Probabilistic Inference (CO-493)

Imperial College London

Gaussian Processes

Marc Deisenroth

Department of Computing
Imperial College London

m.deisenroth@imperial.ac.uk

January 22, 2019

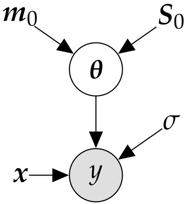
Overview

Bayesian Linear Regression (1-Slide Refresher)

Priors over Functions

Gaussian Processes
- Definition and Derivation
- Inference
- Covariance Functions and Hyper-Parameters
- Training

Bayesian Linear Regression: Model



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

$$\text{Likelihood} \quad p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}\big(y \,|\, \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta},\, \sigma^2\big)$$

$$\implies y = \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta} + \epsilon, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma^2\big)$$

- Parameter $\boldsymbol{\theta}$ becomes a latent (random) variable
- Distribution $p(\boldsymbol{\theta})$ induces a distribution over plausible functions
- Choose a conjugate Gaussian prior
  - Closed-form computations
  - Gaussian posterior

Overview

Bayesian Linear Regression (1-Slide Refresher)

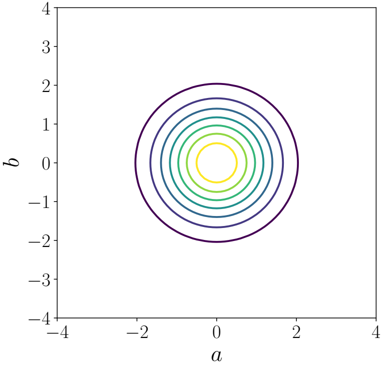
Priors over Functions

Gaussian Processes
- Definition and Derivation
- Inference
- Covariance Functions and Hyper-Parameters
- Training

Distribution over Functions

Consider a linear regression setting

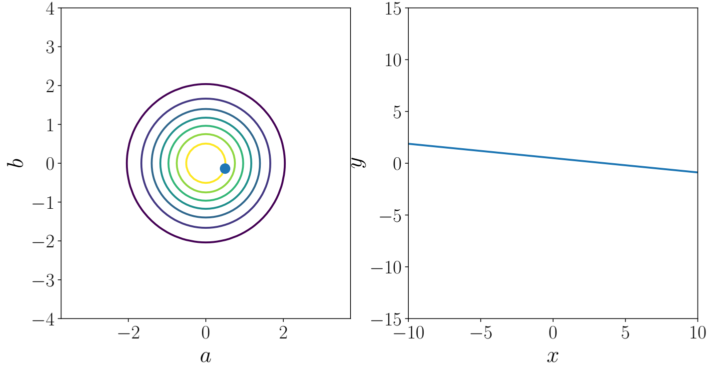
$$y = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$
$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$



Sampling from the Prior over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}(0,\, \sigma_n^2)$$
$$p(a, b) = \mathcal{N}(\mathbf{0},\, \boldsymbol{I})$$
$$f_i(x) = a_i + b_i x, \quad [a_i, b_i] \sim p(a, b)$$



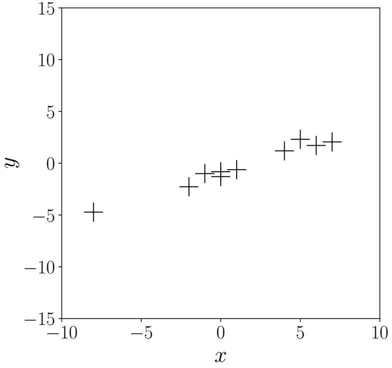
Sampling from the Posterior over Functions

Consider a linear regression setting

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

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

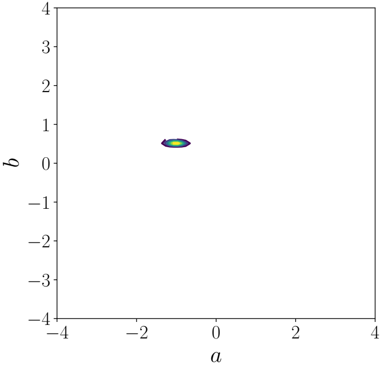
$$\boldsymbol{X} = [x_1, \dots, x_N],\ \boldsymbol{y} = [y_1, \dots, y_N] \quad \text{Training data}$$



Sampling from the Posterior over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$
$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$
$$p(a, b|\boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\big(\boldsymbol{m}_N,\, \boldsymbol{S}_N\big) \qquad \text{Posterior}$$



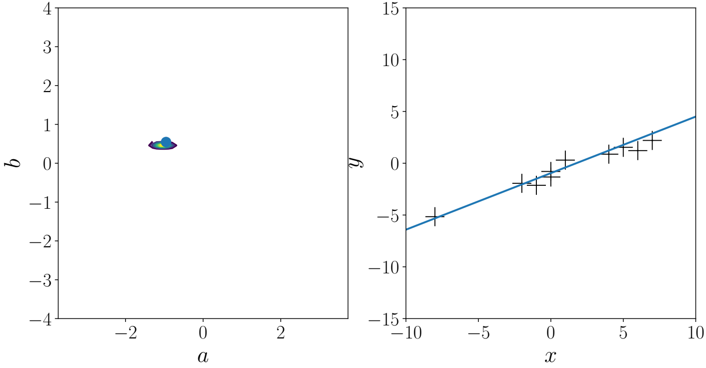
Sampling from the Posterior over Functions

Consider a linear regression setting

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

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

$$f_i = a_i + b_i x$$



Fitting Nonlinear Functions

- Fit nonlinear functions using (Bayesian) linear regression:
  Linear combination of nonlinear features

Fitting Nonlinear Functions

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

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

Fitting Nonlinear Functions

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

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

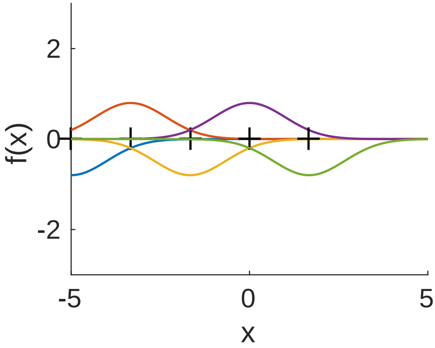
where

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

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

Illustration: Fitting a Radial Basis Function Network

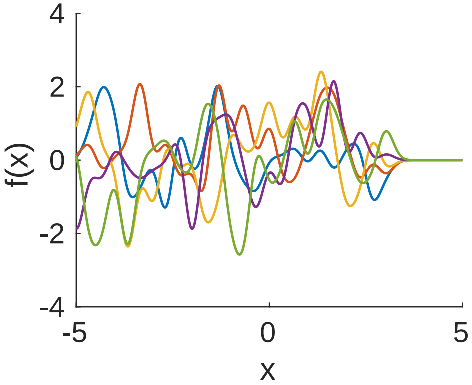
$$\phi_i(\boldsymbol{x}) = \exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top(\boldsymbol{x}-\boldsymbol{\mu}_i)\right)$$



