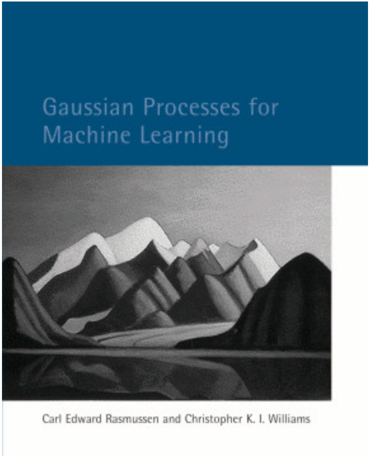
Gaussian Processes

Recommended reading:
Rasmussen/Williams: Chapters 1, 2, 4, 5

Marc Deisenroth

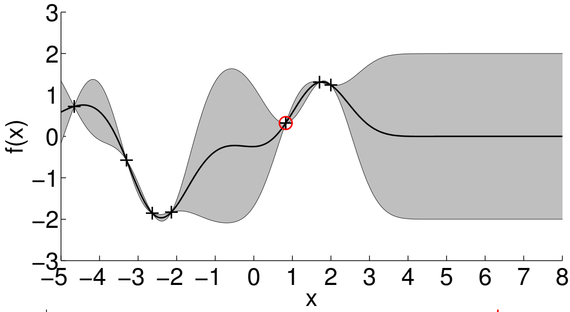
Department of Computing
Imperial College London

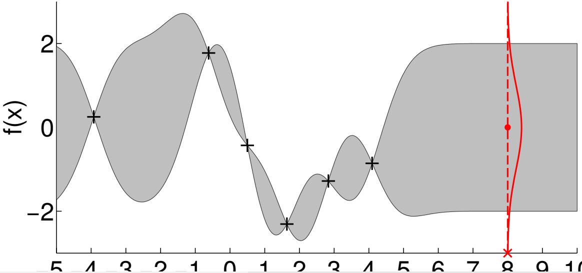
February 13, 2017



http://www.gaussianprocess.org/

Problem Setting





Recap from CO-496: Bayesian Linear Regression

- Linear Regression Model:

$$f(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \boldsymbol{w}, \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_p)$$

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

- Integrating out the parameters when predicting leads to a distribution over functions:

$$p(f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})d\boldsymbol{w}$$

$$= \mathcal{N}(\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$$

$$\mu(\boldsymbol{x}_*) = \phi_*^\top \boldsymbol{\Sigma}_p \boldsymbol{\Phi}(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}\boldsymbol{y}$$

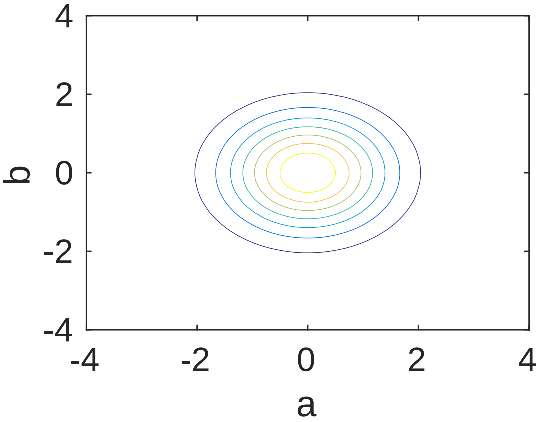
$$\sigma^2(\boldsymbol{x}_*) = \phi_*^\top \boldsymbol{\Sigma}_p \phi_* - \phi_*^\top \boldsymbol{\Sigma}_p \boldsymbol{\Phi}(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}\boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_p \phi_*$$

$$\boldsymbol{K} = \boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_p \boldsymbol{\Phi}$$

Sampling from the Prior over Functions

Consider a linear regression setting

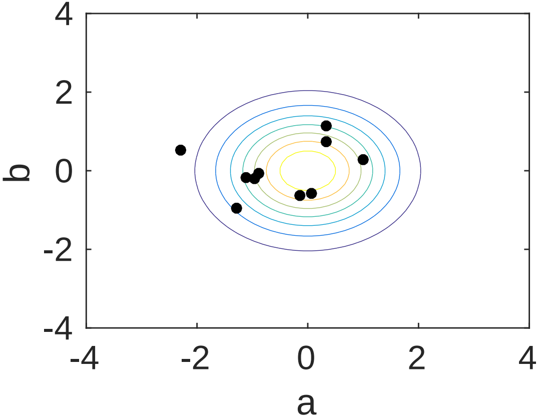
$$y = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$
$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$



Sampling from the Prior over Functions

Consider a linear regression setting

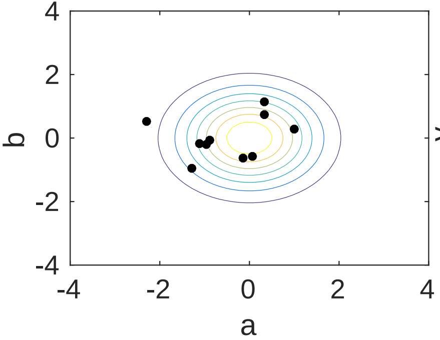
$$y = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$
$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$

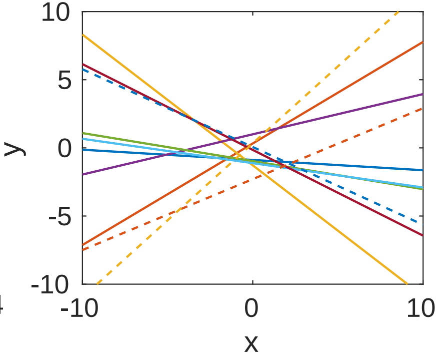


Sampling from the Prior over Functions

Consider a linear regression setting

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

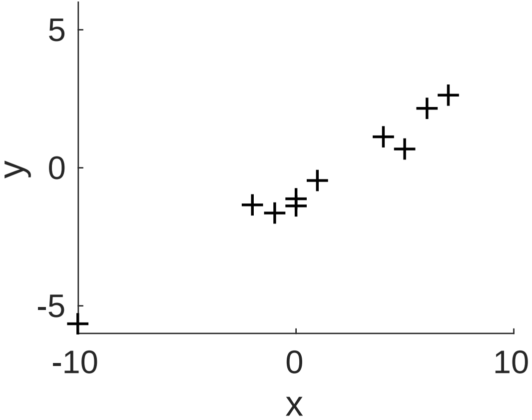




Sampling from the Prior over Functions

Consider a linear regression setting

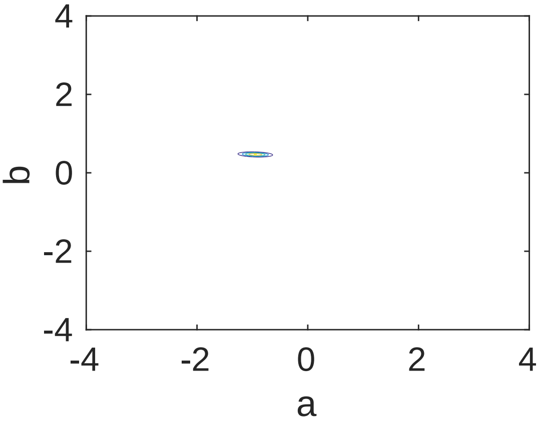
$$y = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$
$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$



