Imperial College London

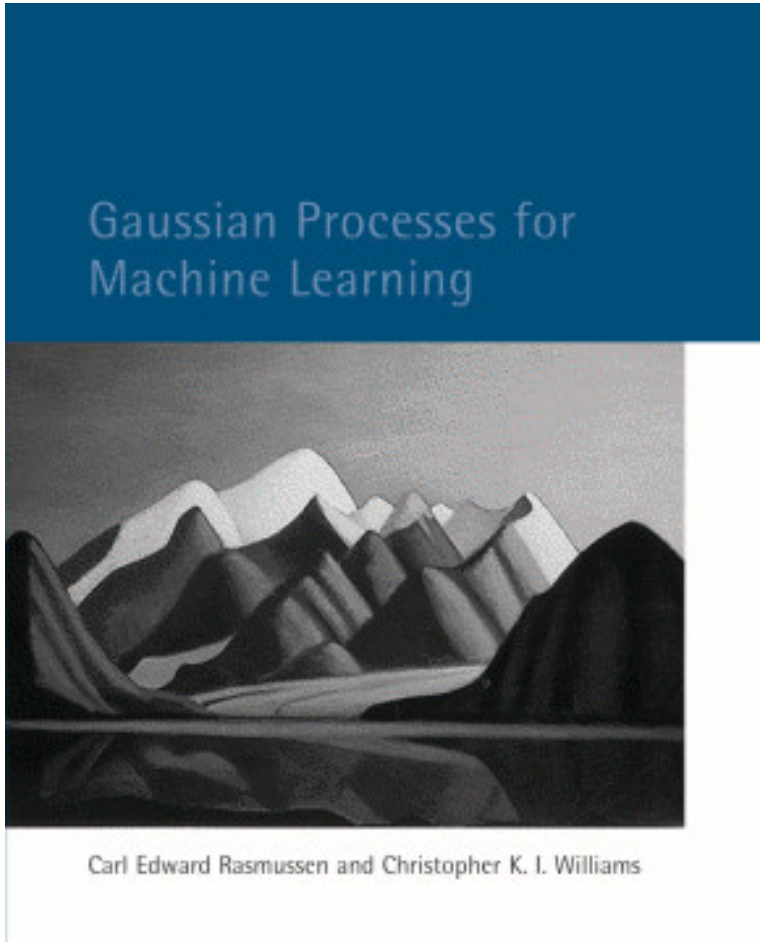
Gaussian Processes for Big Data Problems

Marc Deisenroth

Department of Computing
Imperial College London

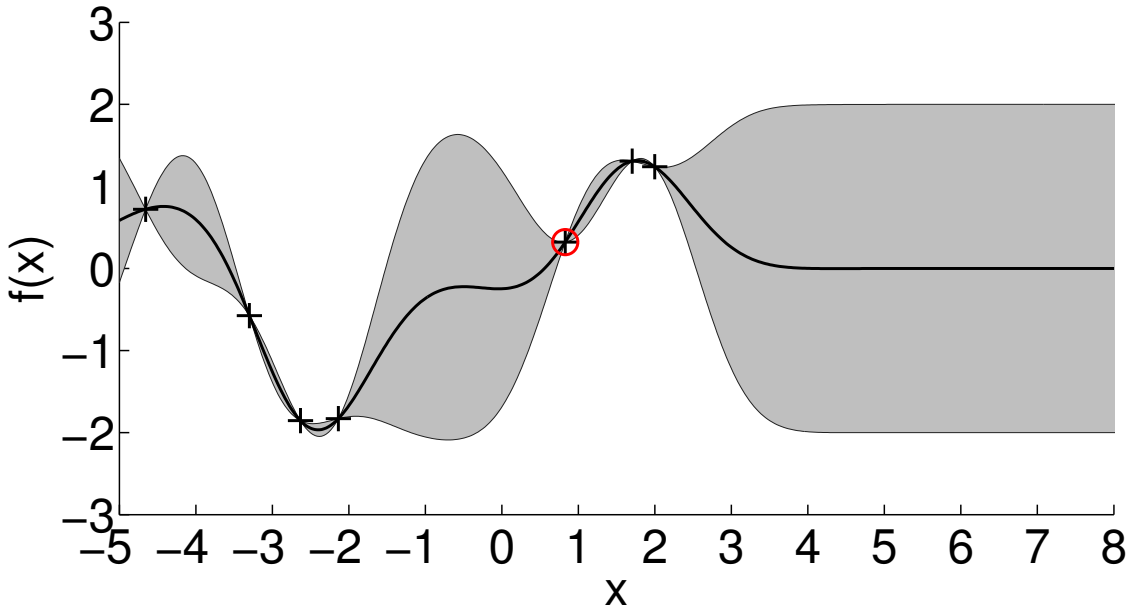
`http://wp.doc.ic.ac.uk/sml/marc-deisenroth`

Machine Learning Summer School, Chalmers University
14 April 2015



http://www.gaussianprocess.org/

Problem Setting

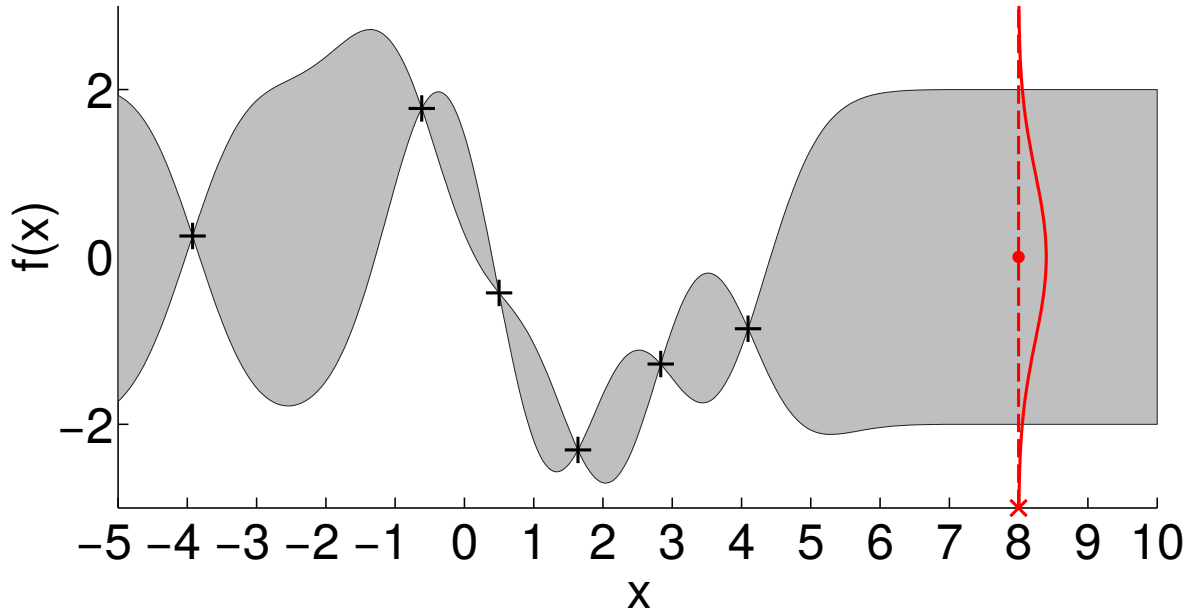


Objective

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

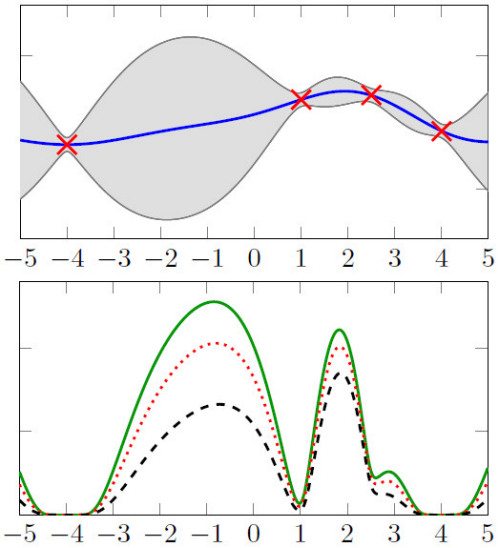


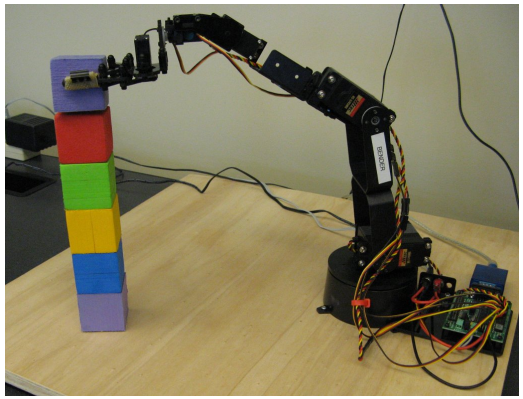
Objective

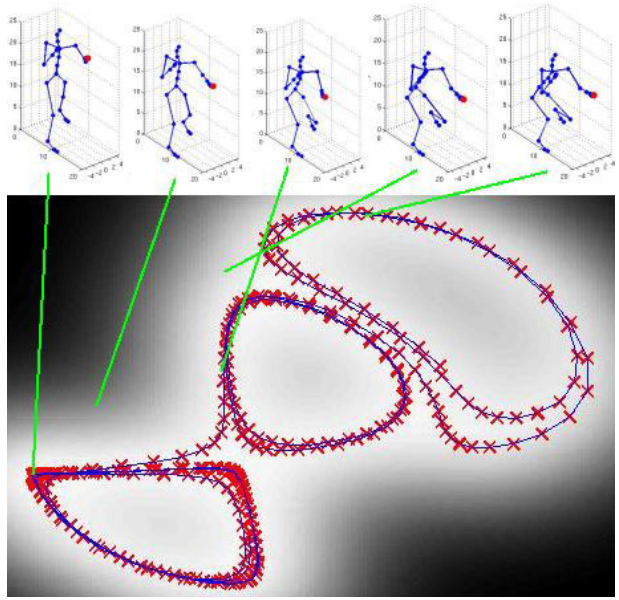
For a set of observations $y_i = f(\boldsymbol{x}_i) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, find a distribution over functions $p(f)$ that explains the data

- Probabilistic regression problem

(Some) Relevant Application Areas







- Global black-box optimization and experimental design
- Autonomous learning in robotics
- Probabilistic dimensionality reduction and data visualization

Table of Contents

Table of Contents

Key Concepts in Probability Theory

Two fundamental rules:

$$p(x) = \int p(x,y)\mathrm{d}y \qquad \text{Sum rule/Marginalization property}$$

$$p(x,y) = p(y|x)p(x) \qquad \text{Product rule}$$

Key Concepts in Probability Theory

Two fundamental rules:

$$p(x) = \int p(x,y)\mathrm{d}y \qquad \text{Sum rule/Marginalization property}$$

$$p(x,y) = p(y|x)p(x) \qquad \text{Product rule}$$

Bayes' Theorem (Probabilistic Inverse)

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})\ p(\boldsymbol{x})}{p(\boldsymbol{y})}, \qquad \boldsymbol{x}: \text{hypothesis}, \quad \boldsymbol{y}: \text{measurement}$$

Key Concepts in Probability Theory

Two fundamental rules:

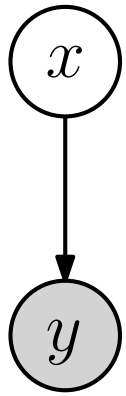
$$p(x) = \int p(x,y)\mathrm{d}y \qquad \text{Sum rule/Marginalization property}$$

$$p(x,y) = p(y|x)p(x) \qquad \text{Product rule}$$

Bayes' Theorem (Probabilistic Inverse)

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})\ p(\boldsymbol{x})}{p(\boldsymbol{y})}, \qquad \boldsymbol{x}: \text{hypothesis}, \quad \boldsymbol{y}: \text{measurement}$$

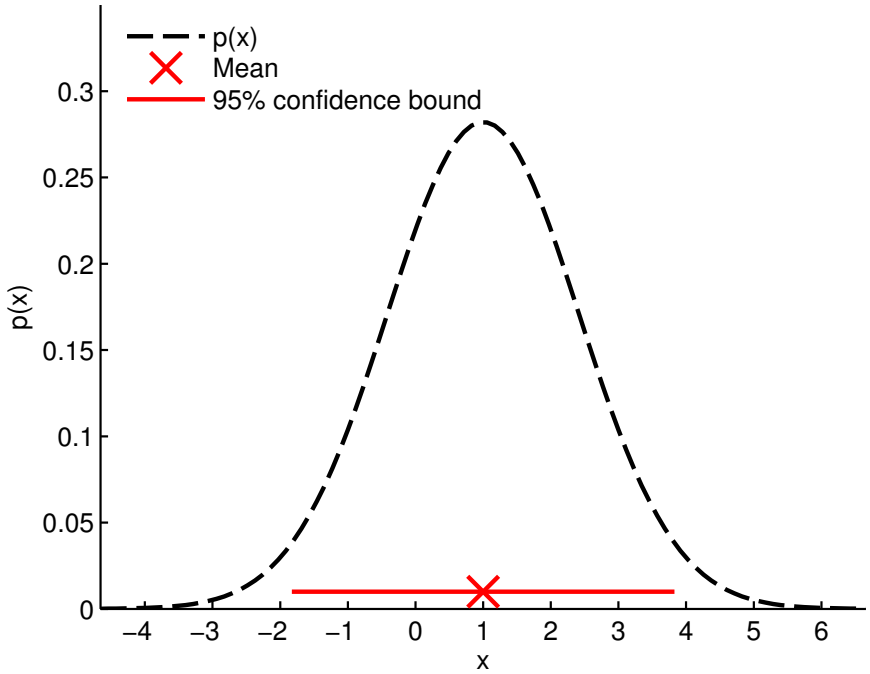
- Posterior belief
- Prior belief
- Likelihood (measurement model)
- Marginal likelihood (normalization constant)

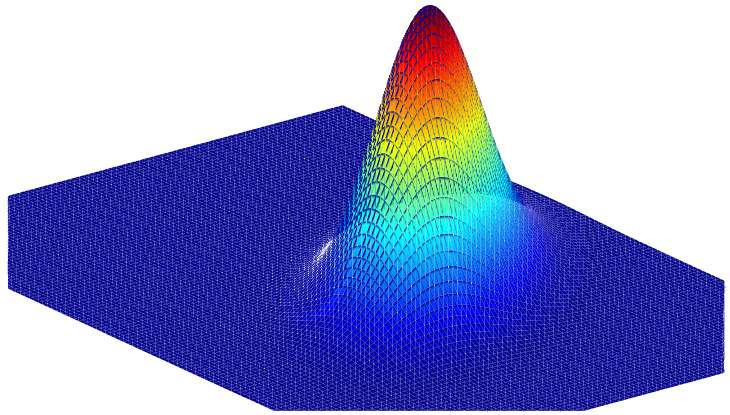


The Gaussian Distribution

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

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

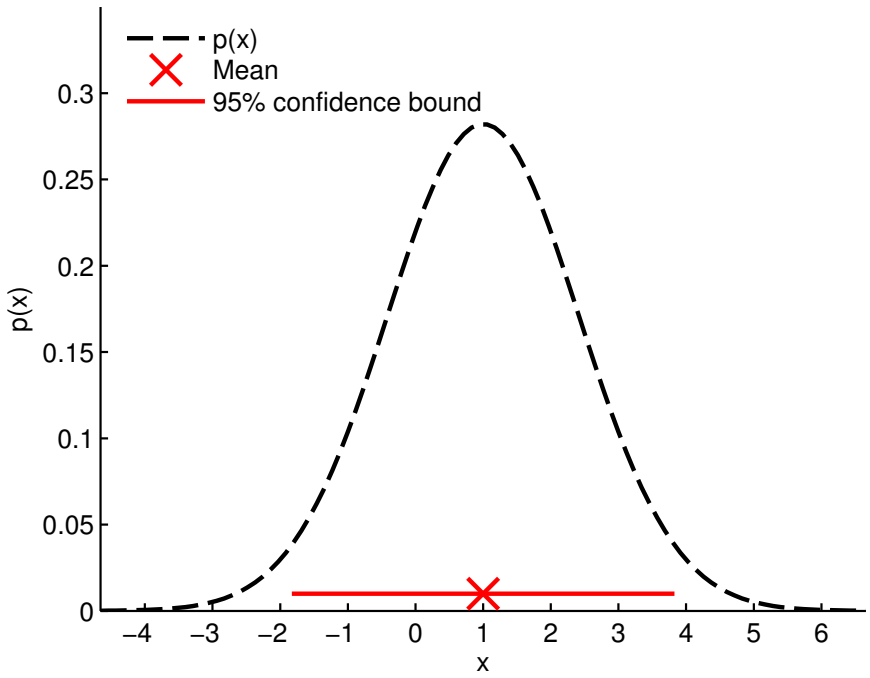


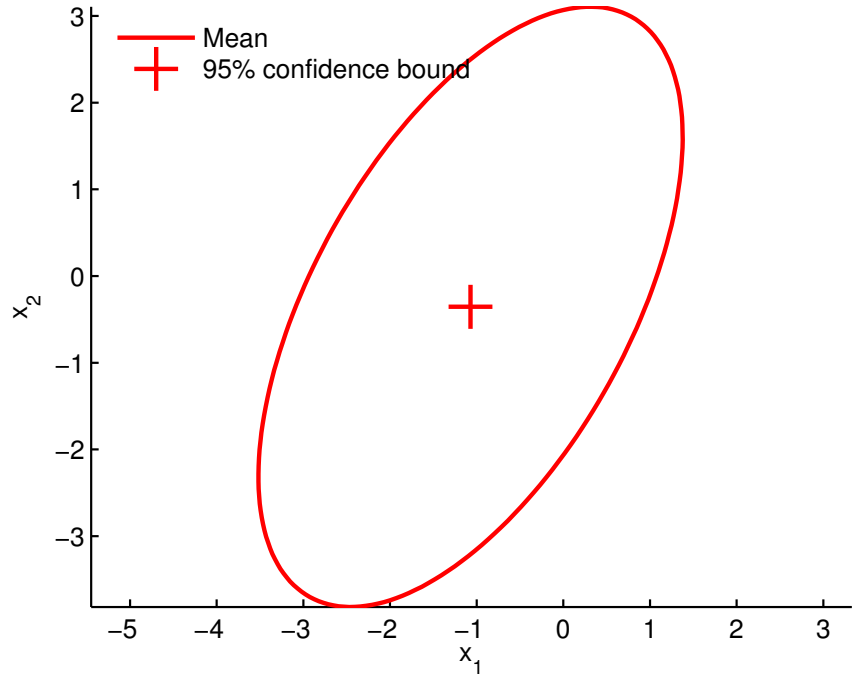


The Gaussian Distribution

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

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

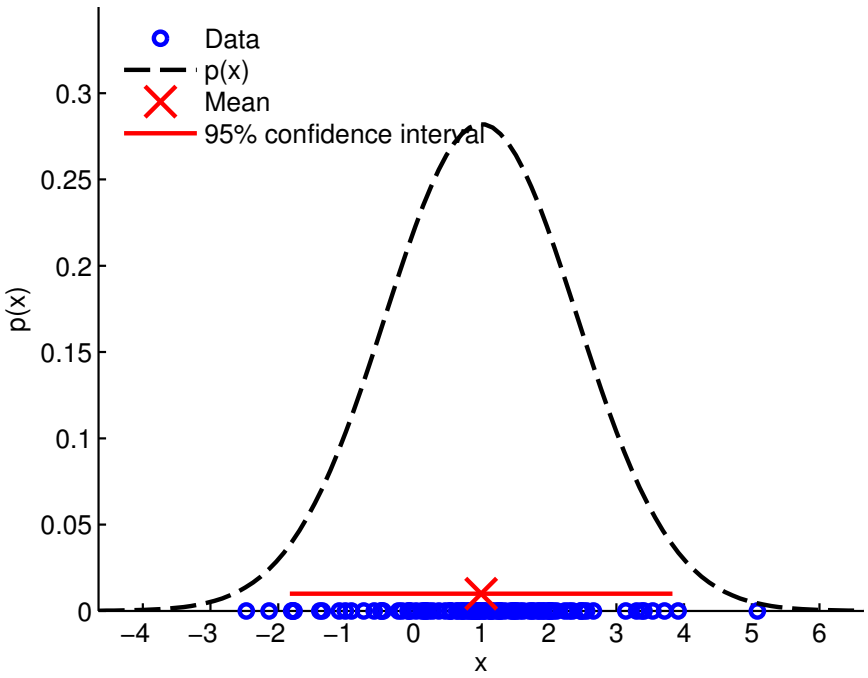


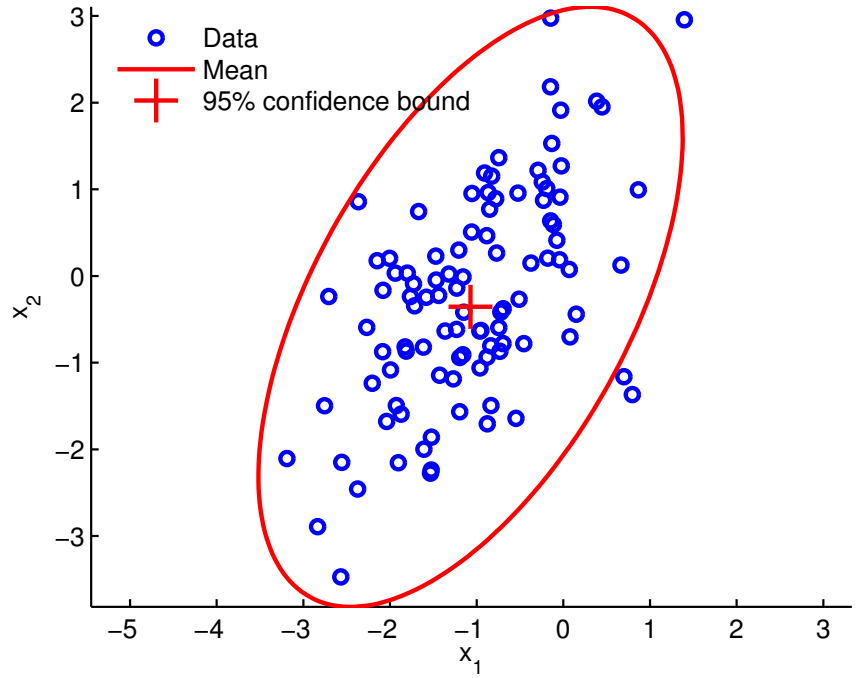


The Gaussian Distribution

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

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





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 $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$...