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

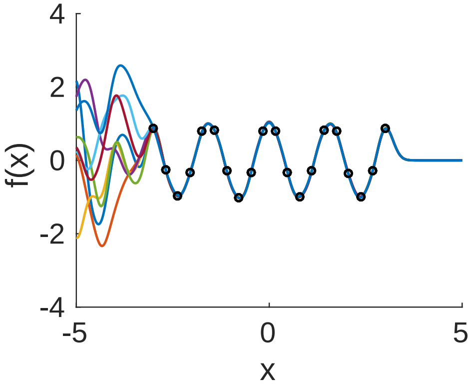
Samples from the RBF Prior

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_i \phi_i(\boldsymbol{x}), \quad p(\boldsymbol{\theta}) = \mathcal{N}\big(\mathbf{0}, \boldsymbol{I}\big)$$

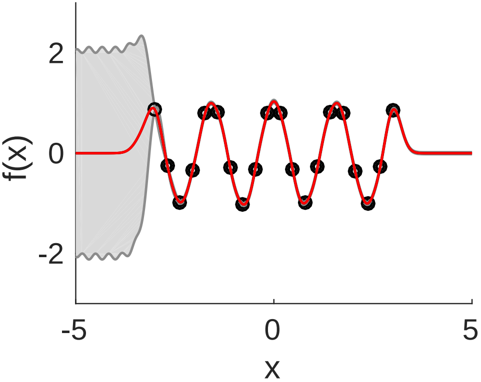


Samples from the RBF Posterior

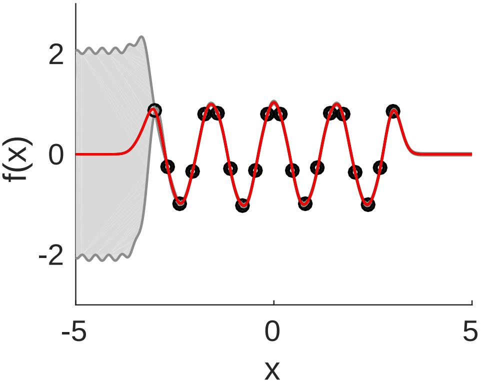
$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_i \phi_i(\boldsymbol{x}), \quad p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\big(\boldsymbol{m}_N, \boldsymbol{S}_N\big)$$



RBF Posterior



Limitations



- Feature engineering (what basis functions to use?)
- 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
  - ⏩ Place a prior on functions
  - ⏩ Make assumptions on the distribution of functions

Approach

- 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

Approach

- 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

Approach

- 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**

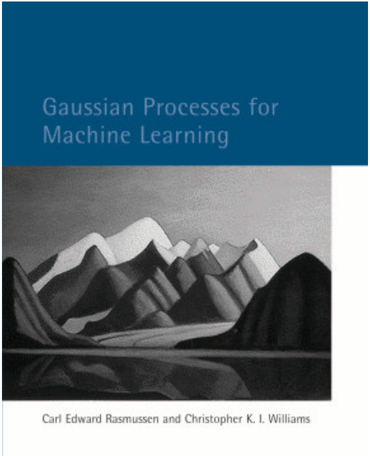
Overview

Bayesian Linear Regression (1-Slide Refresher)

Priors over Functions

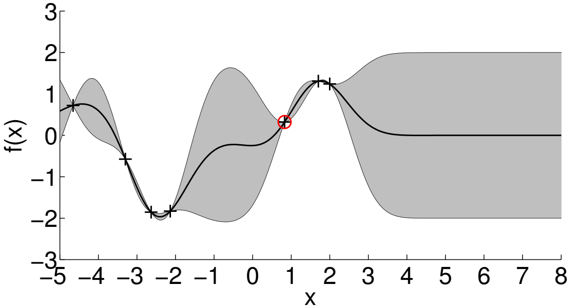
Gaussian Processes
- Definition and Derivation
- Inference
- Covariance Functions and Hyper-Parameters
- Training

Reference



http://www.gaussianprocess.org/

Problem Setting

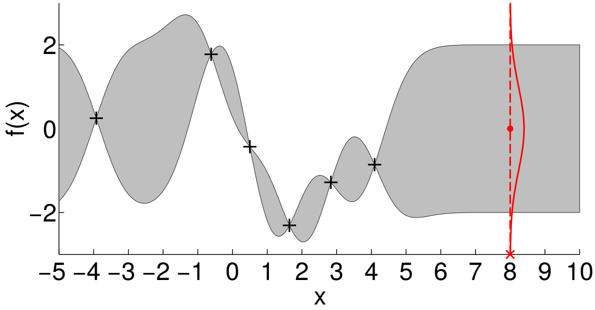


Objective

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

▶▶ Probabilistic regression problem

Problem Setting

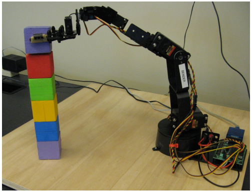
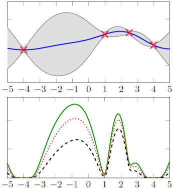


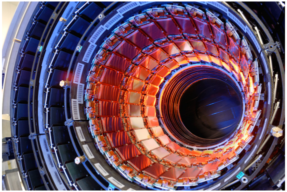
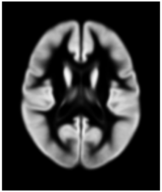
Objective

For a set of observations $y_i = f(\boldsymbol{x}_i) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, 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

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.

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 (Rasmussen & Williams, 2006)

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

Gaussian Process

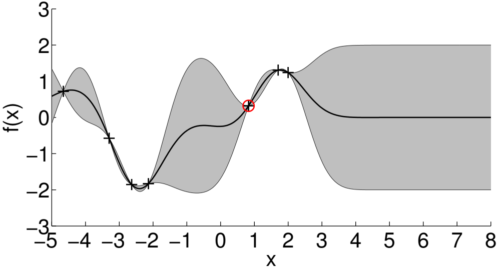
- We will place a distribution $p(f)$ on functions f
- Informally, a function can be considered an infinitely long vector of function values $f = [f_1, f_2, f_3, ...]$
- A Gaussian process is a generalization of a multivariate Gaussian distribution to infinitely many variables.

Definition (Rasmussen & Williams, 2006)

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

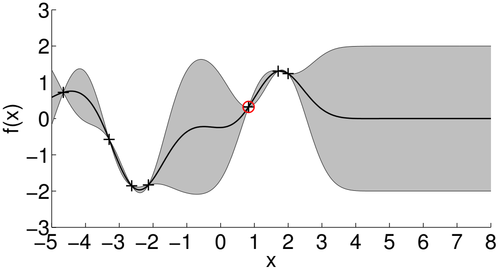
- A Gaussian distribution is specified by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$
- A Gaussian process is specified by a mean function $m(\cdot)$ and a covariance function (kernel) $k(\cdot, \cdot)$

Mean Function



- The "average" function of the distribution over functions
- Allows us to bias the model (can make sense in application-specific settings)
- "Agnostic" mean function in the absence of data or prior knowledge: $m(\cdot) \equiv 0$ everywhere (for symmetry reasons)

Covariance Function



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

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

▶▶ Kernel trick (Schölkopf & Smola, 2002)

GP Regression as a Bayesian Inference Problem

Objective

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

GP Regression as a Bayesian Inference Problem

Objective

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

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

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

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. Here: $\boldsymbol{X}$ training inputs, $\boldsymbol{y}$ training targets

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.

