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 $y_i = f(x_i) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2_\varepsilon)$, find a distribution over functions $p(f)$ that explains the data

Probalistic regression problem
Problem Setting

Objective

For a set of observations $y_i = f(x_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2_\epsilon)$, 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

Introduction
  Gaussian Distribution

Gaussian Processes
  Definition and Derivation
  Inference
  Covariance Functions
  Model Selection
  Limitations

Scaling Gaussian Processes to Large Data Sets
  Sparse Gaussian Processes
  Distributed Gaussian Processes
Table of Contents

Introduction

Gaussian Distribution

Gaussian Processes
Definition and Derivation
Inference
Covariance Functions
Model Selection
Limitations

Scaling Gaussian Processes to Large Data Sets
Sparse Gaussian Processes
Distributed Gaussian Processes
Key Concepts in Probability Theory

Two fundamental rules:

\[ p(x) = \int p(x, y) \, dy \]  \hspace{1cm} \text{Sum rule/Marginalization property}

\[ p(x, y) = p(y|x)p(x) \]  \hspace{1cm} \text{Product rule}
Key Concepts in Probability Theory

Two fundamental rules:

\[ p(x) = \int p(x, y) \, dy \]  \hspace{1cm} \text{Sum rule/Marginalization property}

\[ p(x, y) = p(y|x) p(x) \]  \hspace{1cm} \text{Product rule}

Bayes’ Theorem (Probabilistic Inverse)

\[ p(x|y) = \frac{p(y|x) \, p(x)}{p(y)} , \quad x : \text{hypothesis}, \quad y : \text{measurement} \]
Key Concepts in Probability Theory

Two fundamental rules:

\[ p(x) = \int p(x, y) \, dy \]  
\[ p(x, y) = p(y|x) \, p(x) \]  
**Sum rule/Marginalization property**  
**Product rule**

Bayes’ Theorem (Probabilistic Inverse)

\[ p(x|y) = \frac{p(y|x) \, p(x)}{p(y)} \]  
\[ x : \text{hypothesis}, \quad y : \text{measurement} \]

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

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

- **Mean vector \( \mu \)** ➤ Average of the data
- **Covariance matrix \( \Sigma \)** ➤ Spread of the data
The Gaussian Distribution

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

- **Mean vector** \( \mu \) ➤ Average of the data
- **Covariance matrix** \( \Sigma \) ➤ Spread of the data
The Gaussian Distribution

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

- Mean vector \( \mu \) \( \rightarrow \) Average of the data
- Covariance matrix \( \Sigma \) \( \rightarrow \) Spread of the data
Sampling from a Multivariate Gaussian

Objective

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

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

**Objective**

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

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

Exploit that affine transformations $y = Ax + b$ of Gaussians remain Gaussian

- Mean: $\mathbb{E}_x[Ax + b] = A\mathbb{E}_x[x] + b$
- Covariance: $\mathbb{V}_x[Ax + b] = A\mathbb{V}_x[x]A^\top$
Sampling from a Multivariate Gaussian

**Objective**

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

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

Exploit that affine transformations \( y = Ax + b \) of Gaussians remain Gaussian

- Mean: \( \mathbb{E}_x[Ax + b] = A\mathbb{E}_x[x] + b \)
- Covariance: \( \mathbb{V}_x[Ax + b] = AV_x[x]A^\top \)

1. Find conditions for \( A, b \) to match the mean of \( y \)
2. Find conditions for \( A, b \) to match the covariance of \( y \)
Sampling from a Multivariate Gaussian (2)

**Objective**

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

\[
x = \text{randn}(D, 1); \quad \text{Sample } x \sim \mathcal{N}(0, I)
\]

\[
y = \text{chol}(\Sigma)'x + \mu; \quad \text{Scale } x
\]

Here \( \text{chol}(\Sigma) \) is the Cholesky factor \( L \), such that \( L^\top L = \Sigma \)
Sampling from a Multivariate Gaussian (2)

Objective

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

\[
x = \text{randn}(D,1); \quad \text{Sample } x \sim \mathcal{N}(0, I)
y = \text{chol}(\Sigma)'*x + \mu; \quad \text{Scale } x
\]

Here \( \text{chol}(\Sigma) \) is the Cholesky factor \( L \), such that \( L^\top L = \Sigma \)

Therefore, the mean and covariance of \( y \) are

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

\[
\text{Cov}[y] = \mathbb{E}[(y - \bar{y})(y - \bar{y})^\top] = \mathbb{E}[L^\top xx^\top L] = L^\top \mathbb{E}[xx^\top] L = L^\top L = \Sigma
\]
Conditional

\[ p(x, y) = \mathcal{N} \left( \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right) \]
Conditional

$$p(x, y) = \mathcal{N} \left( \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right)$$
\[
p(x, y) = \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}\right)
\]

\[
p(x|y) = \mathcal{N}(\mu_{x|y}, \Sigma_{x|y})
\]

\[
\mu_{x|y} = \mu_x + \Sigma_{xy} \Sigma_{yy}^{-1} (y - \mu_y)
\]

\[
\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx}
\]

Conditional \( p(x|y) \) is also Gaussian

\[\text{Computationally convenient}\]
The marginal of a joint Gaussian distribution is Gaussian

Intuitively: Ignore (integrate out) everything you are not interested in
Table of Contents

Introduction

Gaussian Distribution

Gaussian Processes
  Definition and Derivation
  Inference
  Covariance Functions
  Model Selection
  Limitations