Sampling from the Posterior over Functions

Consider a linear regression setting

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

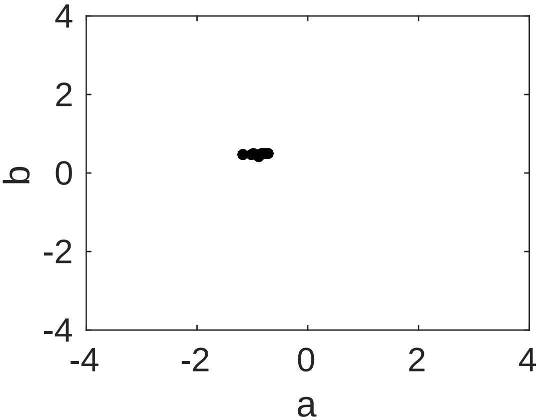


Sampling from the Posterior over Functions

Consider a linear regression setting

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

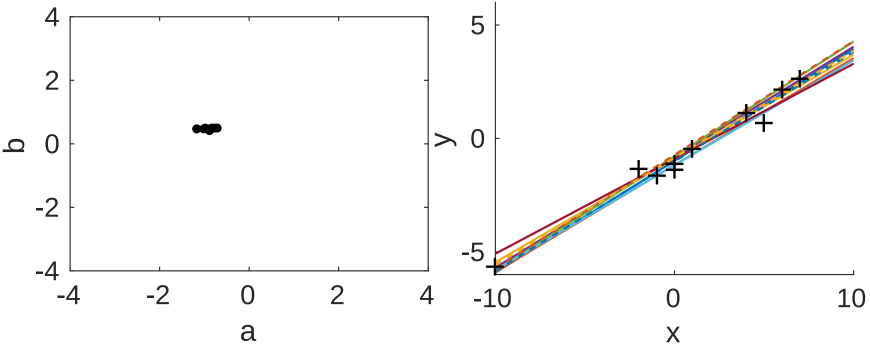
$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$



Sampling from the Posterior over Functions

Consider a linear regression setting

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



Fitting Nonlinear Functions

- Fit nonlinear functions using (Bayesian) linear regression:
  Linear combination of nonlinear features
- Example: Radial-basis-function (RBF) network

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \phi_i(\boldsymbol{x}) \,, \quad w_i \sim \mathcal{N}\big(0,\, \sigma_p^2\big)$$

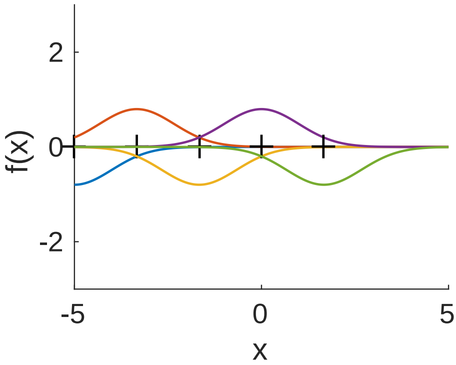
where

$$\phi_i(\boldsymbol{x}) = \exp\big(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top(\boldsymbol{x}-\boldsymbol{\mu}_i)\big)$$

for given "centers" $\boldsymbol{\mu}_i$

Illustration: Fitting a Radial Basis Function Network

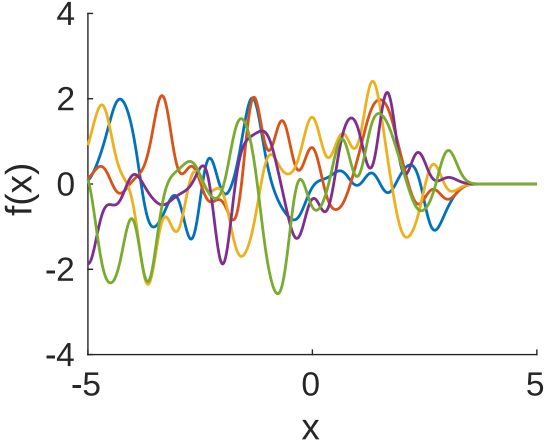
$$\phi_i(\boldsymbol{x}) = \exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top(\boldsymbol{x}-\boldsymbol{\mu}_i)\right)$$



- Place Gaussian-shaped basis functions ϕ_i at 25 input locations μ_i, linearly spaced in the interval $[-5, 3]$

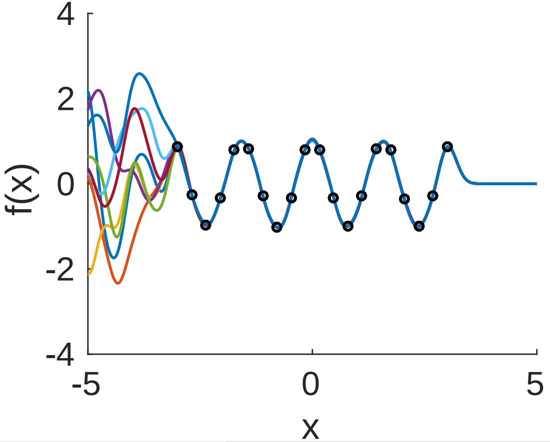
Samples from the RBF Prior

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \phi_i(\boldsymbol{x}), \quad p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

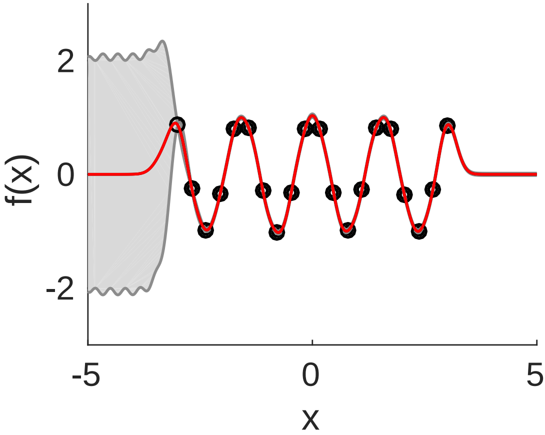


Samples from the RBF Posterior

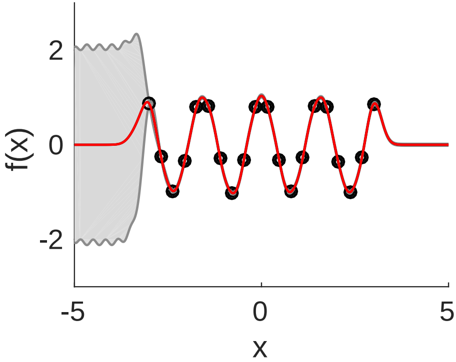
$$f(\boldsymbol{x}) = \sum_{i=1}^{n} w_i \phi_i(\boldsymbol{x}), \quad p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\big(\boldsymbol{m}_N, \boldsymbol{S}_N\big)$$