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. Here: $\boldsymbol{X}$ training inputs, $\boldsymbol{y}$ training targets

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

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. Here: $\boldsymbol{X}$ training inputs, $\boldsymbol{y}$ training targets

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

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

GP Regression as a Bayesian Inference Problem

Objective

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

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

GP Prior

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

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

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

GP Prior

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

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

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

- $\Sigma_{ff}^{(i,j)} = \mathrm{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$

GP Prior

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

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

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

GP Prior (2)

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

GP Prior (2)

- 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

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

▸▸ Computing the joint distribution of an arbitrary number of training and test inputs boils down to manipulating (finite-dimensional) Gaussian distributions

GP Posterior Predictions

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

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

GP Posterior Predictions

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

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

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

GP Posterior Predictions

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

- **Objective:** Find $p(f(\boldsymbol{X}_*)|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*)$ for training data $\boldsymbol{X}, \boldsymbol{y}$ and test inputs $\boldsymbol{X}_*$.
- GP prior at training inputs: $p(f|\boldsymbol{X}) = \mathcal{N}\big(m(\boldsymbol{X}), \boldsymbol{K}\big)$
- 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 Posterior Predictions

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 $p(\boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*)$ at test inputs $\boldsymbol{X}_*$

GP Posterior Predictions

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

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

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

GP Posterior Predictions

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

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

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

$$\begin{aligned} \mathbb{V}[f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*] &= k_{\text{post}}(\boldsymbol{X}_*, \boldsymbol{X}_*) \\ &= \underbrace{k(\boldsymbol{X}_*, \boldsymbol{X}_*)}_{\text{prior variance}} - \underbrace{k(\boldsymbol{X}_*, \boldsymbol{X})(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{X}_*)}_{\geqslant 0} \end{aligned}$$

GP Posterior

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

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

GP Posterior

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

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

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

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

GP Posterior

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

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

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

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

$$= Z \times GP\big(m_{\text{post}}(\cdot), k_{post}(\cdot, \cdot)\big)$$

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

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

GP Posterior

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

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

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

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

$$= Z \times GP\big(m_{\text{post}}(\cdot), k_{post}(\cdot, \cdot)\big)$$

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

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

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 Posterior

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

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

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

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

$$= Z \times GP\big(m_{\text{post}}(\cdot), k_{post}(\cdot, \cdot)\big)$$

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

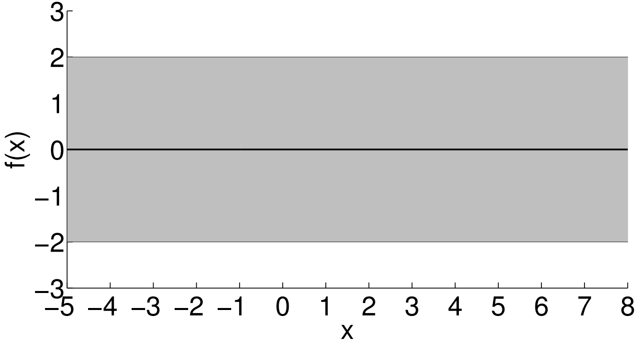
$$k_{\text{post}}(\cdot, \cdot) = k(\cdot, \cdot) - k(\cdot, \boldsymbol{X})(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \cdot)$$

Marginal likelihood:

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

Prediction at $\boldsymbol{x}_*$: $p(f(x_*)|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*) = \mathcal{N}\big(m_{\text{post}}(\boldsymbol{x}_*), \, k_{\text{post}}(\boldsymbol{x}_*, \boldsymbol{x}_*)\big)$

Illustration: Inference with Gaussian Processes

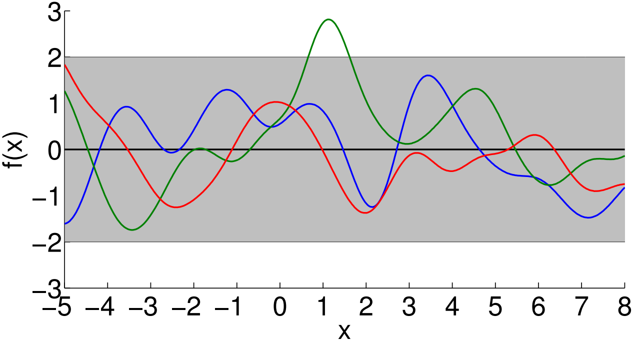


Prior belief about the function

Predictive (marginal) mean and variance:

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

Illustration: Inference with Gaussian Processes

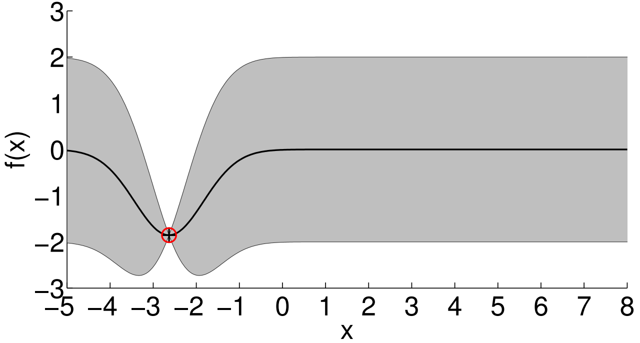


Prior belief about the function

Predictive (marginal) mean and variance:

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

Illustration: Inference with Gaussian Processes

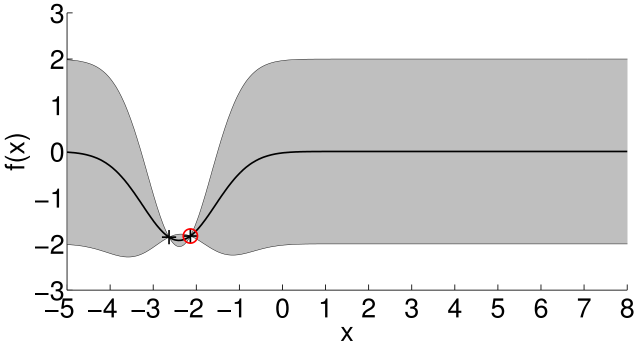


Posterior belief about the function

Predictive (marginal) mean and variance:

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

Illustration: Inference with Gaussian Processes



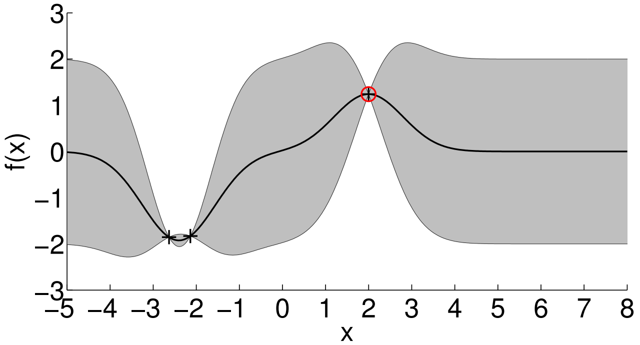
Posterior belief about the function

Predictive (marginal) mean and variance:

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

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

Illustration: Inference with Gaussian Processes

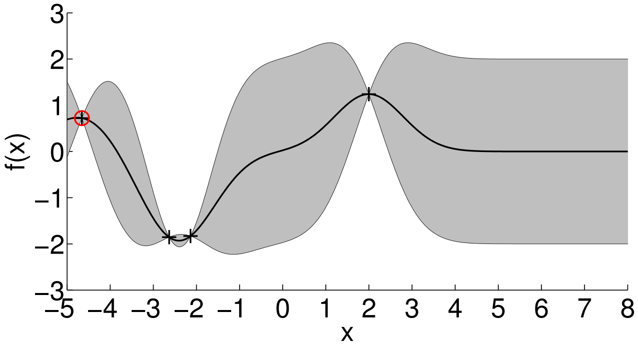


Posterior belief about the function

Predictive (marginal) mean and variance:

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

Illustration: Inference with Gaussian Processes

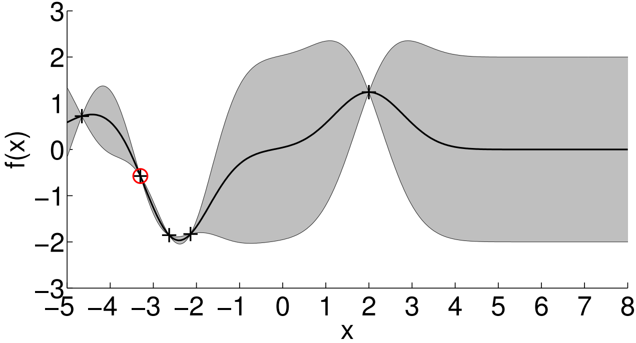


Posterior belief about the function

Predictive (marginal) mean and variance:

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

Illustration: Inference with Gaussian Processes

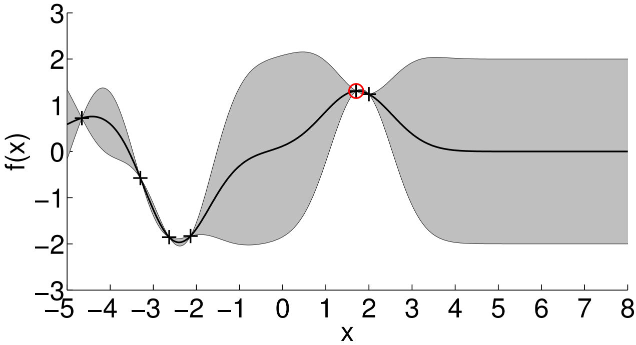


Posterior belief about the function

Predictive (marginal) mean and variance:

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

Illustration: Inference with Gaussian Processes

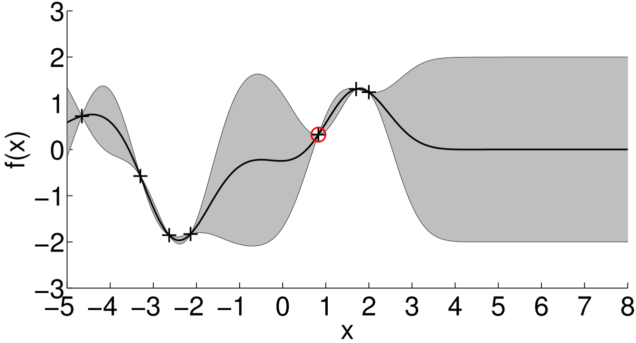


Posterior belief about the function

Predictive (marginal) mean and variance:

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

Illustration: Inference with Gaussian Processes

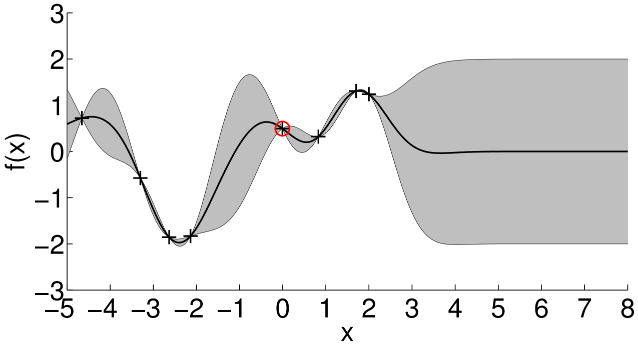


Posterior belief about the function

Predictive (marginal) mean and variance:

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

Illustration: Inference with Gaussian Processes



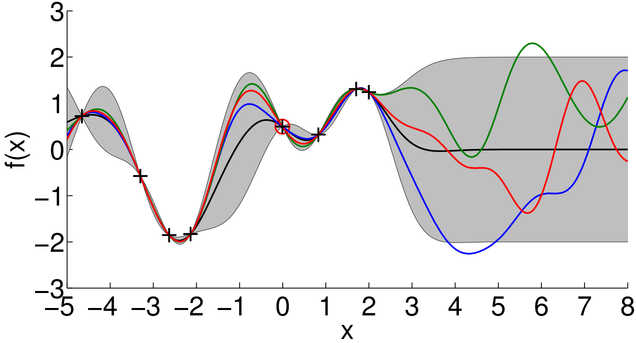
Posterior belief about the function

Predictive (marginal) mean and variance:

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

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

Illustration: Inference with Gaussian Processes



Posterior belief about the function

Predictive (marginal) mean and variance:

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

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

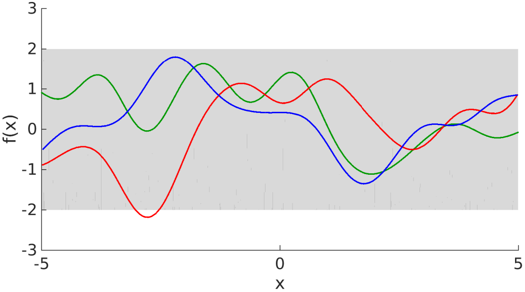
Covariance Function

- 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



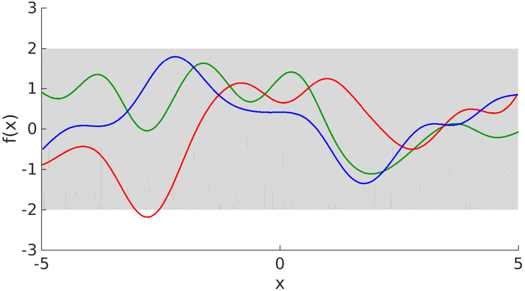
- Assumption on latent function: Smooth (∞ differentiable)

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, i.e., when do function values become uncorrelated?

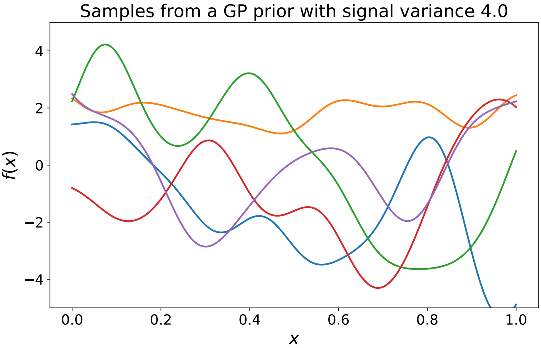
  ▸▸ **Smoothness parameter**



- Assumption on latent function: Smooth (∞ differentiable)

Amplitude Parameter σ_f^2

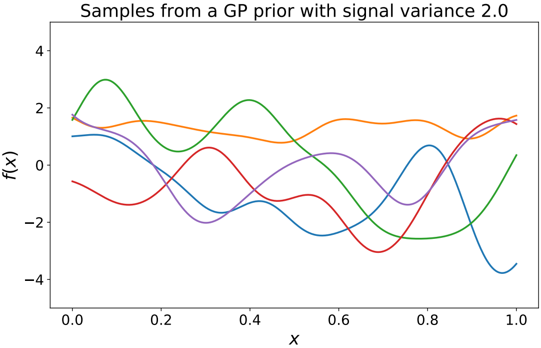
$$k_{Gauss}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\left(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)/\ell^2 \right)$$



