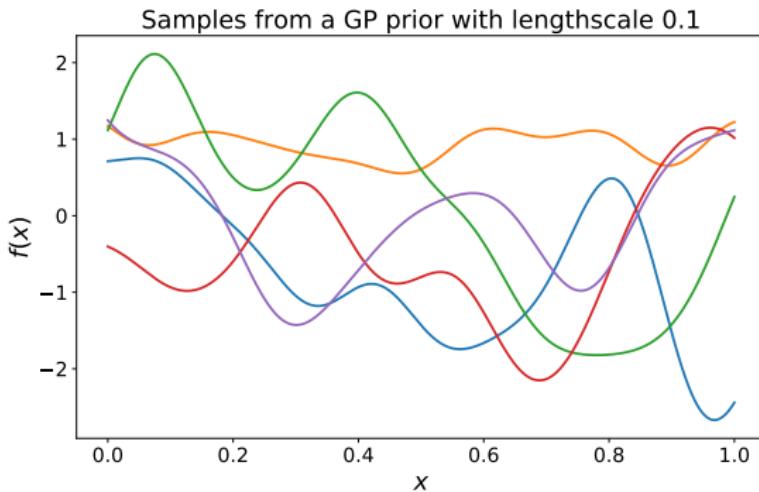
- Correlation between function values $f(\mathbf{x})$ and $f(\mathbf{x}')$ depends on the (scaled) distance $\|\tau\|/\ell = \|\mathbf{x} - \mathbf{x}'\|/\ell$ of the corresponding inputs.
- What does a short/long length-scale ℓ imply?

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$



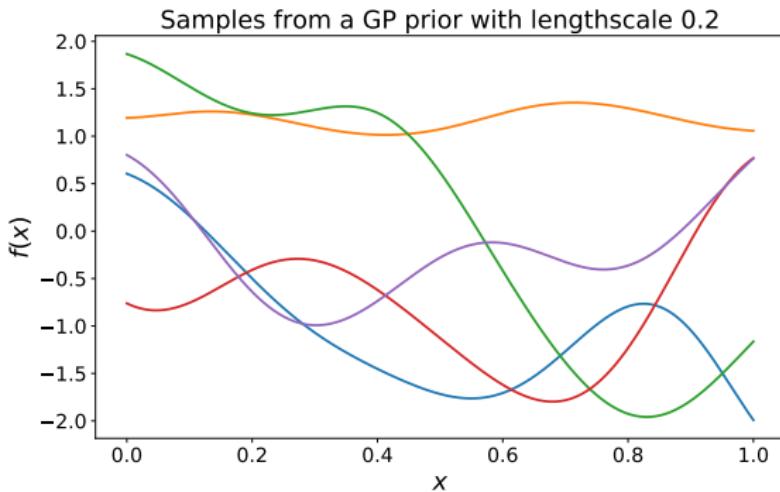
► Explore interactive diagrams at
<https://drafts.distill.pub/gp/>

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$



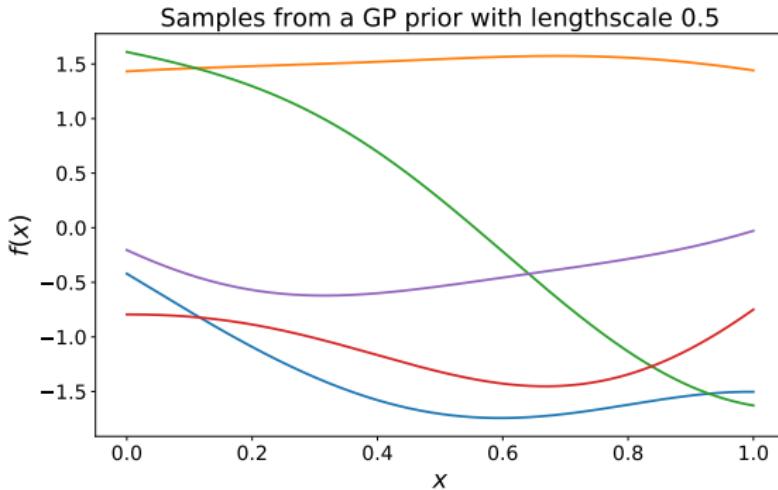
► Explore interactive diagrams at
<https://drafts.distill.pub/gp/>

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$



► Explore interactive diagrams at
<https://drafts.distill.pub/gp/>

$$k_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)/\ell^2\right)$$

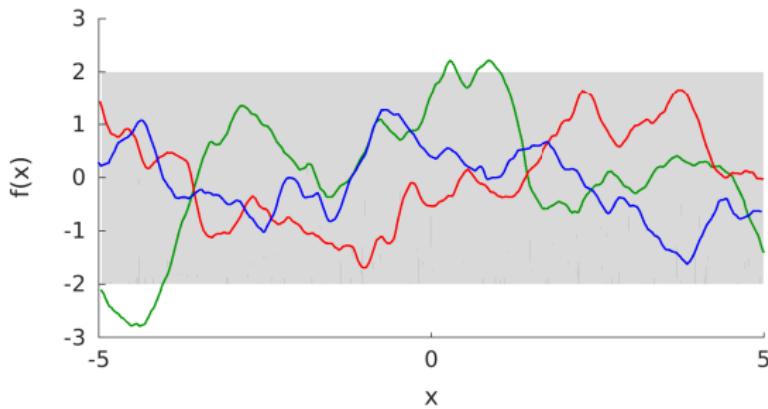


► Explore interactive diagrams at
<https://drafts.distill.pub/gp/>

Matérn Covariance Function

$$k_{Mat,3/2}(x_i, x_j) = \sigma_f^2 \left(1 + \frac{\sqrt{3}\|x_i - x_j\|}{\ell} \right) \exp \left(-\frac{\sqrt{3}\|x_i - x_j\|}{\ell} \right)$$

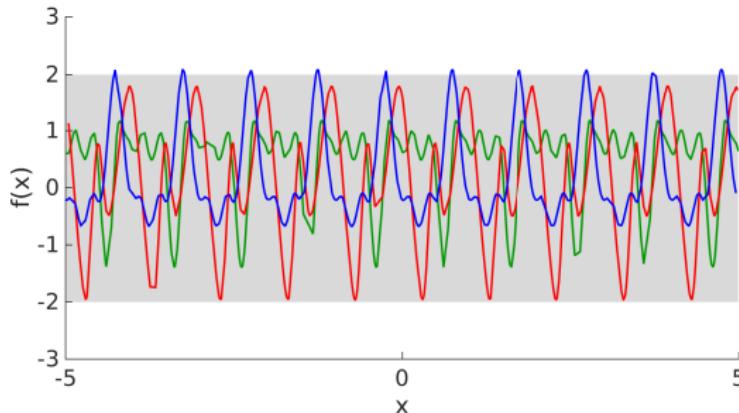
- Assumption on latent function: 1-times differentiable
- σ_f : Amplitude of the latent function
- ℓ : Length-scale. How far do we have to move in input space before the function value changes significantly?



Periodic Covariance Function

$$k_{per}(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{2 \sin^2\left(\frac{\kappa(x_i - x_j)}{2\pi}\right)}{\ell^2}\right)$$
$$= k_{Gauss}(\mathbf{u}(x_i), \mathbf{u}(x_j)), \quad \mathbf{u}(x) = \begin{bmatrix} \cos(\kappa x) \\ \sin(\kappa x) \end{bmatrix}$$

- Assumption on latent function: **periodic**
- **Periodicity parameter** κ



Assume k_1 and k_2 are valid covariance functions and $u(\cdot)$ is a (nonlinear) transformation of the input space. Then

- $k_1 + k_2$ is a valid covariance function

Assume k_1 and k_2 are valid covariance functions and $u(\cdot)$ is a (nonlinear) transformation of the input space. Then

- $k_1 + k_2$ is a valid covariance function
- $k_1 k_2$ is a valid covariance function

Assume k_1 and k_2 are valid covariance functions and $u(\cdot)$ is a (nonlinear) transformation of the input space. Then

- $k_1 + k_2$ is a valid covariance function
- $k_1 k_2$ is a valid covariance function
- $k(u(x), u(x'))$ is a valid covariance function (MacKay, 1998)
 - ▶ Periodic covariance function
 - ▶ Manifold Gaussian process (Calandra et al., 2016)
 - ▶ Deep kernel learning (Wilson et al., 2016)

Assume k_1 and k_2 are valid covariance functions and $u(\cdot)$ is a (nonlinear) transformation of the input space. Then

- $k_1 + k_2$ is a valid covariance function
- $k_1 k_2$ is a valid covariance function
- $k(u(x), u(x'))$ is a valid covariance function (MacKay, 1998)
 - ▶ Periodic covariance function
 - ▶ Manifold Gaussian process (Calandra et al., 2016)
 - ▶ Deep kernel learning (Wilson et al., 2016)
- ▶ Automatic Statistician (Lloyd et al., 2014)

$$p(f(\cdot)|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y}|\mathbf{X})}$$

$$p(f(\cdot)|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y}|\mathbf{X})}$$

Gaussian likelihood in linear regression:

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y | \boldsymbol{\theta}^\top \mathbf{x}, \sigma_n^2)$$

- Function (not a distribution) of the parameters
- Describes how parameters and observed data are connected
- Tells us how to transform parameters into (noisy) data

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Gaussian likelihood in linear regression:

$$p(y | \mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y | \boldsymbol{\theta}^\top \mathbf{x}, \sigma_n^2)$$

- Function (not a distribution) of the parameters
- Describes how parameters and observed data are connected
- Tells us how to transform parameters into (noisy) data

Gaussian likelihood in Gaussian processes:

$$p(y | f(\mathbf{x})) = \mathcal{N}(y | f(\mathbf{x}), \sigma_n^2)$$

- Intuition: Parameters are the function f itself

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Bayesian linear regression with a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$p(\mathbf{y} | \mathbf{X}) = \int p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

- Normalizes the posterior distribution

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Bayesian linear regression with a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$\begin{aligned} p(\mathbf{y} | \mathbf{X}) &= \int p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= \mathcal{N}(\mathbf{y} | \mathbf{0}, \Phi \Phi^\top + \sigma^2 \mathbf{I}) \end{aligned}$$

- Normalizes the posterior distribution
- Can be computed analytically

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Bayesian linear regression with a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$\begin{aligned} p(\mathbf{y} | \mathbf{X}) &= \int p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= \mathcal{N}(\mathbf{y} | \mathbf{0}, \Phi \Phi^\top + \sigma^2 \mathbf{I}) \\ &= \mathbb{E}_{\boldsymbol{\theta}}[p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta})] \end{aligned}$$

- Normalizes the posterior distribution
- Can be computed analytically
- Expected likelihood (under the parameter prior)
- Expected predictive distribution of the training targets \mathbf{y} (under the parameter prior)

Gaussian process marginal likelihood

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|f(\mathbf{X}))p(f(\mathbf{X}))df$$

- Normalizes the posterior distribution

Gaussian process marginal likelihood

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}) &= \int p(\mathbf{y}|f(\mathbf{X}))p(f(\mathbf{X}))df \\ &= \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I}) \end{aligned}$$

- Normalizes the posterior distribution
- Can be computed analytically

Gaussian process marginal likelihood

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}) &= \int p(\mathbf{y}|f(\mathbf{X}))p(f(\mathbf{X}))df \\ &= \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I}) \\ &= \mathbb{E}_f[p(\mathbf{y}|f(\mathbf{X}))] \end{aligned}$$

- Normalizes the posterior distribution
- Can be computed analytically
- Expected likelihood (under the GP prior)
- Expected predictive distribution of the training targets \mathbf{y} (under the GP prior)

Gaussian process marginal likelihood

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}) &= \int p(\mathbf{y}|f(\mathbf{X}))p(f(\mathbf{X}))df \\ &= \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I}) \\ &= \mathbb{E}_f[p(\mathbf{y}|f(\mathbf{X}))] \end{aligned}$$

- Normalizes the posterior distribution
- Can be computed analytically
- Expected likelihood (under the GP prior)
- Expected predictive distribution of the training targets \mathbf{y} (under the GP prior)

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2 \mathbf{I}| - \frac{N}{2}\log(2\pi)$$

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \quad i, j = 1, \dots, N$$

Posterior over functions (with training data \mathbf{X}, \mathbf{y}):

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Posterior over functions (with training data \mathbf{X}, \mathbf{y}):

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Using the properties of Gaussians, we obtain (with $\mathbf{K} := k(\mathbf{X}, \mathbf{X})$)

$$p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot)) = \mathcal{N}(\mathbf{y} | f(\mathbf{X}), \sigma_n^2 \mathbf{I}) GP(m(\cdot), k(\cdot, \cdot))$$

Posterior over functions (with training data \mathbf{X}, \mathbf{y}):

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Using the properties of Gaussians, we obtain (with $\mathbf{K} := k(\mathbf{X}, \mathbf{X})$)

$$\begin{aligned} p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot)) &= \mathcal{N}(\mathbf{y} | f(\mathbf{X}), \sigma_n^2 \mathbf{I}) GP(m(\cdot), k(\cdot, \cdot)) \\ &= Z \times GP(m_{\text{post}}(\cdot), k_{\text{post}}(\cdot, \cdot)) \end{aligned}$$

$$m_{\text{post}}(\cdot) = m(\cdot) + k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}(\mathbf{y} - m(\mathbf{X}))$$

$$k_{\text{post}}(\cdot, \cdot) = k(\cdot, \cdot) - k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}k(\mathbf{X}, \cdot)$$