RBF Posterior



Limitations



- Feature engineering
- Finite number of features:
  - Above: Without basis functions on the right, we cannot express any variability of the function
  - Ideally: Add more (infinitely many) basis functions

Approach

- Instead of sampling parameters, which induce a distribution over functions, sample functions directly
  ⏩ 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

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

Definition

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

- A Gaussian distribution is specified by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$
- A Gaussian process is specified by a mean function $m(\cdot)$ and a covariance function (kernel) $k(\cdot, \cdot)$

Covariance Function

- The covariance function (kernel) is symmetric and positive semi-definite
- It allows us to compute covariances between (unknown) function values by just looking at the corresponding inputs:

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

GP Regression as a Bayesian Inference Problem

Objective

For a set of observations $y_i = f(\boldsymbol{x}_i) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$, find a (posterior) distribution over functions $p(f|\boldsymbol{X}, \boldsymbol{y})$ that explains the data

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

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

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

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

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

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

Prior over Functions

- Treat a function as a long vector of function values:

$$f = [f_1, f_2, \dots]$$

⏩ Look at a distribution over function values $f_i = f(\boldsymbol{x}_i)$

- Consider a finite number of N function values $\boldsymbol{f}$ and all other (infinitely many) function values $\tilde{\boldsymbol{f}}$. Informally:

$$p(\boldsymbol{f}, \tilde{\boldsymbol{f}}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_f \\ \boldsymbol{\mu}_{\tilde{f}} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{ff} & \boldsymbol{\Sigma}_{f\tilde{f}} \\ \boldsymbol{\Sigma}_{\tilde{f}f} & \boldsymbol{\Sigma}_{\tilde{f}\tilde{f}} \end{bmatrix}\right)$$

where $\boldsymbol{\Sigma}_{\tilde{f}\tilde{f}} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{\Sigma}_{f\tilde{f}} \in \mathbb{R}^{N \times m}$, $m \to \infty$.

- $\Sigma_{ff}^{(i,j)} = \text{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- Key property: The marginal remains finite

$$p(\boldsymbol{f}) = \int p(\boldsymbol{f}, \tilde{\boldsymbol{f}}) d\tilde{\boldsymbol{f}} = \mathcal{N}(\boldsymbol{\mu}_f, \boldsymbol{\Sigma}_{ff})$$

Training and Test Marginal

- In practice, we always have finite training and test inputs $\boldsymbol{x}_{\text{train}}, \boldsymbol{x}_{\text{test}}$.
- Define $\boldsymbol{f}_* := \boldsymbol{f}_{\text{test}}, \boldsymbol{f} := \boldsymbol{f}_{\text{train}}$.
- Then, we obtain the finite marginal

$$p(\boldsymbol{f}, \boldsymbol{f}_*) = \int p(\boldsymbol{f}, \boldsymbol{f}_*, \boldsymbol{f}_{\text{other}}) d\boldsymbol{f}_{\text{other}} = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_f \\ \boldsymbol{\mu}_*\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{ff} & \boldsymbol{\Sigma}_{f*} \\ \boldsymbol{\Sigma}_{*f} & \boldsymbol{\Sigma}_{**}\end{bmatrix}\right)$$

GP Regression as a Bayesian Inference Problem (ctd.)

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

$$p(f|\boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y}|f, \boldsymbol{X})\; p(f|\boldsymbol{X})}{p(\boldsymbol{y}|\boldsymbol{X})}$$

Using the properties of Gaussians, we obtain

$$p(\boldsymbol{y}|f, \boldsymbol{X})\; p(f|\boldsymbol{X}) = \mathcal{N}\big(\boldsymbol{y}\,|\,f(\boldsymbol{X}),\, \sigma_n^2 \boldsymbol{I}\big)\; \mathcal{N}\big(f(\boldsymbol{X})\,|\,m(\boldsymbol{X}),\, \boldsymbol{K}\big)$$

$$= Z\mathcal{N}\Big(f(\boldsymbol{X})\,|\,\underbrace{m(\boldsymbol{X}) + \boldsymbol{K}(\boldsymbol{K} + \sigma_n^2\boldsymbol{I})^{-1}(\boldsymbol{y} - m(\boldsymbol{X}))}_{\text{posterior mean}},\, \underbrace{\boldsymbol{K} - \boldsymbol{K}(\boldsymbol{K} + \sigma_n^2\boldsymbol{I})^{-1}\boldsymbol{K}}_{\text{posterior covariance}}\Big)$$

$\boldsymbol{K} = k(\boldsymbol{X}, \boldsymbol{X})$

Marginal likelihood:

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

GP Predictions (1)

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

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

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

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

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

GP Predictions (2)

Prior:

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

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

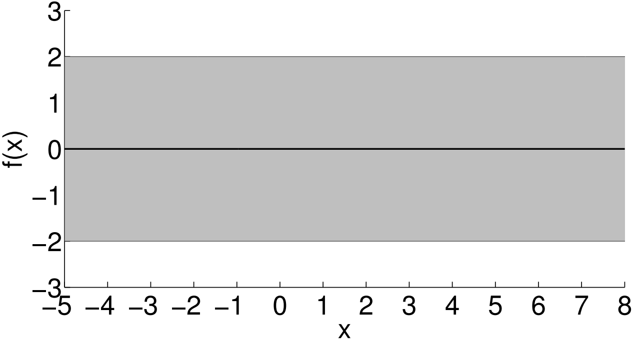
$$p(f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*) = \mathcal{N}\big(\mathbb{E}[f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*], \mathbb{V}[f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*]\big)$$

$$\mathbb{E}[f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*] = m_{\text{post}}(\boldsymbol{X}_*) = \underbrace{m(\boldsymbol{X}_*)}_{\text{prior mean}} + k(\boldsymbol{X}_*, \boldsymbol{X})(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - m(\boldsymbol{X}))$$

$$\begin{aligned} \mathbb{V}[f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*] &= k_{\text{post}}(\boldsymbol{X}_*, \boldsymbol{X}_*) \\ &= \underbrace{k(\boldsymbol{X}_*, \boldsymbol{X}_*)}_{\text{prior variance}} - k(\boldsymbol{X}_*, \boldsymbol{X})(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{X}_*) \end{aligned}$$