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

Amplitude Parameter σ_f^2

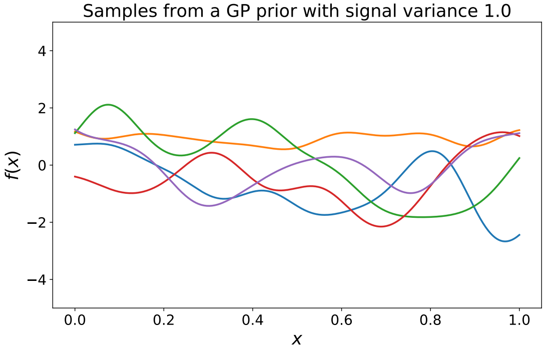
$$k_{Gauss}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\big(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)/\ell^2 \big)$$



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

Amplitude Parameter σ_f^2

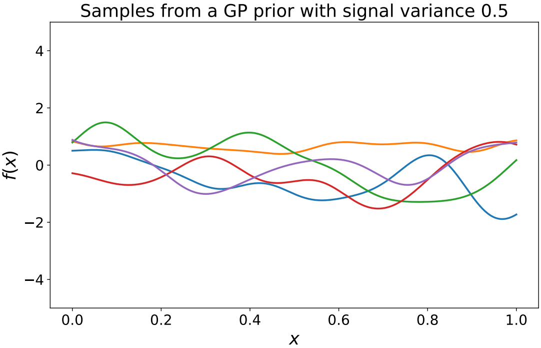
$$k_{Gauss}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\left(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)/\ell^2\right)$$



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

Amplitude Parameter σ_f^2

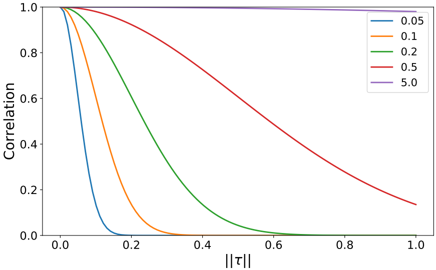
$$k_{Gauss}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\left(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)/\ell^2\right)$$



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

Length-Scale ℓ

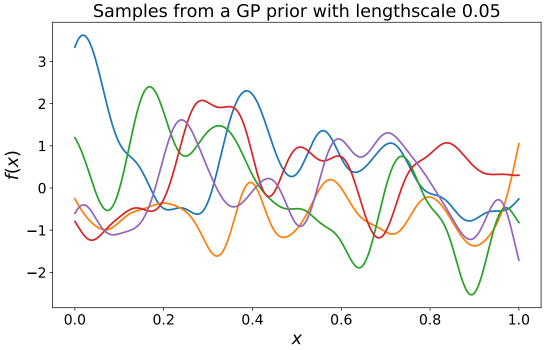
$$k_{Gauss}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\big(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)/\ell^2\big)$$



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

Length-Scale ℓ (2)

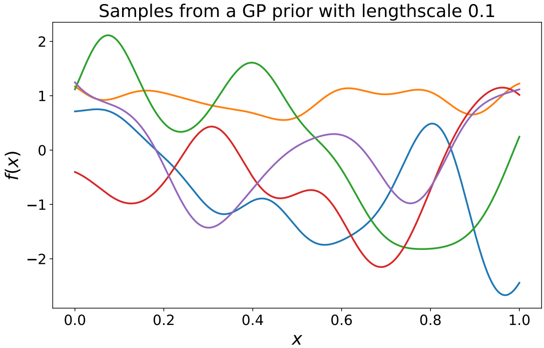
$$k_{Gauss}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\big(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)/\ell^2\big)$$



▶▶ Explore interactive diagrams at https://drafts.distill.pub/gp/

Length-Scale ℓ (2)

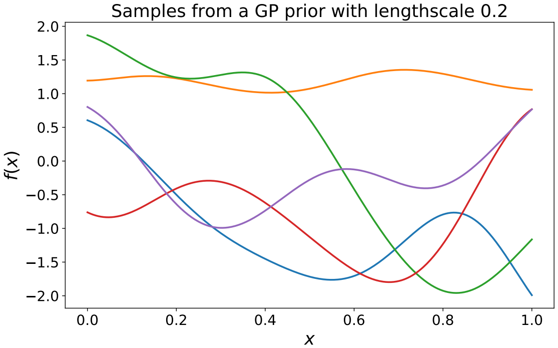
$$k_{Gauss}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\big(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)/\ell^2 \big)$$



▶▶ Explore interactive diagrams at `https://drafts.distill.pub/gp/`

Length-Scale ℓ (2)

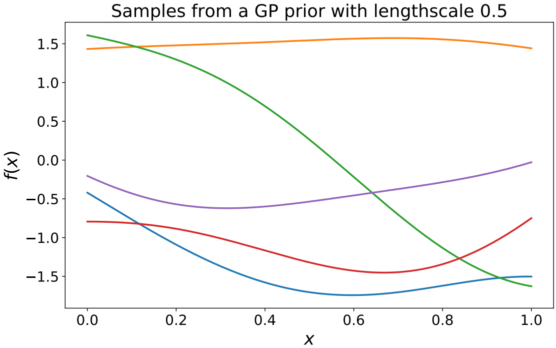
$$k_{Gauss}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\big(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)/\ell^2\big)$$



▶▶ Explore interactive diagrams at https://drafts.distill.pub/gp/

Length-Scale ℓ (2)

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

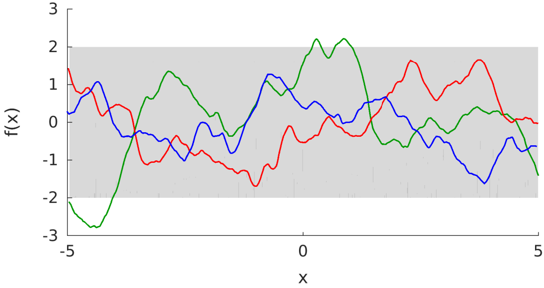


▶▶ Explore interactive diagrams at https://drafts.distill.pub/gp/

Matérn Covariance Function

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

- 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?



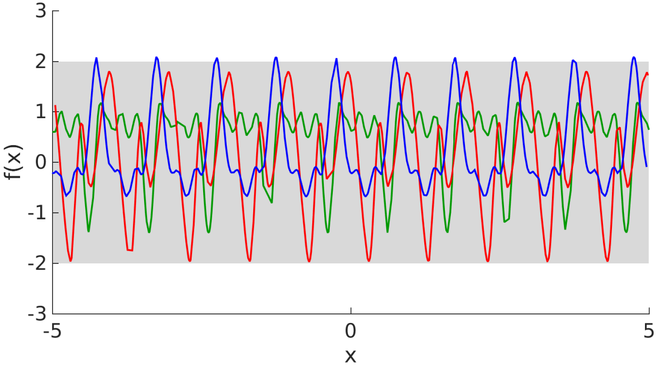
- Assumption on latent function: 1-times differentiable