Posterior over functions (with training data \mathbf{X}, \mathbf{y}):

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot))}{p(\mathbf{y} | \mathbf{X})}$$

Using the properties of Gaussians, we obtain (with $\mathbf{K} := k(\mathbf{X}, \mathbf{X})$)

$$\begin{aligned} p(\mathbf{y} | f(\mathbf{X})) p(f(\cdot)) &= \mathcal{N}(\mathbf{y} | f(\mathbf{X}), \sigma_n^2 \mathbf{I}) GP(m(\cdot), k(\cdot, \cdot)) \\ &= Z \times GP(m_{\text{post}}(\cdot), k_{\text{post}}(\cdot, \cdot)) \end{aligned}$$

$$m_{\text{post}}(\cdot) = m(\cdot) + k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}(\mathbf{y} - m(\mathbf{X}))$$

$$k_{\text{post}}(\cdot, \cdot) = k(\cdot, \cdot) - k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}k(\mathbf{X}, \cdot)$$

Marginal likelihood:

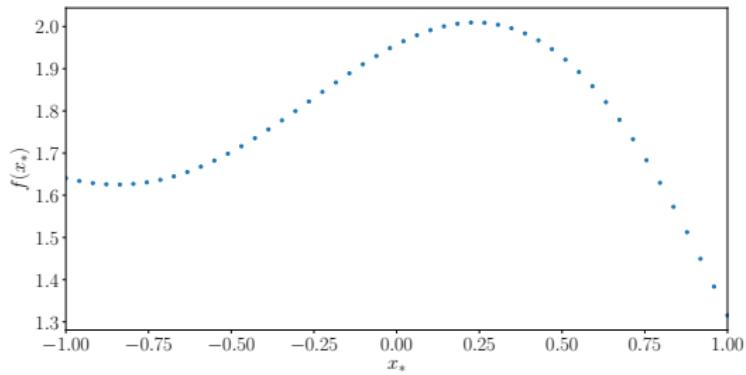
$$Z = p(\mathbf{y} | \mathbf{X}) = \int p(\mathbf{y} | f(\mathbf{X})) p(f(\mathbf{X})) df = \mathcal{N}(\mathbf{y} | m(\mathbf{X}), \mathbf{K} + \sigma_n^2 \mathbf{I})$$

- GP is a distribution over functions
 - ▶ A sample from a GP will be an entire function

- GP is a distribution over functions
 - ▶ A sample from a GP will be an entire function
- In practice, we cannot sample functions directly

Sampling from the GP Prior

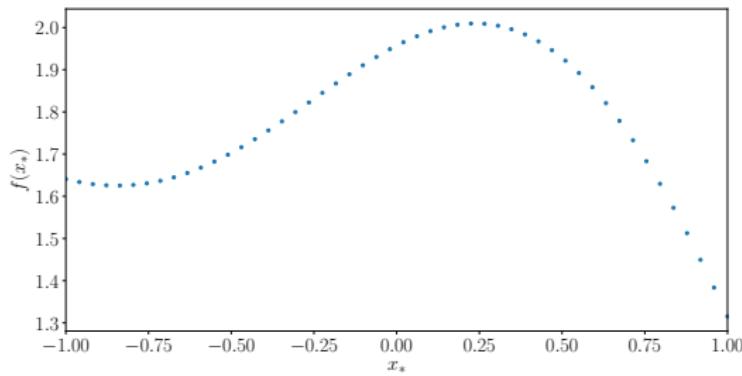
- GP is a distribution over functions
 - ▶ A sample from a GP will be an entire function
- In practice, we cannot sample functions directly
- Instead: function = collection of function values



Sampling from the GP Prior

- GP is a distribution over functions
 - ▶ A sample from a GP will be an entire function
- In practice, we cannot sample functions directly
- Instead: function = collection of function values
- Determine function values at a finite set of input locations

$$\mathbf{X}_* = [\mathbf{x}_*^{(1)}, \dots, \mathbf{x}_*^{(K)}]$$



- Without any training data, the predictive distribution at test points \mathbf{X}_* is

$$\begin{aligned} p(\mathbf{f}(\mathbf{X}_*)|\mathbf{X}_*) &= \mathcal{N}\left(\mathbb{E}_f[f(\mathbf{X}_*)], \mathbb{V}_f[f(\mathbf{X}_*)]\right) \\ &= \mathcal{N}\left(m_{\text{prior}}(\mathbf{X}_*), k_{\text{prior}}(\mathbf{X}_*, \mathbf{X}_*)\right) \end{aligned}$$

- Without any training data, the predictive distribution at test points \mathbf{X}_* is

$$\begin{aligned} p(\mathbf{f}(\mathbf{X}_*)|\mathbf{X}_*) &= \mathcal{N}\left(\mathbb{E}_f[f(\mathbf{X}_*)], \mathbb{V}_f[f(\mathbf{X}_*)]\right) \\ &= \mathcal{N}\left(m_{\text{prior}}(\mathbf{X}_*), k_{\text{prior}}(\mathbf{X}_*, \mathbf{X}_*)\right) \end{aligned}$$

- Exploited: Definition of GP that all function values are jointly Gaussian distributed

- Without any training data, the predictive distribution at test points \mathbf{X}_* is

$$\begin{aligned} p(\mathbf{f}(\mathbf{X}_*)|\mathbf{X}_*) &= \mathcal{N}\left(\mathbb{E}_f[f(\mathbf{X}_*)], \mathbb{V}_f[f(\mathbf{X}_*)]\right) \\ &= \mathcal{N}\left(m_{\text{prior}}(\mathbf{X}_*), k_{\text{prior}}(\mathbf{X}_*, \mathbf{X}_*)\right) \end{aligned}$$

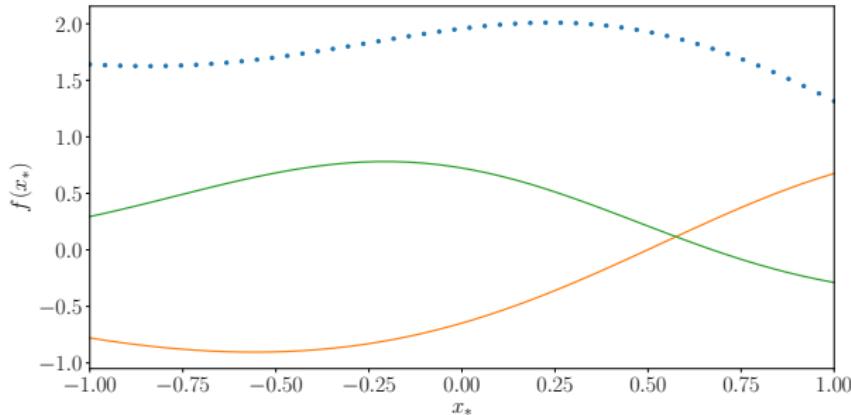
- Exploited: Definition of GP that **all function values are jointly Gaussian distributed**
- Generate “function draws” (samples from the GP prior)

$$f_k(\mathbf{X}_*) \sim \mathcal{N}\left(m_{\text{prior}}(\mathbf{X}_*), k_{\text{prior}}(\mathbf{X}_*, \mathbf{X}_*)\right)$$

- Goal: Generate random functions f_k , so that

$$f_k(\mathbf{X}_*) \sim \mathcal{N}(m_{\text{prior}}(\mathbf{X}_*), k_{\text{prior}}(\mathbf{X}_*, \mathbf{X}_*))$$

Sampling from the GP Prior (3)



- Goal: Generate random functions f_k , so that

$$f_k(\mathbf{X}_*) \sim \mathcal{N}(m_{\text{prior}}(\mathbf{X}_*), k_{\text{prior}}(\mathbf{X}_*, \mathbf{X}_*))$$

- Define $\mathbf{m}_* := m_{\text{prior}}(\mathbf{X}_*)$ and $\mathbf{K}_{**} := k_{\text{prior}}(\mathbf{X}_*, \mathbf{X}_*)$. Then

$$f_k(\mathbf{X}_*) \sim \mathcal{N}(\mathbf{m}_*, \mathbf{K}_{**})$$

► Sample from a multivariate Gaussian

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

- **Objective:** Find $p(f(\mathbf{X}_*)|\mathbf{X}, \mathbf{y}, \mathbf{X}_*)$ for training data \mathbf{X}, \mathbf{y} and test inputs \mathbf{X}_* .
- GP prior at training inputs: $p(f|\mathbf{X}) = \mathcal{N}(m(\mathbf{X}), \mathbf{K})$
- Gaussian Likelihood: $p(\mathbf{y}|f, \mathbf{X}) = \mathcal{N}(f(\mathbf{X}), \sigma_n^2 \mathbf{I})$

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

- **Objective:** Find $p(f(\mathbf{X}_*)|\mathbf{X}, \mathbf{y}, \mathbf{X}_*)$ for training data \mathbf{X}, \mathbf{y} and test inputs \mathbf{X}_* .
- GP prior at training inputs: $p(f|\mathbf{X}) = \mathcal{N}(m(\mathbf{X}), \mathbf{K})$
- Gaussian Likelihood: $p(\mathbf{y}|f, \mathbf{X}) = \mathcal{N}(f(\mathbf{X}), \sigma_n^2 \mathbf{I})$
- With $f \sim GP$ it follows that f, f_* are jointly Gaussian distributed:

$$p(f, f_* | \mathbf{X}, \mathbf{X}_*) = \mathcal{N} \left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix} \right)$$

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

- **Objective:** Find $p(f(\mathbf{X}_*)|\mathbf{X}, \mathbf{y}, \mathbf{X}_*)$ for training data \mathbf{X}, \mathbf{y} and test inputs \mathbf{X}_* .
- GP prior at training inputs: $p(f|\mathbf{X}) = \mathcal{N}(m(\mathbf{X}), \mathbf{K})$
- Gaussian Likelihood: $p(\mathbf{y}|f, \mathbf{X}) = \mathcal{N}(f(\mathbf{X}), \sigma_n^2 \mathbf{I})$
- With $f \sim GP$ it follows that f, f_* are jointly Gaussian distributed:

$$p(\mathbf{f}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*) = \mathcal{N} \left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix} \right)$$

- Due to the Gaussian likelihood, we also get (\mathbf{f} is unobserved)

$$p(\mathbf{y}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*) = \mathcal{N} \left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix} \right)$$

Prior evaluated at \mathbf{X}, \mathbf{X}_* :

$$p(\mathbf{y}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*) = \mathcal{N} \left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix} \right)$$

Posterior predictive distribution $p(\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*)$ at test inputs \mathbf{X}_*

Prior evaluated at \mathbf{X}, \mathbf{X}_* :

$$p(\mathbf{y}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*) = \mathcal{N} \left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix} \right)$$

Posterior predictive distribution $p(\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*)$ at test inputs \mathbf{X}_* obtained by Gaussian conditioning:

$$p(\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*) = \mathcal{N} \left(\mathbb{E}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*], \mathbb{V}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] \right)$$

$$\mathbb{E}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = \underbrace{m(\mathbf{X}_*)}_{\text{prior mean}} + \underbrace{k(\mathbf{X}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}}_{\text{"Kalman gain"}} \underbrace{(\mathbf{y} - m(\mathbf{X}))}_{\text{error}}$$

Prior evaluated at \mathbf{X}, \mathbf{X}_* :

$$p(\mathbf{y}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*) = \mathcal{N} \left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & k(\mathbf{X}, \mathbf{X}_*) \\ k(\mathbf{X}_*, \mathbf{X}) & k(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix} \right)$$

Posterior predictive distribution $p(\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*)$ at test inputs \mathbf{X}_* obtained by Gaussian conditioning:

$$p(\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*) = \mathcal{N} \left(\mathbb{E}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*], \mathbb{V}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] \right)$$

$$\mathbb{E}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = \underbrace{m(\mathbf{X}_*)}_{\text{prior mean}} + \underbrace{k(\mathbf{X}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}}_{\text{"Kalman gain"}} \underbrace{(\mathbf{y} - m(\mathbf{X}))}_{\text{error}}$$

$$\mathbb{V}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = \underbrace{k(\mathbf{X}_*, \mathbf{X}_*)}_{\text{prior variance}} - \underbrace{k(\mathbf{X}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{X}_*)}_{\geq 0}$$

- GP posterior (from earlier):

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = GP\left(m_{\text{post}}(\cdot), k_{\text{post}}(\cdot, \cdot)\right)$$

$$m_{\text{post}}(\cdot) = m(\cdot) + k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}(\mathbf{y} - m(\mathbf{X}))$$

$$k_{\text{post}}(\cdot, \cdot) = k(\cdot, \cdot) - k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}k(\mathbf{X}, \cdot)$$

- GP posterior (from earlier):

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = GP(m_{\text{post}}(\cdot), k_{\text{post}}(\cdot, \cdot))$$

$$m_{\text{post}}(\cdot) = m(\cdot) + k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}(\mathbf{y} - m(\mathbf{X}))$$

$$k_{\text{post}}(\cdot, \cdot) = k(\cdot, \cdot) - k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}k(\mathbf{X}, \cdot)$$