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 Gaussians remain Gaussian

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

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 Gaussians 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 $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
`y = chol(Σ)'*x + μ;`	Scale $\boldsymbol{x}$

Here `chol(Σ)` is the Cholesky factor $\boldsymbol{L}$, such that $\boldsymbol{L}^\top \boldsymbol{L} = \boldsymbol{\Sigma}$

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);
y = chol(Σ)'*x + μ;
```

Sample $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

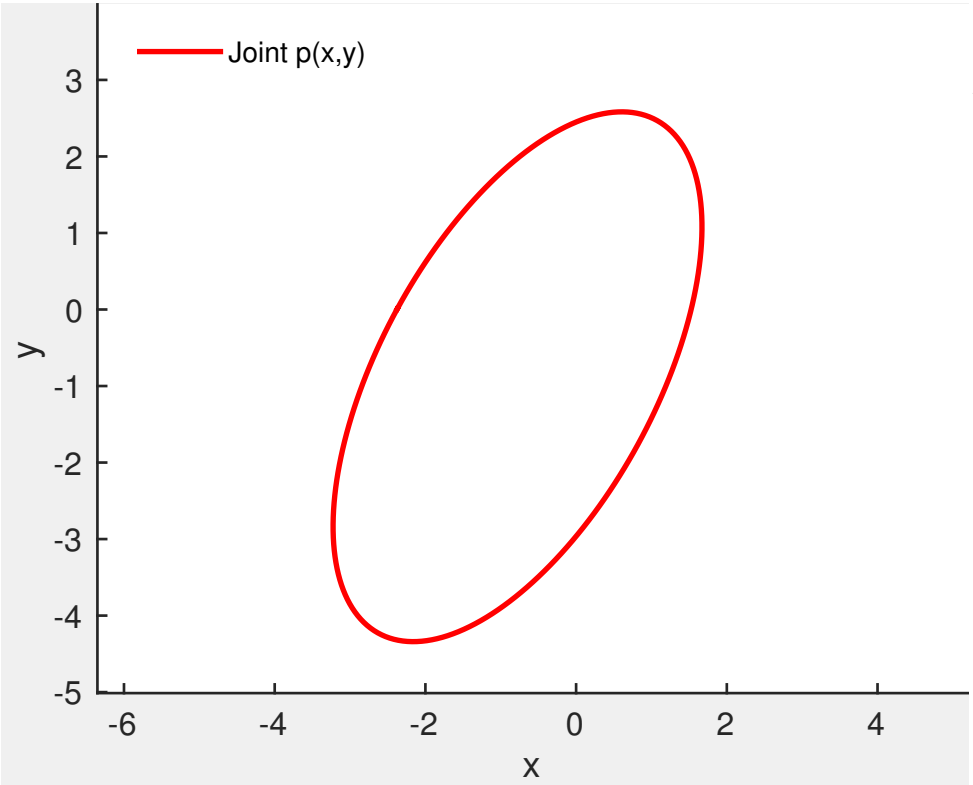
Scale $\boldsymbol{x}$

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

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

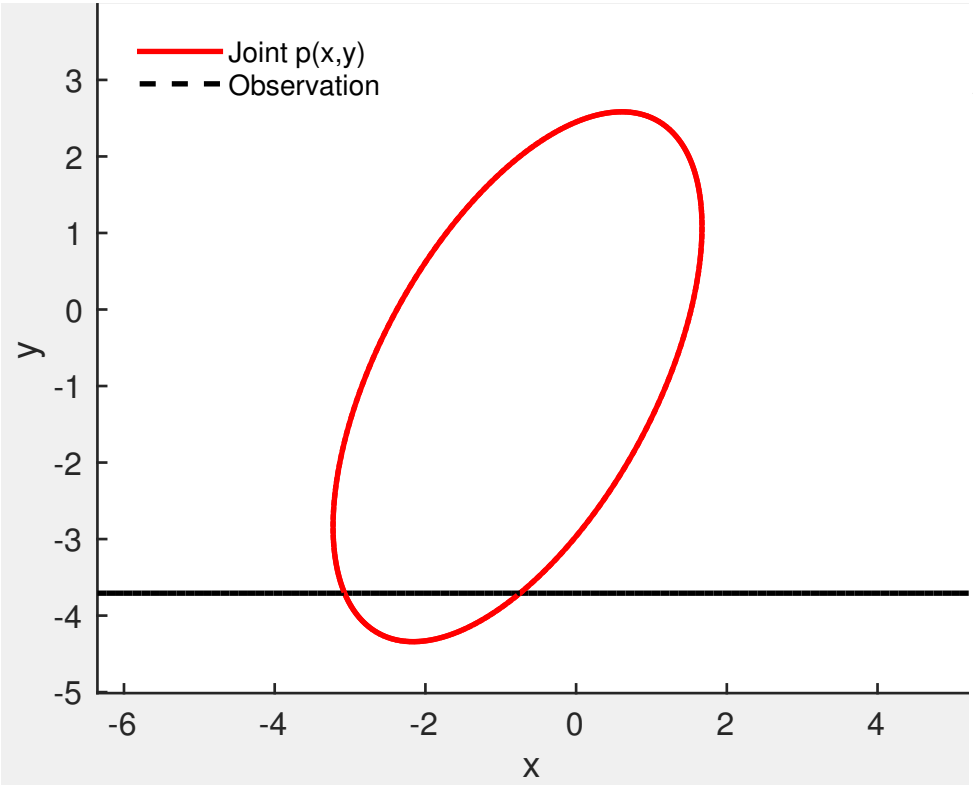
$$\mathrm{Cov}[\boldsymbol{y}] = \mathbb{E}[(\boldsymbol{y} - \bar{\boldsymbol{y}})(\boldsymbol{y} - \bar{\boldsymbol{y}})^\top] = \mathbb{E}[\boldsymbol{L}^\top \boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{L}] = \boldsymbol{L}^\top \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top] \boldsymbol{L} = \boldsymbol{L}^\top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

Conditional



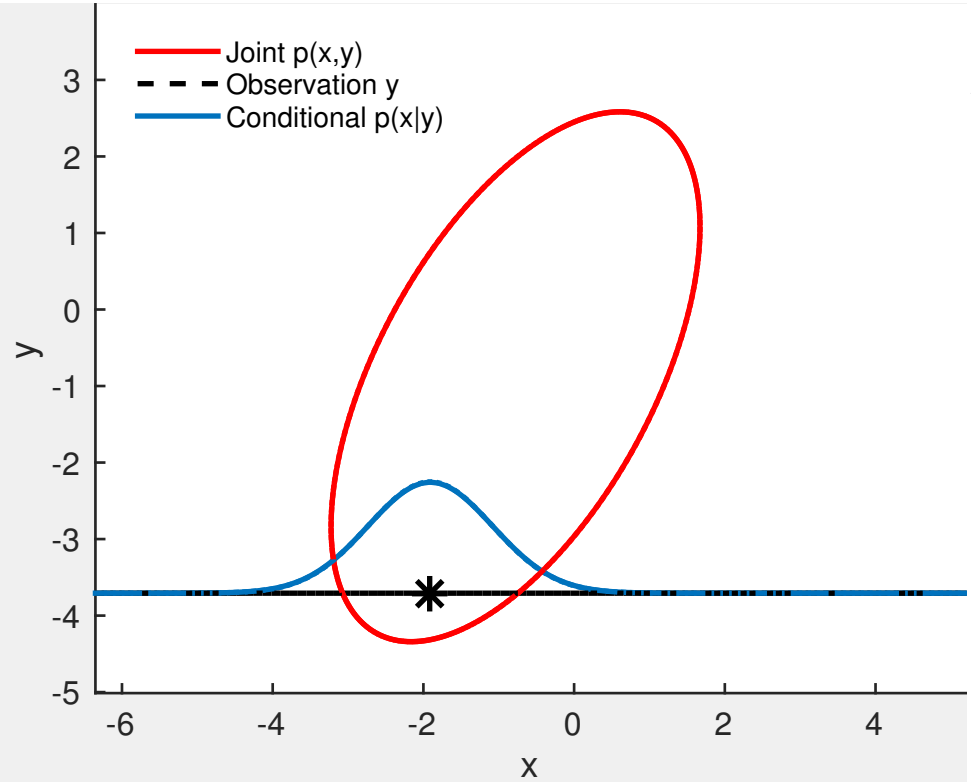
$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}\right)$$

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



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

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

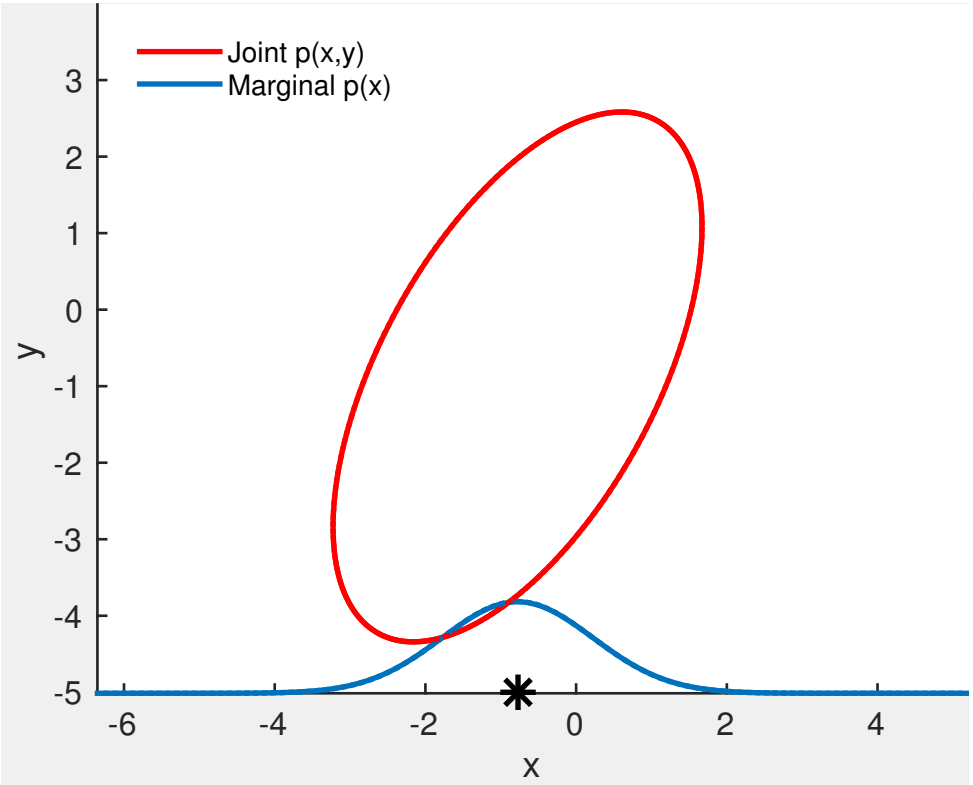
$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y)$$

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

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

⏩ Computationally convenient

Marginal



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

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

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

Table of Contents

Gaussian Process

Definition

A Gaussian process (GP) is a collection of random variables $x_1, x_2, \ldots,$ any finite number of which is Gaussian distributed.

Gaussian Process

Definition

A Gaussian process (GP) is a collection of random variables $x_1, x_2, \ldots$, any finite number of which is Gaussian distributed.

Unexpected (?) example: Linear dynamical system

$$x_{t+1} = Ax_t + w, \quad w \sim \mathcal{N}(0, Q)$$

$x_1, x_2, \ldots$ is a collection of random variables

Gaussian Process

Definition

A Gaussian process (GP) is a collection of random variables $x_1, x_2, \ldots$, any finite number of which is Gaussian distributed.

Unexpected (?) example: Linear dynamical system

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{w}, \quad \boldsymbol{w} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{Q}\right)$$

$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ is a collection of random variables,
any finite number of which is Gaussian distributed

Gaussian Process

Definition

A Gaussian process (GP) is a collection of random variables $x_1, x_2, \ldots$, any finite number of which is Gaussian distributed.

Unexpected (?) example: Linear dynamical system

$$x_{t+1} = Ax_t + w, \quad w \sim \mathcal{N}(0, Q)$$

$x_1, x_2, \ldots$ is a collection of random variables,
any finite number of which is Gaussian distributed
⏩ This is a GP

Gaussian Process

Definition

A Gaussian process (GP) is a collection of random variables $x_1, x_2, \ldots$, any finite number of which is Gaussian distributed.

Unexpected (?) example: Linear dynamical system

$$x_{t+1} = Ax_t + w, \quad w \sim \mathcal{N}(0, Q)$$

$x_1, x_2, \ldots$ is a collection of random variables,
any finite number of which is Gaussian distributed
⏩ This is a GP

But we will use the Gaussian process for a different purpose, not for time series

The Gaussian in the Limit

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

The Gaussian in the Limit

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

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

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

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

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

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

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

Back to Regression: Distribution over Functions

- We are not really interested in a distribution on $\boldsymbol{x}$, but rather in a distribution over function values $f(\boldsymbol{x})$

Back to Regression: Distribution over Functions

- We are not really interested in a distribution on $\boldsymbol{x}$, but rather in a distribution over function values $f(\boldsymbol{x})$
- Let's replace $\boldsymbol{x}$ with function values $f = f(\boldsymbol{x})$
  - ⏩ (Treat a function as a long vector of function values)

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

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

Back to Regression: Distribution over Functions

- We are not really interested in a distribution on $\boldsymbol{x}$, but rather in a distribution over function values $f(\boldsymbol{x})$
- Let's replace $\boldsymbol{x}$ with function values $f = f(\boldsymbol{x})$
  - ⏩ (Treat a function as a long vector of function values)

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

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

- Again, the marginal remains finite

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

Marginal and Conditional over Functions

Define $f_* := f_{\text{test}}, f := f_{\text{train}}$.

Marginal and Conditional over Functions

Define $\boldsymbol{f}_* := \boldsymbol{f}_{\text{test}}, \boldsymbol{f} := \boldsymbol{f}_{\text{train}}$.

- Marginal

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

Marginal and Conditional over Functions

Define $f_* := f_{\text{test}}, f := f_{\text{train}}$.

- Marginal

$$p(f_*, f) = \mathcal{N}\left(\begin{bmatrix}\mu_f \\ \mu_*\end{bmatrix}, \begin{bmatrix}\Sigma_{ff} & \Sigma_{f*} \\ \Sigma_{*f} & \Sigma_{**}\end{bmatrix}\right)$$

- Conditional (predictive distribution)

$$p(f_*|f) = \mathcal{N}(m, S)$$
$$m = \mu_* + \Sigma_{*f}\Sigma_{ff}^{-1}(f - \mu)$$
$$S = \Sigma_{**} - \Sigma_{*f}\Sigma_{ff}^{-1}\Sigma_{f*}$$

Marginal and Conditional over Functions

Define $f_* := f_{\text{test}}, f := f_{\text{train}}$.

- Marginal

$$p(f_*, f) = \mathcal{N}\left(\begin{bmatrix}\mu_f \\ \mu_*\end{bmatrix}, \begin{bmatrix}\Sigma_{ff} & \Sigma_{f*} \\ \Sigma_{*f} & \Sigma_{**}\end{bmatrix}\right)$$

- Conditional (predictive distribution)

$$p(f_*|f) = \mathcal{N}(m, S)$$
$$m = \mu_* + \Sigma_{*f}\Sigma_{ff}^{-1}(f - \mu)$$
$$S = \Sigma_{**} - \Sigma_{*f}\Sigma_{ff}^{-1}\Sigma_{f*}$$

⏩ We need to compute (cross-)covariances between unknown function values