From now: Set prior mean function $m \equiv 0$

Illustration: Inference with Gaussian Processes



Prior belief about the function

Predictive (marginal) mean and variance:

$$
\begin{aligned}
\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \varnothing] &= m(\boldsymbol{x}_*) = 0 \\
\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \varnothing] &= \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*)
\end{aligned}
$$

Covariance Function

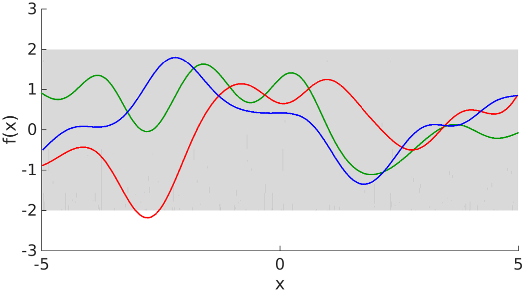
- A Gaussian process is fully specified by a mean function m and a kernel/covariance function k
- The covariance function (kernel) is symmetric and positive semi-definite
- Covariance function encodes high-level structural assumptions about the latent function f (e.g., smoothness, differentiability, periodicity)

Gaussian Covariance Function

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

- 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

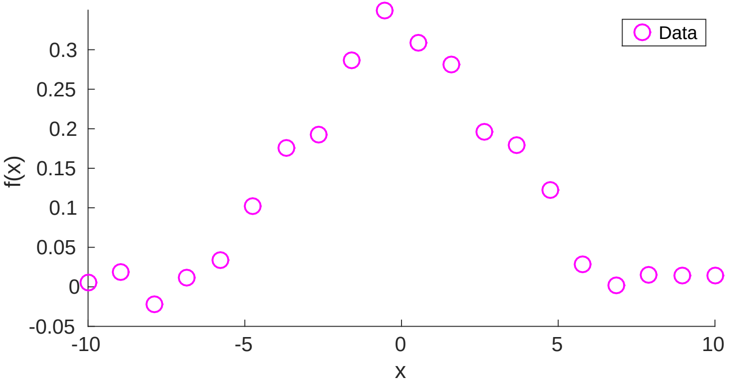
  ▸▸ **Smoothness parameter**

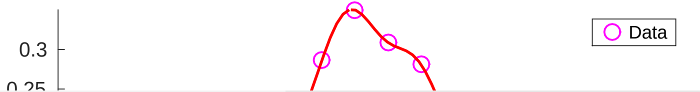


- Assumption on latent function: Smooth (∞ differentiable)

Length-Scales

Length scales determine how wiggly the function is and how much information we can transfer to other function values

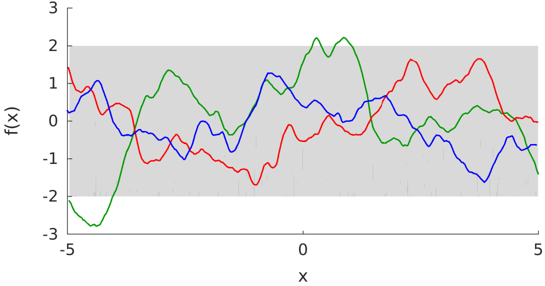




Matérn Covariance Function

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

- 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?

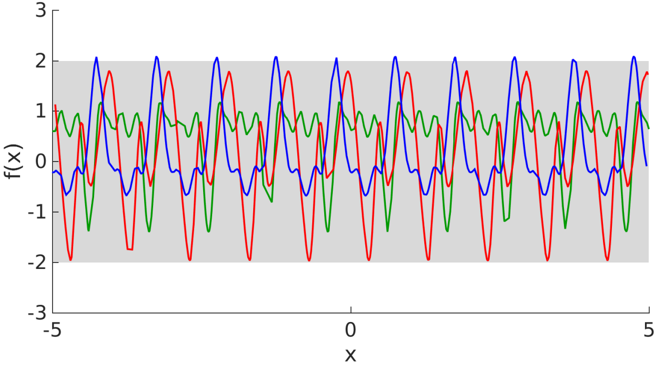


- Assumption on latent function: 1-times differentiable

Periodic Covariance Function

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

κ: Periodicity parameter



Meta-Parameters of a GP

The GP possesses a set of hyper-parameters:

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

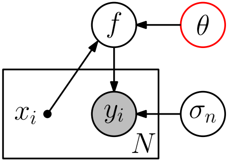
▸▸ Train a GP to find a good set of hyper-parameters

▸▸ Model selection to find good mean and covariance functions
(can also be automated Automatic Statistician (Lloyd et al., 2014))

Gaussian Process Training: Hyper-Parameters

GP Training

Find good GP hyper-parameters $\boldsymbol{\theta}$ (kernel and mean function parameters)



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

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

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

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

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

Training via Marginal Likelihood Maximization

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:

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

$$= \int \mathcal{N}\big(\boldsymbol{y}\,|\,f(\boldsymbol{X}),\,\sigma_n^2\boldsymbol{I}\big)\mathcal{N}\big(f(\boldsymbol{X})\,|\,\boldsymbol{0},\,\boldsymbol{K}\big)df = \mathcal{N}\big(\boldsymbol{y}\,|\,\boldsymbol{0},\,\boldsymbol{K}+\sigma_n^2\boldsymbol{I}\big)$$

Learning the GP hyper-parameters:

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

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

Training via Marginal Likelihood Maximization

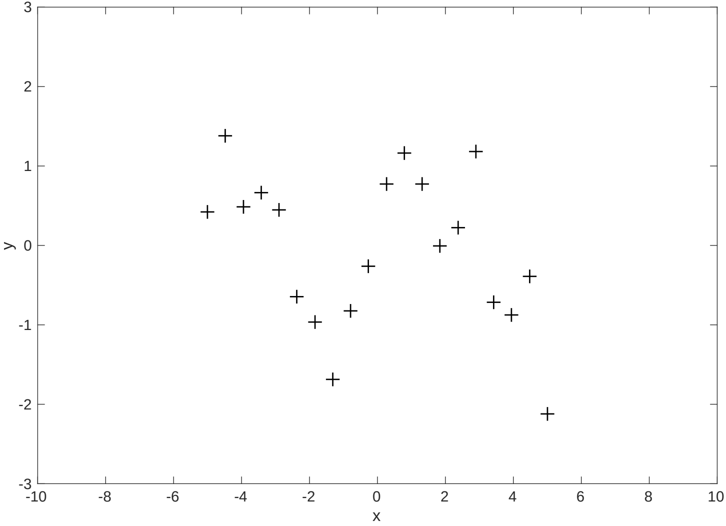
Log-marginal likelihood:

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

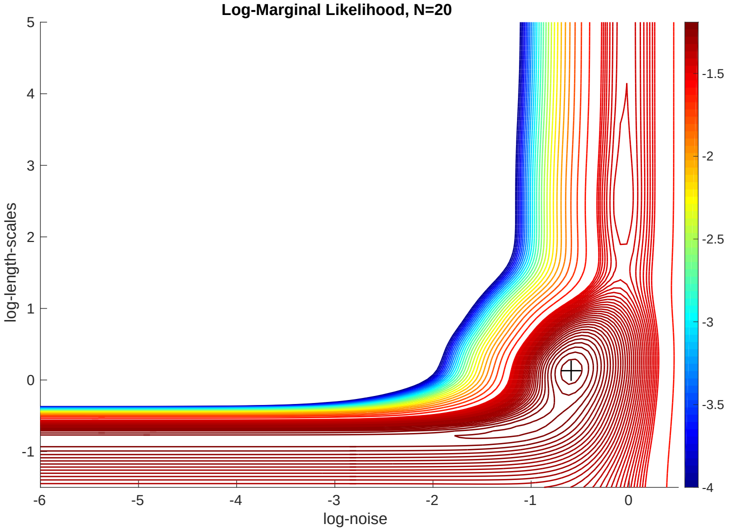
- Automatic trade-off between data fit and model complexity
- Gradient-based optimization of hyper-parameters $\boldsymbol{\theta}$:

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

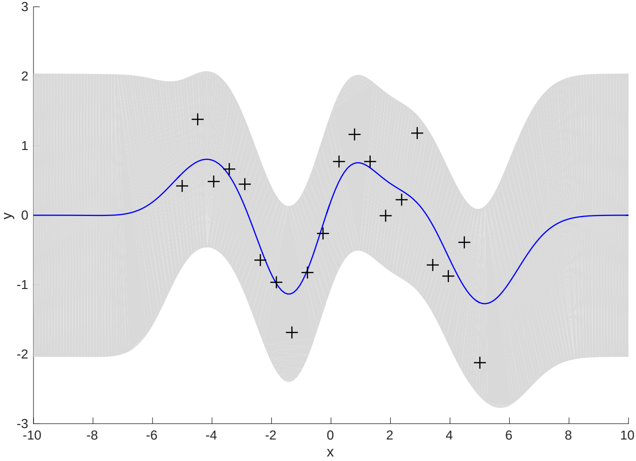
Example: Training Data



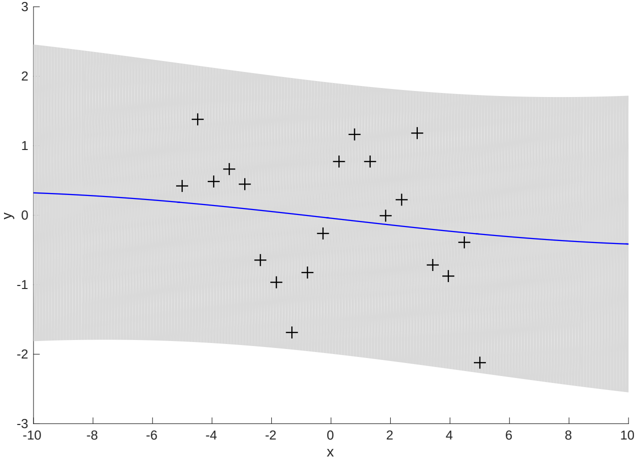
Example: Marginal Likelihood Contour



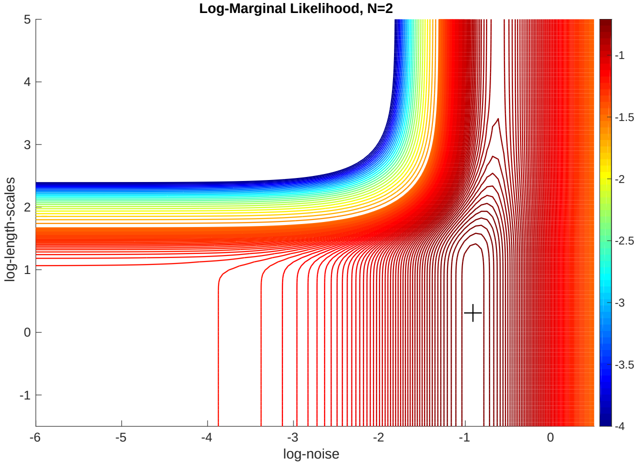
Example: Exploring the Modes (1)



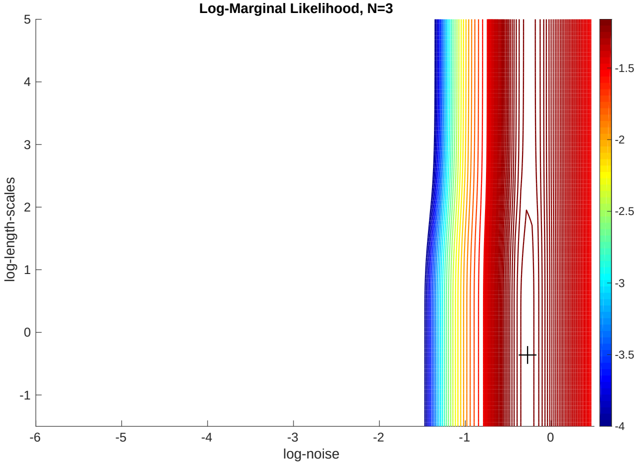
Example: Exploring the Modes (2)



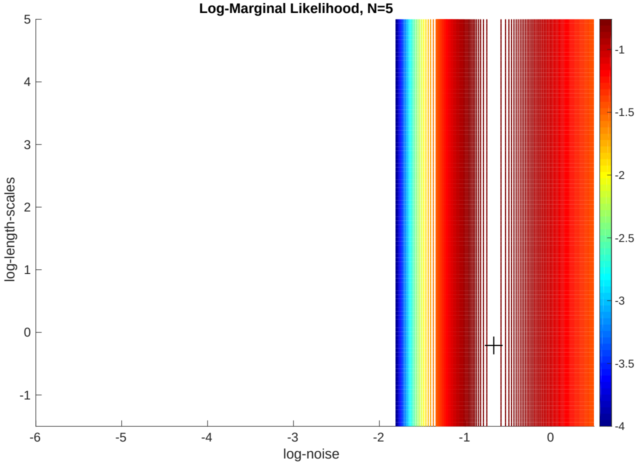
Marginal Likelihood (1)



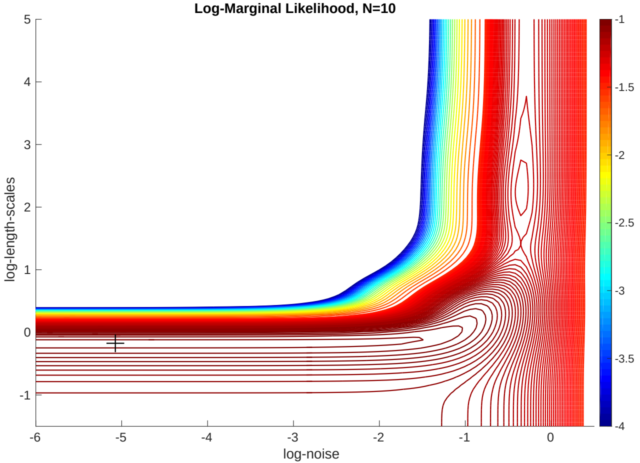
Marginal Likelihood (2)