- GP posterior predictions at \mathbf{X}_* :

$$p(\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*) = \mathcal{N}(\mathbb{E}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*], \mathbb{V}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*])$$

$$\mathbb{E}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = m(\mathbf{X}_*) + k(\mathbf{X}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}(\mathbf{y} - m(\mathbf{X}))$$

$$\mathbb{V}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = k(\mathbf{X}_*, \mathbf{X}_*) - k(\mathbf{X}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}k(\mathbf{X}, \mathbf{X}_*)$$

- GP posterior (from earlier):

$$p(f(\cdot) | \mathbf{X}, \mathbf{y}) = GP(m_{\text{post}}(\cdot), k_{\text{post}}(\cdot, \cdot))$$

$$m_{\text{post}}(\cdot) = m(\cdot) + k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}(\mathbf{y} - m(\mathbf{X}))$$

$$k_{\text{post}}(\cdot, \cdot) = k(\cdot, \cdot) - k(\cdot, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}k(\mathbf{X}, \cdot)$$

- GP posterior predictions at \mathbf{X}_* :

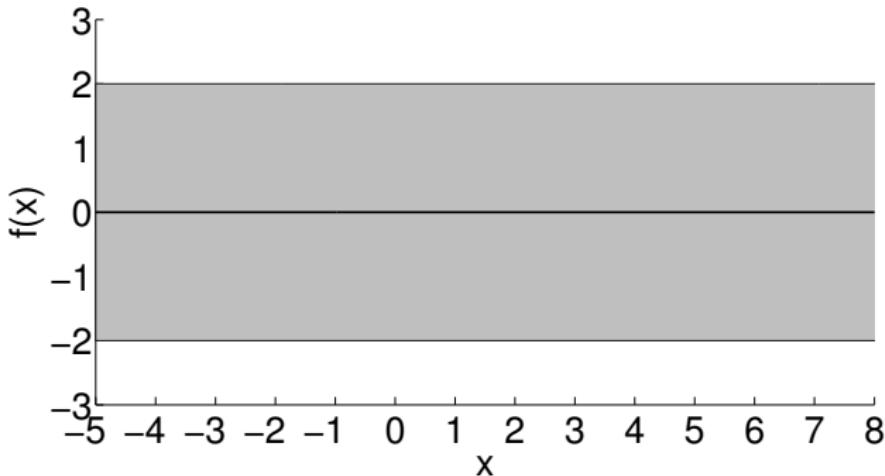
$$p(\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*) = \mathcal{N}(\mathbb{E}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*], \mathbb{V}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*])$$

$$\mathbb{E}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = m(\mathbf{X}_*) + k(\mathbf{X}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}(\mathbf{y} - m(\mathbf{X}))$$

$$\mathbb{V}[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = k(\mathbf{X}_*, \mathbf{X}_*) - k(\mathbf{X}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}k(\mathbf{X}, \mathbf{X}_*)$$

Predictions

Make predictions by evaluating the GP posterior mean and covariance function at a finite number of inputs \mathbf{X}_*

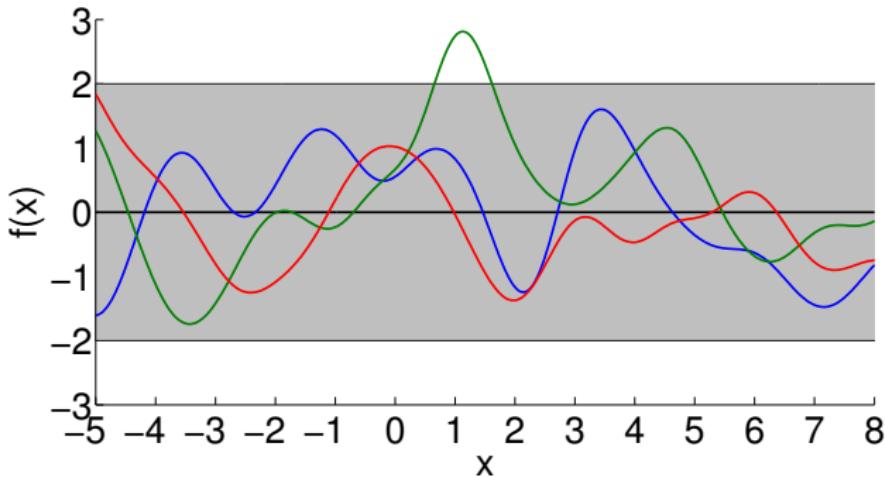


Prior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*) | \boldsymbol{x}_*, \emptyset] = m(\boldsymbol{x}_*) = 0$$

$$\mathbb{V}[f(\boldsymbol{x}_*) | \boldsymbol{x}_*, \emptyset] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$$



Prior belief about the function

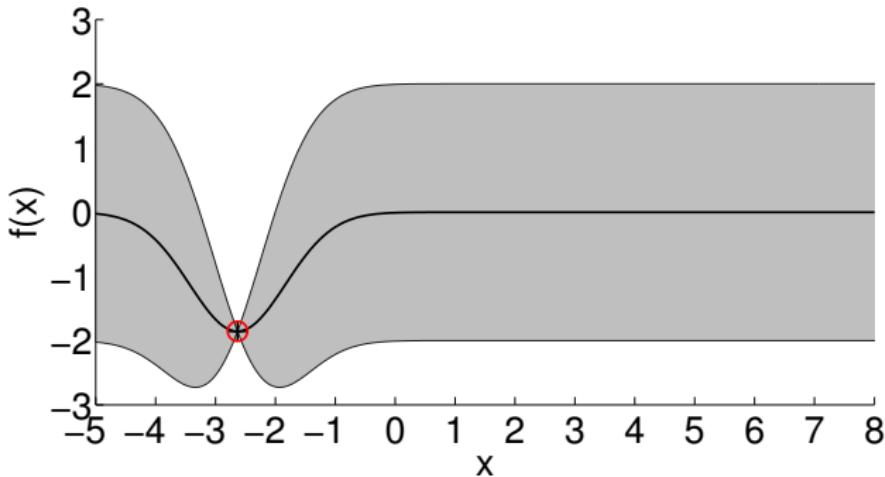
Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*) | \mathbf{x}_*, \emptyset] = m(\mathbf{x}_*) = 0$$

$$\mathbb{V}[f(\mathbf{x}_*) | \mathbf{x}_*, \emptyset] = \sigma^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*)$$

Illustration: Inference with Gaussian Processes

UCL



Posterior belief about the function

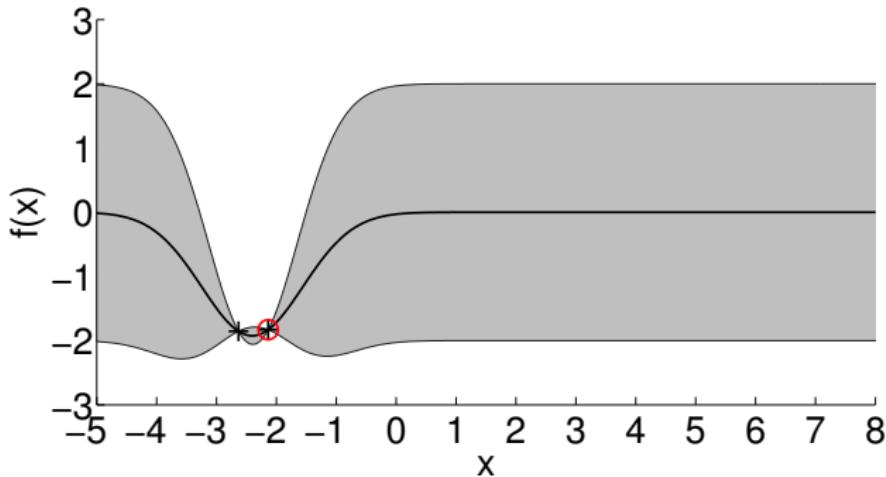
Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = m(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\mathbb{V}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}_*)$$

Illustration: Inference with Gaussian Processes

UCL



Posterior belief about the function

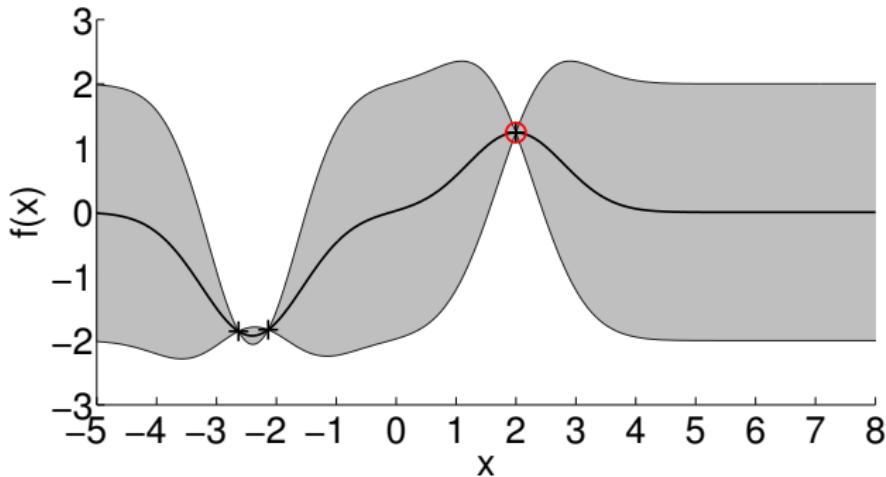
Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = m(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\mathbb{V}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}_*)$$

Illustration: Inference with Gaussian Processes

UCL



Posterior belief about the function

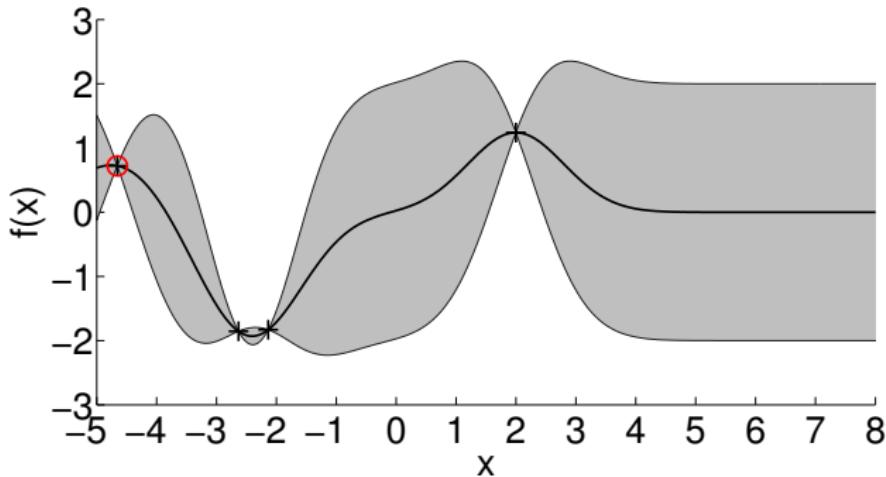
Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = m(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\mathbb{V}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}_*)$$

Illustration: Inference with Gaussian Processes

UCL



Posterior belief about the function

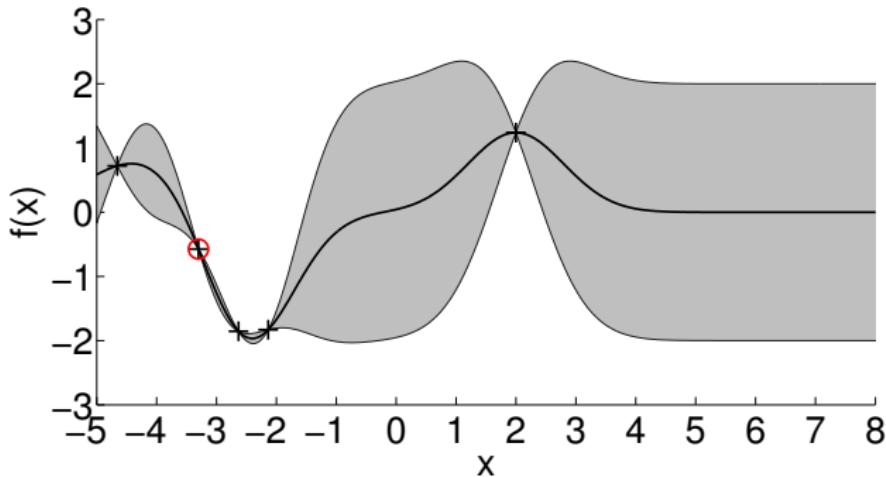
Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = m(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\mathbb{V}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}_*)$$