Marginal and Conditional over Functions

Define $f_* := f_{\text{test}}, f := f_{\text{train}}$.

- Marginal

$$p(f_*, f) = \mathcal{N}\left(\begin{bmatrix}\mu_f \\ \mu_*\end{bmatrix}, \begin{bmatrix}\Sigma_{ff} & \Sigma_{f*} \\ \Sigma_{*f} & \Sigma_{**}\end{bmatrix}\right)$$

- Conditional (predictive distribution)

$$p(f_*|f) = \mathcal{N}(m, S)$$
$$m = \mu_* + \Sigma_{*f}\Sigma_{ff}^{-1}(f - \mu)$$
$$S = \Sigma_{**} - \Sigma_{*f}\Sigma_{ff}^{-1}\Sigma_{f*}$$

- We need to compute (cross-)covariances between unknown function values
- The kernel trick helps us out

Kernelization

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

$$p(\boldsymbol{f}_*|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$$

$$\boldsymbol{m} = \boldsymbol{\mu}_* + \boldsymbol{\Sigma}_{*f}\boldsymbol{\Sigma}_{ff}^{-1}(\boldsymbol{f} - \boldsymbol{\mu})$$

$$\boldsymbol{S} = \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_{*f}\boldsymbol{\Sigma}_{ff}^{-1}\boldsymbol{\Sigma}_{f*}$$

Kernelization

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

$$p(\boldsymbol{f}_*|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$$

$$\boldsymbol{m} = \boldsymbol{\mu}_* + \boldsymbol{\Sigma}_{*f}\boldsymbol{\Sigma}_{ff}^{-1}(\boldsymbol{f} - \boldsymbol{\mu})$$

$$\boldsymbol{S} = \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_{*f}\boldsymbol{\Sigma}_{ff}^{-1}\boldsymbol{\Sigma}_{f*}$$

- A kernel function k is symmetric and positive definite.

Kernelization

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

$$p(\boldsymbol{f}_*|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$$

$$\boldsymbol{m} = \boldsymbol{\mu}_* + \boldsymbol{\Sigma}_{*f}\boldsymbol{\Sigma}_{ff}^{-1}(\boldsymbol{f} - \boldsymbol{\mu})$$

$$\boldsymbol{S} = \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_{*f}\boldsymbol{\Sigma}_{ff}^{-1}\boldsymbol{\Sigma}_{f*}$$

- A kernel function k is symmetric and positive definite.
- Kernel computes covariances between unknown function values $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ by just looking at the corresponding inputs $\boldsymbol{x}_i, \boldsymbol{x}_j$

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

Kernelization

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

$$p(\boldsymbol{f}_*|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$$

$$\boldsymbol{m} = \boldsymbol{\mu}_* + \boldsymbol{\Sigma}_{*f}\boldsymbol{\Sigma}_{ff}^{-1}(\boldsymbol{f} - \boldsymbol{\mu})$$

$$\boldsymbol{S} = \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_{*f}\boldsymbol{\Sigma}_{ff}^{-1}\boldsymbol{\Sigma}_{f*}$$

- A kernel function k is symmetric and positive definite.
- Kernel computes covariances between unknown function values $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ by just looking at the corresponding inputs $\boldsymbol{x}_i, \boldsymbol{x}_j$

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

- This yields the predictive distribution

$$p(\boldsymbol{f}_*|\boldsymbol{f}, \boldsymbol{X}, \boldsymbol{X}_*) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$$

$$\boldsymbol{m} = \boldsymbol{\mu}_* + k(\boldsymbol{X}_*, \boldsymbol{X})k(\boldsymbol{X}, \boldsymbol{X})^{-1}(\boldsymbol{f} - \boldsymbol{\mu})$$

$$\boldsymbol{S} = \boldsymbol{K}_{**} - k(\boldsymbol{X}_*, \boldsymbol{X})k(\boldsymbol{X}, \boldsymbol{X})^{-1}k(\boldsymbol{X}, \boldsymbol{X}_*)$$

GP Regression as a Bayesian Inference Problem

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

GP Regression as a Bayesian Inference Problem

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

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 $y_i = f(\boldsymbol{x}_i) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, find a distribution over functions $p(f)$ 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

GP Regression as a Bayesian Inference Problem

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

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

GP Regression as a Bayesian Inference Problem

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

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

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

GP Regression as a Bayesian Inference Problem

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

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

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

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

GP Regression as a Bayesian Inference Problem

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

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

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

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

$$m_{\text{post}}(\boldsymbol{x}_i) = m(\boldsymbol{x}_i) + k(\boldsymbol{X}, \boldsymbol{x}_i)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - m(\boldsymbol{x}_i))$$

$$k_{\text{post}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) - k(\boldsymbol{X}, \boldsymbol{x}_i)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_j)$$

GP Regression as a Bayesian Inference Problem (2)

Posterior Gaussian process:

$$m_{\text{post}}(\boldsymbol{x}_i) = m(x_i) + k(\boldsymbol{X}, \boldsymbol{x}_i)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - m(\boldsymbol{x}_i))$$

$$k_{\text{post}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) - k(\boldsymbol{X}, \boldsymbol{x}_i)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_j)$$

GP Regression as a Bayesian Inference Problem (2)

Posterior Gaussian process:

$$m_{\text{post}}(\boldsymbol{x}_i) = m(x_i) + k(\boldsymbol{X}, \boldsymbol{x}_i)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - m(\boldsymbol{x}_i))$$
$$k_{\text{post}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) - k(\boldsymbol{X}, \boldsymbol{x}_i)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_j)$$

Predictive distribution $p(f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*)$ at test inputs $\boldsymbol{x}_*$:

$$p(f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*) = \mathcal{N}\big(\mathbb{E}[f_*],\, \mathbb{V}[f_*]\big)$$
$$\mathbb{E}[f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*] = m_{\text{post}}(\boldsymbol{x}_*) = m(\boldsymbol{x}_*) + k(\boldsymbol{X}, \boldsymbol{x}_*)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - m(\boldsymbol{x}_*))$$
$$\mathbb{V}[f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*] = k_{\text{post}}(\boldsymbol{x}_*, \boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)$$

GP Regression as a Bayesian Inference Problem (2)

Posterior Gaussian process:

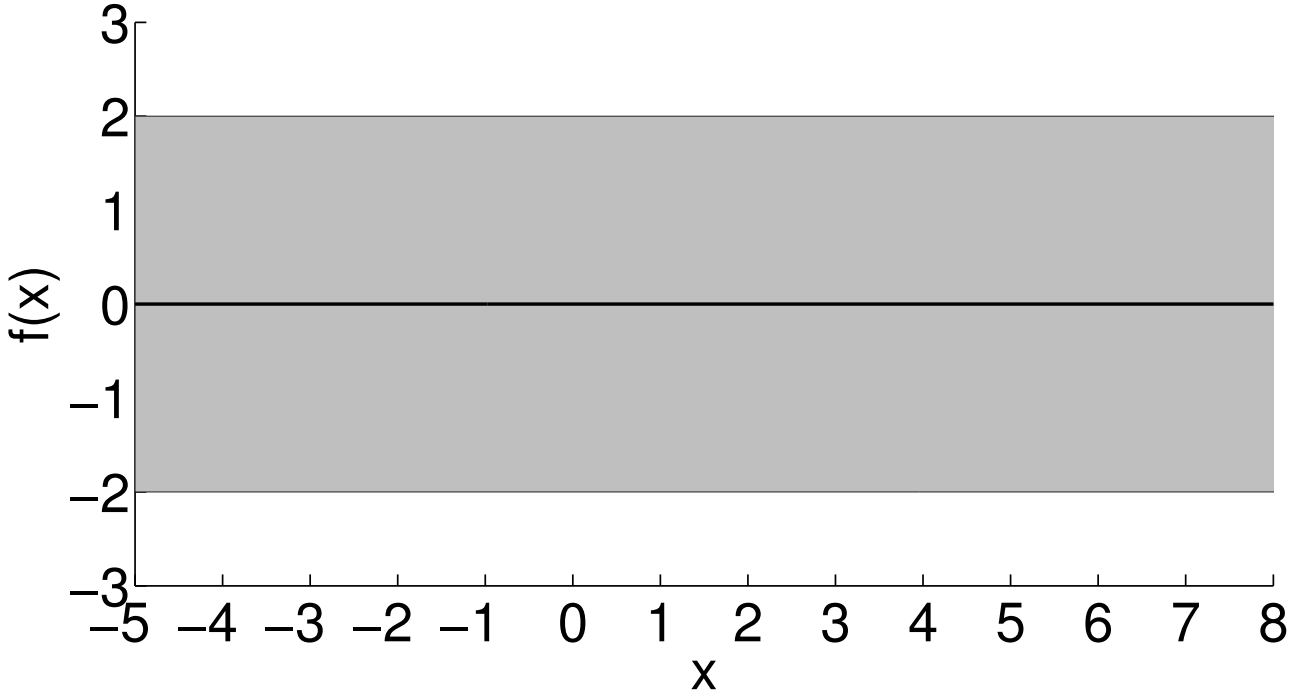