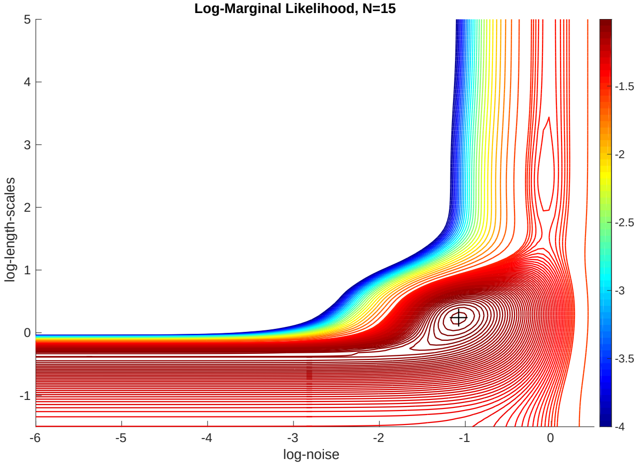
Marginal Likelihood (3)



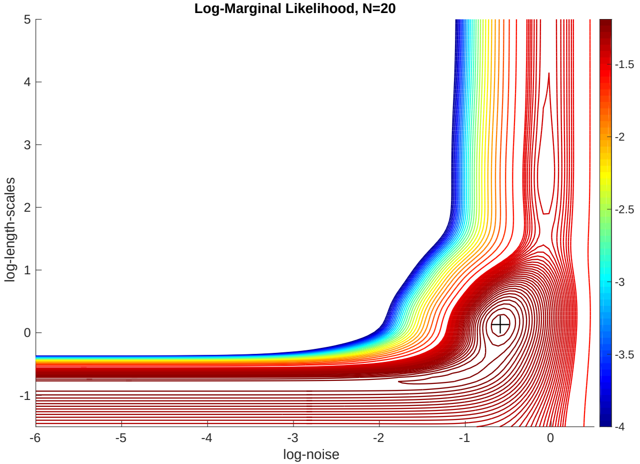
Marginal Likelihood (4)



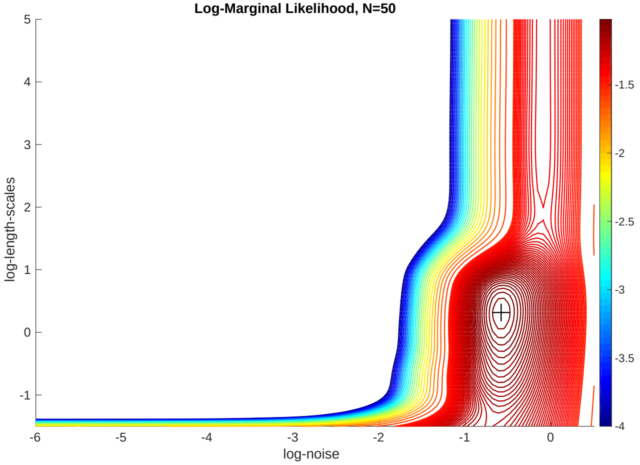
Marginal Likelihood (5)



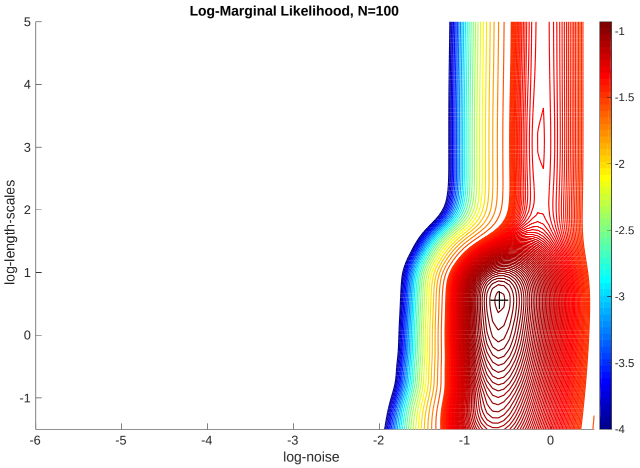
Marginal Likelihood (6)



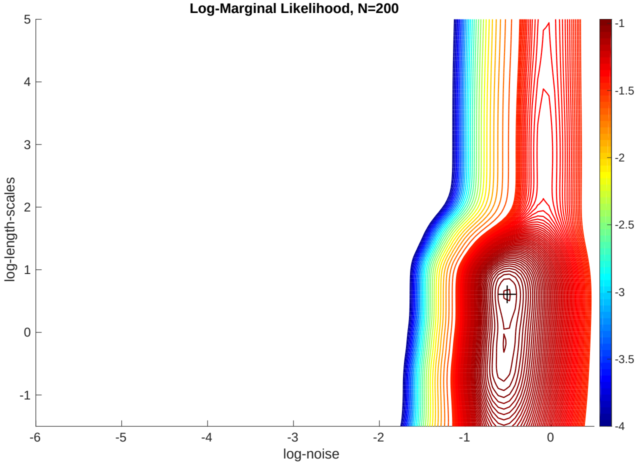
Marginal Likelihood (7)



Marginal Likelihood (8)



Marginal Likelihood (9)



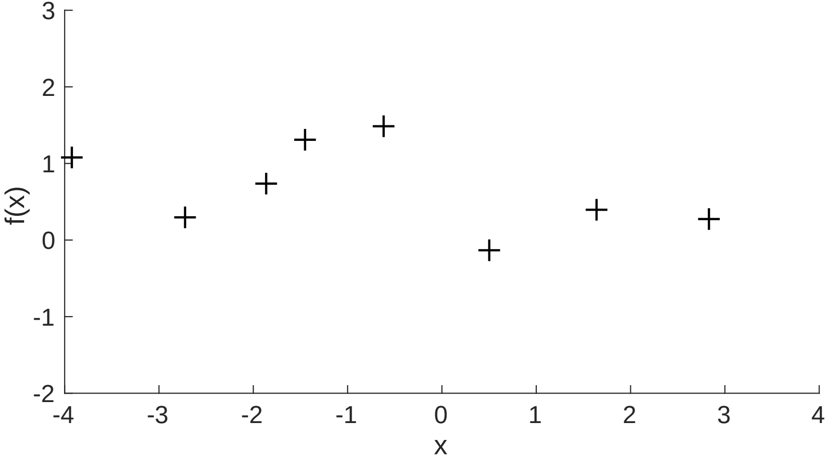
Marginal Likelihood and Parameter Learning

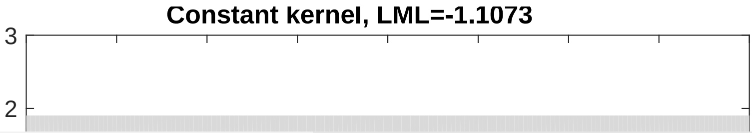
- The marginal likelihood is non-convex
- In particular in the very-small-data regime, a GP can end up in three different modes when optimizing the hyper-parameters:
  - Overfitting (unlikely, but possible)
  - Underfitting (everything is considered noise)
  - Good fit
- Re-start hyper-parameter optimization from random initialization to mitigate the problem
- With increasing data set size the GP typically ends up in the "good-fit" mode. Overfitting (indicator: small length-scales and small noise variance) is very unlikely.
- Ideally, we would integrate the hyper-parameters out
  Why can we do not do this easily?