Illustration: Inference with Gaussian Processes

UCL



Posterior belief about the function

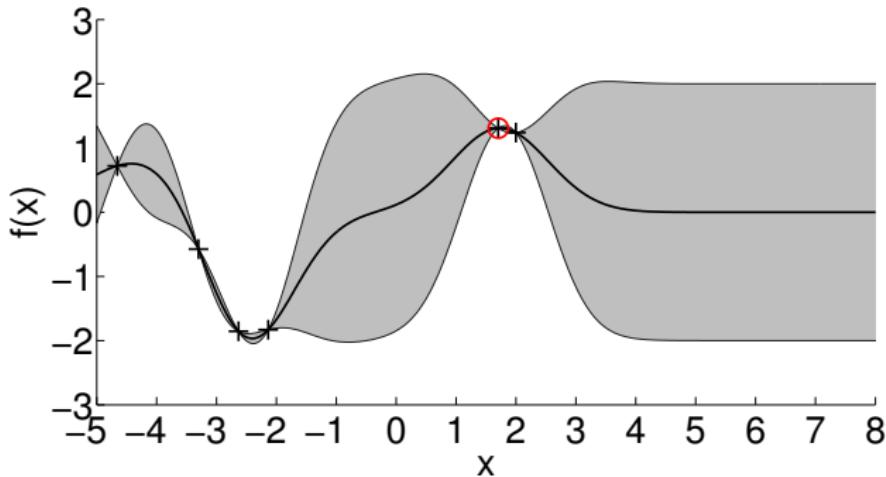
Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = m(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\mathbb{V}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}_*)$$

Illustration: Inference with Gaussian Processes

UCL



Posterior belief about the function

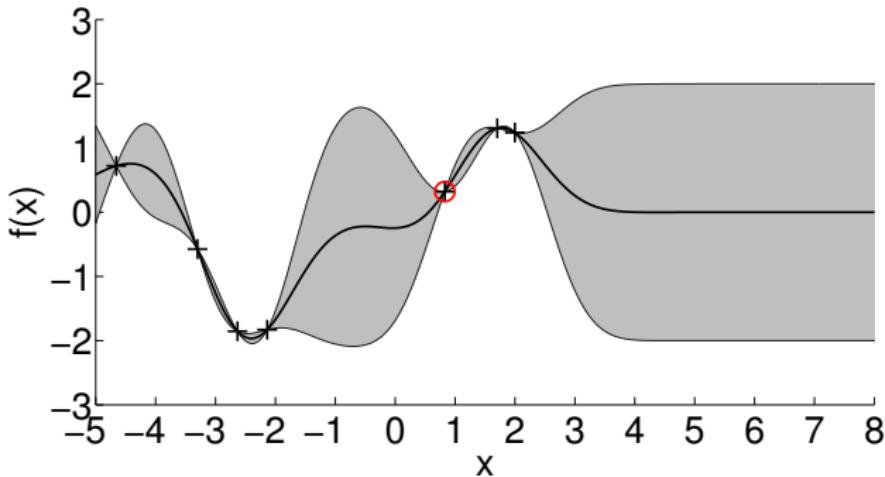
Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = m(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\mathbb{V}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}_*)$$

Illustration: Inference with Gaussian Processes

UCL



Posterior belief about the function

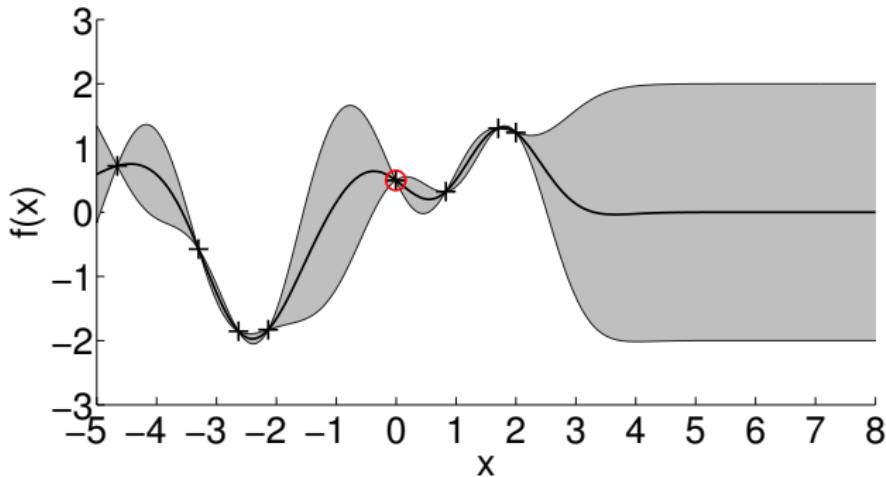
Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = m(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\mathbb{V}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}_*)$$

Illustration: Inference with Gaussian Processes

UCL

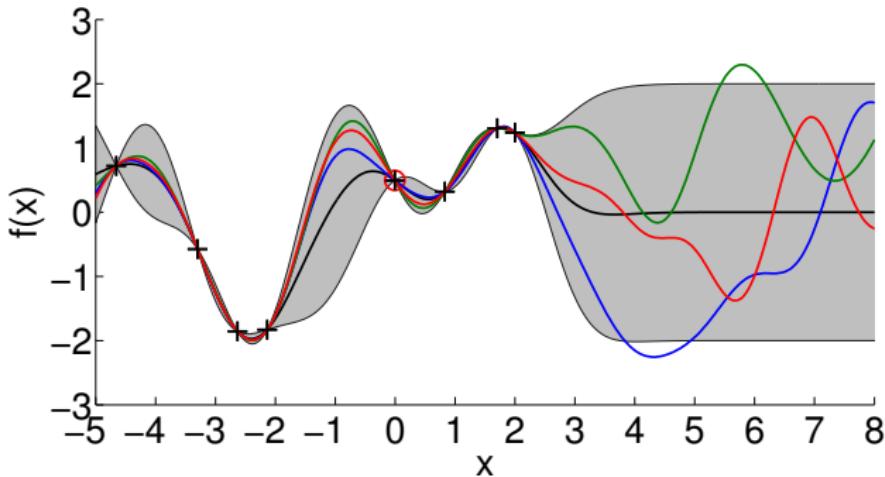


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = m(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\mathbb{V}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}_*)$$



Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = m(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

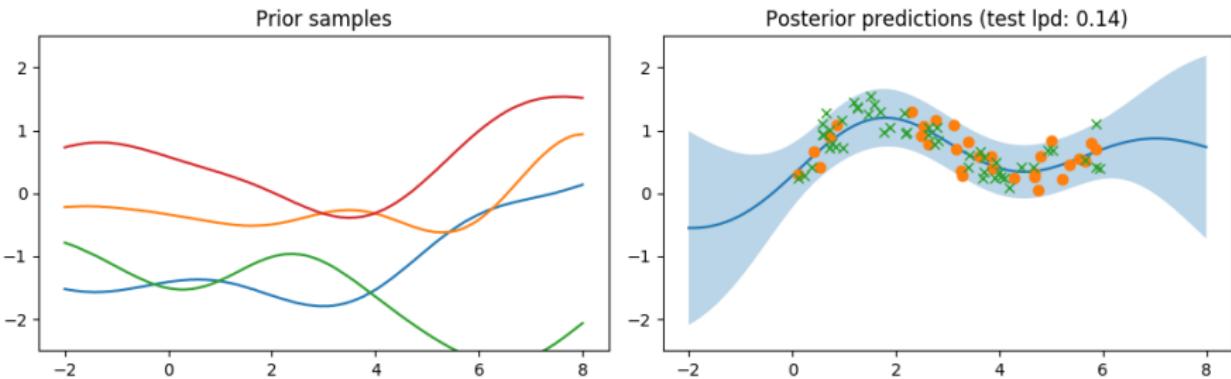
$$\mathbb{V}[f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{X}, \mathbf{y}] = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X})(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}_*)$$

Overview

- 1 Gaussian Process: Definition**
- 2 Regression as Inference**
 - GP Prior
 - Likelihood
 - Marginal Likelihood
 - Posterior
 - Predictions
- 3 Model Selection**
 - GP Training
 - Hyper-Parameters
 - Inspection of the Marginal Likelihood
 - Covariance Function
- 4 Limitations and Guidelines**
- 5 Application Areas**

Model Selection

Influence of Prior on Posterior

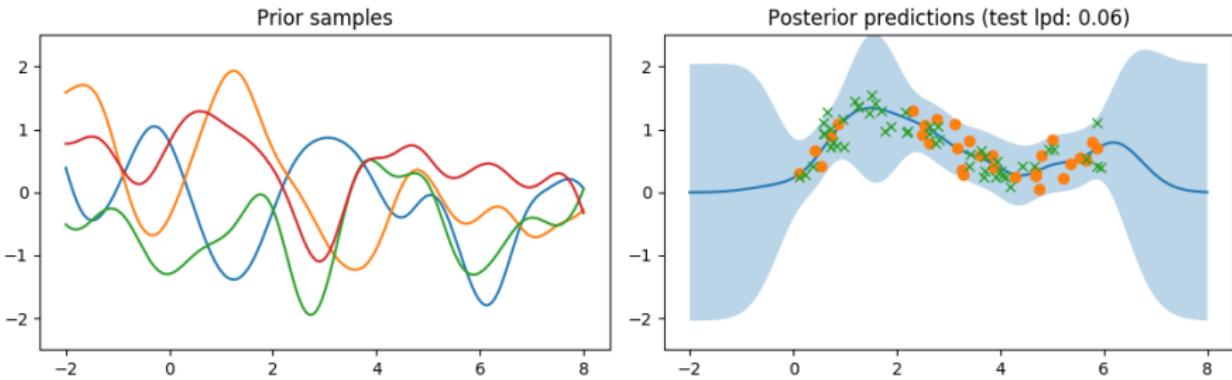


- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \ell)$$

for different length-scales ℓ and different datasets

Influence of Prior on Posterior



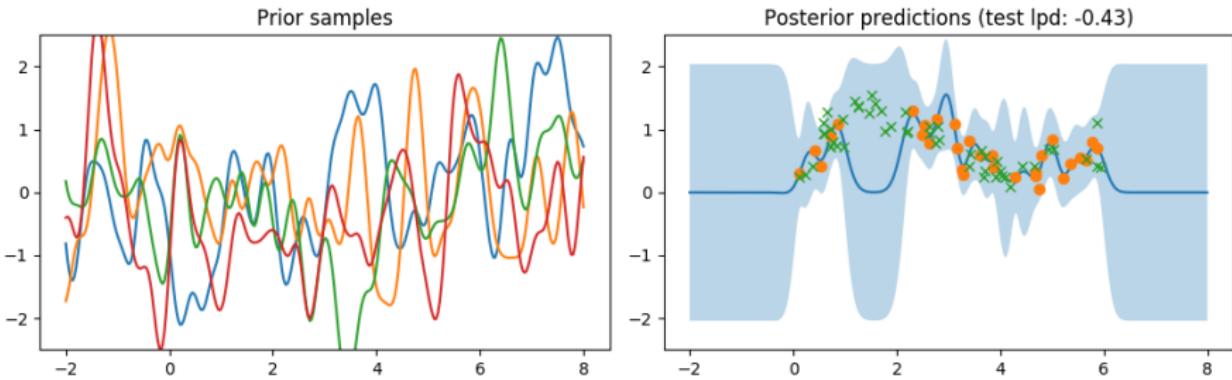
- Generalization error measured by **log-predictive density** (lpd)

$$\text{lpd} = \log p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ➤ More flexible model ➤ Faster increase in uncertainty away from data ➤ Bad generalization properties

Influence of Prior on Posterior



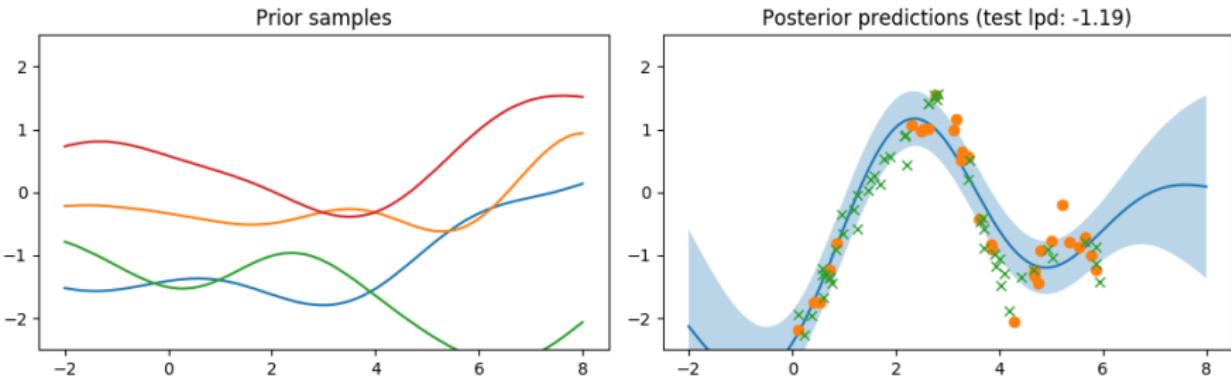
- Generalization error measured by **log-predictive density** (lpd)

$$\text{lpd} = \log p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ➤ More flexible model ➤ Faster increase in uncertainty away from data ➤ Bad generalization properties