$$m_{\text{post}}(\boldsymbol{x}_i) = m(x_i) + k(\boldsymbol{X}, \boldsymbol{x}_i)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - m(\boldsymbol{x}_i))$$
$$k_{\text{post}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j) - k(\boldsymbol{X}, \boldsymbol{x}_i)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_j)$$

Predictive distribution $p(f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*)$ at test inputs $\boldsymbol{x}_*$:

$$p(f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*) = \mathcal{N}\left(\mathbb{E}[f_*],\, \mathbb{V}[f_*]\right)$$
$$\mathbb{E}[f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*] = m_{\text{post}}(\boldsymbol{x}_*) = m(\boldsymbol{x}_*) + k(\boldsymbol{X}, \boldsymbol{x}_*)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - m(\boldsymbol{x}_*))$$
$$\mathbb{V}[f_* | \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_*] = k_{\text{post}}(\boldsymbol{x}_*, \boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top (\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)$$

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

Illustration

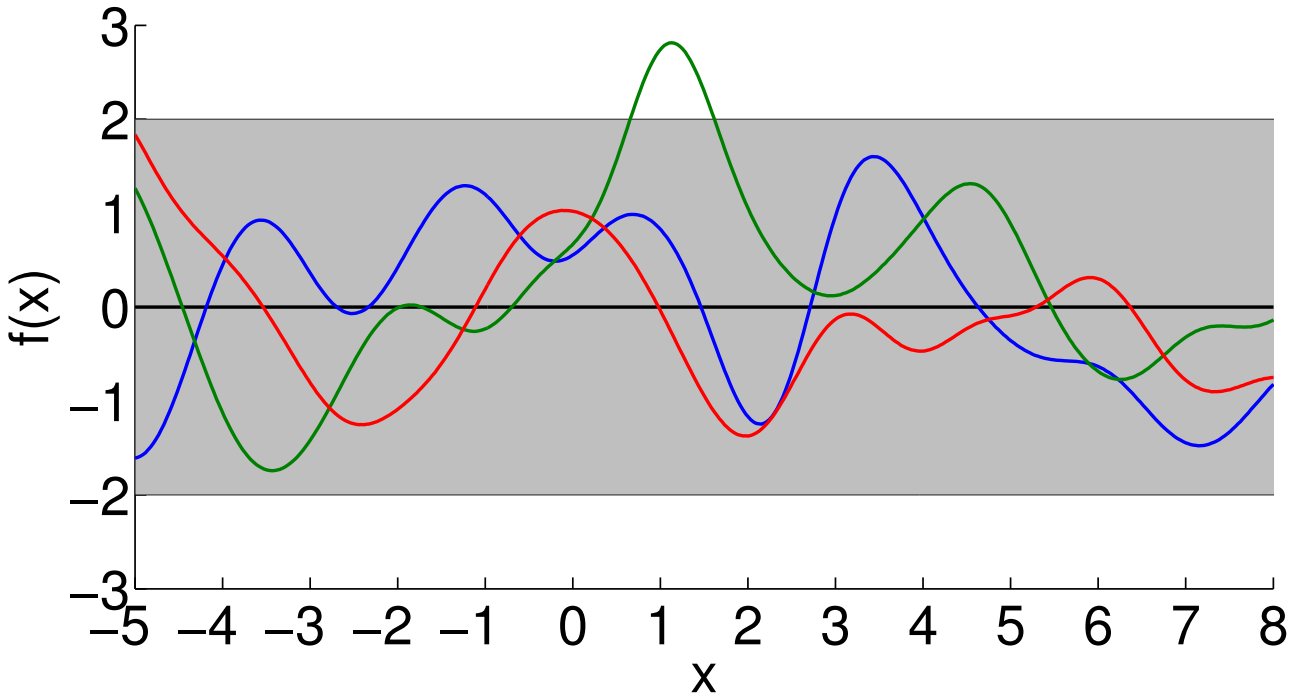


Prior belief about the function

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

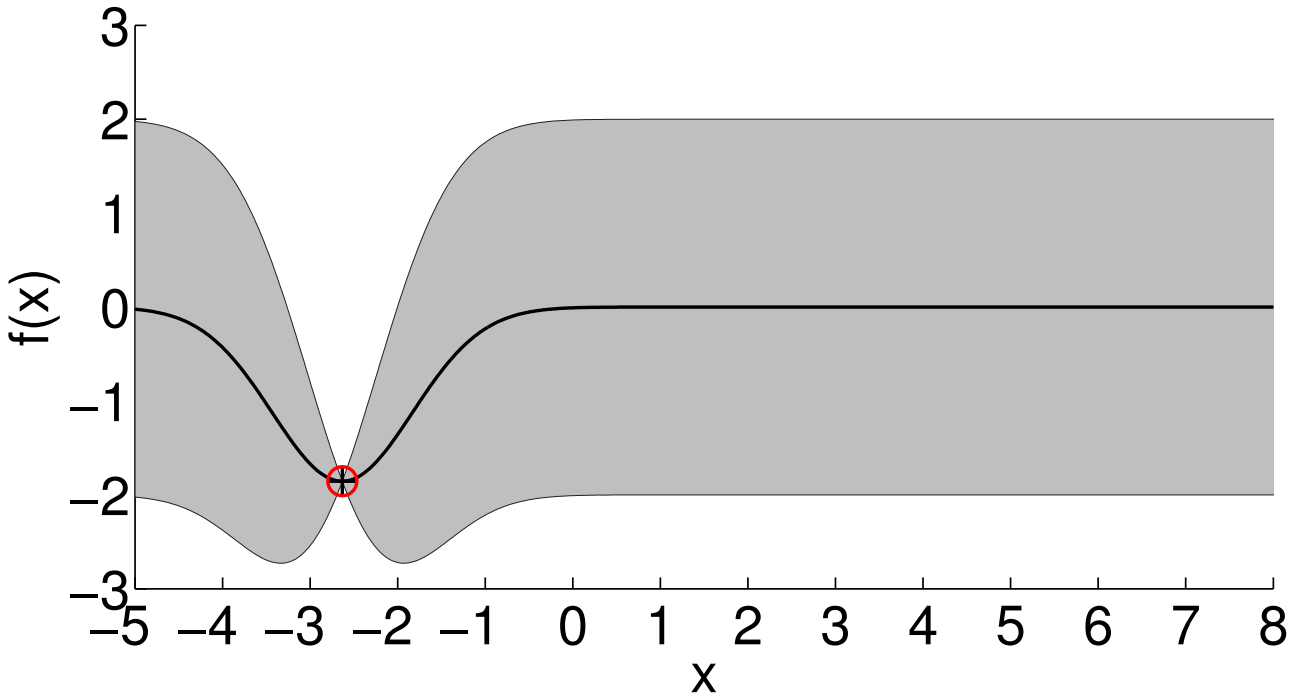


Prior belief about the function

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



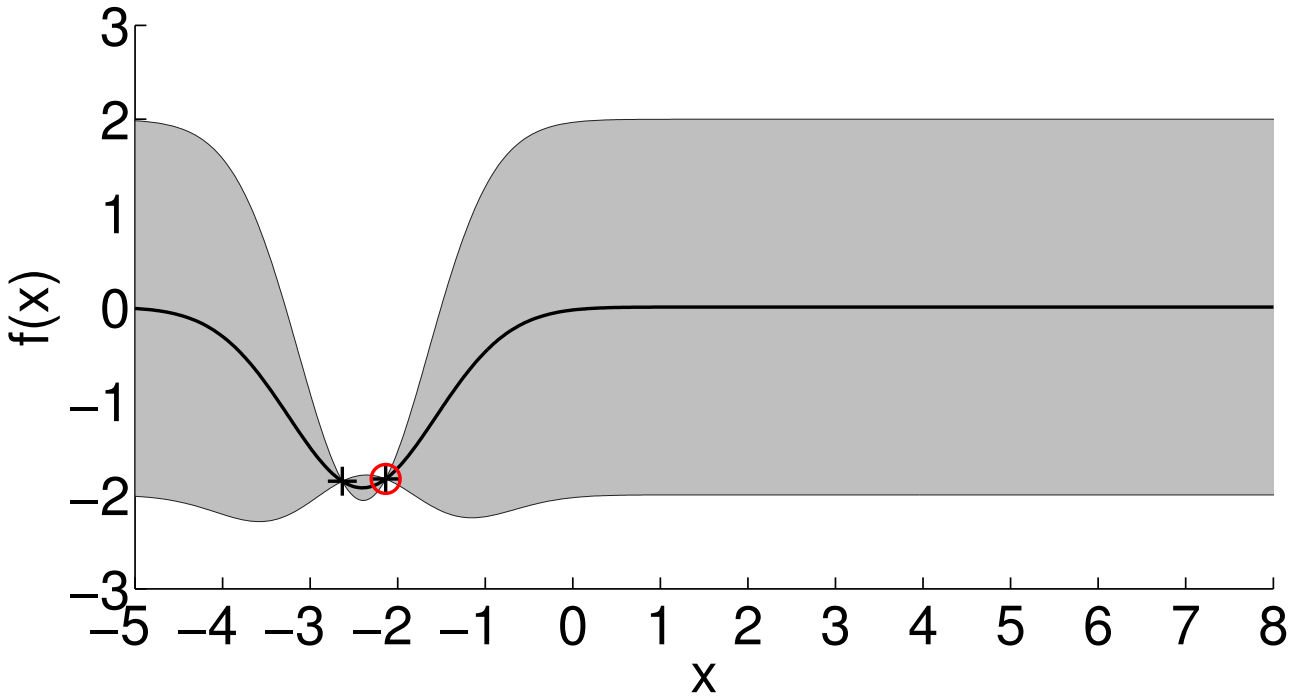
Posterior belief about the function

Predictive (marginal) mean and variance:

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

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

Illustration

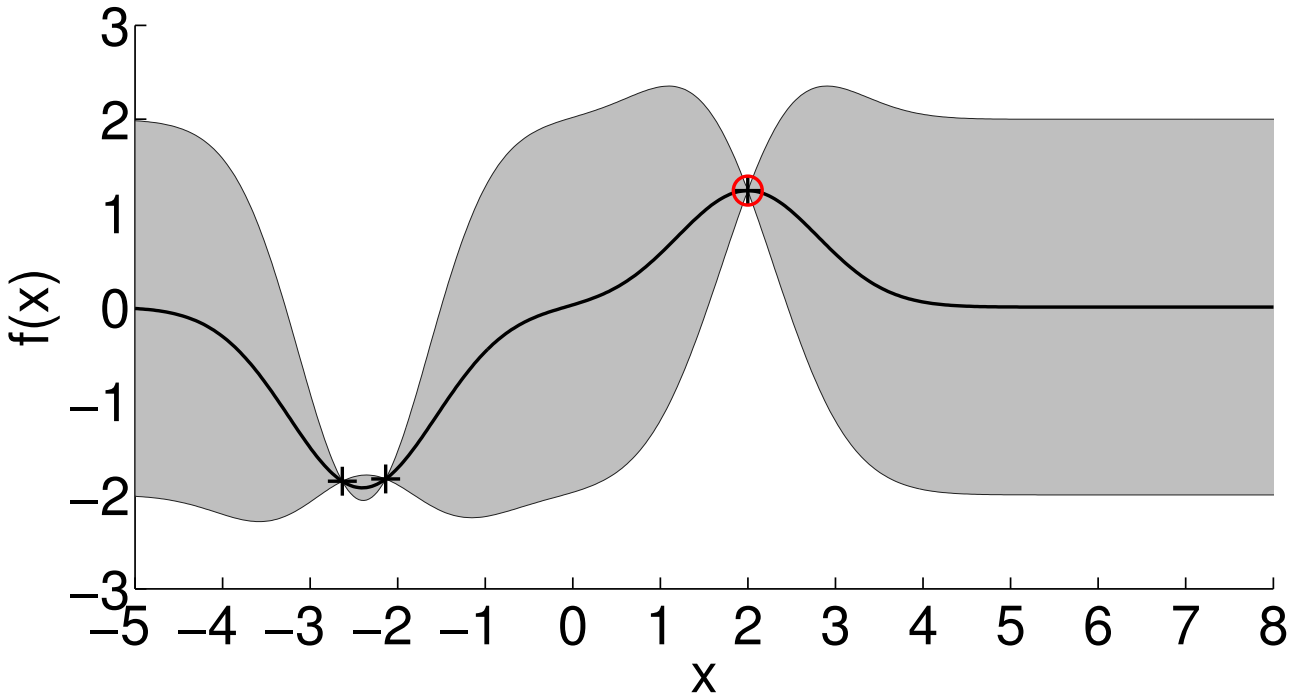


Posterior belief about the function

Predictive (marginal) mean and variance:

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

Illustration



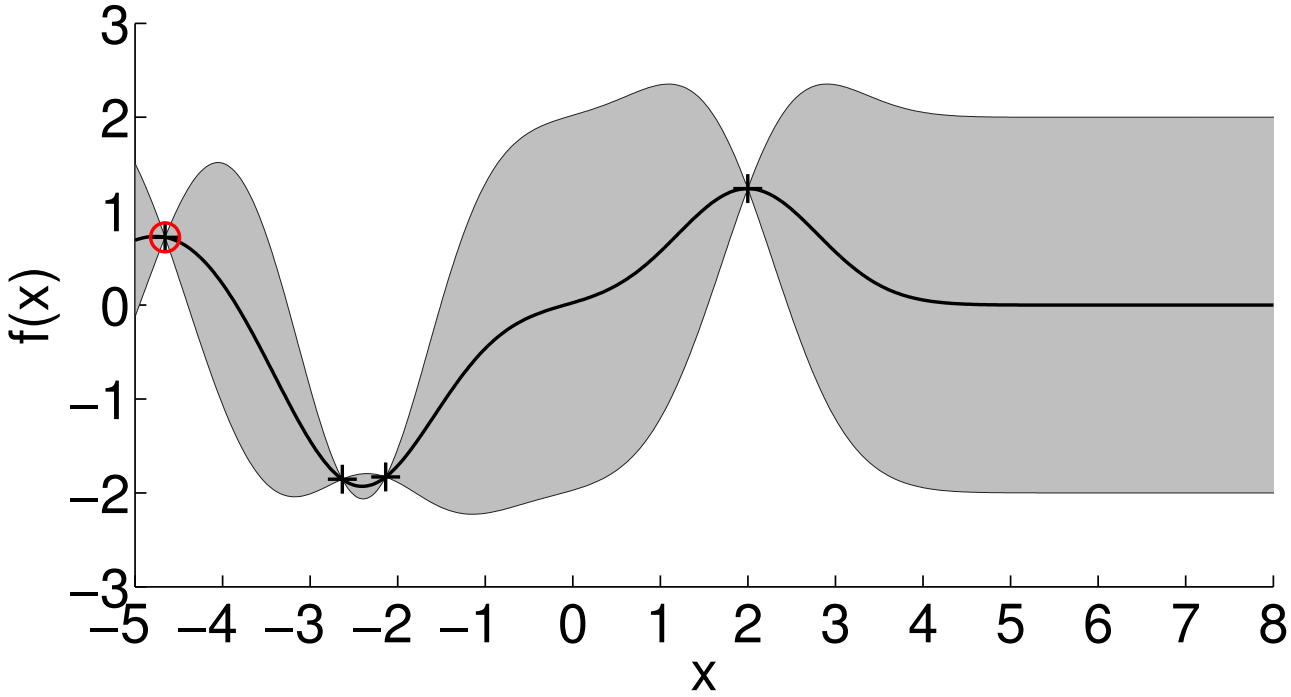
Posterior belief about the function

Predictive (marginal) mean and variance:

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

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

Illustration



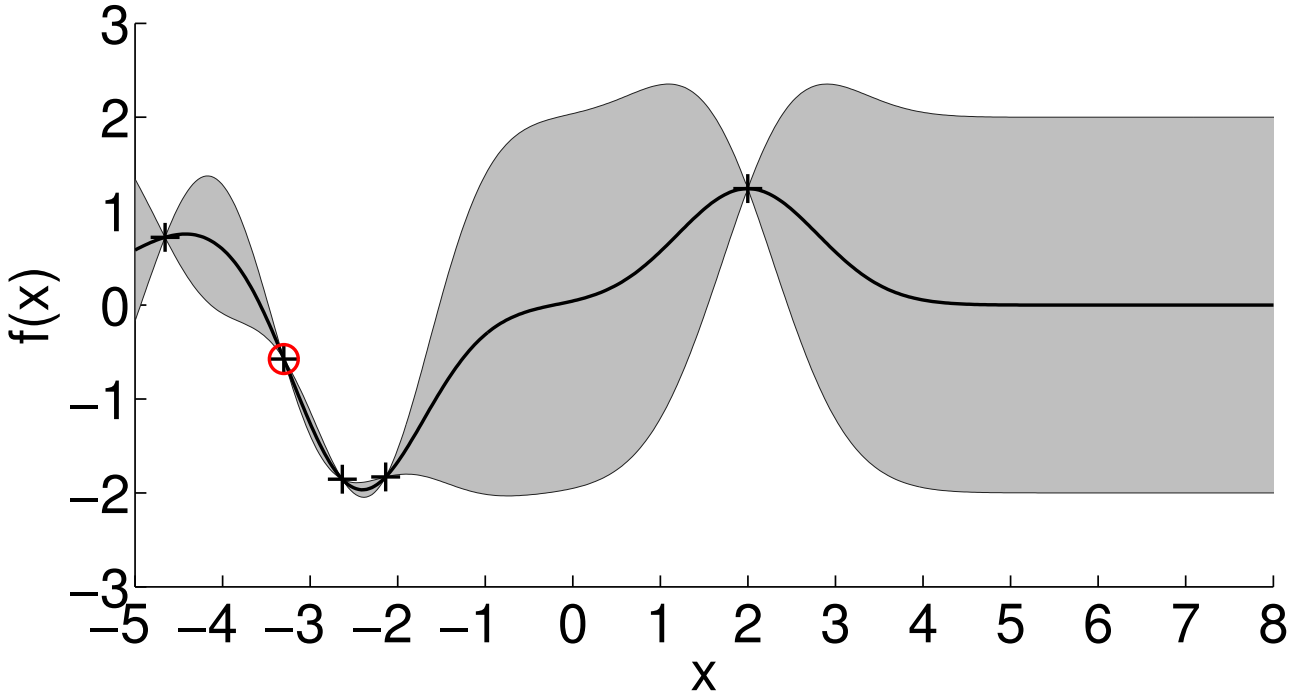
Posterior belief about the function

Predictive (marginal) mean and variance:

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

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

Illustration



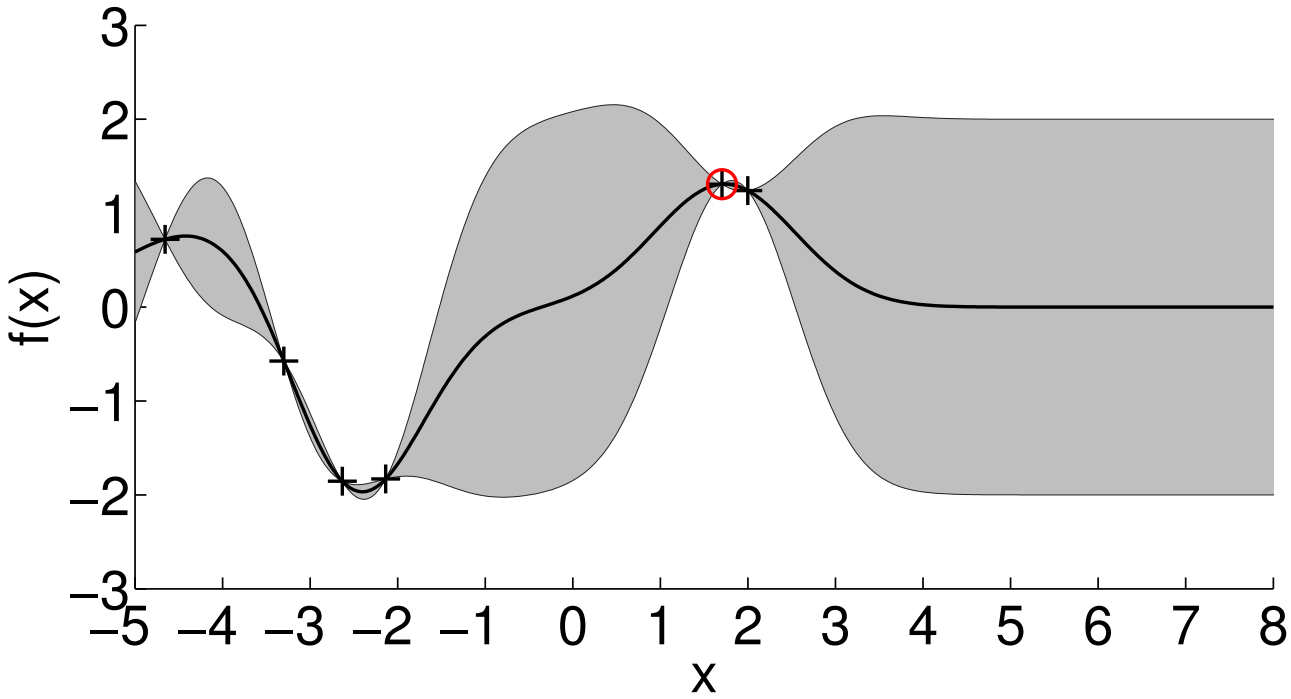
Posterior belief about the function

Predictive (marginal) mean and variance:

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

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

Illustration

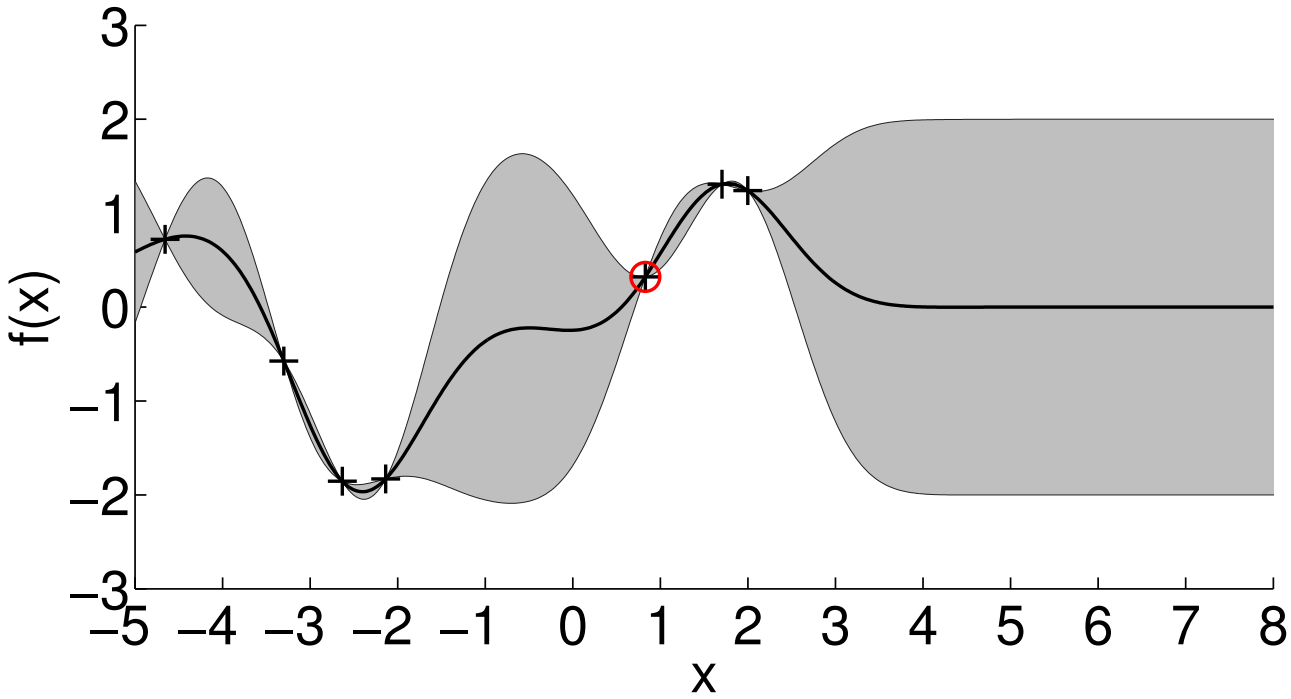


Posterior belief about the function

Predictive (marginal) mean and variance:

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

Illustration



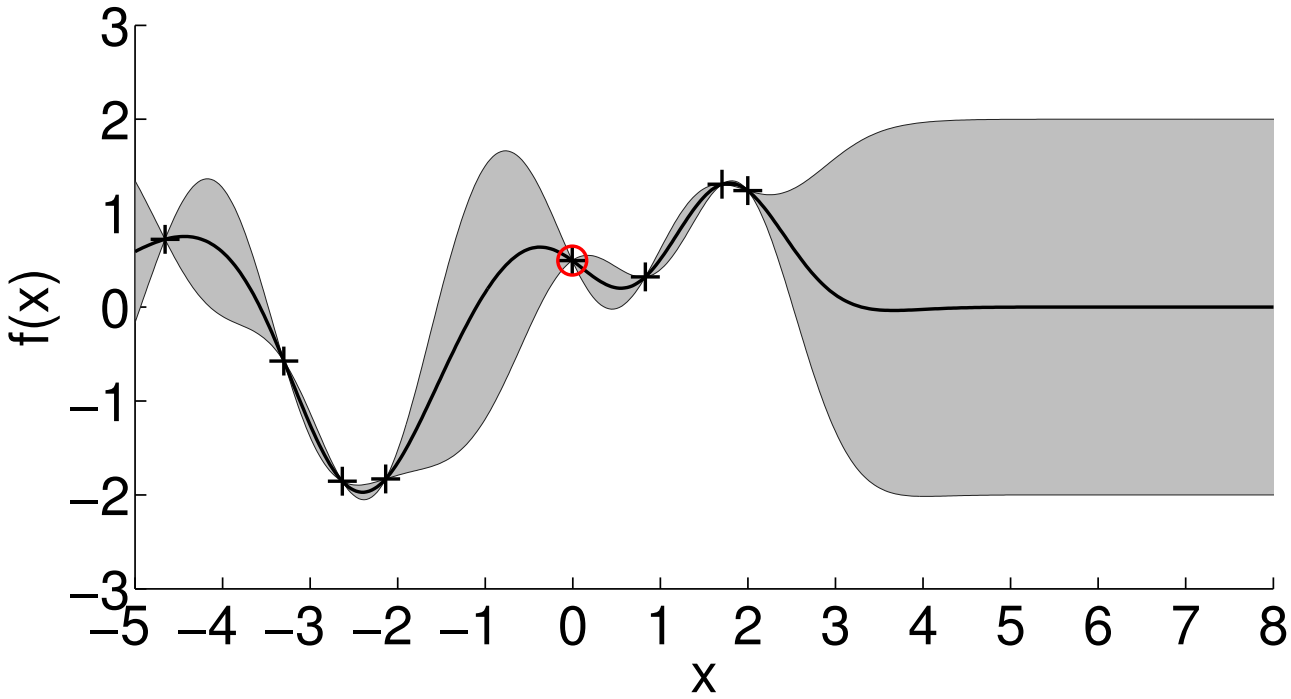
Posterior belief about the function

Predictive (marginal) mean and variance:

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

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

Illustration



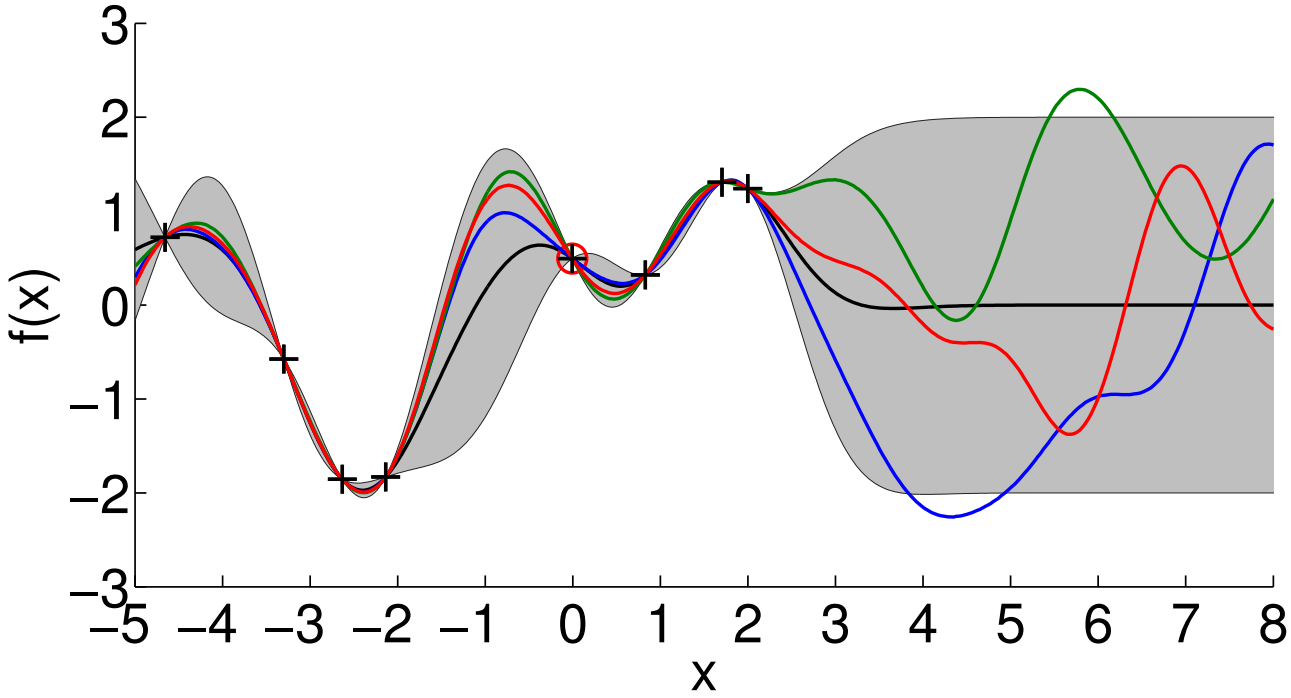
Posterior belief about the function

Predictive (marginal) mean and variance:

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

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

Illustration



Posterior belief about the function

Predictive (marginal) mean and variance:

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

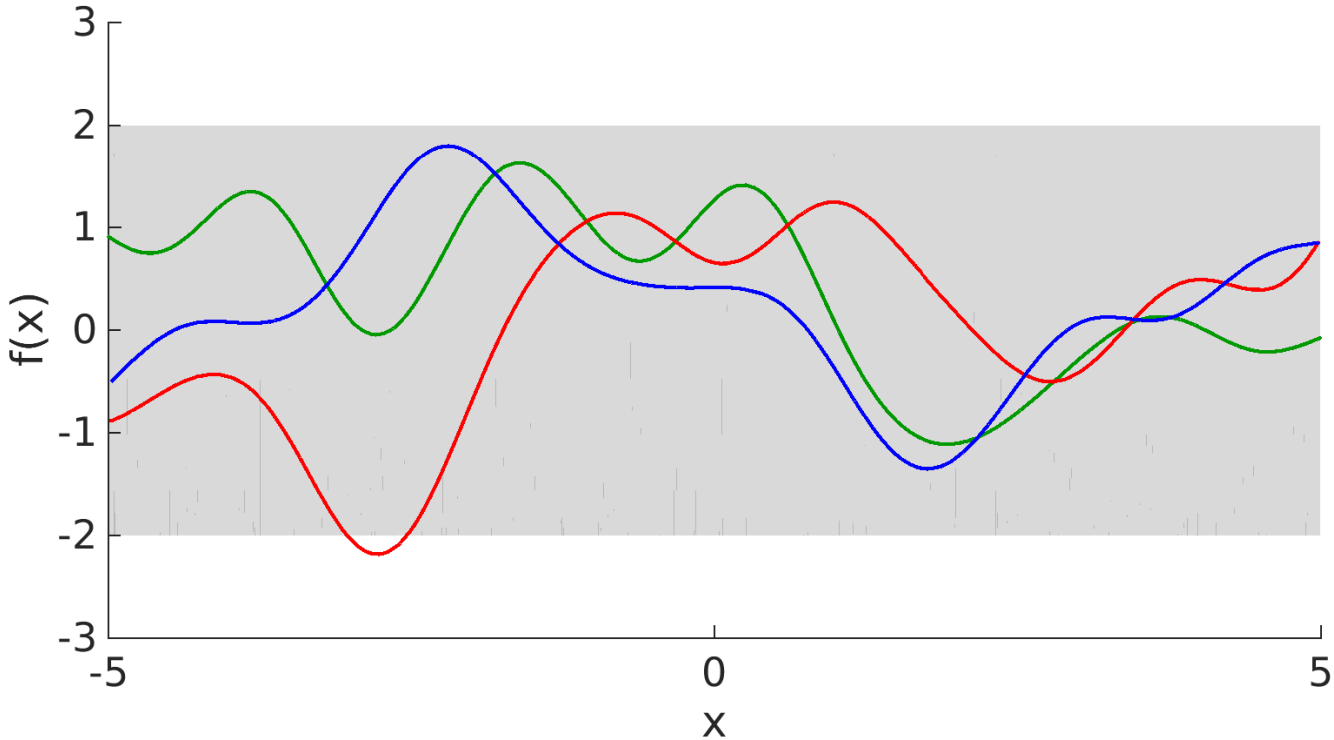
Covariance Function

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

Gaussian Covariance Function

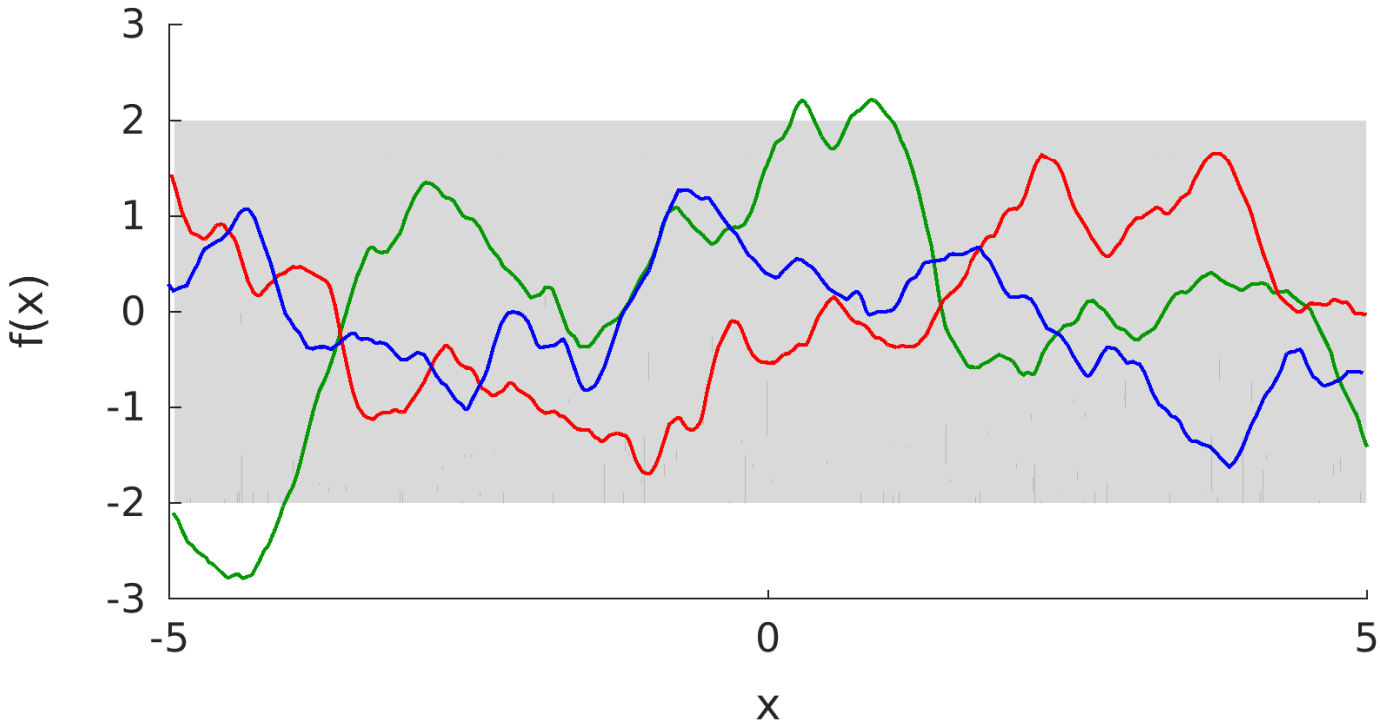
$$k_{Gauss}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta_1^2 \exp\left(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)/\theta_2^2\right)$$

- θ_1: Amplitude of the latent function
- θ_2: Length scale. How far do we have to move in input space before the function value changes significantly
  - ⏩ Smoothness parameter



Matérn Covariance Function

$$k_{Mat,3/2}(x_i, x_j) = \theta_1^2 \left(1 + \frac{\sqrt{3}(x_i - x_j)}{\theta_2}\right) \exp\left(-\frac{\sqrt{3}(x_i - x_j)}{\theta_2}\right)$$

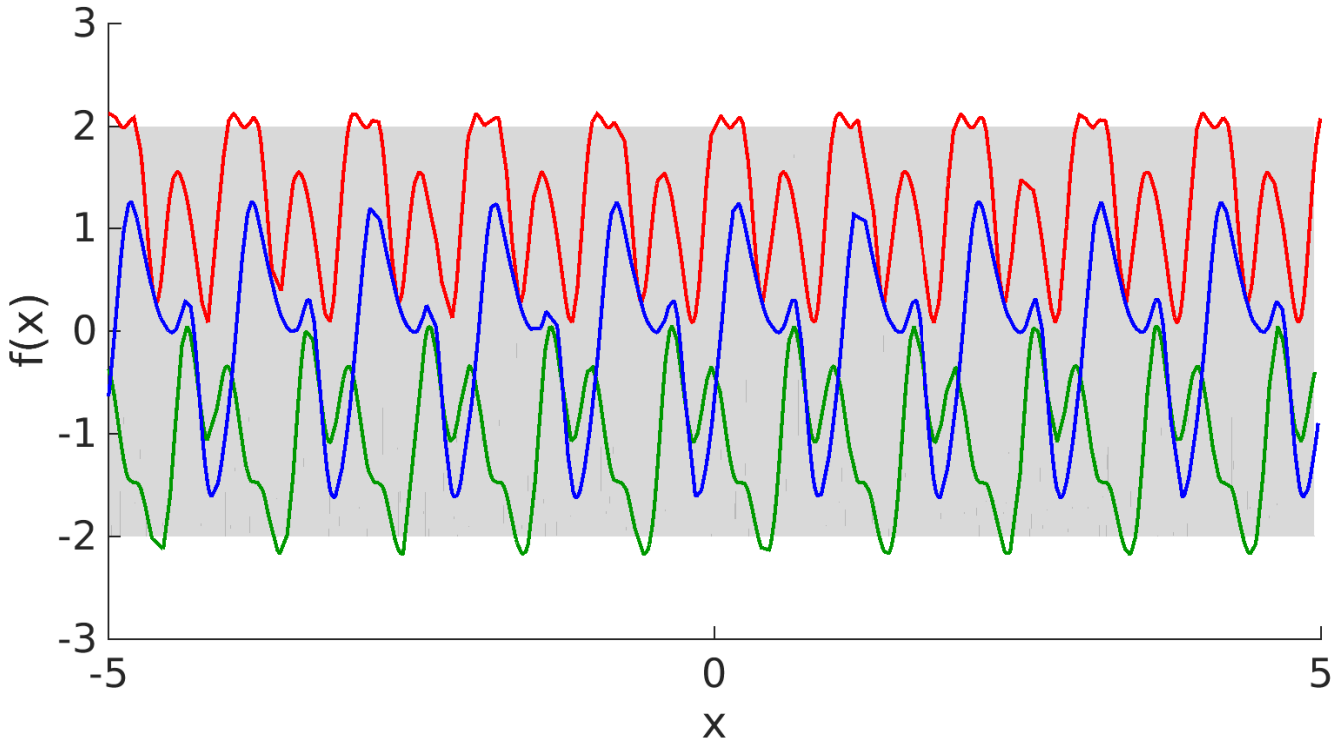


Periodic Covariance Function

$$k_{per}(x_i, x_j) = \theta_1^2 \exp\left(- \frac{2 \sin^2\left(\frac{\theta_3 (x_i - x_j)}{2\pi}\right)}{\theta_2^2} \right)$$

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

θ_3: Periodicity parameter



Model Selection

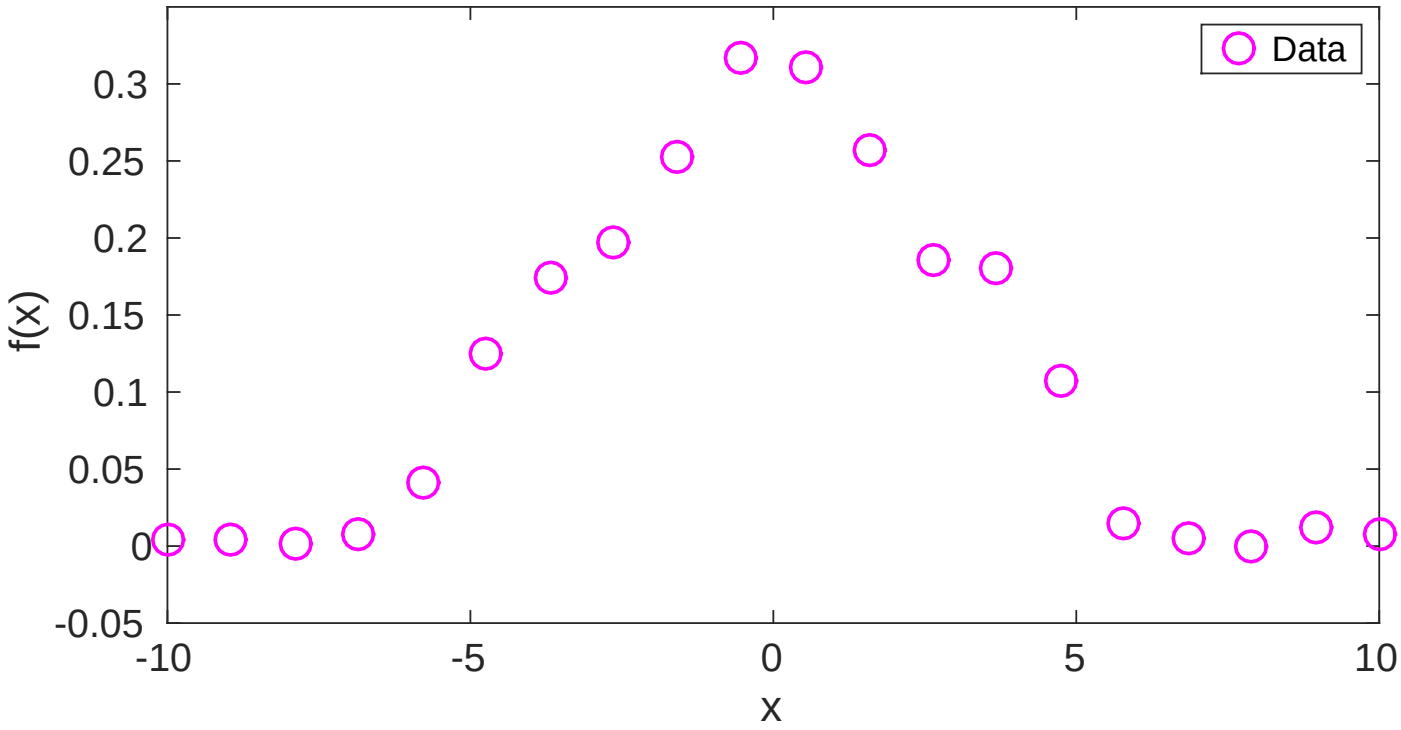
A GP is fully specified by a mean function m and a covariance function (kernel) k. Both functions possess parameters $\{\boldsymbol{\theta}_m, \boldsymbol{\theta}_k\} =: \boldsymbol{\theta}$

- How do we find good parameters $\boldsymbol{\theta}$?
- How do we choose m and k?

⏩ Model selection

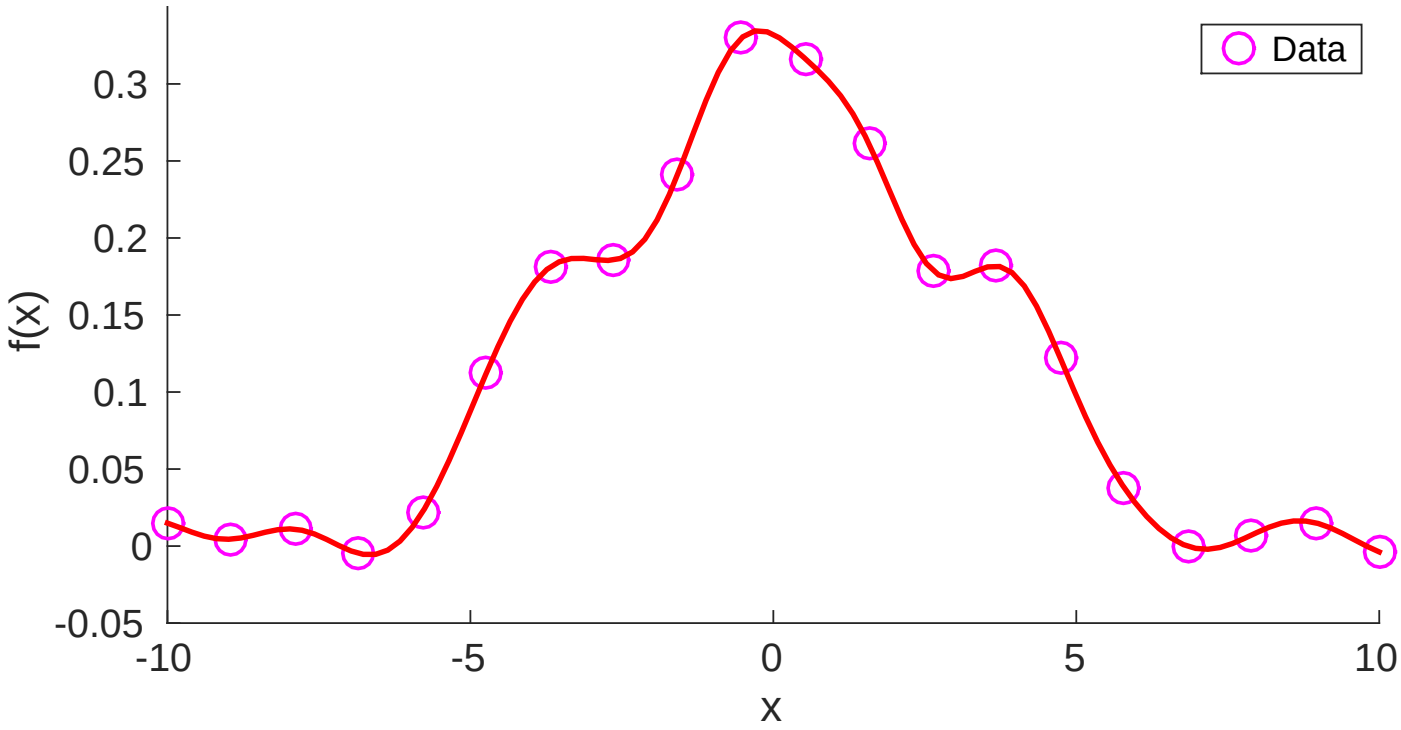
Model Selection I: Length-Scales

Length scales determine how wiggly the function is



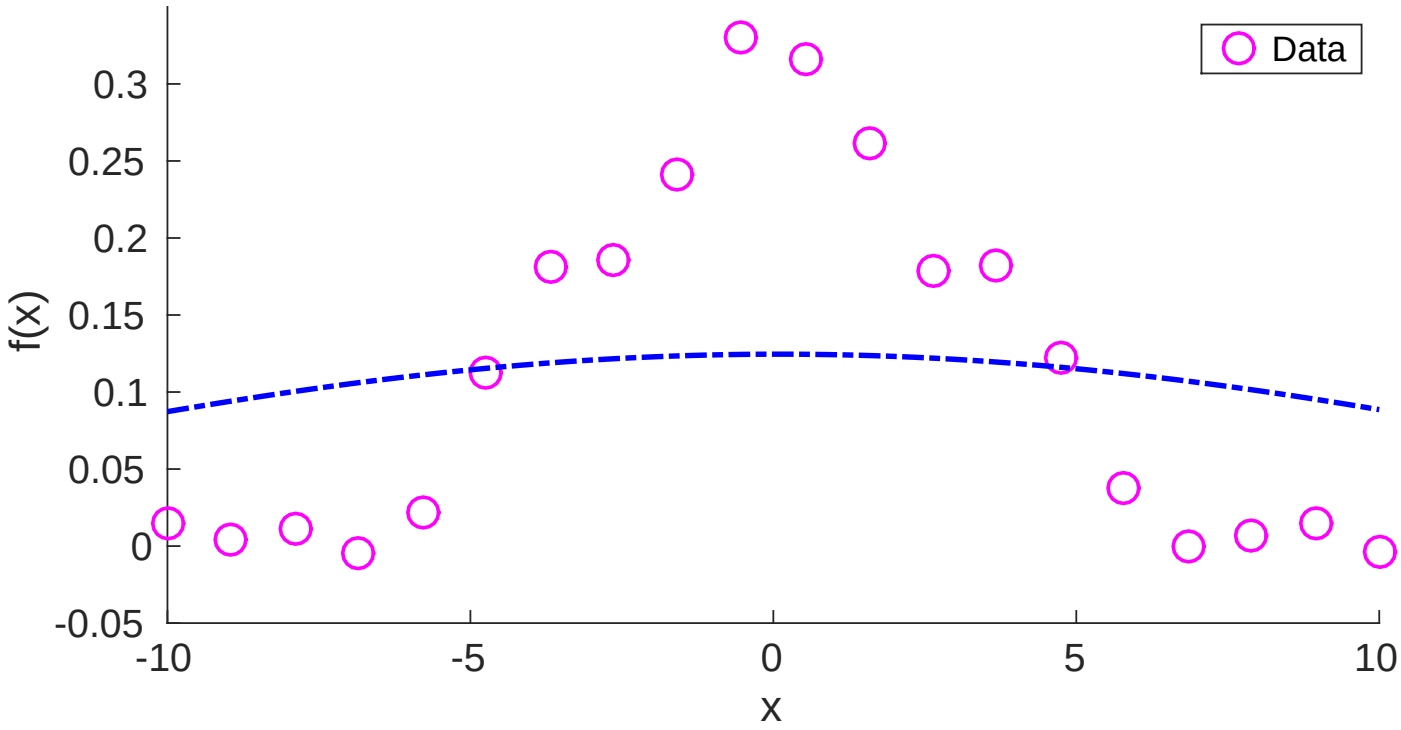
Model Selection I: Length-Scales

Length scales determine how wiggly the function is



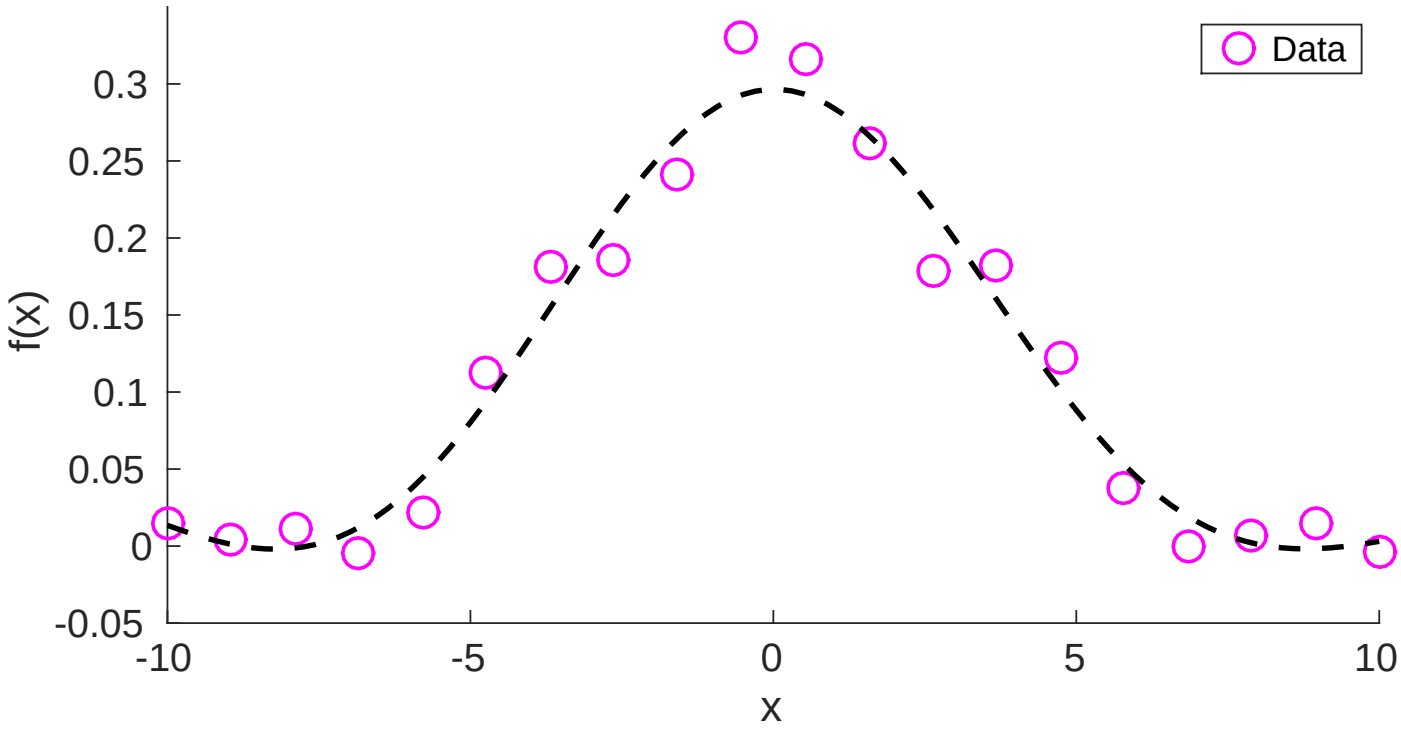
Model Selection I: Length-Scales

Length scales determine how wiggly the function is



Model Selection I: Length-Scales

Length scales determine how wiggly the function is



Model Selection: Hyper-Parameters

GP Training

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

Model Selection: Hyper-Parameters

GP Training

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

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

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

Model Selection: Hyper-Parameters

GP Training

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

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

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

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

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

Model Selection: Hyper-Parameters

GP Training

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

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

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

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

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

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

Training via Marginal Likelihood Maximization

GP Training

Maximize the evidence/marginal likelihood (probability of the data given the hyper-parameters)

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

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

Training via Marginal Likelihood Maximization

GP Training

Maximize the evidence/marginal likelihood (probability of the data given the hyper-parameters)

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

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

- Automatic trade-off between data fit and model complexity

Training via Marginal Likelihood Maximization