Model Selection—Mean Function and Kernel

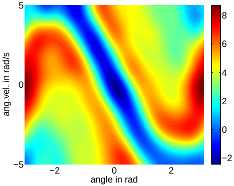
- Assume we have a finite set of models M_i, each one specifying a mean function m_i and a kernel k_i. How do we find the best one?
- Some options:
  - BIC, AIC (see CO-496)
  - Compare marginal likelihood values (assuming a uniform prior on the set of models)

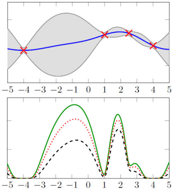
Example

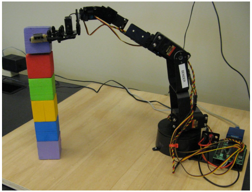




Application Areas







- 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

Limitations of Gaussian Processes

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** $N \approx 10,000$

Tips and Tricks for Practitioners

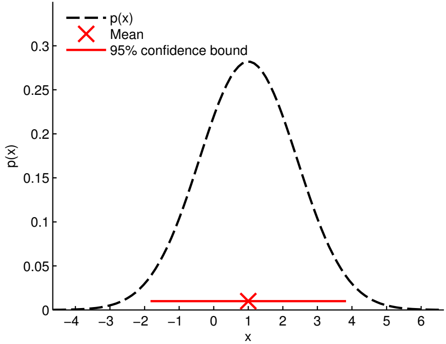
- To set initial hyper-parameters, use domain knowledge if possible.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5.
- Standardize targets y and set initial signal variance to $\sigma_f \approx 1$.
- Often useful: Set initial noise level relatively high (e.g., $\sigma_n \approx 0.5 \times \sigma_f$ 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 $\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$) by penalizing high signal-to-noise ratios σ_f/σ_n

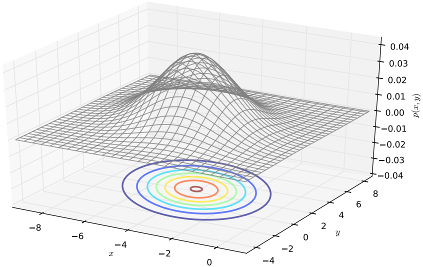
Appendix

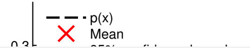
The Gaussian Distribution

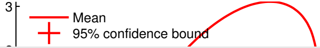
$$p(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}}\exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Mean vector $\boldsymbol{\mu}$ ⏩ Average of the data
- Covariance matrix $\boldsymbol{\Sigma}$ ⏩ Spread of the data









Sampling from a Multivariate Gaussian

Objective

Generate a random sample $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ from a D-dimensional joint Gaussian with covariance matrix $\boldsymbol{\Sigma}$ and mean vector $\boldsymbol{\mu}$.

However, we only have access to a random number generator that can sample $\boldsymbol{x}$ from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$...

Exploit that affine transformations $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ of a Gaussian random variable $\boldsymbol{x}$ remain Gaussian

- Mean: $\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}] = \boldsymbol{A}\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}] + \boldsymbol{b}$
- Covariance: $\mathbb{V}_{\boldsymbol{x}}[\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}] = \boldsymbol{A}\mathbb{V}_{\boldsymbol{x}}[\boldsymbol{x}]\boldsymbol{A}^\top$

1. Find conditions for $\boldsymbol{A}, \boldsymbol{b}$ to match the mean of $\boldsymbol{y}$
2. Find conditions for $\boldsymbol{A}, \boldsymbol{b}$ to match the covariance of $\boldsymbol{y}$

Sampling from a Multivariate Gaussian (2)

Objective

Generate a random sample $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ from a D-dimensional joint Gaussian with covariance matrix $\boldsymbol{\Sigma}$ and mean vector $\boldsymbol{\mu}$.

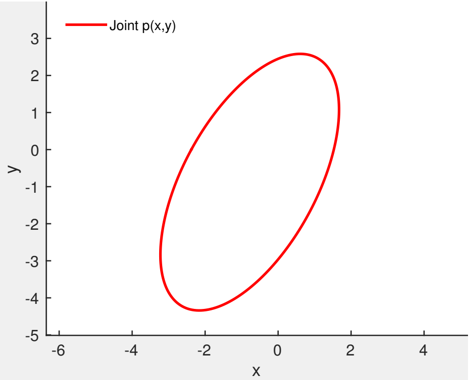
```
x = randn(D,1);          Sample x ~ N(0, I)
y = chol(Σ)'*x + μ;      Scale x and add offset
```

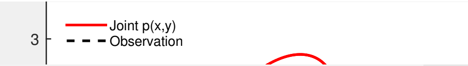
Here `chol(Σ)` is the Cholesky factor $\boldsymbol{L}$, such that $\boldsymbol{L}^\top \boldsymbol{L} = \boldsymbol{\Sigma}$
Therefore, the mean and covariance of $\boldsymbol{y}$ are

$$\mathbb{E}[\boldsymbol{y}] = \bar{\boldsymbol{y}} = \mathbb{E}[\boldsymbol{L}^\top \boldsymbol{x} + \boldsymbol{\mu}] = \boldsymbol{L}^\top \mathbb{E}[\boldsymbol{x}] + \boldsymbol{\mu} = \boldsymbol{\mu}$$

$$\mathrm{Cov}[\boldsymbol{y}] = \mathbb{E}[(\boldsymbol{y} - \bar{\boldsymbol{y}})(\boldsymbol{y} - \bar{\boldsymbol{y}})^\top] = \mathbb{E}[\boldsymbol{L}^\top \boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{L}] = \boldsymbol{L}^\top \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]\boldsymbol{L} = \boldsymbol{L}^\top \boldsymbol{L} = \boldsymbol{\Sigma}$$