Influence of Prior on Posterior



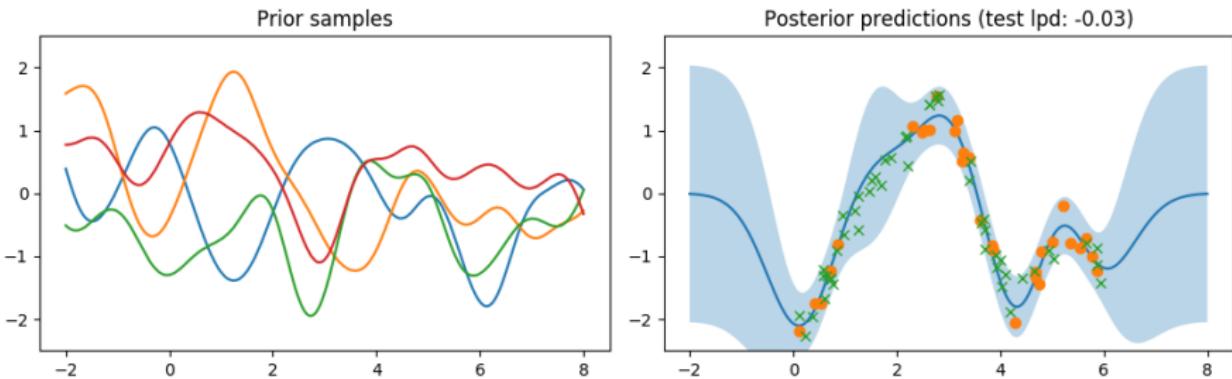
- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ➤ More flexible model ➤ Faster increase in uncertainty away from data ➤ Bad generalization properties

Influence of Prior on Posterior



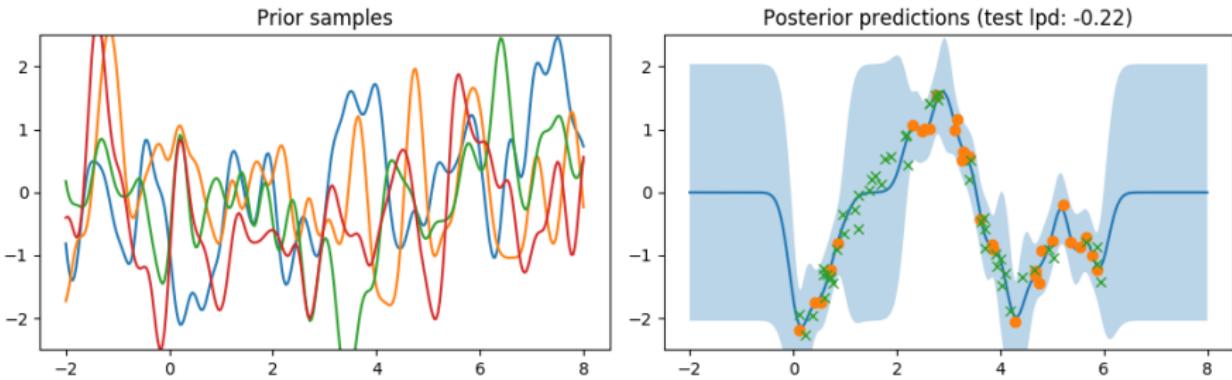
- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ➤ More flexible model ➤ Faster increase in uncertainty away from data ➤ Bad generalization properties

Influence of Prior on Posterior



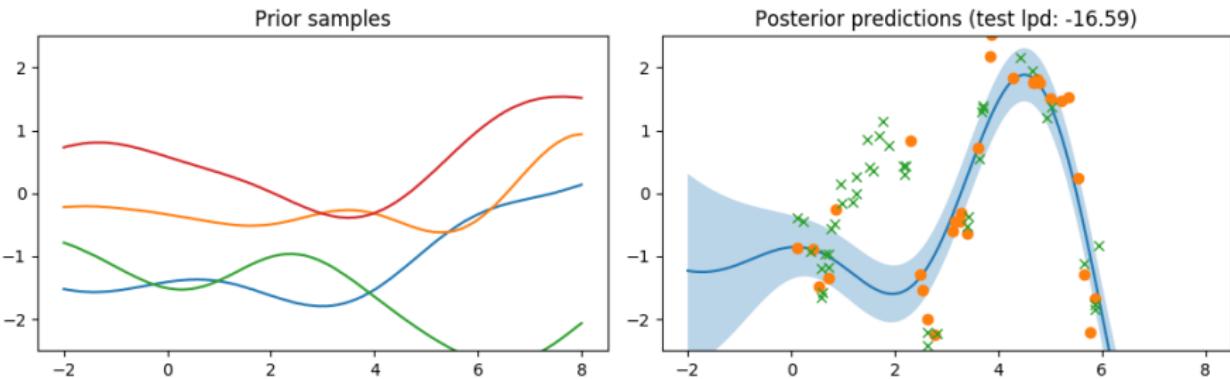
- Generalization error measured by **log-predictive density** (lpd)

$$\text{lpd} = \log p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ➤ More flexible model ➤ Faster increase in uncertainty away from data ➤ Bad generalization properties

Influence of Prior on Posterior



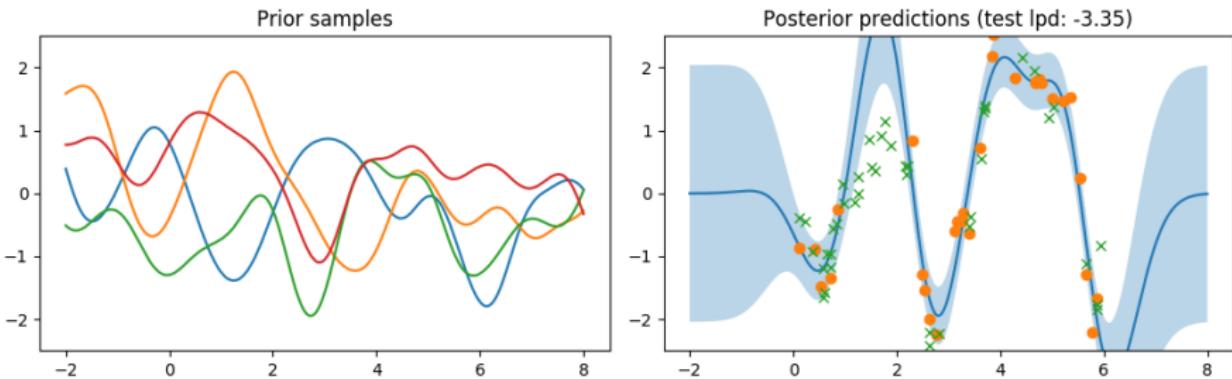
- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ➤ More flexible model ➤ Faster increase in uncertainty away from data ➤ Bad generalization properties

Influence of Prior on Posterior



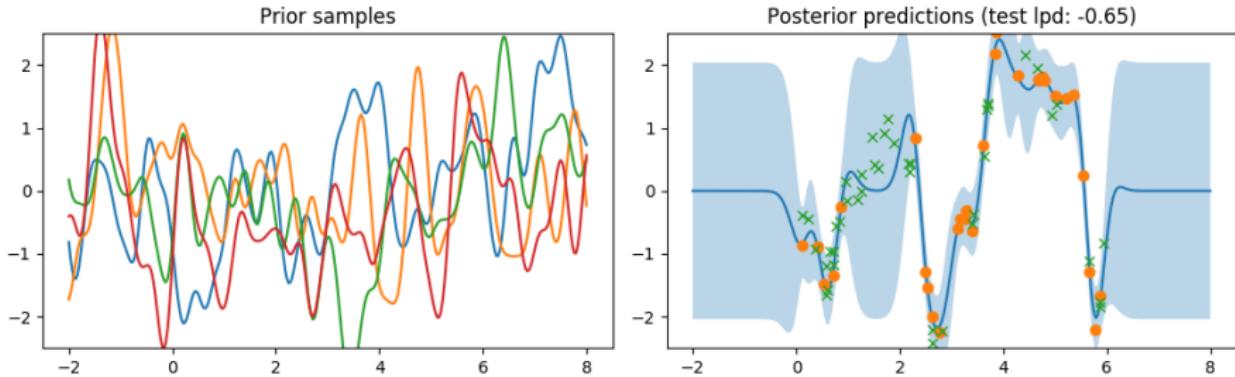
- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ➤ More flexible model ➤ Faster increase in uncertainty away from data ➤ Bad generalization properties

Influence of Prior on Posterior



- Generalization error measured by **log-predictive density** (lpd)

$$\text{lpd} = \log p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ➤ More flexible model ➤ Faster increase in uncertainty away from data ➤ Bad generalization properties

- Make predictions equipped with uncertainty
- Choice of prior (e.g., length-scale) influences predictions
- Different tasks require different priors

- Make predictions equipped with uncertainty
- Choice of prior (e.g., length-scale) influences predictions
- Different tasks require different priors

How do we select a good prior?

- Make predictions equipped with uncertainty
- Choice of prior (e.g., length-scale) influences predictions
- Different tasks require different priors

How do we select a good prior?

Model Selection in GPs

- ▶ Choose hyper-parameters of the GP
- ▶ Choose good mean function and kernel

The GP possesses a set of **hyper-parameters**:

- Parameters of the mean function
- Parameters of the covariance function (e.g., length-scales and signal variance)
- Likelihood parameters (e.g., noise variance σ_n^2)

The GP possesses a set of **hyper-parameters**:

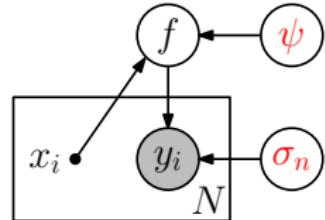
- Parameters of the mean function
 - Parameters of the covariance function (e.g., length-scales and signal variance)
 - Likelihood parameters (e.g., noise variance σ_n^2)
- Train a GP to find a good set of hyper-parameters

The GP possesses a set of **hyper-parameters**:

- Parameters of the mean function
 - Parameters of the covariance function (e.g., length-scales and signal variance)
 - Likelihood parameters (e.g., noise variance σ_n^2)
- Train a GP to find a good set of hyper-parameters
- Higher-level **model selection** to find good mean and covariance functions
(can also be automated: Automatic Statistician (Lloyd et al., 2014))

GP Training

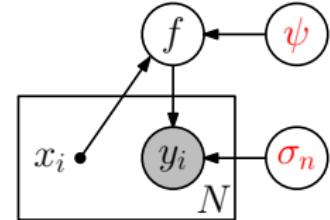
Find good hyper-parameters θ (kernel/mean function parameters ψ , noise variance σ_n^2)



GP Training

Find good hyper-parameters θ (kernel/mean function parameters ψ , noise variance σ_n^2)

- Place a prior $p(\theta)$ on hyper-parameters
- Posterior over hyper-parameters:



$$p(\boldsymbol{\theta} | \mathbf{X}, \mathbf{y}) = \frac{p(\boldsymbol{\theta}) p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{y} | \mathbf{X})}$$

$$p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y} | f, \mathbf{X}) p(f | \mathbf{X}, \boldsymbol{\theta}) df$$

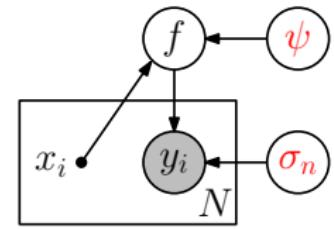
Gaussian Process Training: Hyper-Parameters



- Posterior over hyper-parameters:

$$p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}) = \frac{p(\boldsymbol{\theta}) p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{X})}$$

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}|f(\mathbf{X})) p(f(\mathbf{X})|\boldsymbol{\theta}) df$$

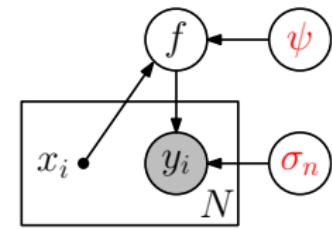


Gaussian Process Training: Hyper-Parameters



- Posterior over hyper-parameters:

$$p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}) = \frac{p(\boldsymbol{\theta}) p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{X})}$$
$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}|f(\mathbf{X})) p(f(\mathbf{X})|\boldsymbol{\theta}) \mathrm{d}f$$



- Choose hyper-parameters $\boldsymbol{\theta}^*$, such that

$$\boldsymbol{\theta}^* \in \arg \max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

Gaussian Process Training: Hyper-Parameters



- Posterior over hyper-parameters:

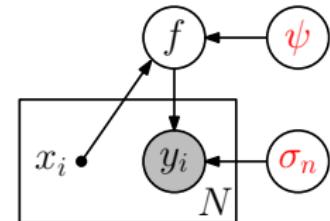
$$p(\boldsymbol{\theta}|\mathbf{X}, \mathbf{y}) = \frac{p(\boldsymbol{\theta}) p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{X})}$$

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}|f(\mathbf{X})) p(f(\mathbf{X})|\boldsymbol{\theta}) df$$

- Choose hyper-parameters $\boldsymbol{\theta}^*$, such that

$$\boldsymbol{\theta}^* \in \arg \max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

- ▶ Maximize marginal likelihood if $p(\boldsymbol{\theta}) = \mathcal{U}$ (uniform prior)



GP Training

Maximize the evidence/marginal likelihood (probability of the data given the hyper-parameters, where the unwieldy f has been integrated out) ➤ Also called Maximum Likelihood Type-II

GP Training

Maximize the evidence/marginal likelihood (probability of the data given the hyper-parameters, where the unwieldy f has been integrated out) ➤ Also called Maximum Likelihood Type-II

Marginal likelihood (with a prior mean function $m(\cdot) \equiv 0$):

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) &= \int p(\mathbf{y}|f(\mathbf{X})) p(f(\mathbf{X})|\boldsymbol{\theta}) df \\ &= \int \mathcal{N}(\mathbf{y} | f(\mathbf{X}), \sigma_n^2 \mathbf{I}) \mathcal{N}(f(\mathbf{X}) | \mathbf{0}, \mathbf{K}) df \\ &= \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K} + \sigma_n^2 \mathbf{I}) \end{aligned}$$