Periodic Covariance Function

$$k_{per}(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{2\sin^2\left(\frac{\kappa(x_i - x_j)}{2\pi}\right)}{\ell^2}\right)$$

$$= k_{Gauss}(\boldsymbol{u}(x_i), \boldsymbol{u}(x_j)), \quad \boldsymbol{u}(x) = \begin{bmatrix}\cos(\kappa x)\\ \sin(\kappa x)\end{bmatrix}$$

κ: Periodicity parameter



Creating New Covariance Functions

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

Creating New Covariance Functions

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

Creating New Covariance Functions

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
- $k(u(\boldsymbol{x}), u(\boldsymbol{x}'))$ is a valid covariance function (MacKay, 1998)
  ▶▶ Periodic covariance function and Manifold Gaussian Process (Calandra et al., 2016)

Creating New Covariance Functions

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
- $k(u(\boldsymbol{x}), u(\boldsymbol{x}'))$ is a valid covariance function (MacKay, 1998)
  ▶▶ Periodic covariance function and Manifold Gaussian Process (Calandra et al., 2016)

▶▶ Automatic Statistician (Lloyd et al., 2014)

Hyper-Parameters of a GP

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)

Hyper-Parameters of a GP

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

Hyper-Parameters of a GP

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)

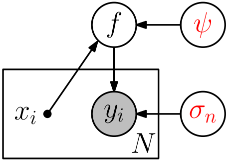
▸▸ Train a GP to find a good set of hyper-parameters

▸▸ 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

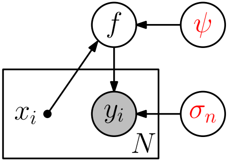
Find good hyper-parameters $\boldsymbol{\theta}$ (kernel/mean function parameters $\boldsymbol{\psi}$, noise variance σ_n^2)



Gaussian Process Training: Hyper-Parameters

GP Training

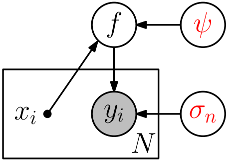
Find good hyper-parameters $\boldsymbol{\theta}$ (kernel/mean function parameters $\boldsymbol{\psi}$, noise variance σ_n^2)



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

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

Gaussian Process Training: Hyper-Parameters



GP Training

Find good hyper-parameters $\boldsymbol{\theta}$ (kernel/mean function parameters $\boldsymbol{\psi}$, noise variance σ_n^2)

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

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

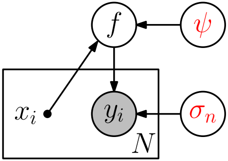
- Choose hyper-parameters $\boldsymbol{\theta}^*$, such that

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

Gaussian Process Training: Hyper-Parameters

GP Training

Find good hyper-parameters $\boldsymbol{\theta}$ (kernel/mean function parameters $\boldsymbol{\psi}$, noise variance σ_n^2)



- 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

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

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

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

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

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

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

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:

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

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

Training via Marginal Likelihood Maximization

Log-marginal likelihood:

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

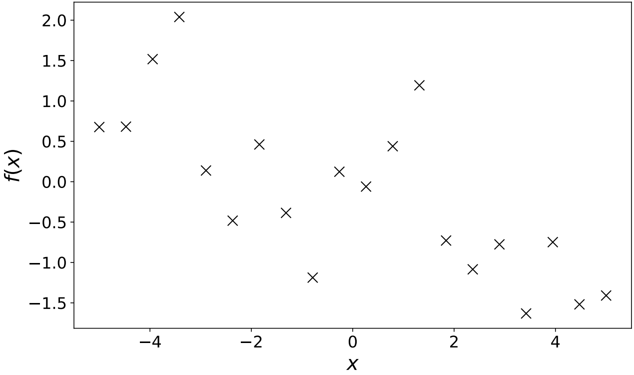
- Automatic trade-off between data fit and model complexity
- Gradient-based optimization of hyper-parameters $\boldsymbol{\theta}$:

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

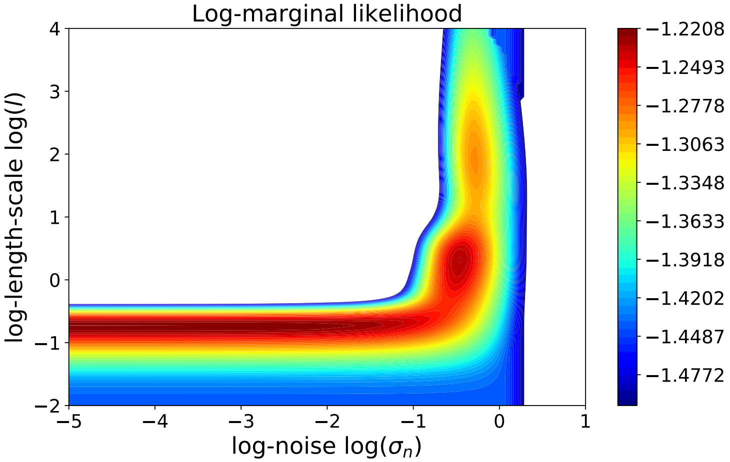
$$= \tfrac{1}{2}\text{tr}\big((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \boldsymbol{K}_{\boldsymbol{\theta}}^{-1})\frac{\partial \boldsymbol{K}_{\boldsymbol{\theta}}}{\partial \theta_i}\big)\,,$$

$$\boldsymbol{\alpha} := \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y}$$

Example: Training Data



Example: Marginal Likelihood Contour



- Three local optima. What do you expect?

Demo

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

Marginal Likelihood and Parameter Learning

- The marginal likelihood is non-convex

Marginal Likelihood and Parameter Learning

- 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:

Marginal Likelihood and Parameter Learning

- 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)

Marginal Likelihood and Parameter Learning

- 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)

Marginal Likelihood and Parameter Learning

- 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

Marginal Likelihood and Parameter Learning

- 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

Marginal Likelihood and Parameter Learning

- 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.

Marginal Likelihood and Parameter Learning

- 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

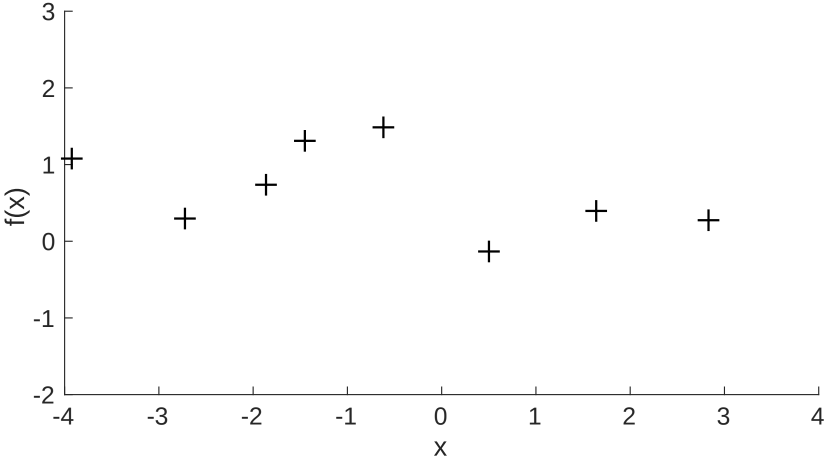
Model Selection—Mean Function and Kernel

- 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?