GP Training

Maximize the evidence/marginal likelihood (probability of the data given the hyper-parameters)

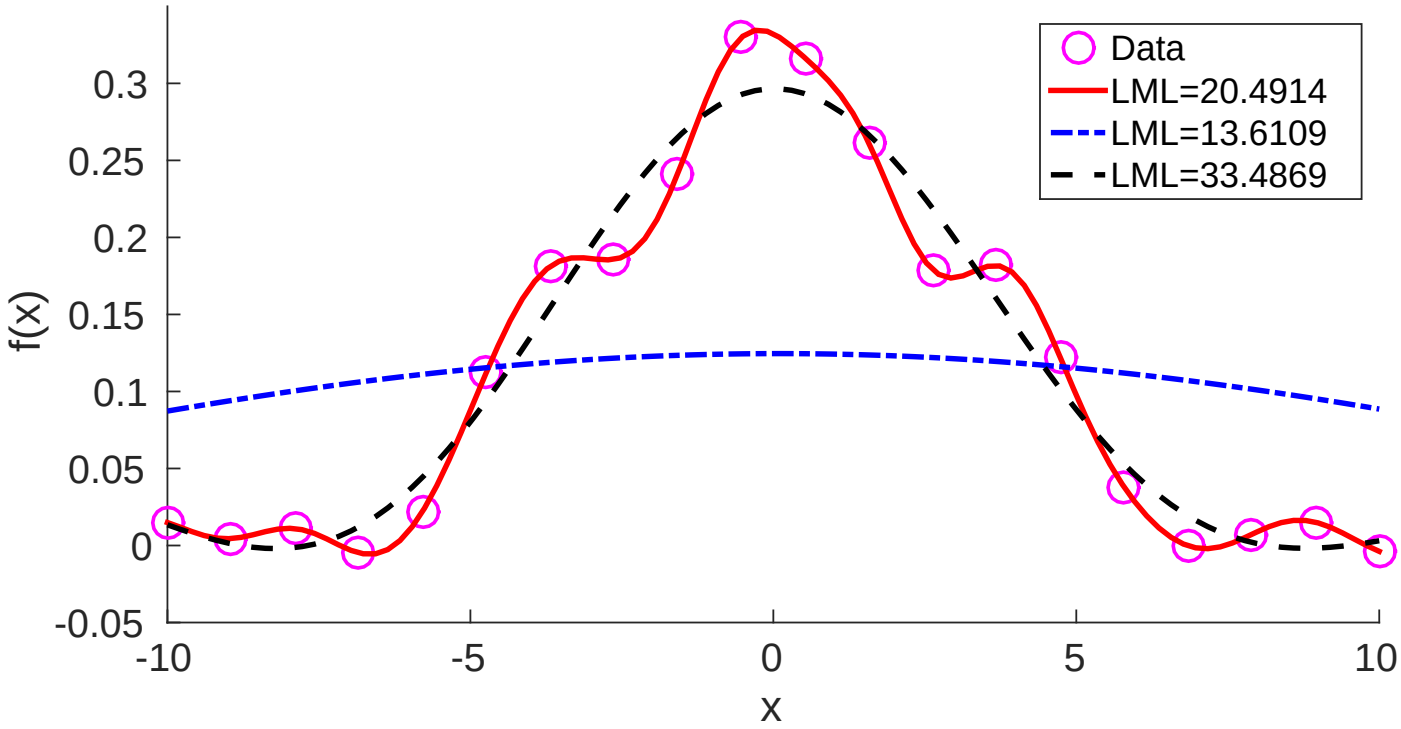
$$\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$$

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

- Automatic trade-off between data fit and model complexity
- Gradient-based optimization possible:

$$\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\theta}}\boldsymbol{K}^{-1}\boldsymbol{y} - \tfrac{1}{2}\mathrm{tr}\left(\boldsymbol{K}^{-1}\frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\theta}}\right)$$

Example



- Compromise between fitting the data and simplicity of the model
- Not fitting the data is explained by a larger noise variance (treated as an additional hyper-parameter and learned jointly)s

Model Selection II—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 II—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?
- Assign a prior $p(M_i)$ for each model
- Posterior model probability:

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

Model Selection II—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?
- Assign a prior $p(M_i)$ for each model
- Posterior model probability:

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

- Choose MAP model M_*, such that

$$M_* \in \arg\max_M \log p(M) + \log p(\boldsymbol{y}|\boldsymbol{X}, M)$$

Model Selection II—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?
- Assign a prior $p(M_i)$ for each model
- Posterior model probability:

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

- Choose MAP model M_*, such that

$$M_* \in \arg\max_M \log p(M) + \log p(\boldsymbol{y}|\boldsymbol{X}, M)$$

⏩ Compare marginal likelihoods if $p(M_i) = 1/|M|$

Model Selection II—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?
- Assign a prior $p(M_i)$ for each model
- Posterior model probability:

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

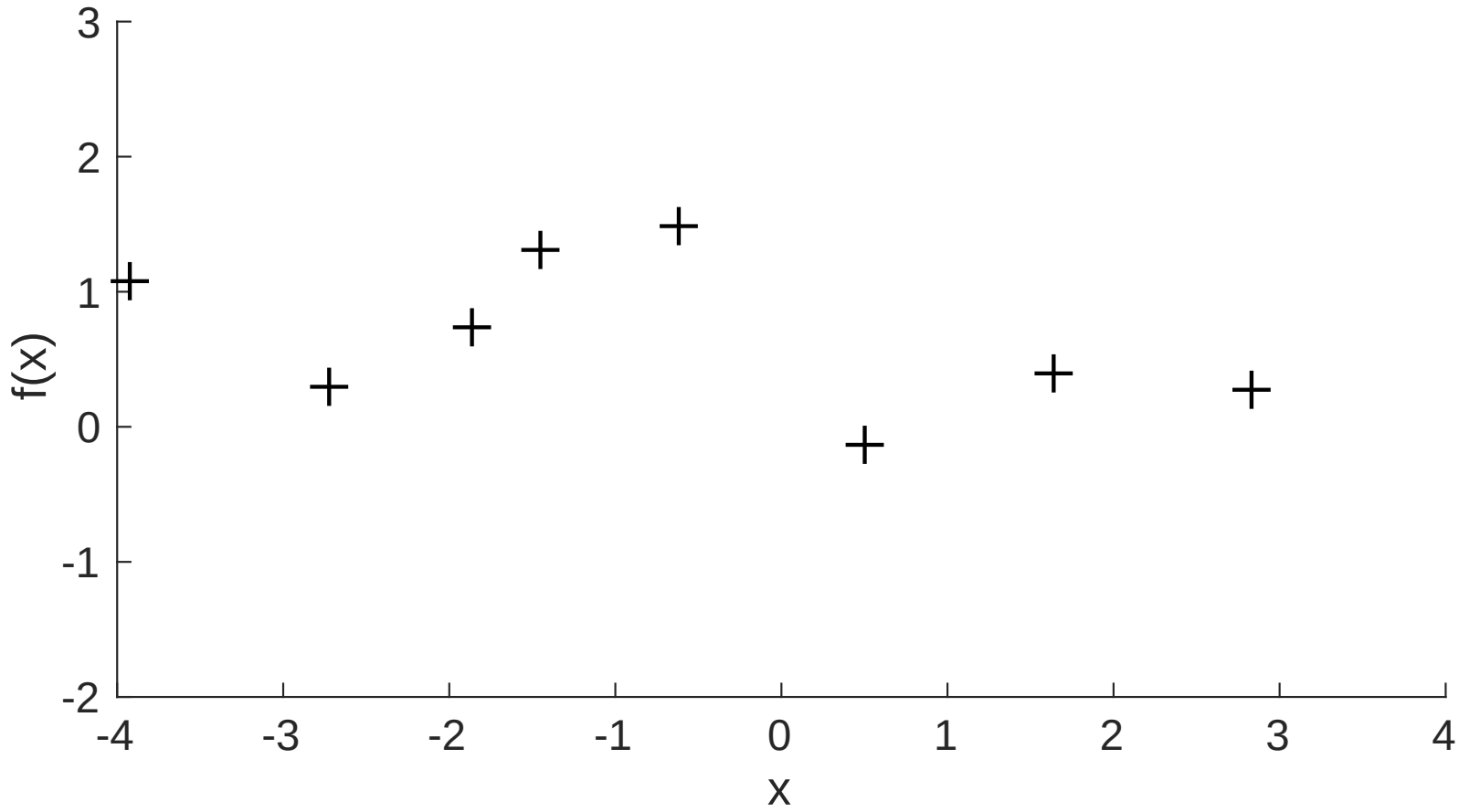
- Choose MAP model M_*, such that

$$M_* \in \arg\max_M \log p(M) + \log p(\boldsymbol{y}|\boldsymbol{X}, M)$$

⏩ Compare marginal likelihoods if $p(M_i) = 1/|M|$

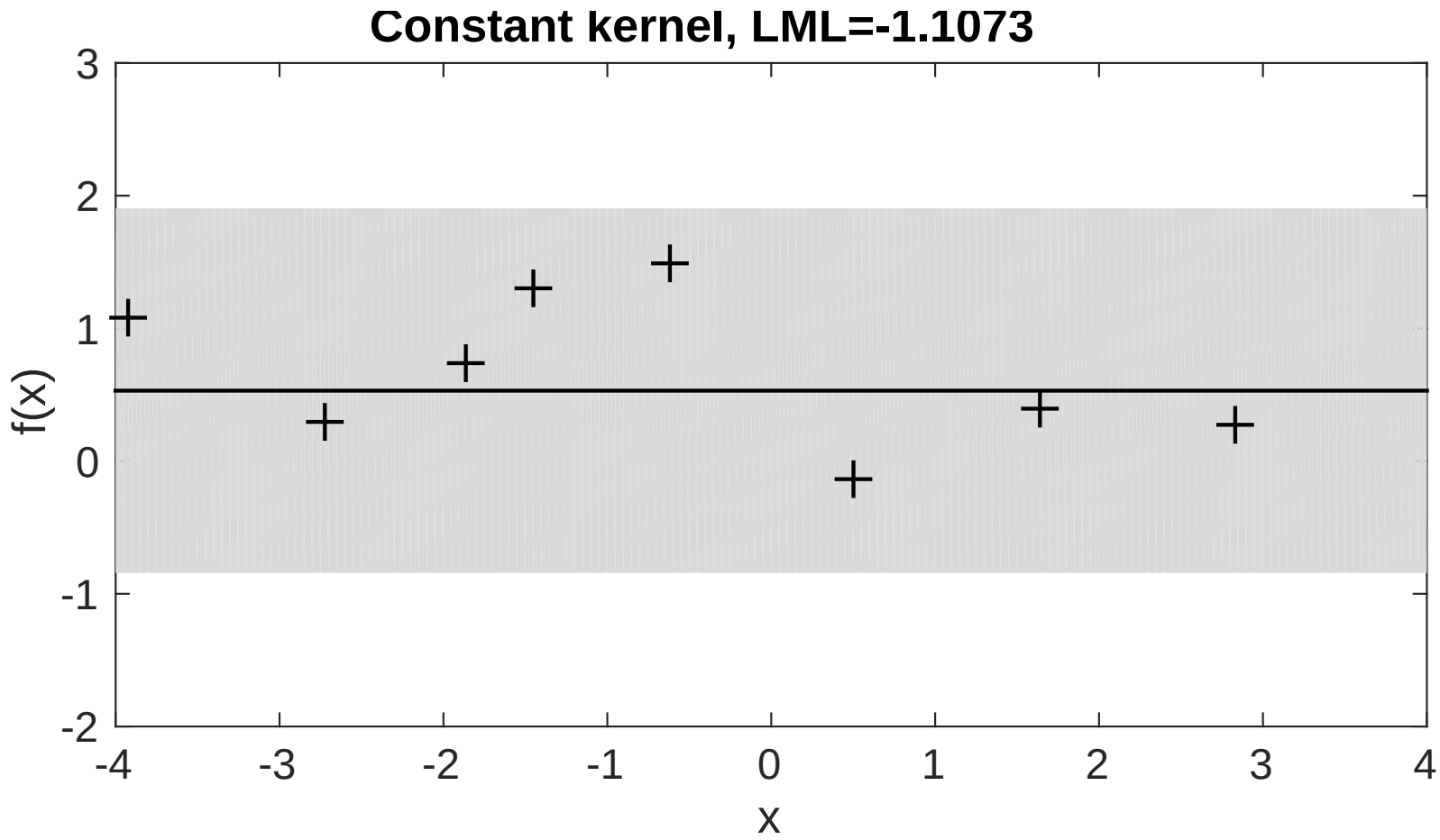
⏩ Marginal likelihood is central for model selection

Example



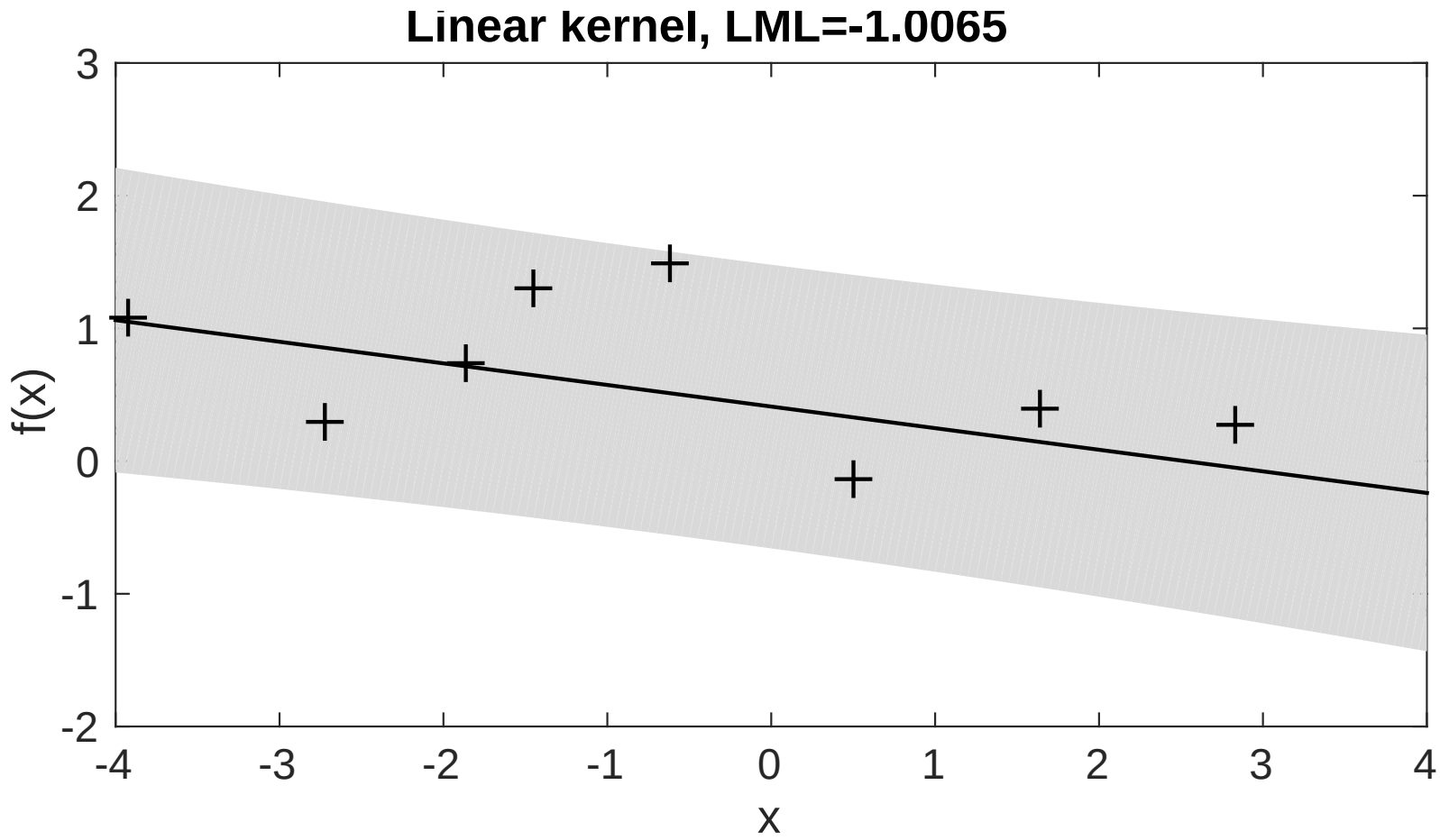
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



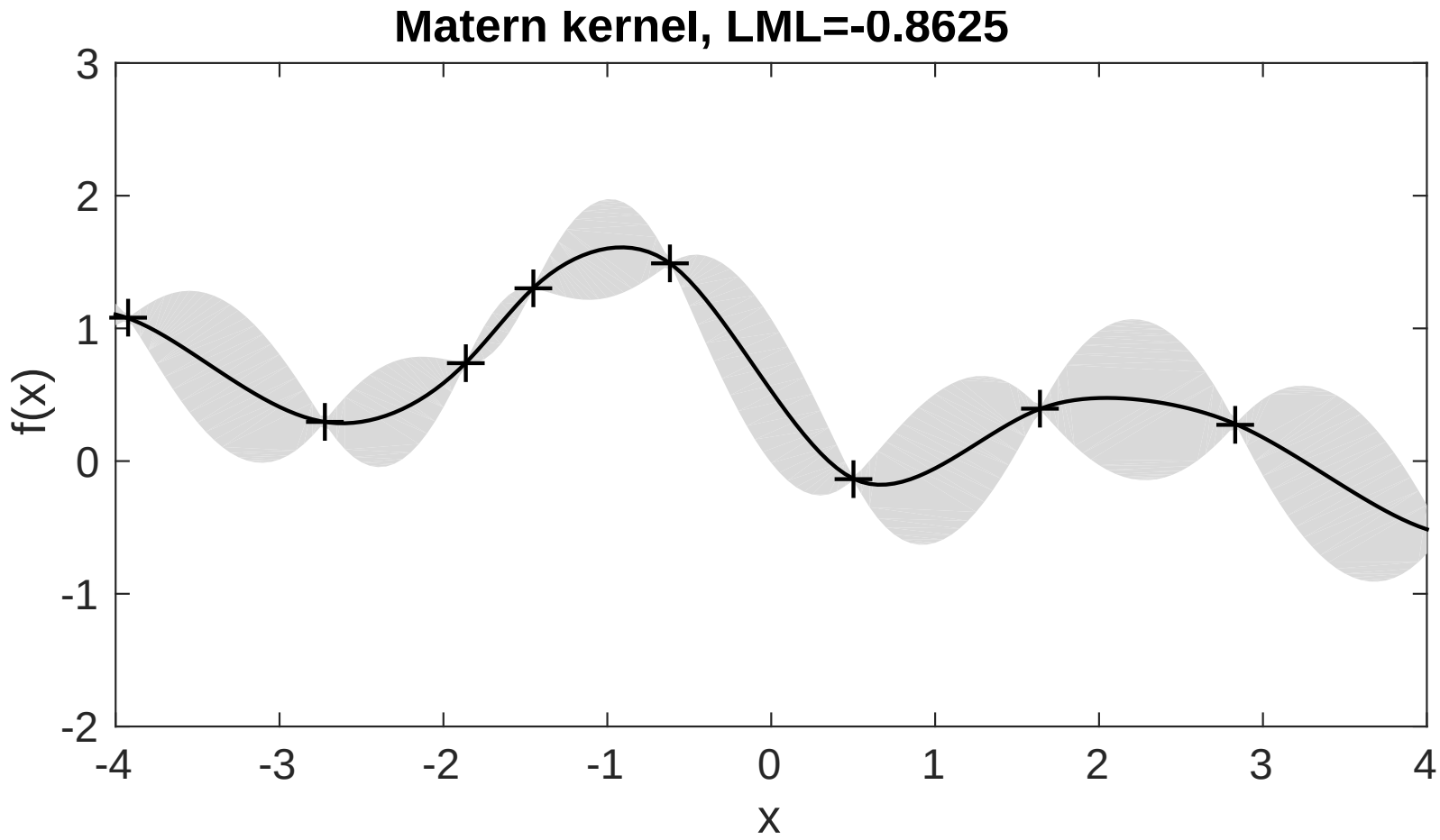
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



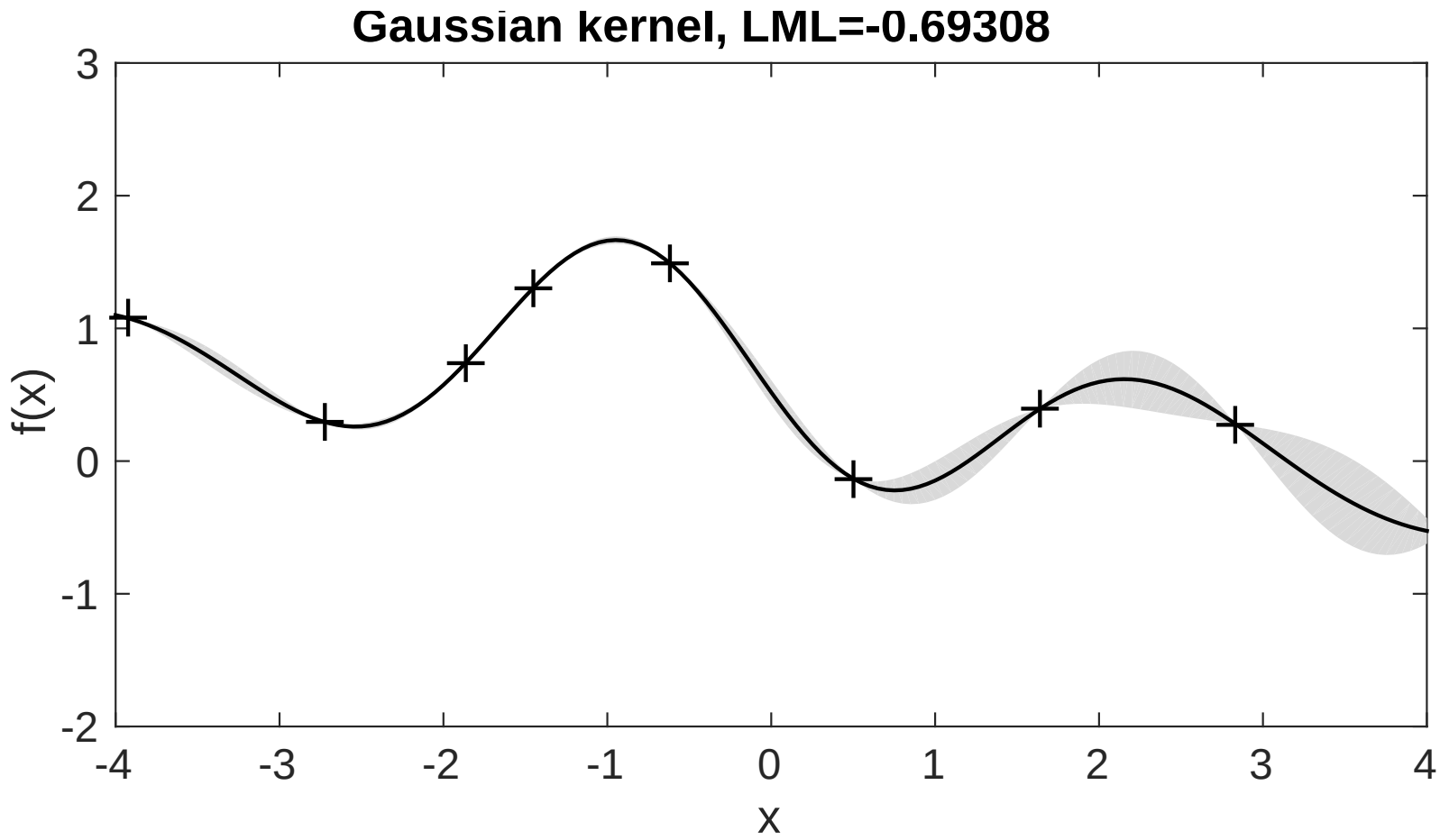
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



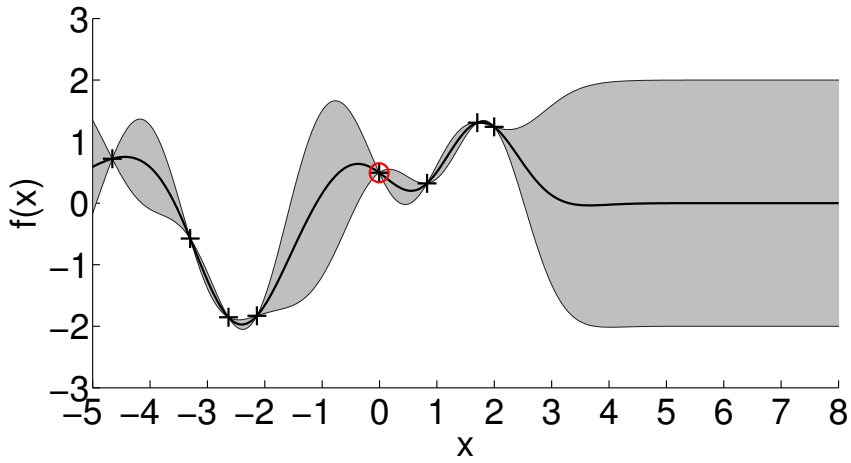
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



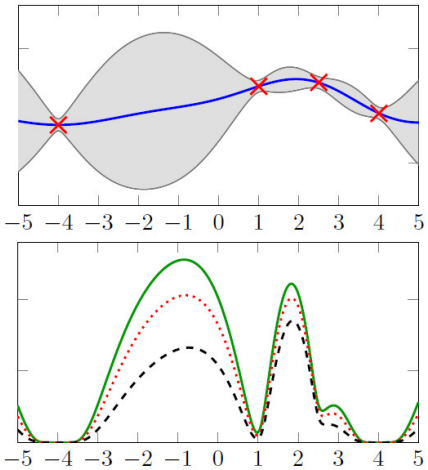
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

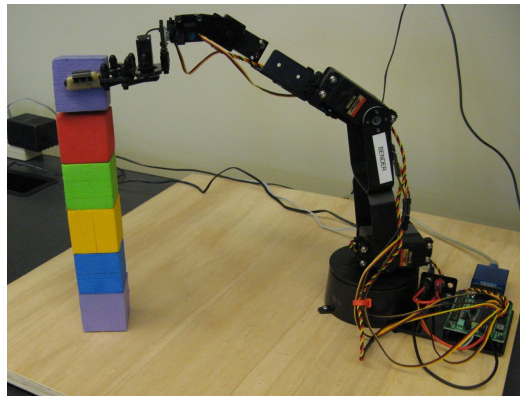
Quick Summary



- Gaussian processes are extremely flexible models
- Consistent confidence bounds!
- Based on simple manipulations of plain Gaussian distributions
- Few hyper-parameters ⏩ Generally easy to train
- First choice for black-box regression

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

Limitations of Gaussian Processes

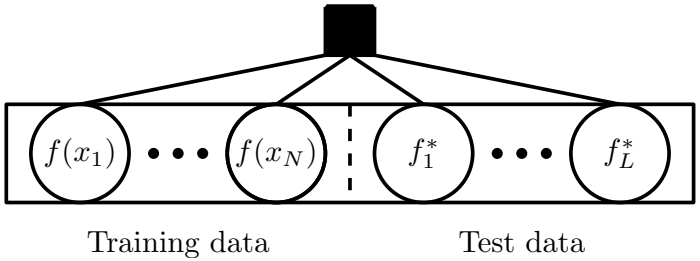
Computational and memory complexity

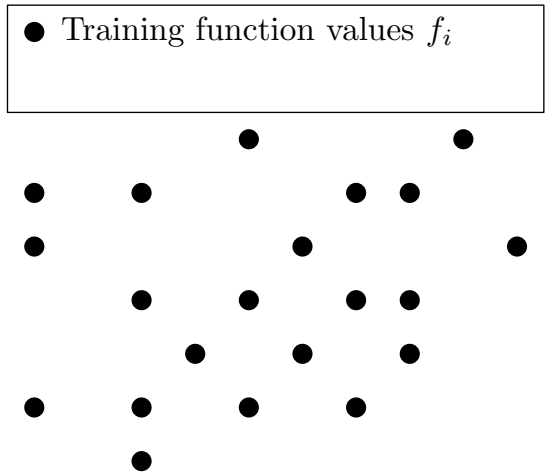
- Training scales in $\mathcal{O}(N^3)$
- Prediction (variances) scales in $\mathcal{O}(N^2)$
- Memory requirement: $\mathcal{O}(ND + N^2)$

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

Table of Contents

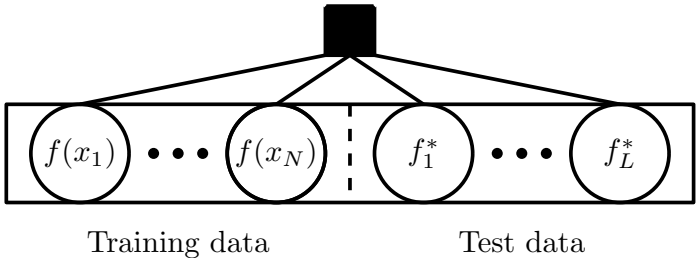
GP Factor Graph

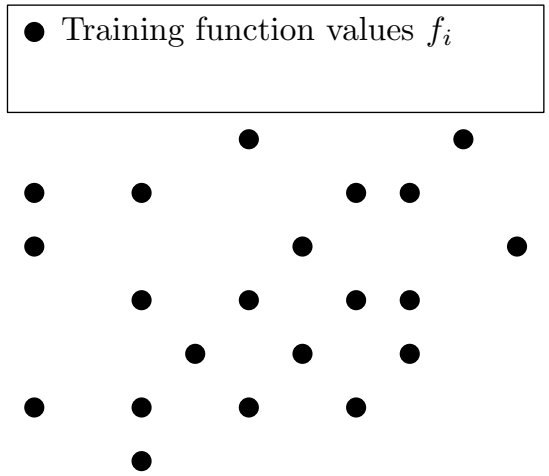




- Probabilistic graphical model (factor graph) of a GP
- All function values are jointly Gaussian distributed (e.g., training and test function values)

GP Factor Graph

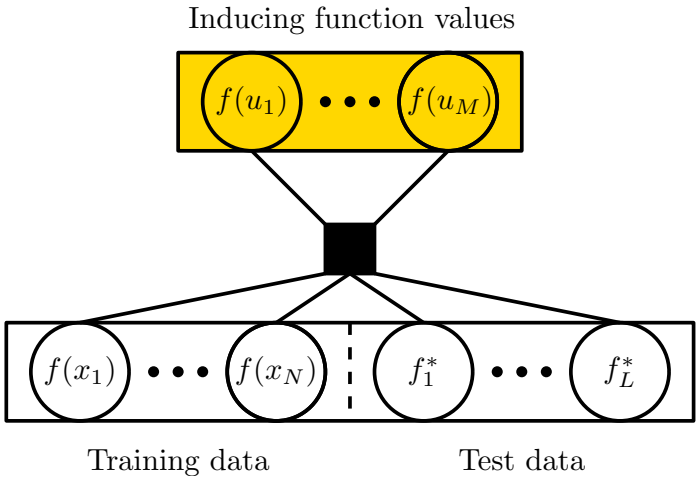


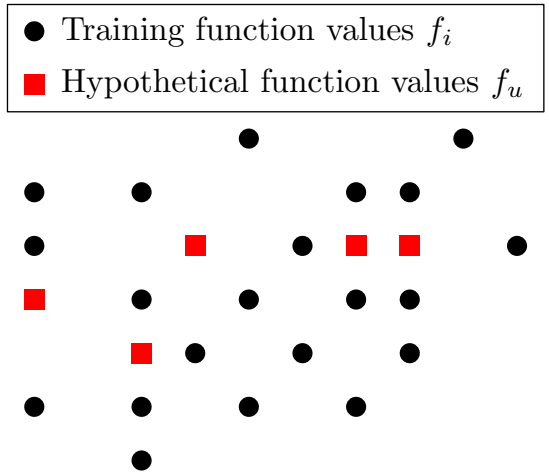


- Probabilistic graphical model (factor graph) of a GP
- All function values are jointly Gaussian distributed (e.g., training and test function values)
- GP prior

$$p(f, f_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{K}_{ff} & \boldsymbol{K}_{f*}\\ \boldsymbol{K}_{*f} & \boldsymbol{K}_{**}\end{bmatrix}\right)$$

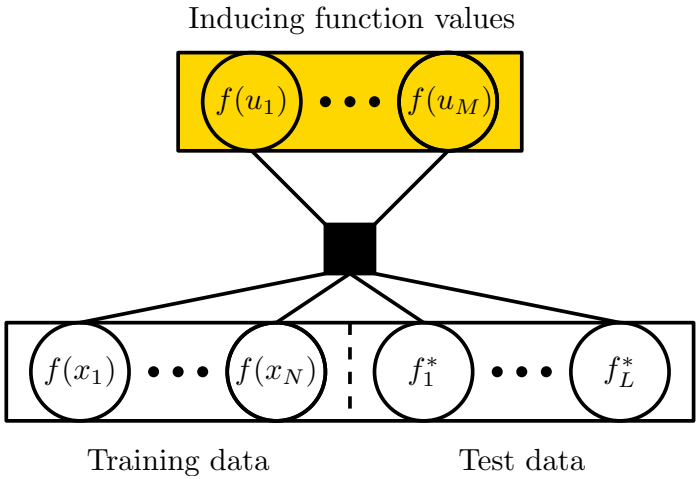
Inducing Variables

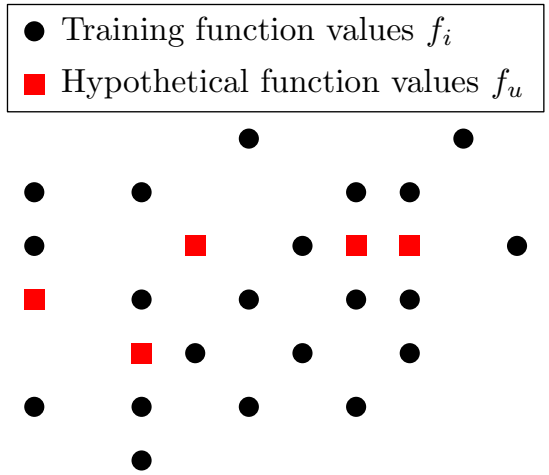




- Introduce inducing function values f_u
  - "Hypothetical" function values

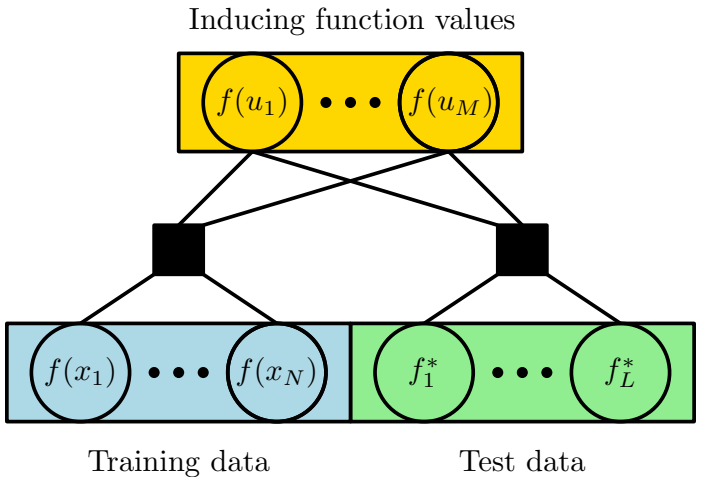
Inducing Variables





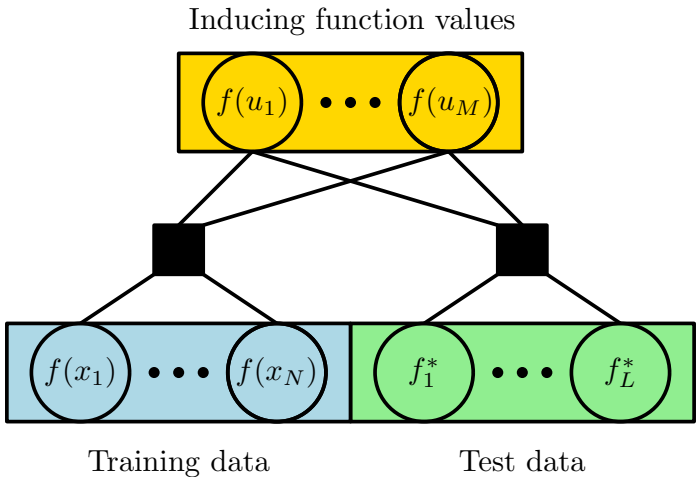
- Introduce inducing function values f_u
  - ⏩ "Hypothetical" function values
- All function values are still jointly Gaussian distributed (e.g., training, test and inducing function values)

Central Approximation Scheme



- Approximation: Training and test set are conditionally independent given the inducing function values: $f \perp\!\!\!\perp f_* | f_u$

Central Approximation Scheme

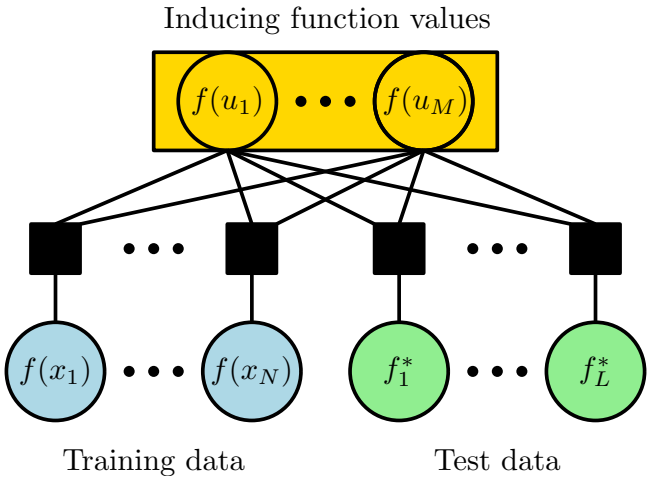


- Approximation: Training and test set are conditionally independent given the inducing function values: $f \perp\!\!\!\perp f_* | f_u$
- Then, the effective GP prior is

$$q(f, f_*) = \int p(f|f_u)p(f_*|f_u)p(f_u)df_u$$

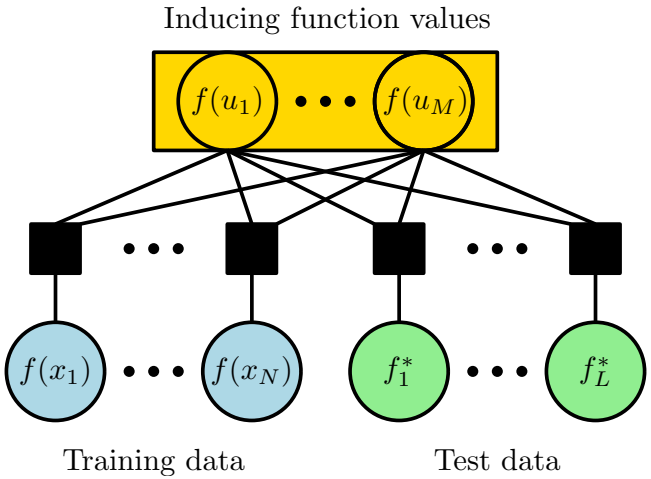
$$= \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{K}_{ff} & \boldsymbol{Q}_{f*}\\ \boldsymbol{Q}_{*f} & \boldsymbol{K}_{**}\end{bmatrix}\right), \qquad \boldsymbol{Q}_{ab} := \boldsymbol{K}_{af_u}\boldsymbol{K}_{f_uf_u}^{-1}\boldsymbol{K}_{f_ub}$$

FI(T)C Sparse Approximation



- Assume that training (and test sets) are fully independent given the inducing variables (Snelson & Ghahramani, 2006)

FI(T)C Sparse Approximation



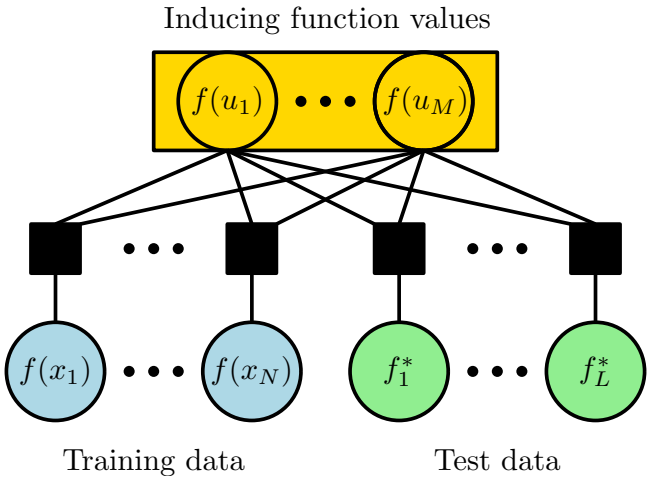
- Assume that training (and test sets) are fully independent given the inducing variables (Snelson & Ghahramani, 2006)

- Effective GP prior with this approximation

$$q(f, f_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix} \boldsymbol{Q}_{ff} - \mathrm{diag}(\boldsymbol{Q}_{ff} - \boldsymbol{K}_{ff}) & \boldsymbol{Q}_{f*} \\ \boldsymbol{Q}_{*f} & \boldsymbol{K}_{**} \end{bmatrix}\right)$$

- $\boldsymbol{Q}_{**} - \mathrm{diag}(\boldsymbol{Q}_{**} - \boldsymbol{K}_{**})$ can be used instead of $\boldsymbol{K}_{**}$ ⏩ FIC

FI(T)C Sparse Approximation



- Assume that training (and test sets) are fully independent given the inducing variables (Snelson & Ghahramani, 2006)

- Effective GP prior with this approximation

$$q(f, f_*) = \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix} \boldsymbol{Q}_{ff} - \mathrm{diag}(\boldsymbol{Q}_{ff} - \boldsymbol{K}_{ff}) & \boldsymbol{Q}_{f*} \\ \boldsymbol{Q}_{*f} & \boldsymbol{K}_{**} \end{bmatrix}\right)$$

- $\boldsymbol{Q}_{**} - \mathrm{diag}(\boldsymbol{Q}_{**} - \boldsymbol{K}_{**})$ can be used instead of $\boldsymbol{K}_{**}$ ⏩ FIC
- Training: $\mathcal{O}(NM^2)$, Prediction: $\mathcal{O}(M^2)$