GP Training

Maximize the evidence/marginal likelihood (probability of the data given the hyper-parameters, where the unwieldy f has been integrated out) ➤ Also called Maximum Likelihood Type-II

Marginal likelihood (with a prior mean function $m(\cdot) \equiv 0$):

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) &= \int p(\mathbf{y}|f(\mathbf{X})) p(f(\mathbf{X})|\boldsymbol{\theta}) df \\ &= \int \mathcal{N}(\mathbf{y} | f(\mathbf{X}), \sigma_n^2 \mathbf{I}) \mathcal{N}(f(\mathbf{X}) | \mathbf{0}, \mathbf{K}) df \\ &= \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K} + \sigma_n^2 \mathbf{I}) \end{aligned}$$

Learning the GP hyper-parameters:

$$\boldsymbol{\theta}^* \in \arg \max_{\boldsymbol{\theta}} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

■ Log-marginal likelihood:

$$\log p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2} \mathbf{y}^\top \mathbf{K}_{\boldsymbol{\theta}}^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_{\boldsymbol{\theta}}| + \text{const}$$

$$\mathbf{K}_{\boldsymbol{\theta}} := \mathbf{K} + \sigma_n^2 \mathbf{I}$$

- Log-marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top \mathbf{K}_{\boldsymbol{\theta}}^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_{\boldsymbol{\theta}}| + \text{const}$$

$$\mathbf{K}_{\boldsymbol{\theta}} := \mathbf{K} + \sigma_n^2 \mathbf{I}$$

- Gradient-based optimization to get hyper-parameters $\boldsymbol{\theta}^*$:

$$\begin{aligned} \frac{\partial \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})}{\partial \theta_i} &= \frac{1}{2} \mathbf{y}^\top \mathbf{K}_{\boldsymbol{\theta}}^{-1} \frac{\partial \mathbf{K}_{\boldsymbol{\theta}}}{\partial \theta_i} \mathbf{K}_{\boldsymbol{\theta}}^{-1} \mathbf{y} - \frac{1}{2} \text{tr}(\mathbf{K}_{\boldsymbol{\theta}}^{-1} \frac{\partial \mathbf{K}_{\boldsymbol{\theta}}}{\partial \theta_i}) \\ &= \frac{1}{2} \text{tr}((\boldsymbol{\alpha} \boldsymbol{\alpha}^\top - \mathbf{K}_{\boldsymbol{\theta}}^{-1}) \frac{\partial \mathbf{K}_{\boldsymbol{\theta}}}{\partial \theta_i}), \\ \boldsymbol{\alpha} &:= \mathbf{K}_{\boldsymbol{\theta}}^{-1} \mathbf{y} \end{aligned}$$

- “ELBO” refers to the log-marginal likelihood
- Data-fit term gets worse, but marginal likelihood increases

¹Thanks to Mark van der Wilk

Log-marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top \mathbf{K}_{\boldsymbol{\theta}}^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_{\boldsymbol{\theta}}| + \text{const}, \quad \mathbf{K}_{\boldsymbol{\theta}} := \mathbf{K} + \sigma_n^2 \mathbf{I}$$

Log-marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top \mathbf{K}_{\boldsymbol{\theta}}^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_{\boldsymbol{\theta}}| + \text{const}, \quad \mathbf{K}_{\boldsymbol{\theta}} := \mathbf{K} + \sigma_n^2 \mathbf{I}$$

- Quadratic term measures whether observation \mathbf{y} is within the variation allowed by the prior

Log-marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top \mathbf{K}_{\boldsymbol{\theta}}^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_{\boldsymbol{\theta}}| + \text{const}, \quad \mathbf{K}_{\boldsymbol{\theta}} := \mathbf{K} + \sigma_n^2 \mathbf{I}$$

- Quadratic term measures whether observation \mathbf{y} is within the variation allowed by the prior
- Determinant is the product of the variances of the prior (volume of the prior) ➤ Volume \approx richness of model class

Log-marginal likelihood:

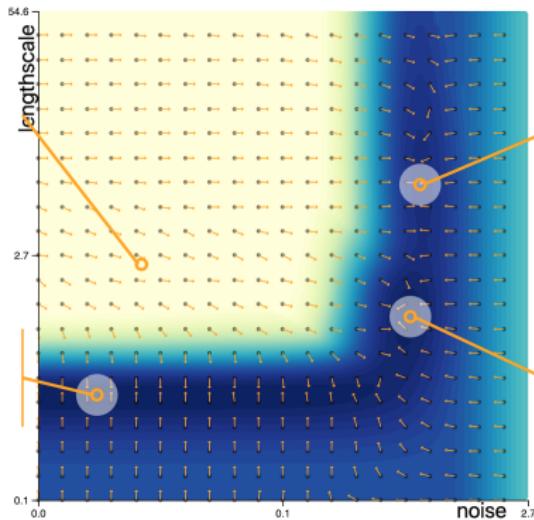
$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top \mathbf{K}_{\boldsymbol{\theta}}^{-1}\mathbf{y} - \frac{1}{2}\log |\mathbf{K}_{\boldsymbol{\theta}}| + \text{const}, \quad \mathbf{K}_{\boldsymbol{\theta}} := \mathbf{K} + \sigma_n^2 \mathbf{I}$$

- Quadratic term measures whether observation \mathbf{y} is within the variation allowed by the prior
- Determinant is the product of the variances of the prior (volume of the prior) ➤ Volume \approx richness of model class

Marginal likelihood

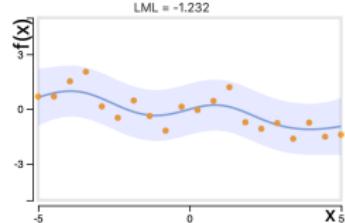
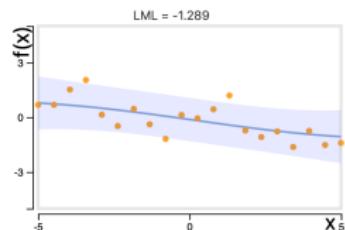
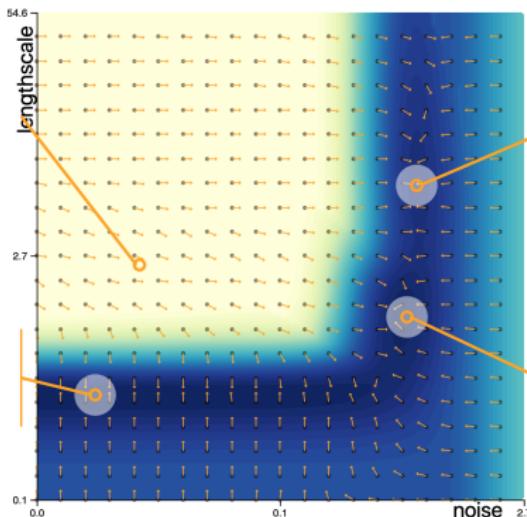
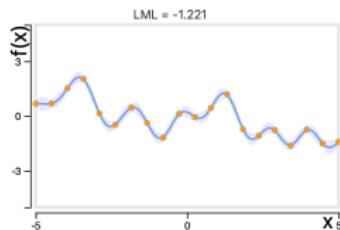
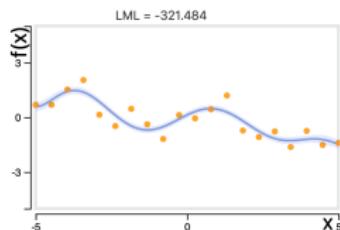
➤ Automatic trade-off between data fit and model complexity

Marginal Likelihood Surface



- Several plausible hyper-parameters (local optima)
- What do you expect to happen in each local optimum?

Marginal Likelihood Surface



- Several plausible hyper-parameters (local optima)
- What do you expect to happen in each local optimum?

<https://drafts.distill.pub/gp/>

- The marginal likelihood is **non-convex**

- The marginal likelihood is **non-convex**
- Especially in the very-small-data regime, a GP can end up in **three different situations** when optimizing the hyper-parameters:

- The marginal likelihood is **non-convex**
- Especially in the very-small-data regime, a GP can end up in **three different situations** when optimizing the hyper-parameters:
 - Short length-scales, low noise (highly nonlinear mean function with little noise)

- The marginal likelihood is **non-convex**
- Especially in the very-small-data regime, a GP can end up in **three different situations** when optimizing the hyper-parameters:
 - Short length-scales, low noise (highly nonlinear mean function with little noise)
 - Long length-scales, high noise (everything is considered noise)

- The marginal likelihood is **non-convex**
- Especially in the very-small-data regime, a GP can end up in **three different situations** when optimizing the hyper-parameters:
 - Short length-scales, low noise (highly nonlinear mean function with little noise)
 - Long length-scales, high noise (everything is considered noise)
 - Hybrid

- The marginal likelihood is **non-convex**
- Especially in the very-small-data regime, a GP can end up in **three different situations** when optimizing the hyper-parameters:
 - Short length-scales, low noise (highly nonlinear mean function with little noise)
 - Long length-scales, high noise (everything is considered noise)
 - Hybrid
- **Re-start** hyper-parameter optimization from random initialization to mitigate the problem

- The marginal likelihood is **non-convex**
- Especially in the very-small-data regime, a GP can end up in **three different situations** when optimizing the hyper-parameters:
 - Short length-scales, low noise (highly nonlinear mean function with little noise)
 - Long length-scales, high noise (everything is considered noise)
 - Hybrid
- **Re-start** hyper-parameter optimization from random initialization to mitigate the problem
- With increasing data set size the GP typically ends up in the “hybrid” mode. Other modes are unlikely.

- The marginal likelihood is **non-convex**
- Especially in the very-small-data regime, a GP can end up in **three different situations** when optimizing the hyper-parameters:
 - Short length-scales, low noise (highly nonlinear mean function with little noise)
 - Long length-scales, high noise (everything is considered noise)
 - Hybrid
- **Re-start** hyper-parameter optimization from random initialization to mitigate the problem
- With increasing data set size the GP typically ends up in the “hybrid” mode. Other modes are unlikely.
- Ideally, we would integrate the hyper-parameters out
No closed-form solution ➔ Markov chain Monte Carlo

Why Does the Marginal Likelihood Work?

- Overall goal: Good generalization performance on unseen test data

- Overall goal: Good generalization performance on unseen test data
- Minimizing training error is not a good idea (e.g., maximum likelihood) ► Overfitting
- Just adding uncertainty does not help either if the model is wrong, but it makes predictions more cautious

- Overall goal: Good generalization performance on unseen test data
- Minimizing training error is not a good idea (e.g., maximum likelihood) ► Overfitting
- Just adding uncertainty does not help either if the model is wrong, but it makes predictions more cautious
- Marginal likelihood seems to find a good balance between fitting the data and finding a simple model (Occam's razor)

Why does the marginal likelihood lead to models that generalize well?

- “Probability of the training data” given the parameters
- General factorization (ignoring inputs X):

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(y_1, \dots, y_N|\boldsymbol{\theta})$$

- “Probability of the training data” given the parameters
- General factorization (ignoring inputs X):

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(y_1, \dots, y_N|\boldsymbol{\theta})$$

$$= p(y_1|\boldsymbol{\theta})p(y_2|y_1, \boldsymbol{\theta})p(y_3|y_1, y_2, \boldsymbol{\theta}) \cdot \dots \cdot p(y_N|y_1, \dots, y_{N-1}, \boldsymbol{\theta})$$

- “Probability of the training data” given the parameters
- General factorization (ignoring inputs X):

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(y_1, \dots, y_N|\boldsymbol{\theta})$$

$$= p(y_1|\boldsymbol{\theta})p(y_2|y_1, \boldsymbol{\theta})p(y_3|y_1, y_2, \boldsymbol{\theta}) \cdot \dots \cdot p(y_N|y_1, \dots, y_{N-1}, \boldsymbol{\theta})$$

$$= p(y_1|\boldsymbol{\theta}) \prod_{n=2}^N p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^N p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^N p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

- If we think of this as a sequence model (where data arrives sequentially), the marginal likelihood predicts the n th training observation given all “previous” observations

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^N p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

- If we think of this as a sequence model (where data arrives sequentially), the marginal likelihood predicts the n th training observation given all “previous” observations
- Predict training data y_n that has not been accounted for (we only condition on y_1, \dots, y_{n-1}) ➤ Treat next data point as test data

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^N p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

- If we think of this as a sequence model (where data arrives sequentially), the marginal likelihood predicts the n th training observation given all “previous” observations
- Predict training data y_n that has not been accounted for (we only condition on y_1, \dots, y_{n-1}) ➤ Treat next data point as test data
- Intuition: If it continuously predicted well on all N previous points, it probably will do well next time

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^N p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

- If we think of this as a sequence model (where data arrives sequentially), the marginal likelihood predicts the n th training observation given all “previous” observations
- Predict training data y_n that has not been accounted for (we only condition on y_1, \dots, y_{n-1}) ➤ Treat next data point as test data
- Intuition: If it continuously predicted well on all N previous points, it probably will do well next time
 - Proxy for generalization error on unseen test data

$$p(\mathbf{y}|\boldsymbol{\theta}) = p(y_1, \dots, y_N|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^N p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