Model Selection—Mean Function and Kernel

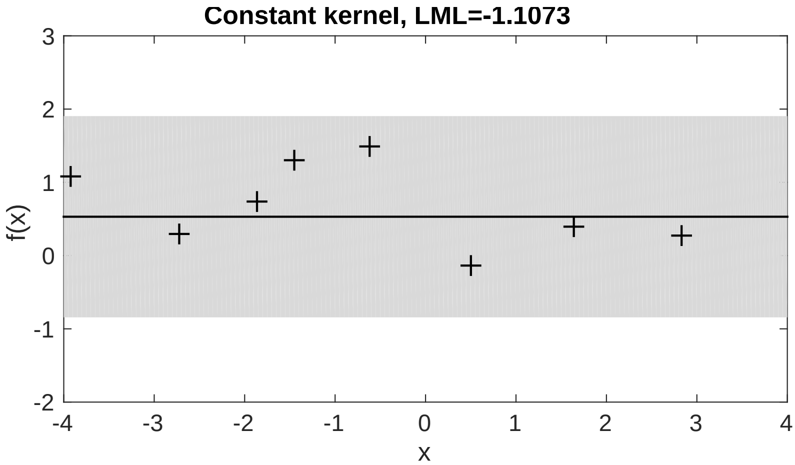
- Assume we have a finite set of models M_i, each one specifying a mean function m_i and a kernel k_i. How do we find the best one?
- Some options:
  - Cross validation
  - Bayesian Information Criterion, Akaike Information Criterion
  - Compare marginal likelihood values (assuming a uniform prior on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

Example



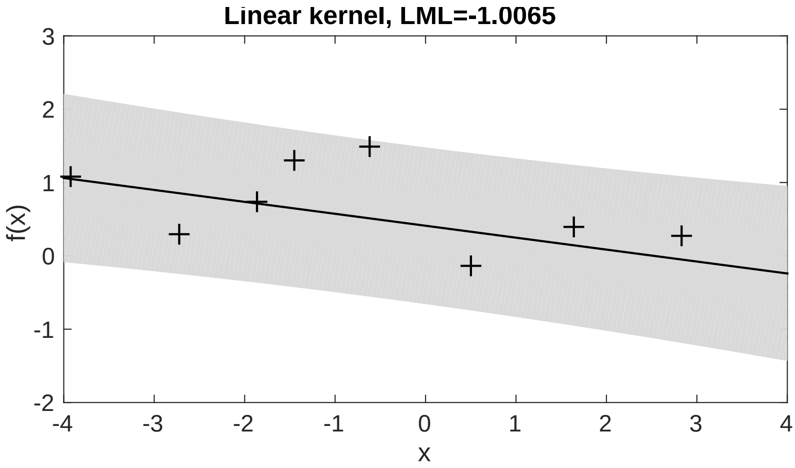
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



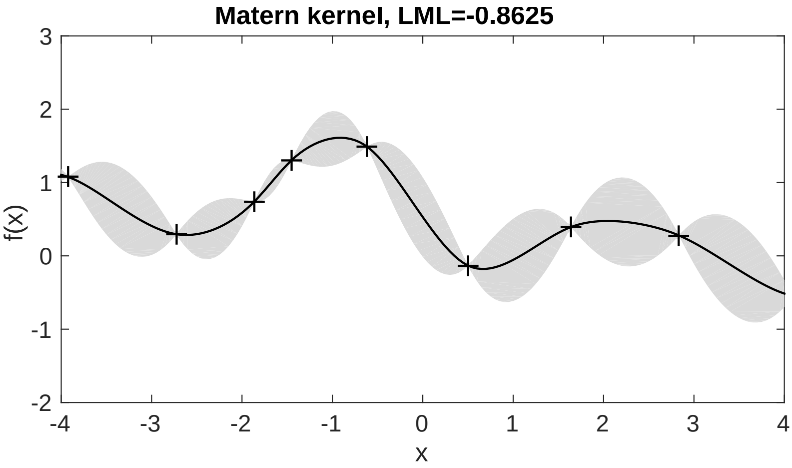
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



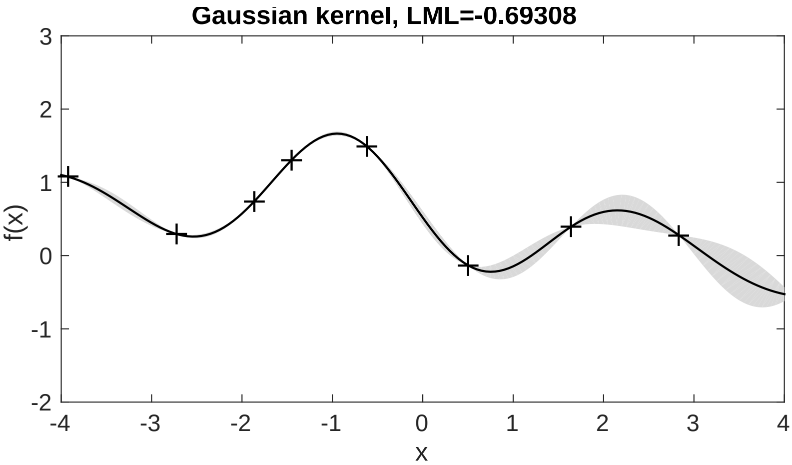
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



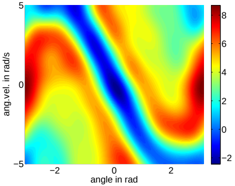
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

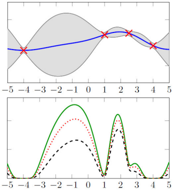
Example

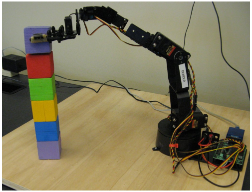


- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

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 $N \approx 10,000$

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** $N \approx 10,000$

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)

Tips and Tricks for Practitioners

- To set initial hyper-parameters, use domain knowledge.

⏩ https://drafts.distill.pub/gp

Tips and Tricks for Practitioners

- 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`

Tips and Tricks for Practitioners

- 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`

Tips and Tricks for Practitioners

- 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`

Tips and Tricks for Practitioners

- 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`

Tips and Tricks for Practitioners

- 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 $\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$) by penalizing high signal-to-noise ratios σ_f / σ_n