Inducing Inputs

- FI(T)C sparse approximation relies on inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs

Inducing Inputs

- FI(T)C sparse approximation relies on inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs
- These **inputs are unknown** a priori ⏩ Find "optimal" ones

Inducing Inputs

- FI(T)C sparse approximation relies on inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs
- These **inputs are unknown** a priori ⏩ Find "optimal" ones
- Find them by maximizing the FI(T)C marginal likelihood with respect to the inducing inputs
  (and the standard hyper-parameters):

$$\boldsymbol{u}_{1:M}^* \in \arg\max_{\boldsymbol{u}_{1:M}} q_{\text{FITC}}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{u}_{1:M}, \boldsymbol{\theta})$$

Inducing Inputs

- FI(T)C sparse approximation relies on inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs
- These **inputs are unknown** a priori ⏩ Find "optimal" ones
- Find them by maximizing the FI(T)C marginal likelihood with respect to the inducing inputs
  (and the standard hyper-parameters):

$$\boldsymbol{u}_{1:M}^* \in \arg\max_{\boldsymbol{u}_{1:M}} q_{\text{FITC}}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{u}_{1:M}, \boldsymbol{\theta})$$

- Intuitively: The marginal likelihood is not only parameterized by the hyper-parameters $\boldsymbol{\theta}$, but also by the inducing inputs $\boldsymbol{u}_{1:M}$.

Inducing Inputs

- FI(T)C sparse approximation relies on inducing function values $f(\boldsymbol{u}_i)$, where $\boldsymbol{u}_i$ are the corresponding inputs
- These **inputs are unknown** a priori ⏩ Find "optimal" ones
- Find them by maximizing the FI(T)C marginal likelihood with respect to the inducing inputs
  (and the standard hyper-parameters):

$$\boldsymbol{u}_{1:M}^* \in \arg\max_{\boldsymbol{u}_{1:M}} q_{\text{FITC}}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{u}_{1:M}, \boldsymbol{\theta})$$

- Intuitively: The marginal likelihood is not only parameterized by the hyper-parameters $\boldsymbol{\theta}$, but also by the inducing inputs $\boldsymbol{u}_{1:M}$.
- End up with a high-dimensional non-convex optimization problem with MD additional parameters

FITC Example

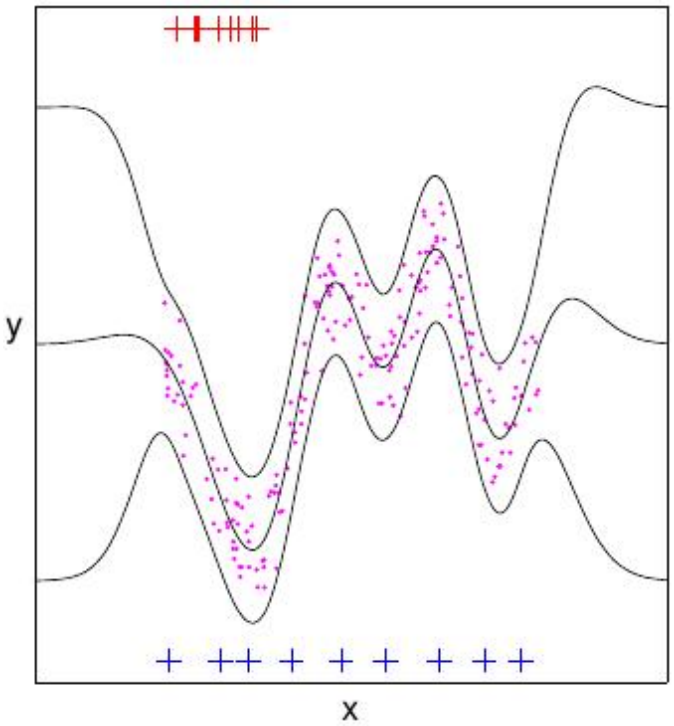


Figure from Ed Snelson

- Pink: Original data
- Red crosses: Initialization of inducing inputs
- Blue crosses: Location of inducing inputs after optimization

- Efficient compression of the original data set

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka et al., 2013)

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka et al., 2013)
- Computational complexity: $\mathcal{O}(M^3)$ or $O(NM^2)$ for training

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka et al., 2013)
- Computational complexity: $\mathcal{O}(M^3)$ or $O(NM^2)$ for training
- Practical limit $M \leqslant 10^4$. Often: $M \in \mathcal{O}(10^2)$ in the case of inducing variables

Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka et al., 2013)
- Computational complexity: $\mathcal{O}(M^3)$ or $O(NM^2)$ for training
- Practical limit $M \leqslant 10^4$. Often: $M \in \mathcal{O}(10^2)$ in the case of inducing variables
- If we set $M = N/100$, i.e., each inducing function value summarizes 100 real function values, our practical limit is $N \in \mathcal{O}(10^6)$

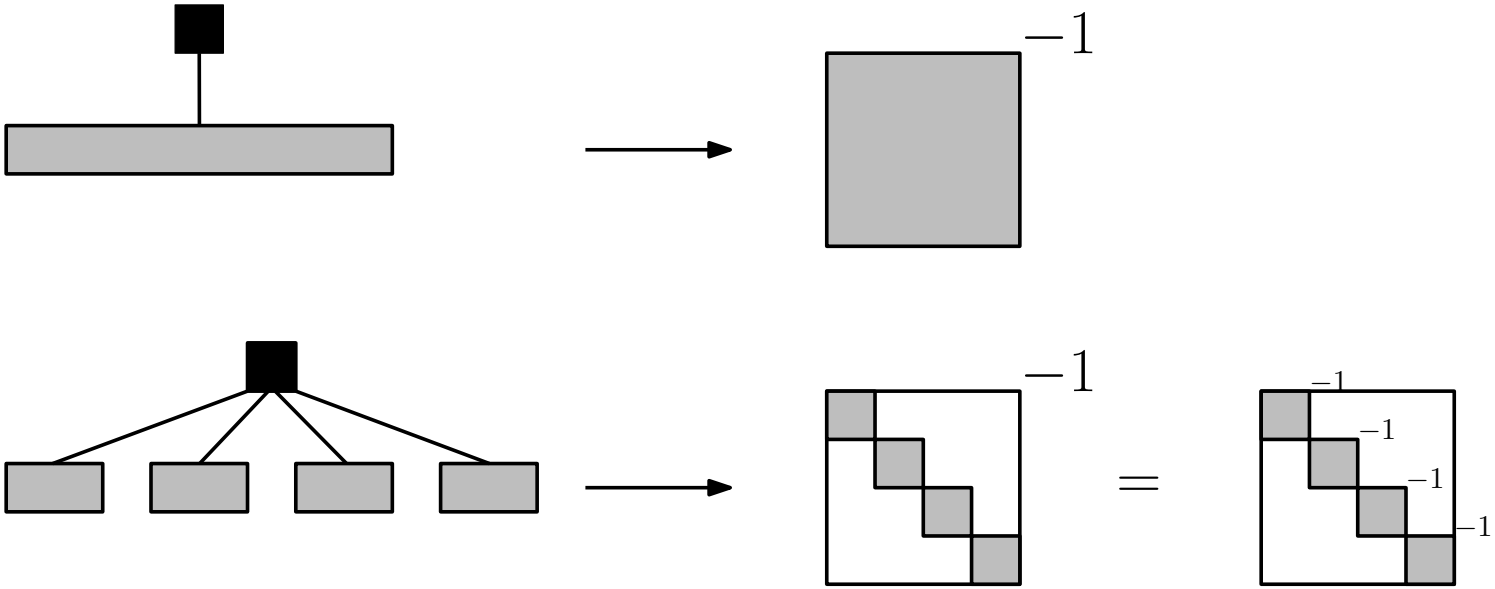
Summary Sparse Gaussian Processes

- Sparse approximations typically approximate a GP with N data points by a model with $M \ll N$ data points
- Selection of these M data points can be tricky and may involve non-trivial computations (e.g., optimizing inducing inputs)
- Simple (random) subset selection is fast and generally robust (Chalupka et al., 2013)
- Computational complexity: $\mathcal{O}(M^3)$ or $O(NM^2)$ for training
- Practical limit $M \leqslant 10^4$. Often: $M \in \mathcal{O}(10^2)$ in the case of inducing variables
- If we set $M = N/100$, i.e., each inducing function value summarizes 100 real function values, our practical limit is $N \in \mathcal{O}(10^6)$

⏩ Let's try something different

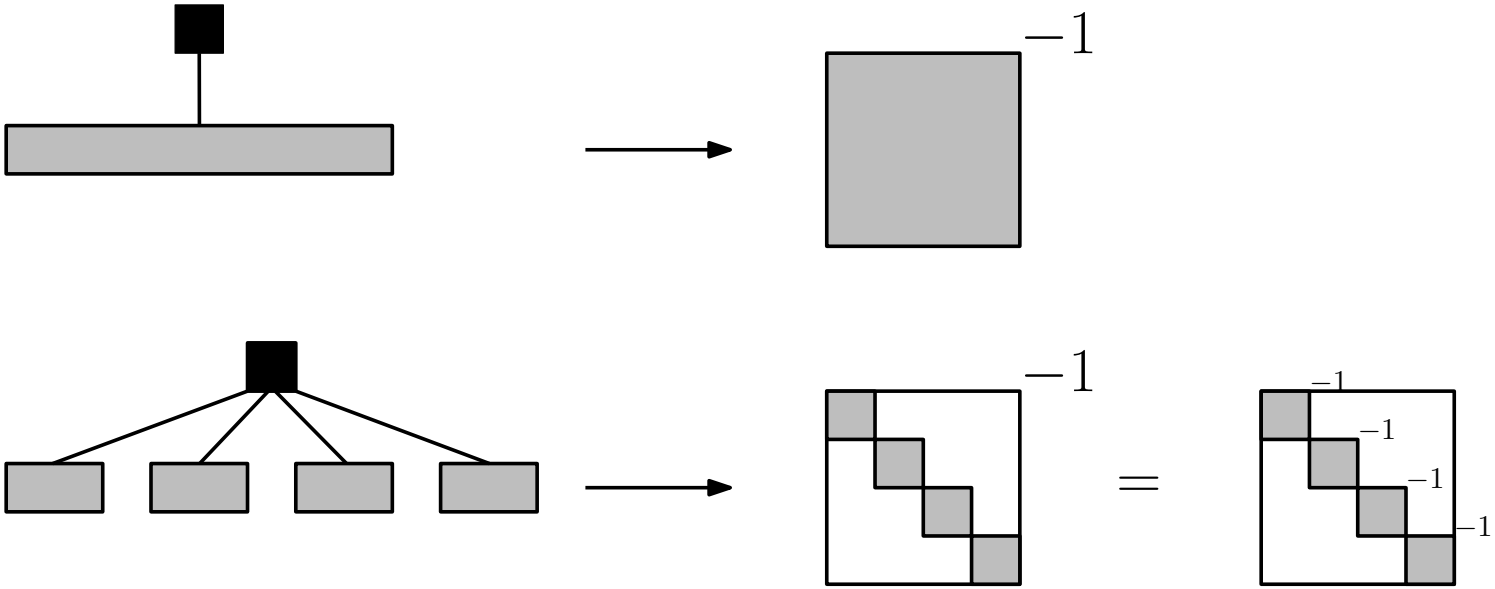
An Orthogonal Approximation: Distributed GPs

An Orthogonal Approximation: Distributed GPs



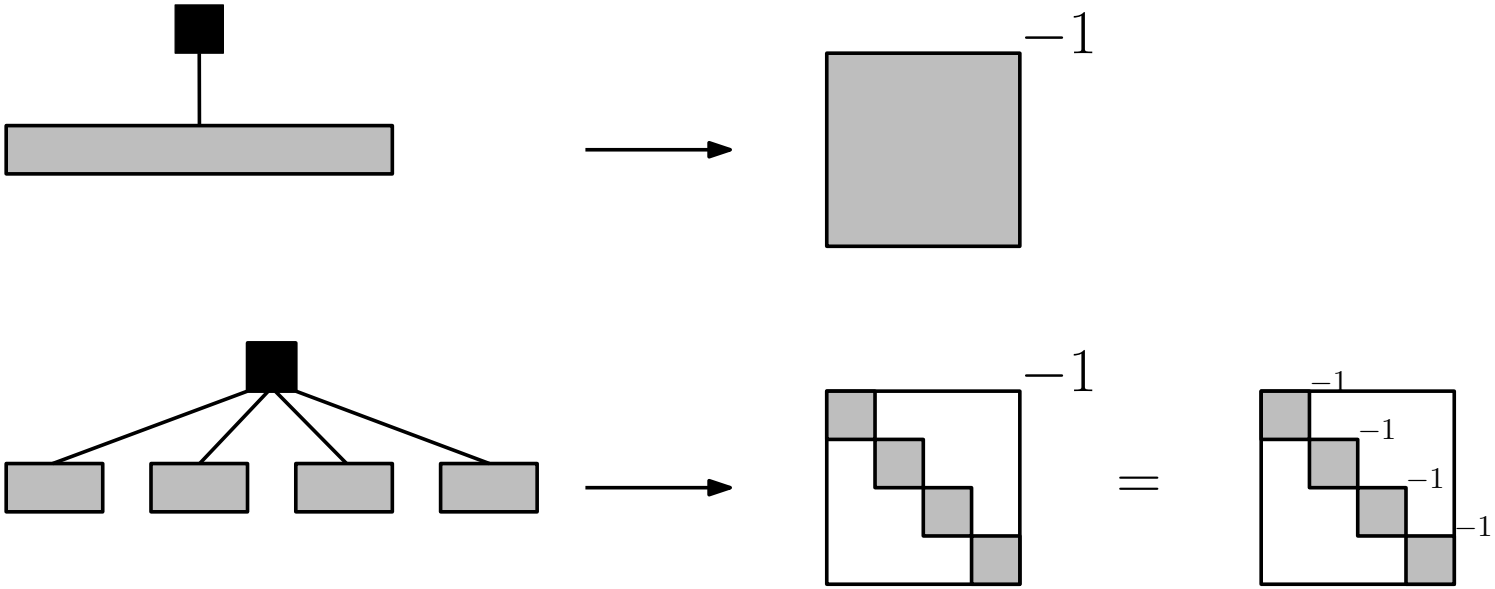
- Randomly split the full data set into M chunks

An Orthogonal Approximation: Distributed GPs



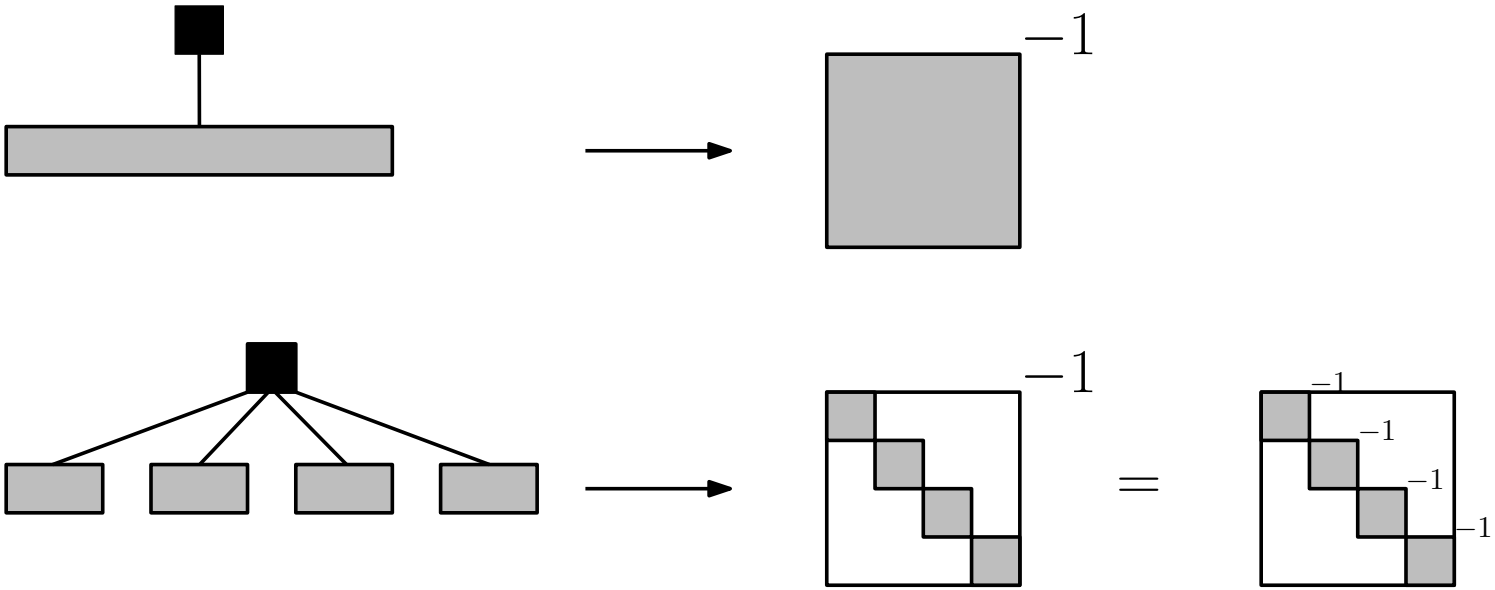
- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks

An Orthogonal Approximation: Distributed GPs



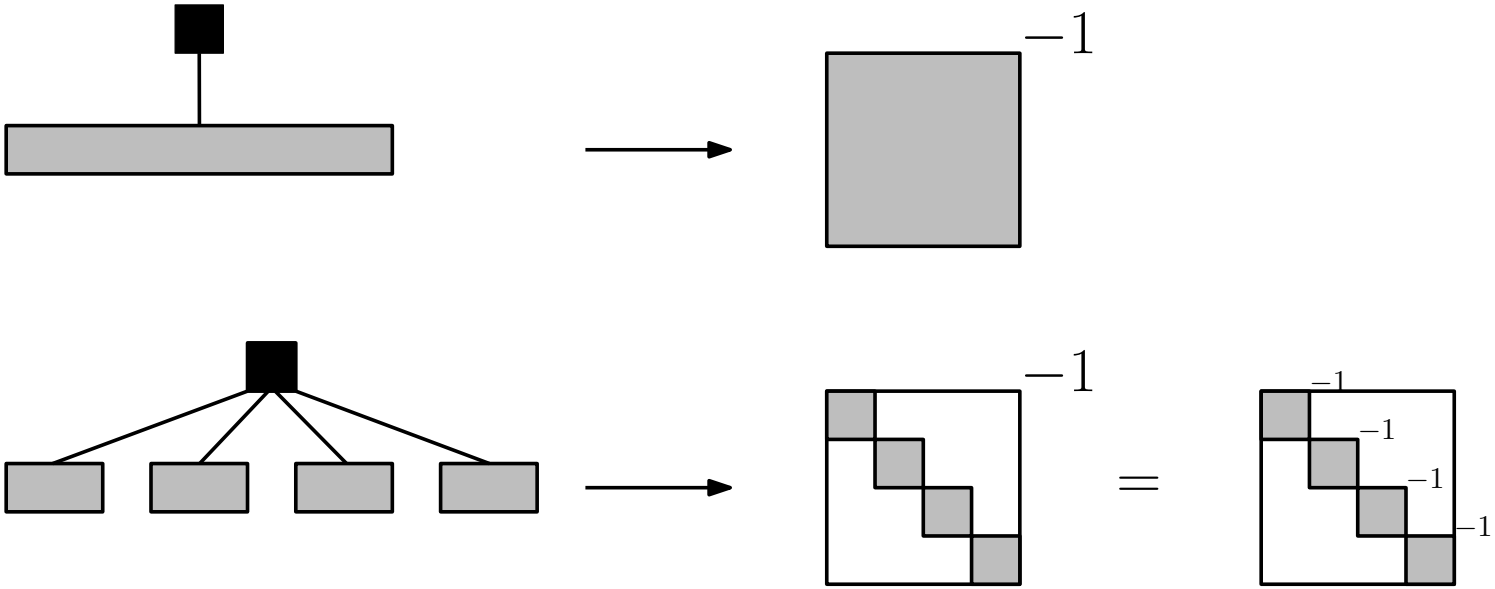
- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks
- Independent computations can be distributed

An Orthogonal Approximation: Distributed GPs



- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks
- Independent computations can be distributed
- Block-diagonal approximation of kernel matrix $\boldsymbol{K}$

An Orthogonal Approximation: Distributed GPs



- Randomly split the full data set into M chunks
- Place M independent GP models (experts) on these small chunks
- Independent computations can be distributed
- Block-diagonal approximation of kernel matrix $\boldsymbol{K}$
- Combine independent computations to an overall result

Training the Distributed GP

- Split data set of size N into M chunks of size P
- Independence of experts ⏩ Factorization of marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) \approx \sum_{k=1}^{M} \log p_k(\boldsymbol{y}^{(k)}|\boldsymbol{X}^{(k)},\boldsymbol{\theta})$$

Training the Distributed GP

- Split data set of size N into M chunks of size P
- Independence of experts ⏩ Factorization of marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) \approx \sum_{k=1}^{M} \log p_k(\boldsymbol{y}^{(k)}|\boldsymbol{X}^{(k)},\boldsymbol{\theta})$$

- Distributed optimization and training straightforward

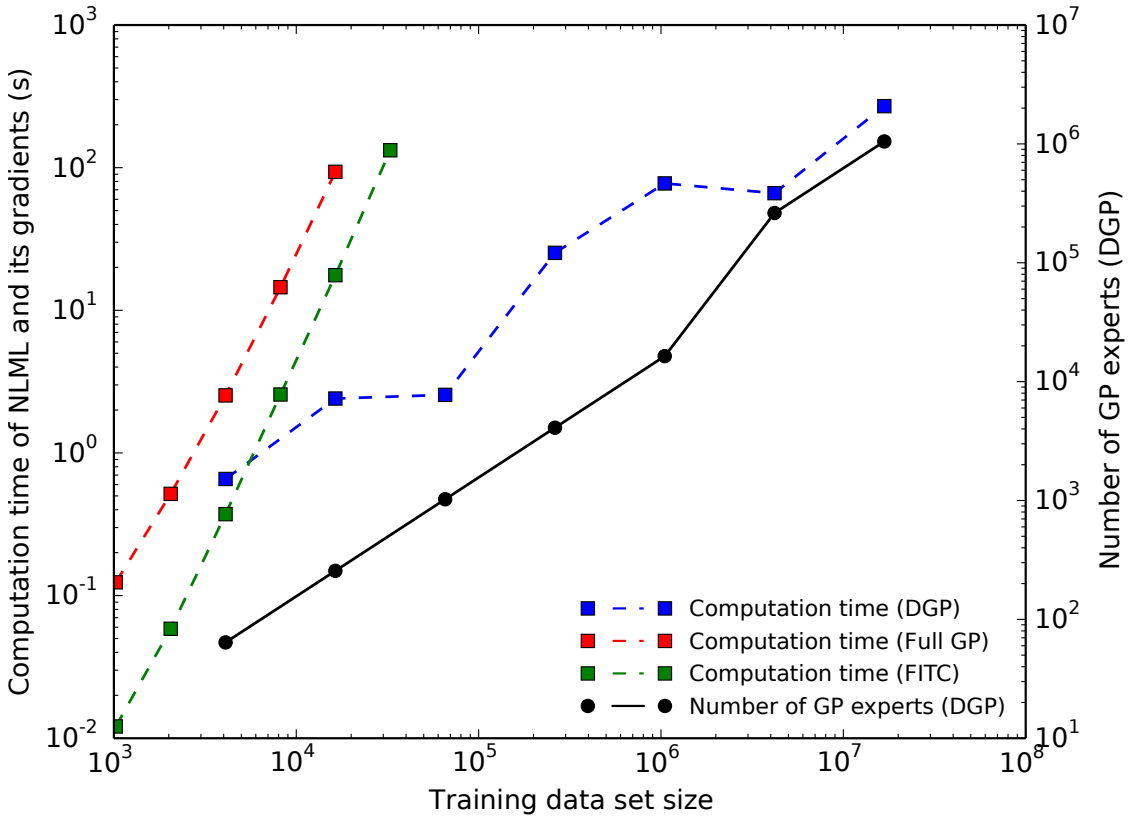
Training the Distributed GP

- Split data set of size N into M chunks of size P
- Independence of experts ⏩ Factorization of marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) \approx \sum_{k=1}^{M} \log p_k(\boldsymbol{y}^{(k)}|\boldsymbol{X}^{(k)},\boldsymbol{\theta})$$

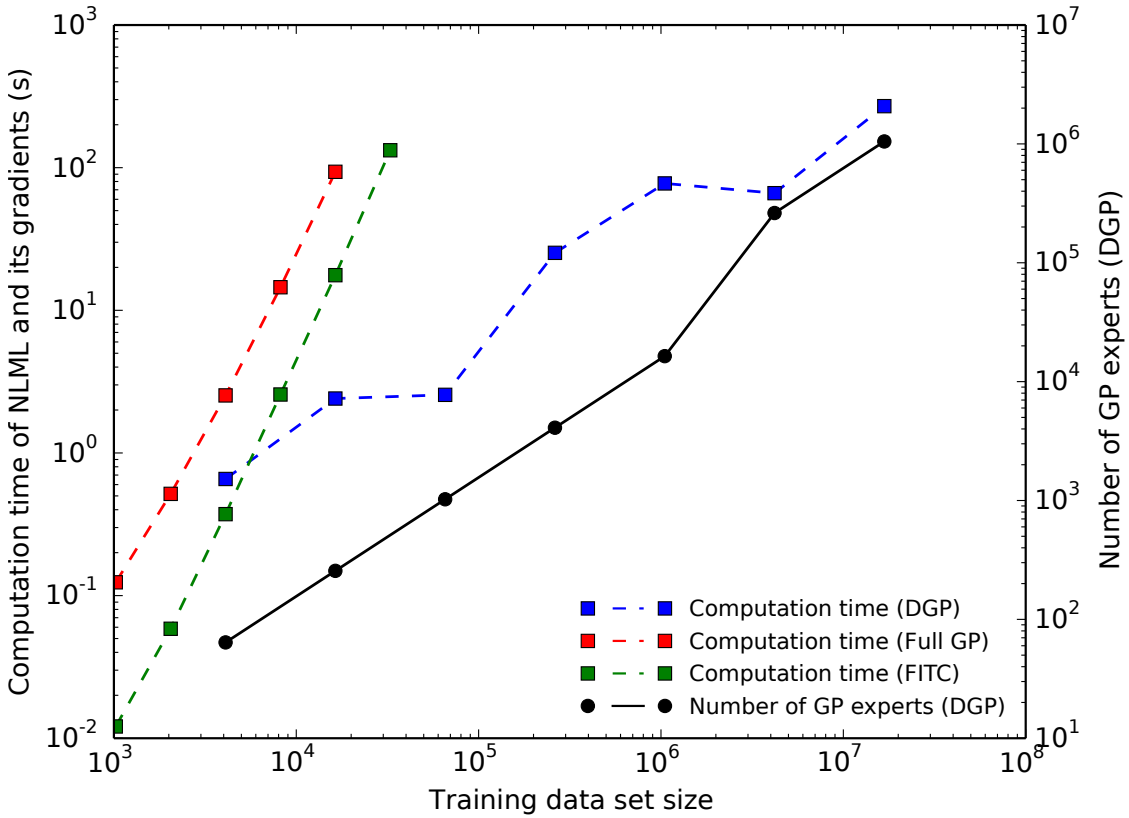
- Distributed optimization and training straightforward
- Computational complexity: $\mathcal{O}(MP^3)$ [instead of $\mathcal{O}(N^3)$]
  But distributed over many machines
- Memory footprint: $\mathcal{O}(MP^2 + ND)$ [instead of $\mathcal{O}(N^2 + ND)$]

Empirical Training Time



- NLML is proportional to training time

Empirical Training Time



- NLML is proportional to training time
- Full GP (16K training points) ≈ sparse GP (50K training points) ≈ distributed GP (16M training points)

⏩ Push practical limit by order(s) of magnitude

Practical Training Times