- Short length-scale

²Thanks to Mark van der Wilk

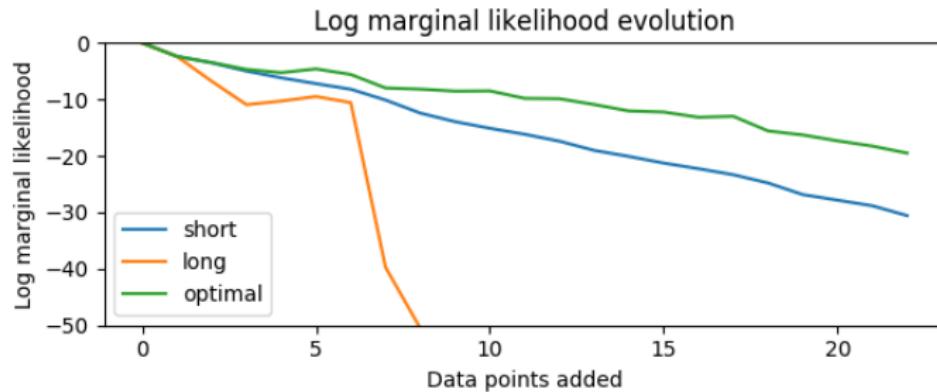
- Long length-scale

³Thanks to Mark van der Wilk

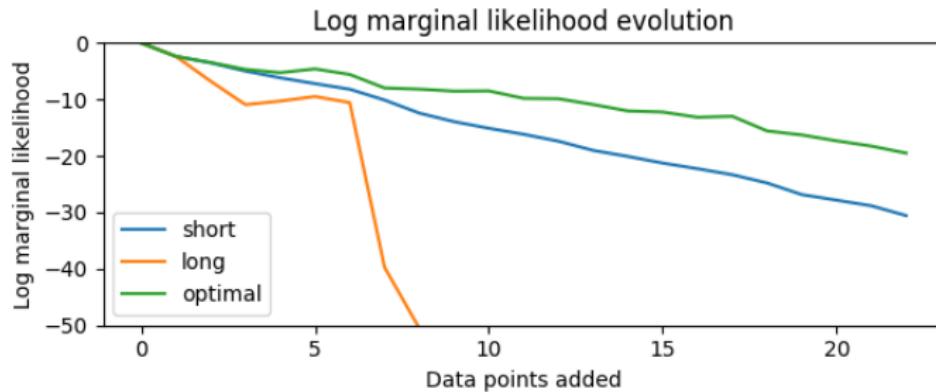
- Optimal length-scale

⁴Thanks to Mark van der Wilk

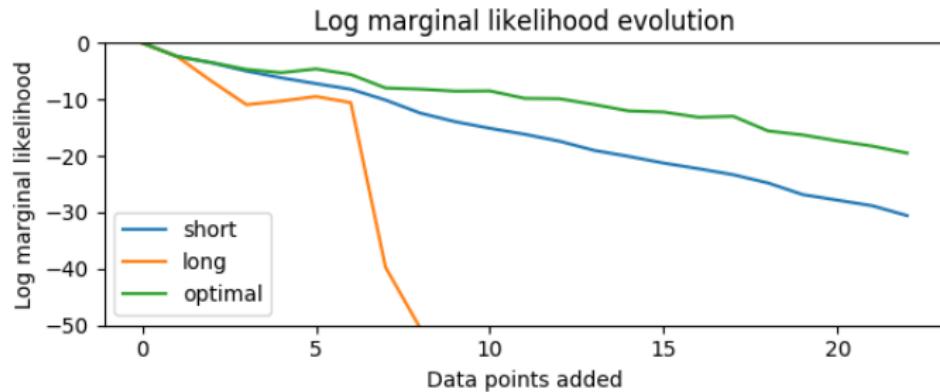
Marginal Likelihood Evolution



- Short lengthscale: consistently **overestimates variance**
 - ▶ No high density, even with observations inside the error bars



- Short lengthscale: consistently **overestimates variance**
 - ▶ No high density, even with observations inside the error bars
- Long lengthscale: consistently **underestimates variance**
 - ▶ Low density because observations are outside the error bars

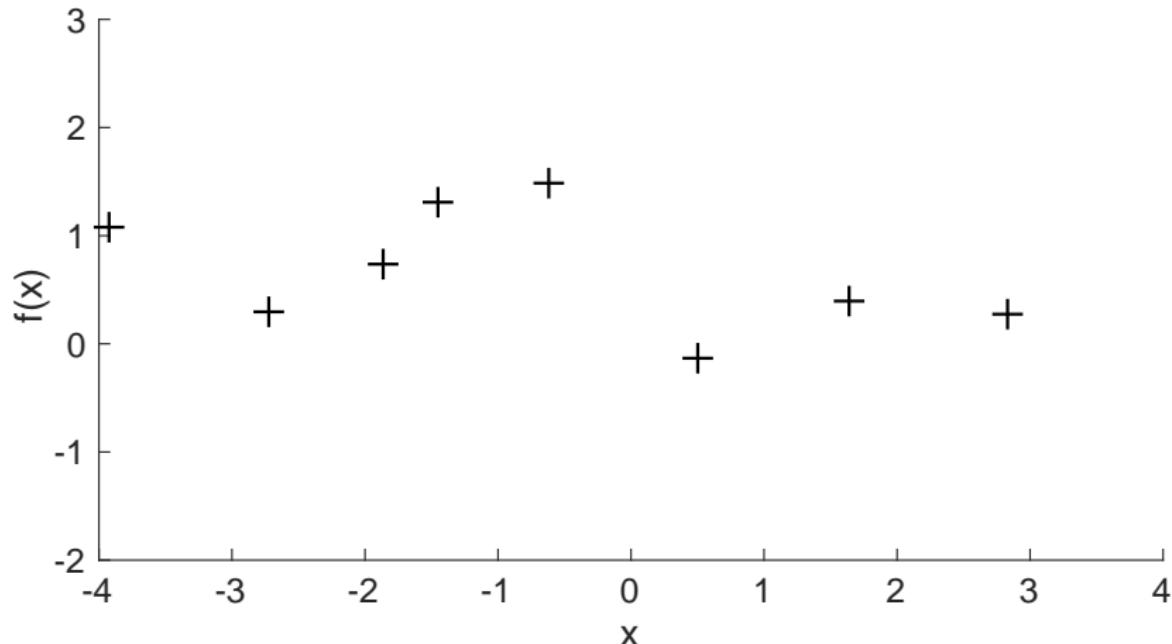


- Short lengthscale: consistently **overestimates variance**
 - ▶ No high density, even with observations inside the error bars
- Long lengthscale: consistently **underestimates variance**
 - ▶ Low density because observations are outside the error bars
- Optimal lengthscale: **trades off both behaviors reasonably well**

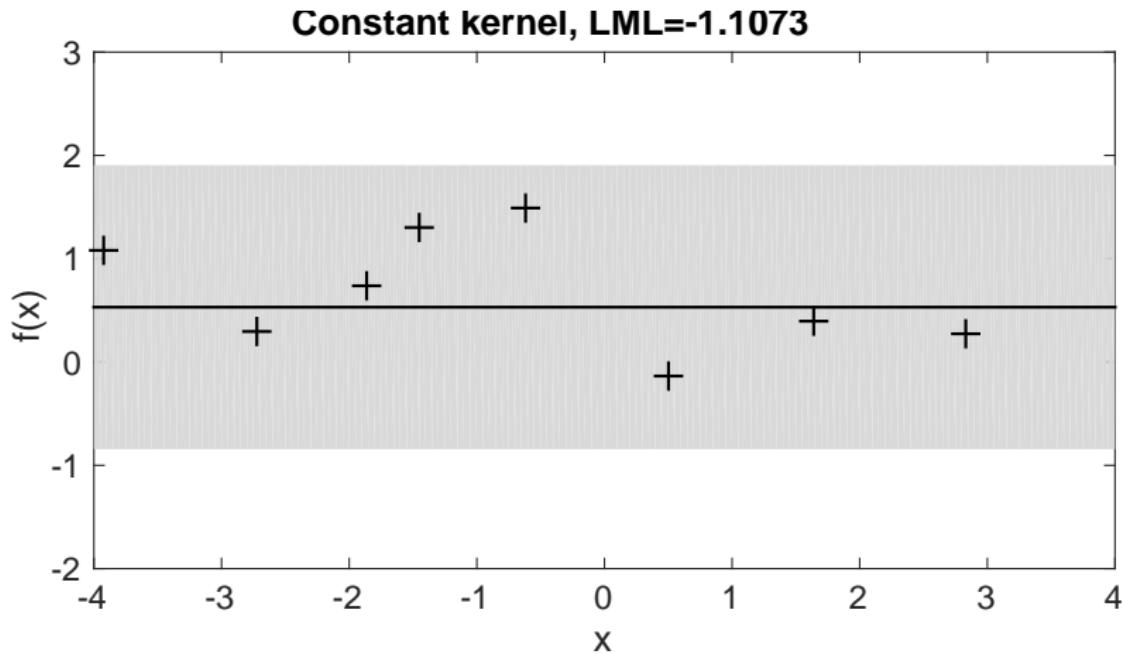
- Assume we have a finite set of models M_i , each one specifying a mean function m_i and a kernel k_i . How do we find the best one?

- Assume we have a finite set of models M_i , each one specifying a mean function m_i and a kernel k_i . How do we find the best one?
- Some options:
 - Cross validation
 - Bayesian Information Criterion, Akaike Information Criterion
 - Compare marginal likelihood values (assuming a uniform prior on the set of models)

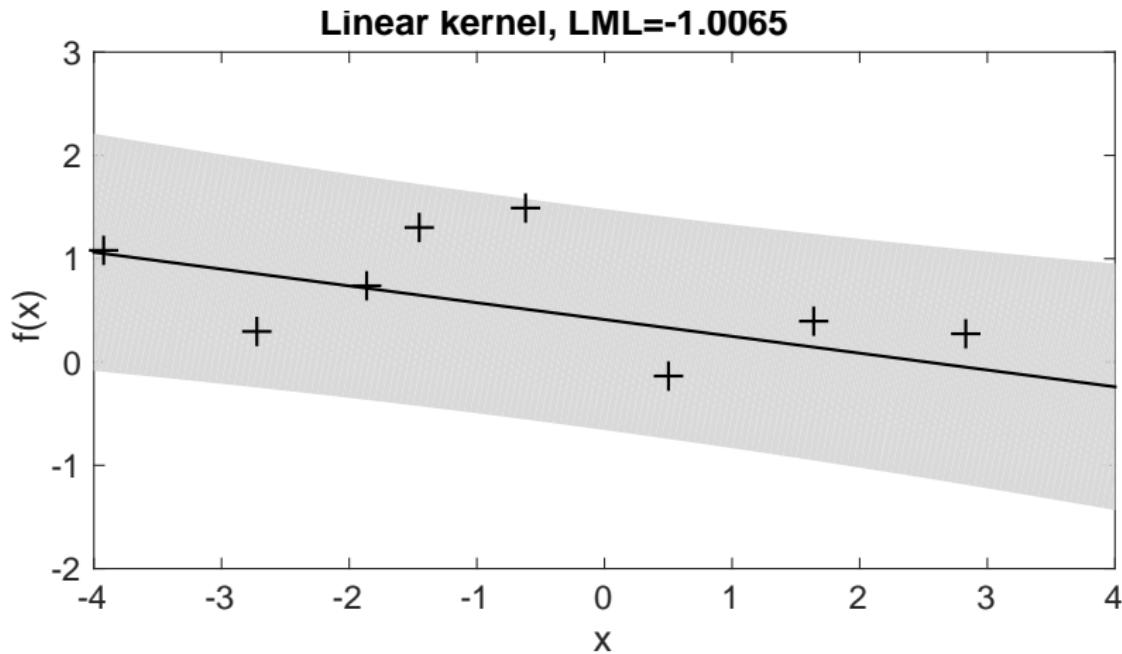
Example



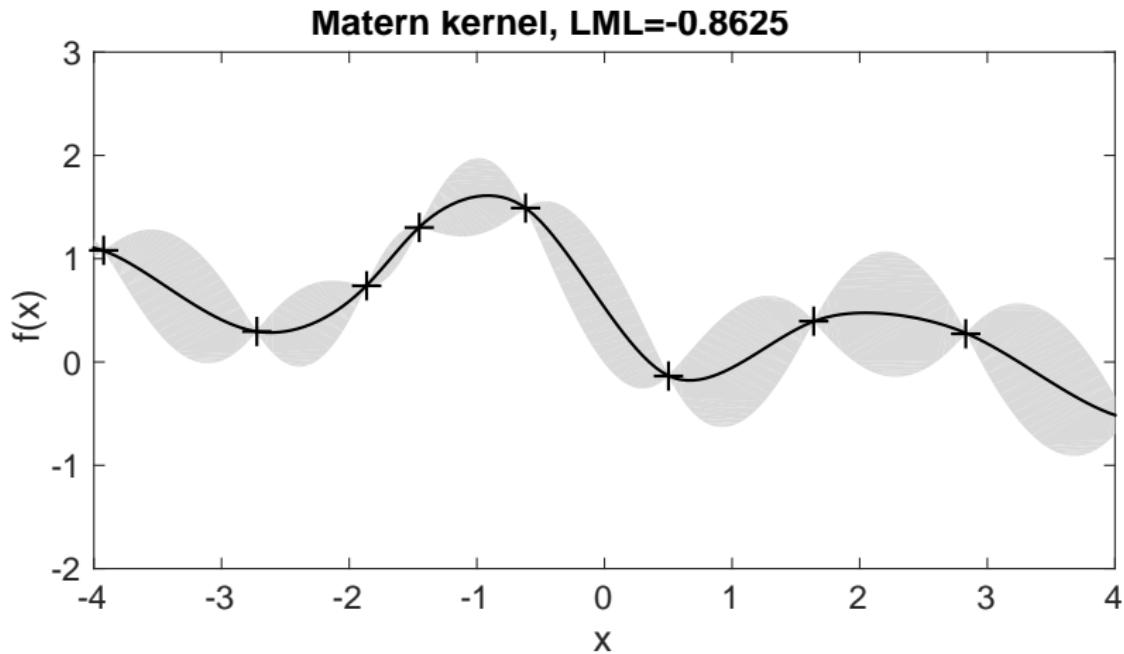
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model