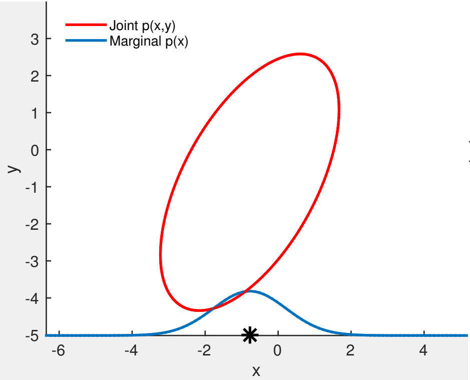
Conditional





$$p(\mathbf{x},\mathbf{y}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy}\end{bmatrix}\right)$$

Marginal



$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}\right)$$

Marginal distribution:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y}$$
$$= \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx})$$

- The marginal of a joint Gaussian distribution is Gaussian
- Intuitively: Ignore (integrate out) everything you are not interested in

The Gaussian Distribution in the Limit

Consider the joint Gaussian distribution $p(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, where $\boldsymbol{x} \in \mathbb{R}^D$ and $\tilde{\boldsymbol{x}} \in \mathbb{R}^k, k \to \infty$ are random variables.
Then

$$p(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_{\tilde{x}} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{x\tilde{x}} \\ \boldsymbol{\Sigma}_{\tilde{x}x} & \boldsymbol{\Sigma}_{\tilde{x}\tilde{x}} \end{bmatrix}\right)$$

where $\boldsymbol{\Sigma}_{\tilde{x}\tilde{x}} \in \mathbb{R}^{k\times k}$ and $\boldsymbol{\Sigma}_{x\tilde{x}} \in \mathbb{R}^{D\times k}$, $k \to \infty$.
However, the marginal remains finite

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \tilde{\boldsymbol{x}}) d\tilde{\boldsymbol{x}} = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx})$$

where we integrate out an infinite number of random variables $\tilde{x}_i$.

Marginal and Conditional in the Limit

- In practice, we consider finite training and test data $\boldsymbol{x}_{\text{train}}, \boldsymbol{x}_{\text{test}}$
- Then, $\boldsymbol{x} = \{\boldsymbol{x}_{\text{train}}, \boldsymbol{x}_{\text{test}}, \boldsymbol{x}_{\text{other}}\}$
  ($\boldsymbol{x}_{\text{other}}$ plays the role of $\tilde{\boldsymbol{x}}$ from previous slide)

$$p(\boldsymbol{x}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{\text{train}} \\ \boldsymbol{\mu}_{\text{test}} \\ \boldsymbol{\mu}_{\text{other}} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{\text{train}} & \boldsymbol{\Sigma}_{\text{train,test}} & \boldsymbol{\Sigma}_{\text{train,other}} \\ \boldsymbol{\Sigma}_{\text{test,train}} & \boldsymbol{\Sigma}_{\text{test}} & \boldsymbol{\Sigma}_{\text{test,other}} \\ \boldsymbol{\Sigma}_{\text{other,train}} & \boldsymbol{\Sigma}_{\text{other,test}} & \boldsymbol{\Sigma}_{\text{other}} \end{bmatrix}\right)$$

$$p(\boldsymbol{x}_{\text{train}}, \boldsymbol{x}_{\text{test}}) = \int p(\boldsymbol{x}_{\text{train}}, \boldsymbol{x}_{\text{test}}, \boldsymbol{x}_{\text{other}}) d\boldsymbol{x}_{\text{other}}$$

$$p(\boldsymbol{x}_{\text{test}}|\boldsymbol{x}_{\text{train}}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

$$\boldsymbol{\mu}_* = \boldsymbol{\mu}_{\text{test}} + \boldsymbol{\Sigma}_{\text{test,train}} \boldsymbol{\Sigma}_{\text{train}}^{-1} (\boldsymbol{x}_{\text{train}} - \boldsymbol{\mu}_{\text{train}})$$

$$\boldsymbol{\Sigma}_* = \boldsymbol{\Sigma}_{\text{test}} - \boldsymbol{\Sigma}_{\text{test,train}} \boldsymbol{\Sigma}_{\text{train}}^{-1} \boldsymbol{\Sigma}_{\text{train,test}}$$

Gaussian Process Training: Hierarchical Inference

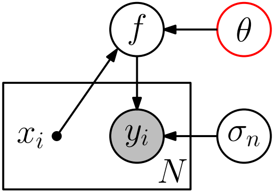
- Level-1 inference (posterior on f):

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

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

- Level-2 inference (posterior on $\boldsymbol{\theta}$)

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



GP as the Limit of an Infinite RBF Network

Consider the universal function approximator

$$f(x) = \sum_{i\in\mathbb{Z}} \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \gamma_n \exp\left(-\frac{(x-(i+\frac{n}{N}))^2}{\lambda^2}\right), \quad x \in \mathbb{R}, \quad \lambda \in \mathbb{R}^+$$

with $\gamma_n \sim \mathcal{N}(0, 1)$ (random weights)

▶▶ Gaussian-shaped basis functions (with variance $\lambda^2/2$) everywhere on the real axis

$$f(x) = \sum_{i\in\mathbb{Z}} \int_i^{i+1} \gamma(s) \exp\left(-\frac{(x-s)^2}{\lambda^2}\right) \mathrm{d}s = \int_{-\infty}^{\infty} \gamma(s) \exp\left(-\frac{(x-s)^2}{\lambda^2}\right) \mathrm{d}s$$

- Mean: $\mathbb{E}[f(x)] = 0$
- Covariance: $\mathrm{Cov}[f(x), f(x')] = \theta_1^2 \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)$ for suitable $\boldsymbol{\theta}_1^2$

▶▶ GP with mean 0 and Gaussian covariance function

References I

[1] E. Brochu, V. M. Cora, and N. de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. Technical Report TR-2009-023, Department of Computer Science, University of British Columbia, 2009.

[2] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley-Interscience, 1993.

[3] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, February 2015.

[4] M. P. Deisenroth and S. Mohamed. Expectation Propagation in Gaussian Process Dynamical Systems. In *Advances in Neural Information Processing Systems*, pages 2618–2626, 2012.

[5] M. P. Deisenroth and J. W. Ng. Distributed Gaussian Processes. In *Proceedings of the International Conference on Machine Learning*, 2015.

[6] M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian Process Dynamic Programming. *Neurocomputing*, 72(7–9):1508–1524, March 2009.

[7] 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.

[8] 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 C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 3156–3164. Curran Associates, Inc., 2013.

[9] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard. Gaussian Process Model Based Predictive Control. In *Proceedings of the 2004 American Control Conference (ACC 2004)*, pages 2214–2219, Boston, MA, USA, June–July 2004.

[10] 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, February 2008.

[11] N. Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, 6:1783–1816, November 2005.

References II

[12] 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.

[13] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian Processes for Global Optimization. In *Proceedings of the International Conference on Learning and Intelligent Optimization*, 2009.

[14] 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.

[15] 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.

[16] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 2006.

[17] 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), February 2013.

[18] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.