- Training* with $N = 10^6, D = 1$ on a laptop: ≈ 30 min
- Training* with $N = 5 \times 10^6, D = 8$ on a workstation: ≈ 4 hours

Practical Training Times

- Training* with $N = 10^6, D = 1$ on a laptop: ≈ 30 min
- Training* with $N = 5 \times 10^6, D = 8$ on a workstation: ≈ 4 hours

*: Maximize the marginal likelihood, stop when converged**

: Convergence often after 30–80 line searches*

Practical Training Times

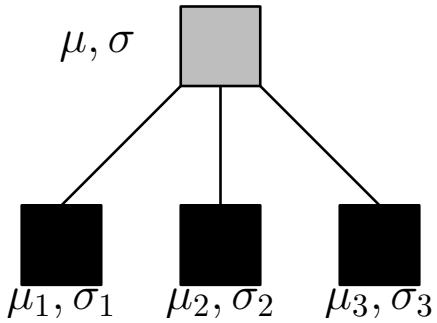
- Training* with $N = 10^6, D = 1$ on a laptop: $\approx$ 30 min
- Training* with $N = 5 \times 10^6, D = 8$ on a workstation: $\approx$ 4 hours

*: Maximize the marginal likelihood, stop when converged**

: Convergence often after 30–80 line searches*

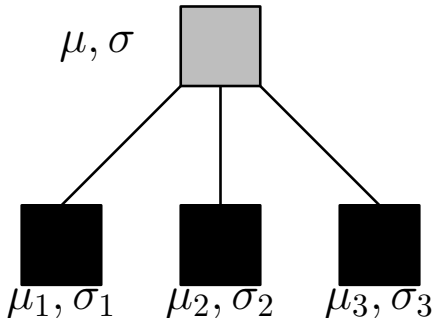
***: Line search $\approx$ 2–3 evaluations of marginal likelihood and its gradient (usually $\mathcal{O}(N^3)$)

Predictions with the Distributed GP



- Prediction of each GP expert is Gaussian $\mathcal{N}\left(\mu_i, \sigma_i^2\right)$
- How to combine them to an overall prediction $\mathcal{N}\left(\mu, \sigma^2\right)$?

Predictions with the Distributed GP



- Prediction of each GP expert is Gaussian $\mathcal{N}\left(\mu_i, \sigma_i^2\right)$
- How to combine them to an overall prediction $\mathcal{N}\left(\mu, \sigma^2\right)$?

⏩ Product-of-GP-experts

- PoE (product of experts) ⏩ (Ng & Deisenroth, 2014)
- gPoE (generalized product of experts) ⏩ (Cao & Fleet, 2014)
- BCM (Bayesian Committee Machine) ⏩ (Tresp, 2000)
- rBCM (robust BCM) ⏩ (Deisenroth & Ng, 2015)

Objectives

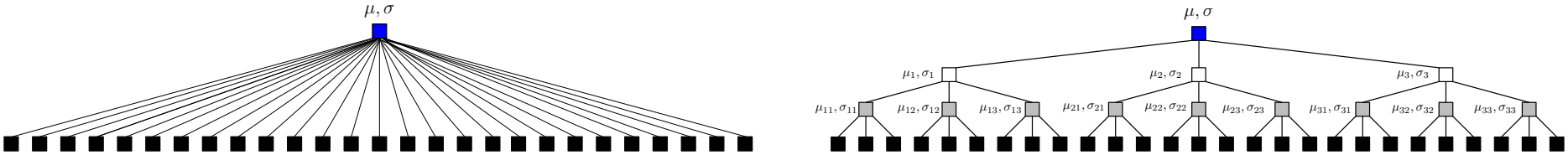


Figure: Two computational graphs

- Scale to large data sets ✓

Objectives

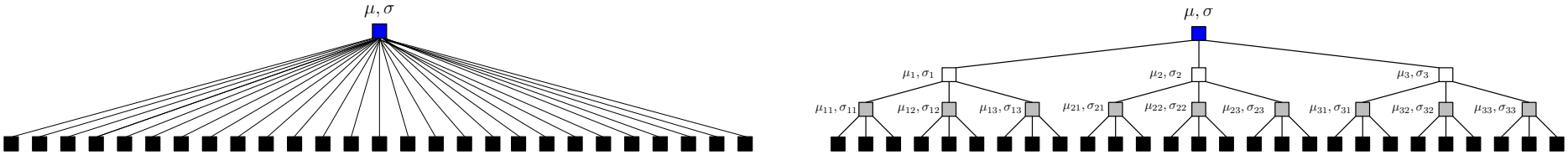


Figure: Two computational graphs

- Scale to large data sets ✓
- Good approximation of full GP ("ground truth")

Objectives

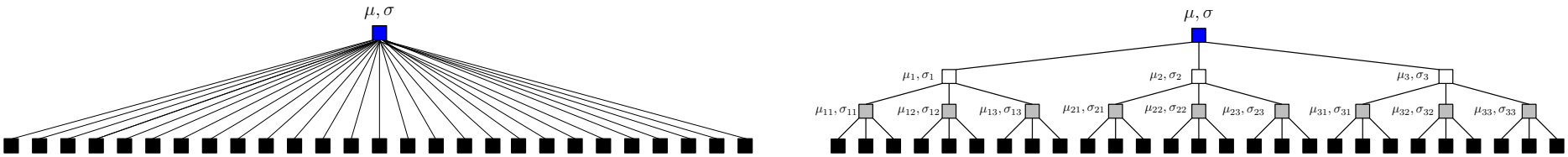


Figure: Two computational graphs

- Scale to large data sets ✓
- Good approximation of full GP ("ground truth")
- Predictions independent of computational graph

  ⏩ Runs on heterogeneous computing infrastructures (laptop, cluster, ...)

Objectives

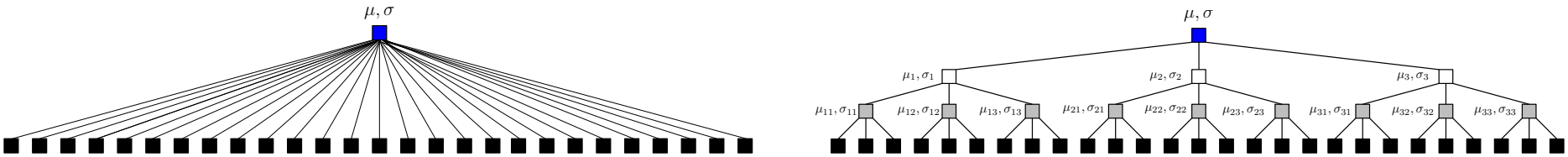
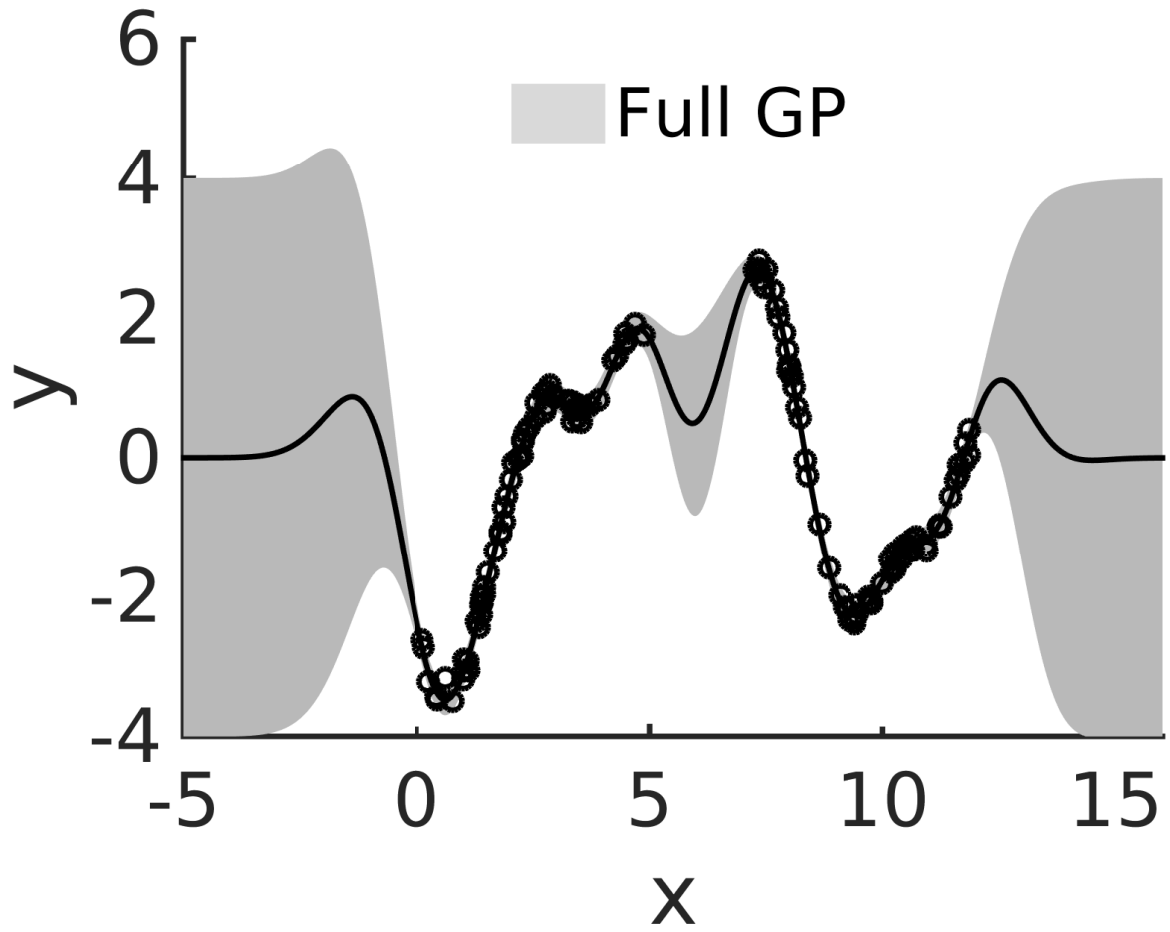


Figure: Two computational graphs

- Scale to large data sets ✓
- Good approximation of full GP ("ground truth")
- Predictions independent of computational graph

  ⏩ Runs on heterogeneous computing infrastructures (laptop, cluster, ...)
- Reasonable predictive variances

Running Example



- Investigate various product-of-experts models
  Same training procedure, but different mechanisms for predictions

Product of GP Experts

- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}),$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \,|\, \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

Product of GP Experts

- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}),$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \,|\, \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

- Predictive precision (inverse variance) and mean:

$$(\sigma_*^{\text{poe}})^{-2} = \sum\nolimits_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

$$\mu_*^{\text{poe}} = (\sigma_*^{\text{poe}})^2 \sum\nolimits_k \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

Product of GP Experts

- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}),$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \,|\, \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

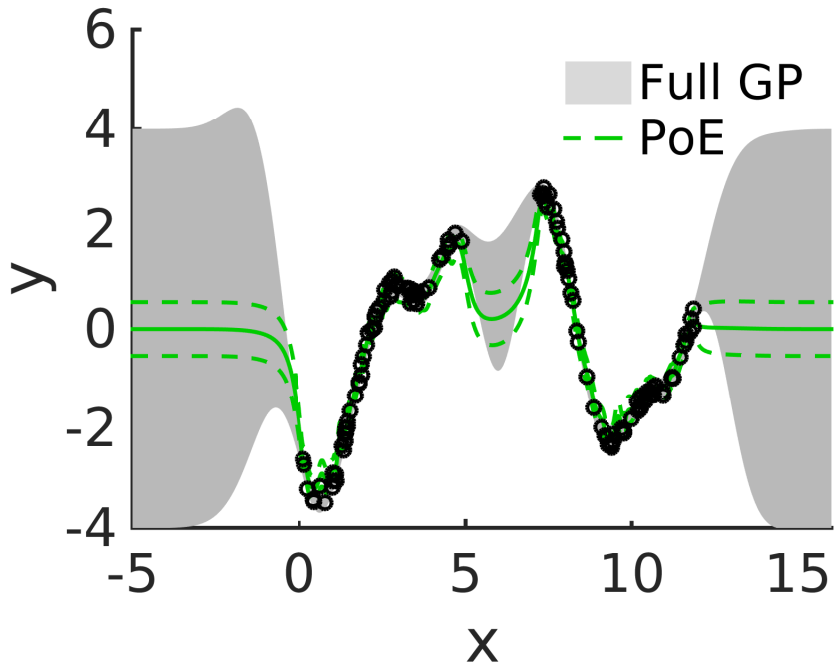
- Predictive precision (inverse variance) and mean:

$$(\sigma_*^{\text{poe}})^{-2} = \sum\nolimits_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

$$\mu_*^{\text{poe}} = (\sigma_*^{\text{poe}})^2 \sum\nolimits_k \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

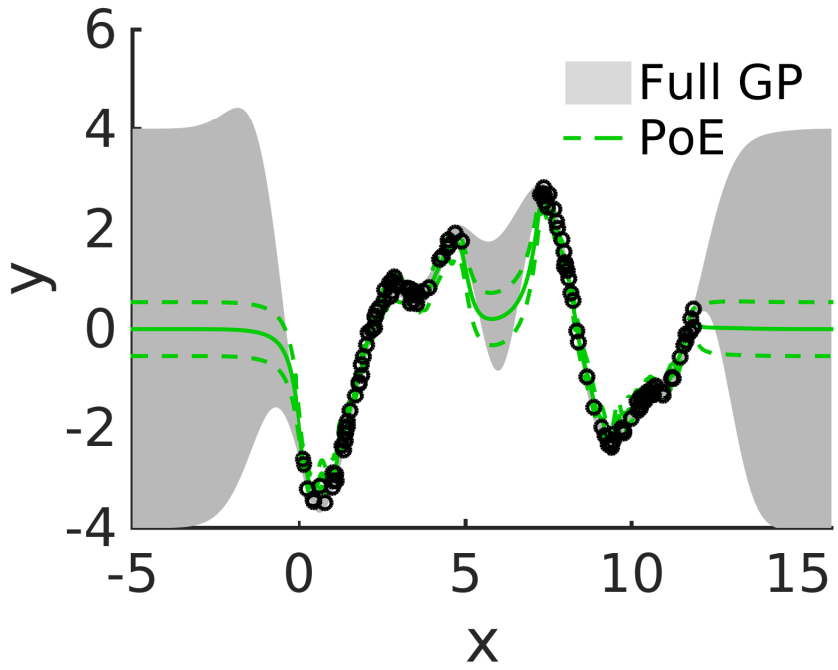
- Independent of the computational graph ✓

Product of GP Experts



- Unreasonable variances for $M > 1$:

Product of GP Experts



- Unreasonable variances for $M > 1$:

$$(\sigma_*^{\text{poe}})^{-2} = \sum_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

- The more experts the more certain the prediction, even if every expert itself is very uncertain ✗ ⏩ Cannot fall back to the prior

Generalized Product of GP Experts

- Weight the responsiblity of each expert in PoE with β_k

Generalized Product of GP Experts

- Weight the responsiblity of each expert in PoE with β_k
- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \,|\, \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

Generalized Product of GP Experts

- Weight the responsiblity of each expert in PoE with β_k
- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \mid \mu_k(\boldsymbol{x}_*), \sigma_k^2(\boldsymbol{x}_*)\right)$$

- Predictive precision and mean:

$$(\sigma_*^{\text{gpoe}})^{-2} = \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

$$\mu_*^{\text{gpoe}} = (\sigma_*^{\text{gpoe}})^2 \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)\, \mu_k(\boldsymbol{x}_*)$$

Generalized Product of GP Experts

- Weight the responsiblity of each expert in PoE with β_k
- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \mid \mu_k(\boldsymbol{x}_*),\, \sigma_k^2(\boldsymbol{x}_*)\right)$$

- Predictive precision and mean:

$$(\sigma_*^{\text{gpoe}})^{-2} = \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

$$\mu_*^{\text{gpoe}} = (\sigma_*^{\text{gpoe}})^2 \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)\, \mu_k(\boldsymbol{x}_*)$$

- With $\sum_k \beta_k = 1$, the model can fall back to the prior ✓
  "Log-opinion pool" model (Heskes, 1998)

Generalized Product of GP Experts

- Weight the responsiblity of each expert in PoE with β_k
- Prediction model (independent predictors):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})$$

$$p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)}) = \mathcal{N}\left(f_* \mid \mu_k(\boldsymbol{x}_*), \sigma_k^2(\boldsymbol{x}_*)\right)$$

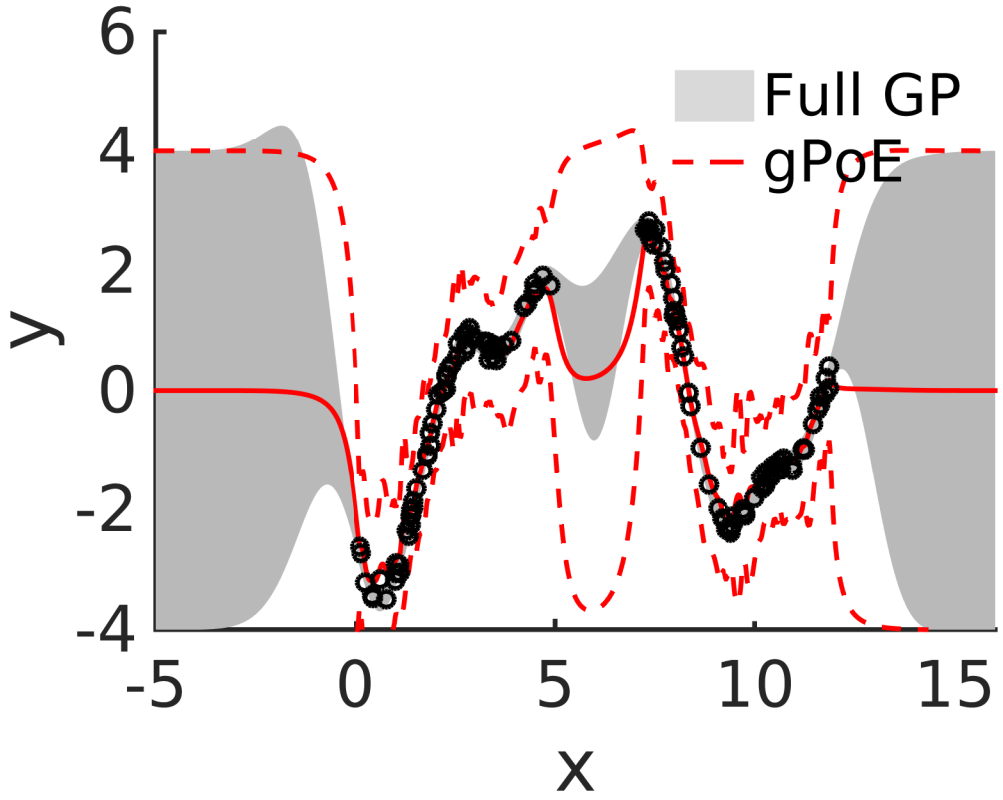
- Predictive precision and mean:

$$(\sigma_*^{\text{gpoe}})^{-2} = \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)$$

$$\mu_*^{\text{gpoe}} = (\sigma_*^{\text{gpoe}})^2 \sum\nolimits_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)\, \mu_k(\boldsymbol{x}_*)$$

- With $\sum_k \beta_k = 1$, the model can fall back to the prior ✓
  "Log-opinion pool" model (Heskes, 1998)
- Independent of computational graph for $\beta_k = 1/M$ ✓

Generalized Product of GP Experts



- Same mean as PoE
- Model no longer overconfident and falls back to prior ✓
- Very conservative variances ✗

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)
- Prediction model ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)
- Prediction model ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