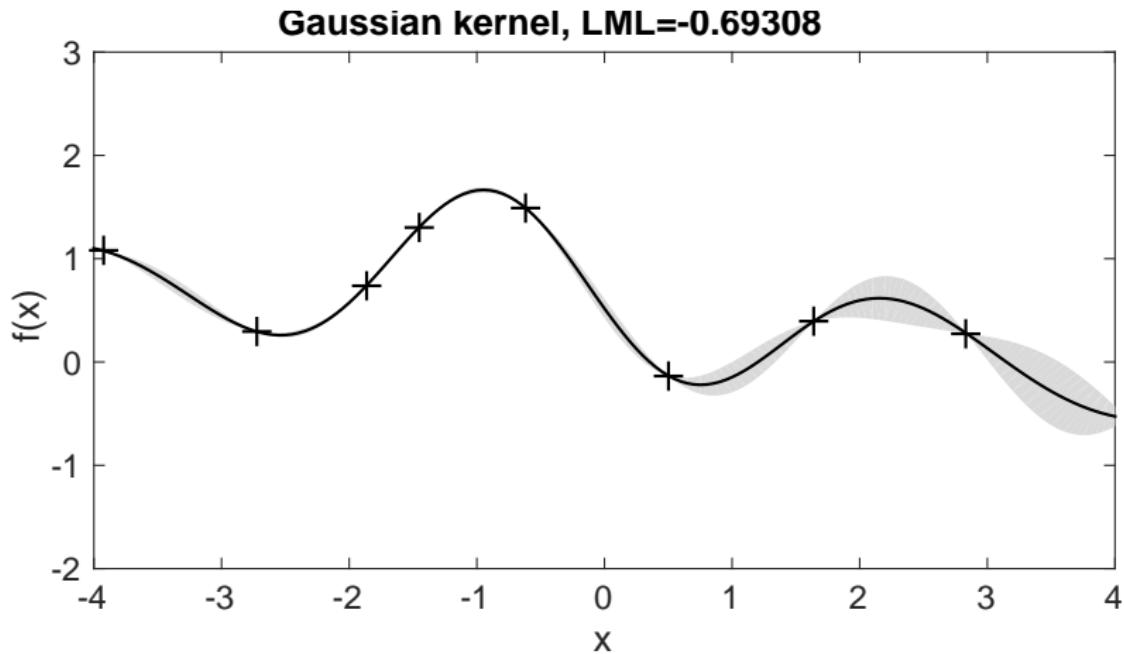
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model



- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model



- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model



- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

- Prior: $f(\boldsymbol{x}) = \theta_s f_{\text{smooth}}(\boldsymbol{x}) + \theta_p f_{\text{periodic}}(\boldsymbol{x})$, with smooth and periodic GP priors, respectively.

⁵Thanks to Mark van der Wilk

- Prior: $f(\mathbf{x}) = \theta_s f_{\text{smooth}}(\mathbf{x}) + \theta_p f_{\text{periodic}}(\mathbf{x})$, with smooth and periodic GP priors, respectively.
- Amount of periodicity vs. smoothness is automatically chosen by selecting hyper-parameters θ_s, θ_p .
- Marginal likelihood learns how to generalize, not just to fit the data

⁵Thanks to Mark van der Wilk

Limitations and Guidelines

Computational and memory complexity

Training set size: N

- Training scales in $\mathcal{O}(N^3)$
- Prediction (variances) scales in $\mathcal{O}(N^2)$
- Memory requirement: $\mathcal{O}(ND + N^2)$

► **Practical limit** $N \approx 10,000$

Computational and memory complexity

Training set size: N

- Training scales in $\mathcal{O}(N^3)$
- Prediction (variances) scales in $\mathcal{O}(N^2)$
- Memory requirement: $\mathcal{O}(ND + N^2)$

► **Practical limit** $N \approx 10,000$

Some solution approaches:

- Sparse GPs with **inducing variables** (e.g., Snelson & Ghahramani, 2006; Quiñonero-Candela & Rasmussen, 2005; Titsias 2009; Hensman et al., 2013; Matthews et al., 2016)
- Combination of **local GP expert models** (e.g., Tresp 2000; Cao & Fleet 2014; Deisenroth & Ng, 2015)
- **Variational Fourier features** (Hensman et al., 2018)

- To set initial hyper-parameters, use [domain knowledge](#).

► <https://drafts.distill.pub/gp>

- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5 .

► <https://drafts.distill.pub/gp>

- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5 .
- Standardize targets y and set initial signal variance to $\sigma_f \approx 1$.

► <https://drafts.distill.pub/gp>

- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5 .
- Standardize targets y and set initial signal variance to $\sigma_f \approx 1$.
- Often useful: Set initial noise level relatively high (e.g., $\sigma_n \approx 0.5 \times \sigma_f$ amplitude), even if you think your data have low noise. The optimization surface for your other parameters will be easier to move in.

► <https://drafts.distill.pub/gp>

- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5 .
- Standardize targets y and set initial signal variance to $\sigma_f \approx 1$.
- Often useful: Set initial noise level relatively high (e.g., $\sigma_n \approx 0.5 \times \sigma_f$ amplitude), even if you think your data have low noise. The optimization surface for your other parameters will be easier to move in.
- When optimizing hyper-parameters, try random restarts or other tricks to avoid local optima are advised.

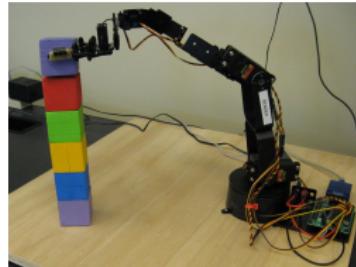
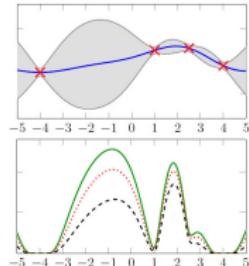
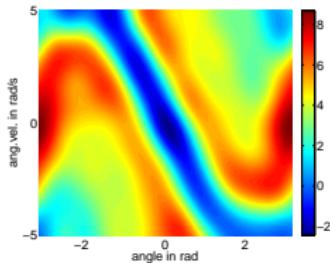
► <https://drafts.distill.pub/gp>

- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5 .
- Standardize targets y and set initial signal variance to $\sigma_f \approx 1$.
- Often useful: Set initial noise level relatively high (e.g., $\sigma_n \approx 0.5 \times \sigma_f$ amplitude), even if you think your data have low noise. The optimization surface for your other parameters will be easier to move in.
- When optimizing hyper-parameters, try random restarts or other tricks to avoid local optima are advised.
- Mitigate the problem of numerical instability (Cholesky decomposition of $\mathbf{K} + \sigma_n^2 \mathbf{I}$) by penalizing high signal-to-noise ratios σ_f/σ_n

► <https://drafts.distill.pub/gp>

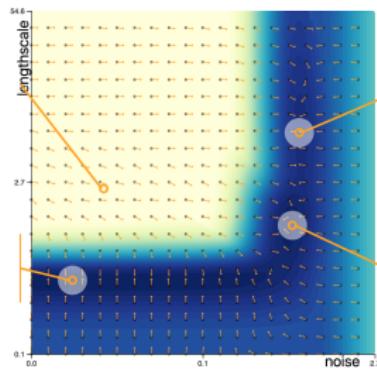
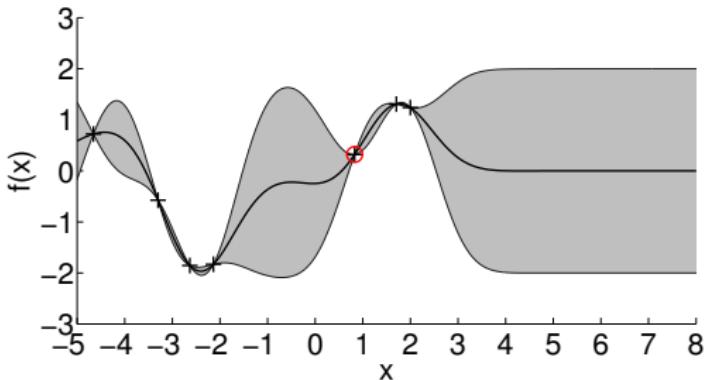
Application Areas

Application Areas



- Reinforcement learning and robotics
 - ▶ Model value functions and/or dynamics with GPs
- Bayesian optimization (Experimental Design)
 - ▶ Model unknown utility functions with GPs
- Geostatistics
 - ▶ Spatial modeling (e.g., landscapes, resources)
- Sensor networks
- Time-series modeling and forecasting

Summary



- Gaussian processes are the **gold-standard** for regression
- Closely related to Bayesian linear regression
- Computations boil down to **manipulating multivariate Gaussian distributions**
- Marginal likelihood objective automatically trades off data fit and model complexity

References I

- [1] G. Bertone, M. P. Deisenroth, J. S. Kim, S. Liem, R. R. de Austri, and M. Welling. Accelerating the BSM Interpretation of LHC Data with Machine Learning. arXiv preprint arXiv:1611.02704, 2016.
- [2] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian Processes for Regression. In *Proceedings of the International Joint Conference on Neural Networks*, 2016.
- [3] Y. Cao and D. J. Fleet. Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process Predictions. <http://arxiv.org/abs/1410.7827>, 2014.
- [4] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley-Interscience, 1993.
- [5] M. Cutler and J. P. How. Efficient Reinforcement Learning for Robots using Informative Simulated Priors. In *Proceedings of the International Conference on Robotics and Automation*, 2015.
- [6] M. P. Deisenroth and J. W. Ng. Distributed Gaussian Processes. In *Proceedings of the International Conference on Machine Learning*, 2015.
- [7] M. P. Deisenroth, C. E. Rasmussen, and D. Fox. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, 2011.
- [8] M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian Process Dynamic Programming. *Neurocomputing*, 72(7–9):1508–1524, Mar. 2009.
- [9] M. P. Deisenroth, R. Turner, M. Huber, U. D. Hanebeck, and C. E. Rasmussen. Robust Filtering and Smoothing with Gaussian Processes. *IEEE Transactions on Automatic Control*, 57(7):1865–1871, 2012.
- [10] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen. Bayesian Inference and Learning in Gaussian Process State-Space Models with Particle MCMC. In *Advances in Neural Information Processing Systems*. 2013.
- [11] N. HajiGhassemi and M. P. Deisenroth. Approximate Inference for Long-Term Forecasting with Periodic Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2014.
- [12] J. Hensman, N. Durrande, and A. Solin. Variational Fourier Features for Gaussian Processes. *Journal of Machine Learning Research*, pages 1–52, 2018.

References II

- [13] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian Processes for Big Data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2013.
- [14] A. Krause, A. Singh, and C. Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9:235–284, Feb. 2008.
- [15] M. C. H. Lee, H. Salimbeni, M. P. Deisenroth, and B. Glocker. Patch Kernels for Gaussian Processes in High-Dimensional Imaging Problems. In *NIPS Workshop on Practical Bayesian Nonparametrics*, 2016.
- [16] J. R. Lloyd, D. Duvenaud, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Automatic Construction and Natural-Language Description of Nonparametric Regression Models. In *AAAI Conference on Artificial Intelligence*, pages 1–11, 2014.
- [17] D. J. C. MacKay. Introduction to Gaussian Processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, volume 168, pages 133–165. Springer, 1998.
- [18] A. G. d. G. Matthews, J. Hensman, R. Turner, and Z. Ghahramani. On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2016.
- [19] M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings. Towards Real-Time Information Processing of Sensor Network Data Using Computationally Efficient Multi-output Gaussian Processes. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 109–120. IEEE Computer Society, 2008.
- [20] J. Quiñonero-Candela and C. E. Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6(2):1939–1960, 2005.
- [21] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [22] S. Roberts, M. A. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian Processes for Time Series Modelling. *Philosophical Transactions of the Royal Society (Part A)*, 371(1984), 2013.

References III

- [23] B. Schölkopf and A. J. Smola. *Learning with Kernels—Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 2002.
- [24] E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. The MIT Press, Cambridge, MA, USA, 2006.
- [25] M. K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.
- [26] V. Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000.
- [27] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep Kernel Learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2016.