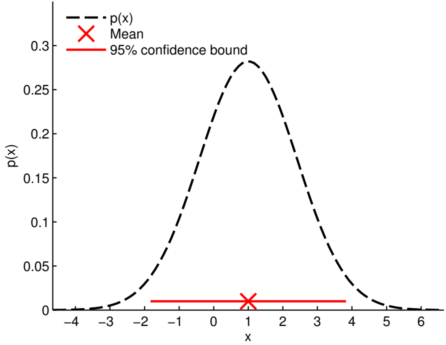
▸▸ `https://drafts.distill.pub/gp`

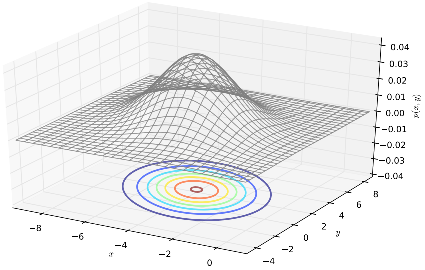
Appendix

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

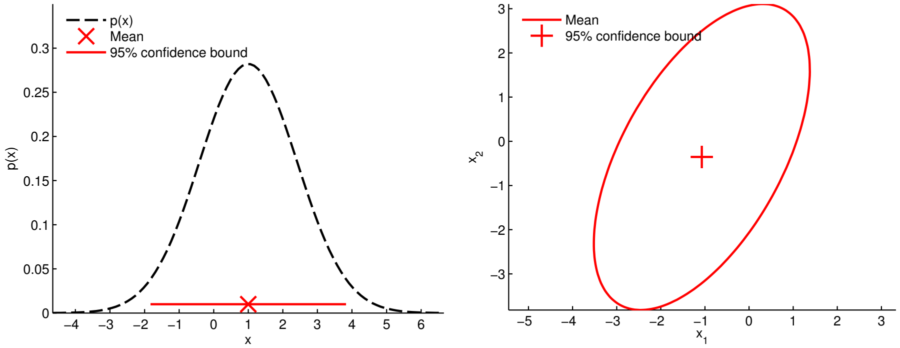




The Gaussian Distribution

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

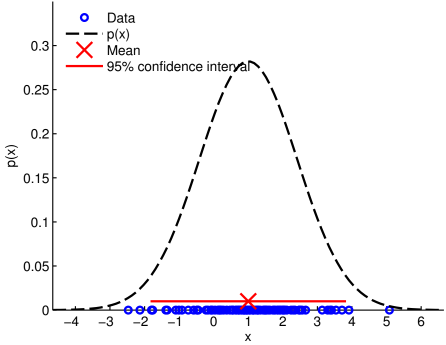
- Mean vector $\boldsymbol{\mu}$ ▶▶ Average of the data
- Covariance matrix $\boldsymbol{\Sigma}$ ▶▶ Spread of the data

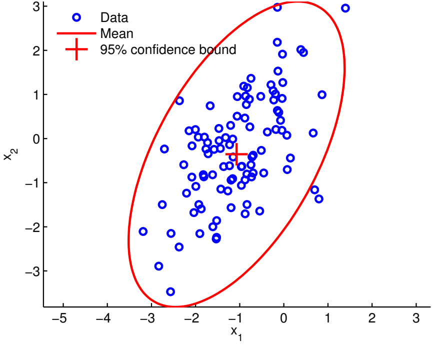


The Gaussian Distribution

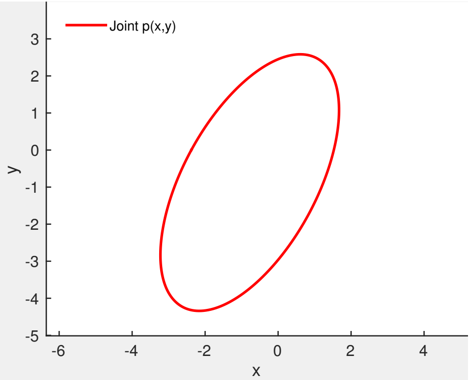
$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

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



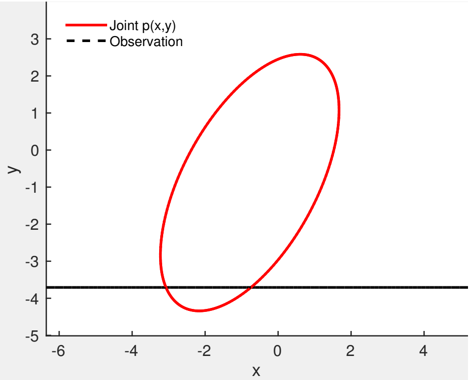


Conditional



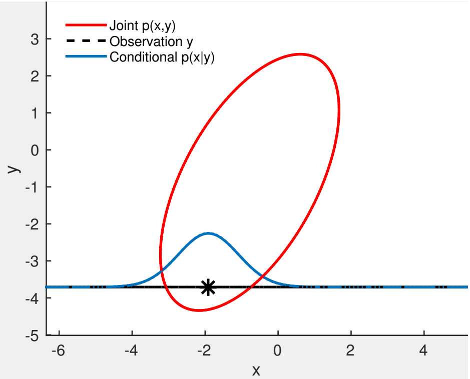
$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}\right)$$

Conditional



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

Conditional



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

$$p(\boldsymbol{x}|\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{\mu}_{x|y}, \boldsymbol{\Sigma}_{x|y}\right)$$

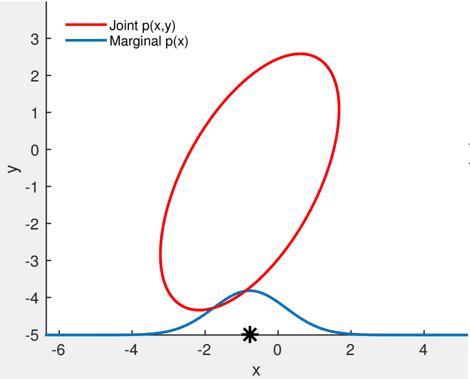
$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y)$$

$$\boldsymbol{\Sigma}_{x|y} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}$$

Conditional $p(\boldsymbol{x}|\boldsymbol{y})$ is also Gaussian

▶▶ Computationally convenient

Marginal

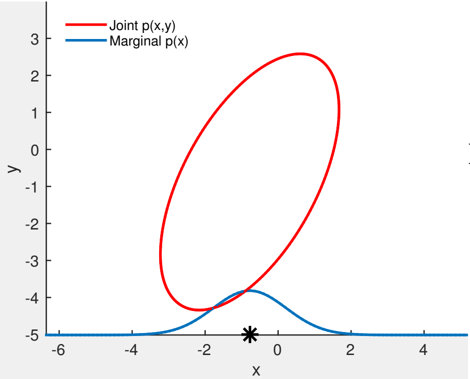


$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}\right)$$

Marginal distribution:

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

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.

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 $\boldsymbol{\Sigma}_{x\tilde{x}} \in \mathbb{R}^{D \times k}, k \to \infty$.

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 $\boldsymbol{x}_{\text{train}}, \boldsymbol{x}_{\text{test}}$

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)

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)

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

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)

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

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

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)

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

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

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

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

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

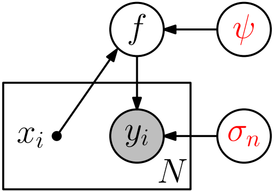
Gaussian Process Training: Hierarchical Inference

$\boldsymbol{\theta}$: Collection of all hyper-parameters

- Level-1 inference (posterior on f):

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

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



Gaussian Process Training: Hierarchical Inference

$\boldsymbol{\theta}$: Collection of all hyper-parameters

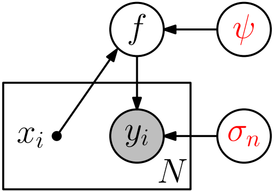
- Level-1 inference (posterior on f):

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

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

- 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

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

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

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] 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 IEEE 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 *IEEE International Conference on Robotics and Automation*, Seattle, WA, May 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*, Los Angeles, CA, USA, June 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 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.

[11] N. HajiGhassemi and **Marc P. Deisenroth**. Approximate Inference for Long-Term Forecasting with Periodic Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, April 2014. Acceptance rate: 36%.

[12] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian Processes for Big Data. In A. Nicholson and P. Smyth, editors, *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2013.

References II

[13] 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.

[14] 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.

[15] 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.

[16] D. J. C. MacKay. Introduction to Gaussian Processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, volume 168, pages 133–165. Springer, Berlin, Germany, 1998.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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), Feb. 2013.

[22] 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.

[23] 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.

[24] M. K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.

[25] V. Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000.