- Predictive precision and mean:

$$(\sigma_*^{\text{bcm}})^{-2} = \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*) - (M-1)\sigma_{**}^{-2}$$

$$\mu_*^{\text{bcm}} = (\sigma_*^{\text{bcm}})^2 \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)
- Prediction model ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

- Predictive precision and mean:

$$(\sigma_*^{\text{bcm}})^{-2} = \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*) - (M-1)\sigma_{**}^{-2}$$

$$\mu_*^{\text{bcm}} = (\sigma_*^{\text{bcm}})^2 \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

- Product of GP experts, divided by $M-1$ times the prior

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)
- Prediction model ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

- Predictive precision and mean:

$$(\sigma_*^{\text{bcm}})^{-2} = \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*) - (M-1)\sigma_{**}^{-2}$$

$$\mu_*^{\text{bcm}} = (\sigma_*^{\text{bcm}})^2 \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

- Product of GP experts, divided by $M-1$ times the prior
- Guaranteed to fall back to the prior outside data regime ✓

Bayesian Committee Machine

- Apply Bayes' theorem when combining predictions (and not only for computing predictions)
- Prediction model ($\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}$$

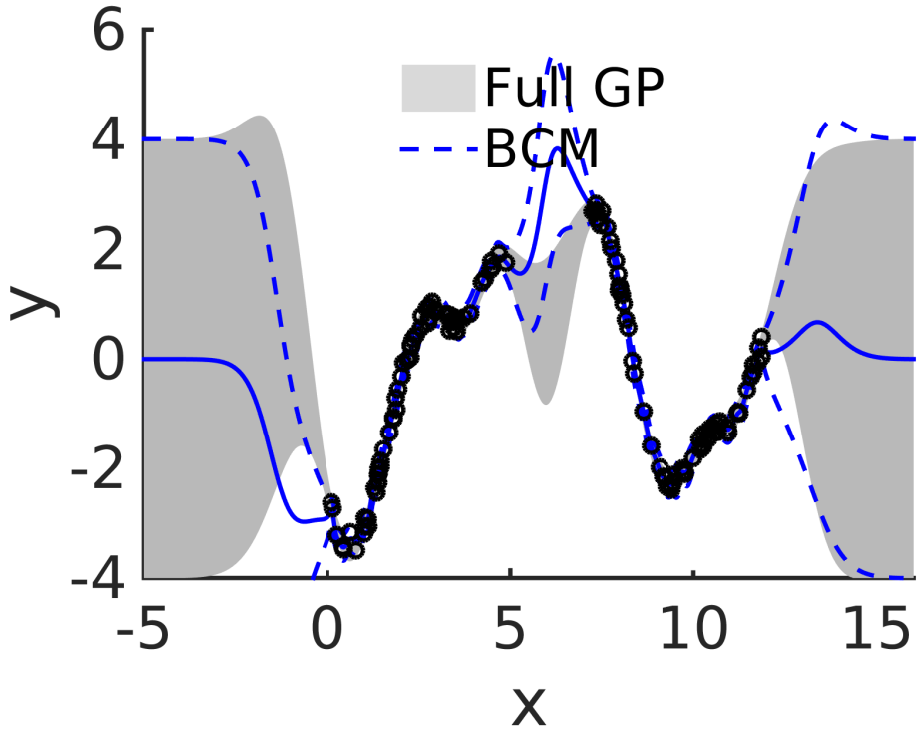
- Predictive precision and mean:

$$(\sigma_*^{\text{bcm}})^{-2} = \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*) - (M-1)\sigma_{**}^{-2}$$

$$\mu_*^{\text{bcm}} = (\sigma_*^{\text{bcm}})^2 \sum_{k=1}^{M} \sigma_k^{-2}(\boldsymbol{x}_*)\mu_k(\boldsymbol{x}_*)$$

- Product of GP experts, divided by $M-1$ times the prior
- Guaranteed to fall back to the prior outside data regime ✓
- Independent of computational graph ✓

Bayesian Committee Machine



- Variance estimates are about right ✓
- When leaving the data regime, the BCM can produce junk ✗

  ⏩ **Robustify**

Robust Bayesian Committee Machine

- Merge gPoE (weighting of experts) with the BCM (Bayes' theorem when combining predictions)

Robust Bayesian Committee Machine

- Merge gPoE (weighting of experts) with the BCM (Bayes' theorem when combining predictions)
- Prediction model (conditional independence $\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{\sum_k \beta_k - 1}(f_*)}$$

Robust Bayesian Committee Machine

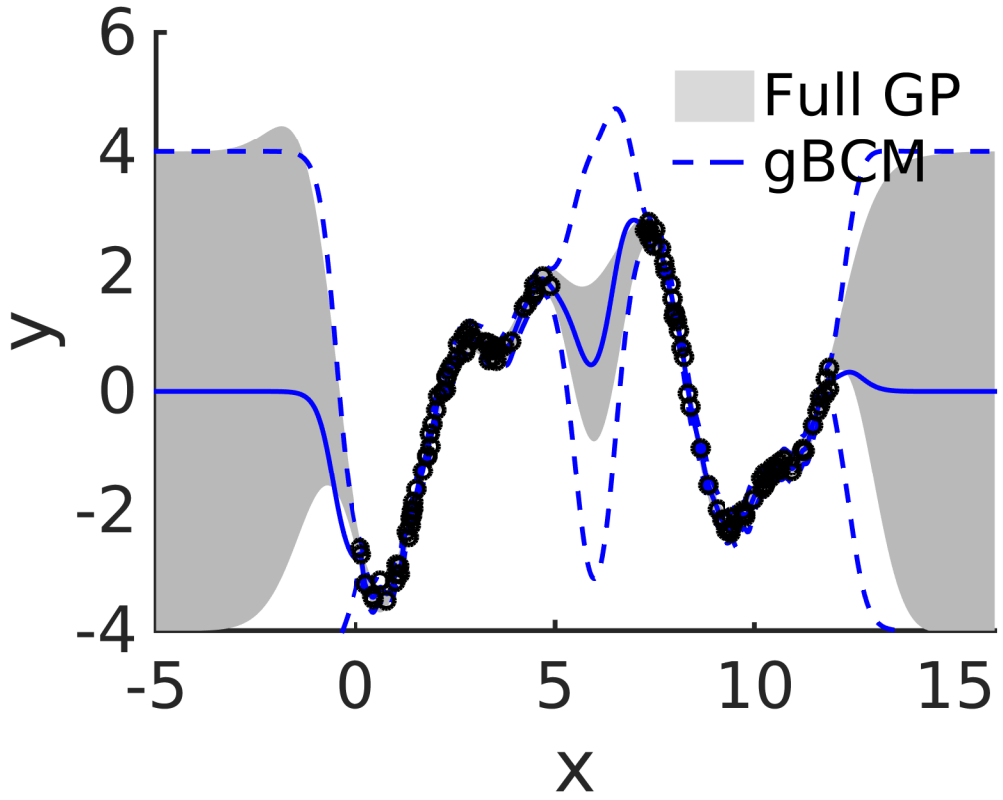
- Merge gPoE (weighting of experts) with the BCM (Bayes' theorem when combining predictions)
- Prediction model (conditional independence $\mathcal{D}^{(j)} \perp\!\!\!\perp \mathcal{D}^{(k)} | f_*$):

$$p(f_*|\boldsymbol{x}_*, \mathcal{D}) = \frac{\prod_{k=1}^M p_k^{\beta_k}(f_*|\boldsymbol{x}_*, \mathcal{D}^{(k)})}{p^{\sum_k \beta_k - 1}(f_*)}$$

- Predictive precision and mean:

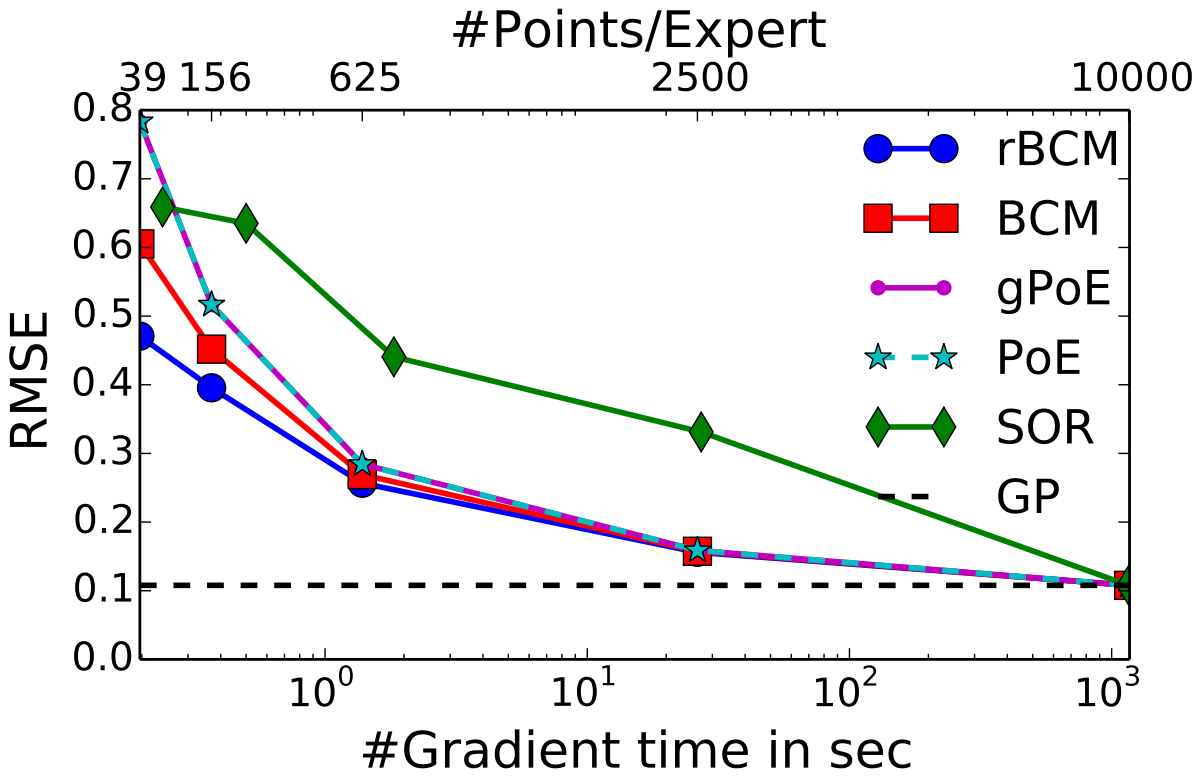
$$(\sigma_*^{\text{rbcm}})^{-2} = \sum_{k=1}^M \beta_k \sigma_k^{-2}(\boldsymbol{x}_*) + (1 - \sum_{k=1}^M \beta_k)\sigma_{**}^{-2},$$
$$\mu_*^{\text{rbcm}} = (\sigma_*^{\text{rbcm}})^2 \sum_k \beta_k \sigma_k^{-2}(\boldsymbol{x}_*)\, \mu_k(\boldsymbol{x}_*)$$

Robust Bayesian Committee Machine



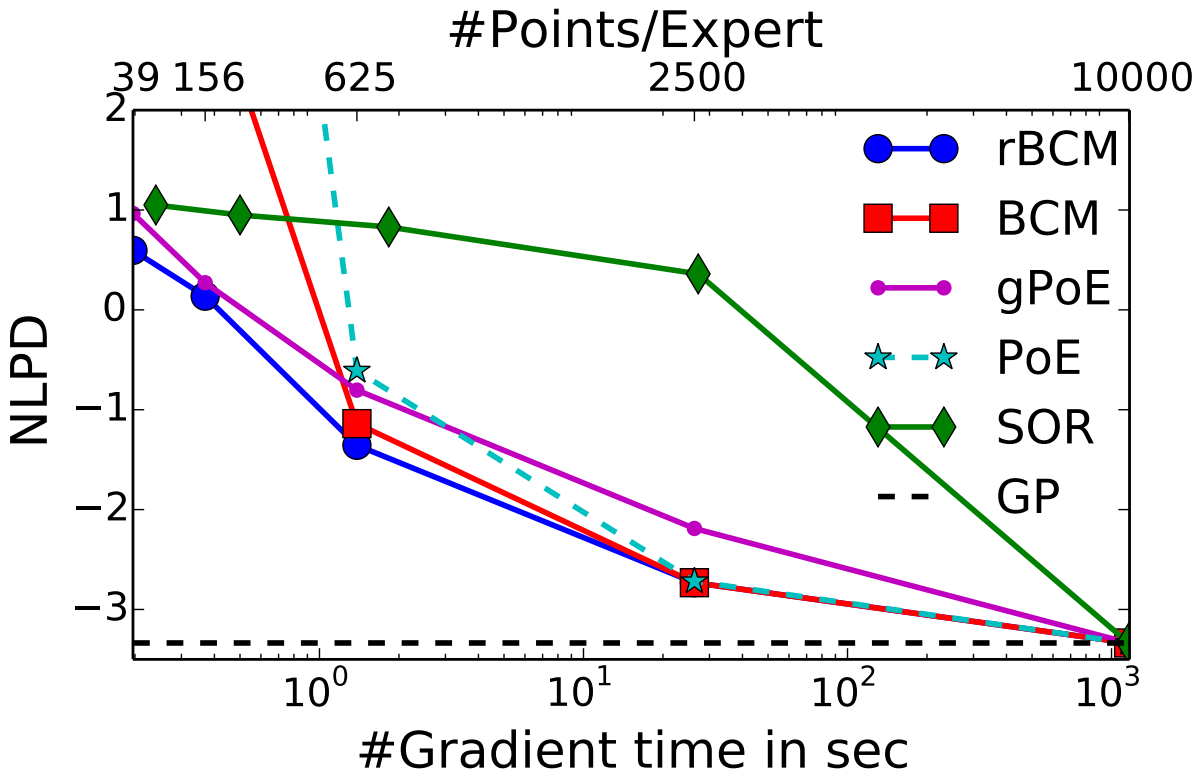
- Does not break down in case of weak experts ⏩ Robustified ✓
- Robust version of BCM ⏩ Reasonable predictions ✓
- Independent of computational graph (for all choices of β_k) ✓

Empirical Approximation Error



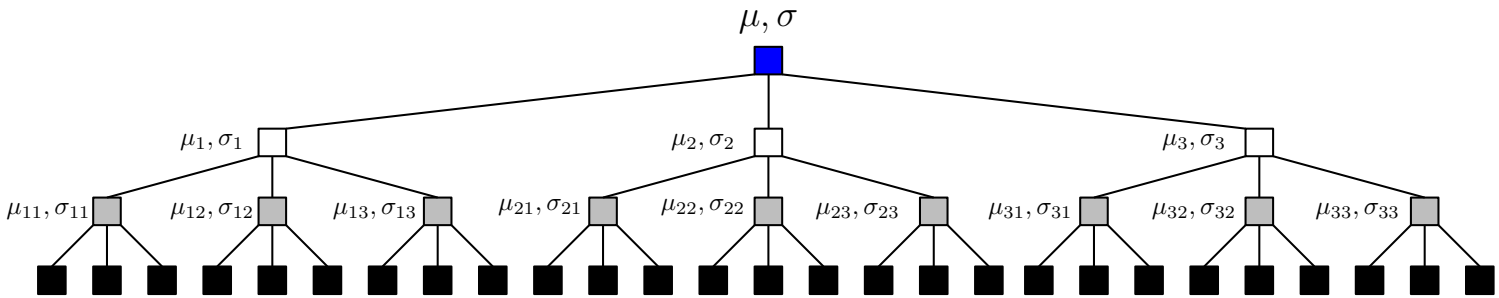
- Simulated robot arm data (10K training, 10K test)
- Hyper-parameters of ground-truth full GP
- RMSE as a function of the training time
- Sparse GP (SOR) performs worse than any distributed GP
- rBCM performs best with "weak" GP experts

Empirical Approximation Error (2)



- NLPD as a function of the training time ⏩ Mean and variance
- BCM and PoE are not robust for weak experts
- gPoE suffers from too conservative variances
- rBCM consistently outperforms other methods

Summary: Distributed Gaussian Processes



- Scale Gaussian processes to large data (beyond 10^6)
- Model conceptually straightforward and easy to train
- Key: Distributed computation
- Currently tested with $N \in \mathcal{O}(10^7)$
- Scales to arbitrarily large data sets (with enough computing power)

`m.deisenroth@imperial.ac.uk`

Thank you for your attention

Expressiveness of the Standard Model

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

Expressiveness of the Standard Model

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

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

Expressiveness of the Standard Model

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 θ_1^2

⏩ GP with mean 0 and Gaussian covariance function

Two-level inference

- level-1 inference:

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

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

- level-2 inference

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

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] 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`, October 2014.

[3] K. Chalupka, C. K. I. Williams, and I. Murray. A Framework for Evaluating Approximate Methods for Gaussian Process Regression. *Journal of Machine Learning Research*, 14:333–350, February 2013.

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

[5] M. P. Deisenroth. *Efficient Reinforcement Learning using Gaussian Processes*, volume 9 of *Karlsruhe Series on Intelligent Sensor-Actuator-Systems*. KIT Scientific Publishing, November 2010. ISBN 978-3-86644-569-7.

[6] 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.

[7] M. P. Deisenroth and J. Ng. Distributed Gaussian Processes. `http://arxiv.org/abs/1502.02843`, February 2015.

[8] Y. Gal, M. van der Wilk, and C. E. Rasmussen. Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models. In *Advances in Neural Information Processing Systems*. 2014.

[9] 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.

[10] T. Heskes. Selecting Weighting Factors in Logarithmic Opinion Pools. In *Advances in Neural Information Processing Systems*, pages 266–272. Morgan Kaufman, 1998.

[11] J. Kern. *Bayesian Process-Convolution Approaches to Specifying Spatial Dependence Structure*. PhD thesis, Institue of Statistics and Decision Sciences, Duke University, 2000.

[12] 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.

References II

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

[14] 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.

[15] J. Ng and M. P. Deisenroth. Hierarchical Mixture-of-Experts Model for Large-Scale Gaussian Process Regression. `http://arxiv.org/abs/1412.3078`, December 2014.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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