Scaling Gaussian Processes to Large Data Sets
  Sparse Gaussian Processes
  Distributed Gaussian Processes
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

$$x_t = Ax_t + w, \quad w \sim N(0, Q)$$

$x_1, x_2, \ldots$ 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
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

\[
x_{t+1} = Ax_t + w, \quad w \sim \mathcal{N}(0, \Sigma)
\]

\( x_1, x_2, \ldots \) 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
Consider the joint Gaussian distribution \( p(x, \tilde{x}) \), where \( x \in \mathbb{R}^D \) and \( \tilde{x} \in \mathbb{R}^k, k \to \infty \).
The Gaussian in the Limit

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

Then

$$p(x, \tilde{x}) = \mathcal{N} \left( \begin{bmatrix} \mu_x \\ \mu_{\tilde{x}} \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{x\tilde{x}} \\ \Sigma_{\tilde{x}x} & \Sigma_{\tilde{x}\tilde{x}} \end{bmatrix} \right)$$

where $\Sigma_{\tilde{x}\tilde{x}} \in \mathbb{R}^{k \times k}$ and $\Sigma_{x\tilde{x}} \in \mathbb{R}^{D \times k}$, $k \to \infty$. 

The Gaussian in the Limit

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

Then

$$p(x, \tilde{x}) = \mathcal{N} \left( \begin{bmatrix} \mu_x \\ \mu_{\tilde{x}} \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{x\tilde{x}} \\ \Sigma_{\tilde{x}x} & \Sigma_{\tilde{x}\tilde{x}} \end{bmatrix} \right)$$

where $\Sigma_{\tilde{x}\tilde{x}} \in \mathbb{R}^{k \times k}$ and $\Sigma_{x\tilde{x}} \in \mathbb{R}^{D \times k}, k \to \infty$

However, the marginal remains finite

$$p(x) = \int p(x, \tilde{x})d\tilde{x} = \mathcal{N} \left( \mu_x, \Sigma_{xx} \right)$$

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 \( x_{\text{train}}, x_{\text{test}} \)
Marginal and Conditional in the Limit

- In practice, we consider *finite training and test data* $x_{\text{train}}, x_{\text{test}}$
- Then, $x = \{x_{\text{train}}, x_{\text{test}}, x_{\text{other}}\}$

($x_{\text{other}}$ plays the role of $\tilde{x}$ from previous slide)
Marginal and Conditional in the Limit

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

$$p(x) = \mathcal{N} \left( \begin{bmatrix} \mu_{\text{train}} \\ \mu_{\text{test}} \\ \mu_{\text{other}} \end{bmatrix}, \begin{bmatrix} \Sigma_{\text{train}} & \Sigma_{\text{train,test}} & \Sigma_{\text{train,other}} \\ \Sigma_{\text{test,train}} & \Sigma_{\text{test}} & \Sigma_{\text{test,other}} \\ \Sigma_{\text{other,train}} & \Sigma_{\text{other,test}} & \Sigma_{\text{other}} \end{bmatrix} \right)$$

Gaussian Processes

Marc Deisenroth @MLSS, 14 April 2015
In practice, we consider finite training and test data $x_{\text{train}}, x_{\text{test}}$.

Then, $x = \{x_{\text{train}}, x_{\text{test}}, x_{\text{other}}\}$

($x_{\text{other}}$ plays the role of $\tilde{x}$ from previous slide)

$$p(x) = \mathcal{N}\left(\begin{bmatrix} \mu_{\text{train}} \\ \mu_{\text{test}} \\ \mu_{\text{other}} \end{bmatrix}, \begin{bmatrix} \Sigma_{\text{train}} & \Sigma_{\text{train, test}} \\ \Sigma_{\text{test, train}} & \Sigma_{\text{test}} \\ \Sigma_{\text{other, train}} & \Sigma_{\text{other, test}} \end{bmatrix}\right)$$

$$p(x_{\text{train}}, x_{\text{test}}) = \int p(x_{\text{train}}, x_{\text{test}}, x_{\text{other}}) d x_{\text{other}}$$
Marginal and Conditional in the Limit

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

\[
p(x) = \mathcal{N}\left( \begin{bmatrix} \mu_{\text{train}} \\ \mu_{\text{test}} \\ \mu_{\text{other}} \end{bmatrix}, \begin{bmatrix} \Sigma_{\text{train}} & \Sigma_{\text{train,test}} & \Sigma_{\text{train,other}} \\ \Sigma_{\text{test,train}} & \Sigma_{\text{test}} & \Sigma_{\text{test,other}} \\ \Sigma_{\text{other,train}} & \Sigma_{\text{other,test}} & \Sigma_{\text{other}} \end{bmatrix} \right)\]

\[
p(x_{\text{train}}, x_{\text{test}}) = \int p(x_{\text{train}}, x_{\text{test}}, x_{\text{other}}) d x_{\text{other}}
\]

\[
p(x_{\text{test}} | x_{\text{train}}) = \mathcal{N}(\mu_*, \Sigma_*)
\]

\[
\mu_* = \mu_{\text{test}} + \Sigma_{\text{test,train}} \Sigma_{\text{train}}^{-1} (x_{\text{train}} - \mu_{\text{train}})
\]

\[
\Sigma_* = \Sigma_{\text{test}} - \Sigma_{\text{test,train}} \Sigma_{\text{train}}^{-1} \Sigma_{\text{train,test}}
\]
Back to Regression: Distribution over Functions

- We are not really interested in a distribution on $x$, but rather in a distribution over function values $f(x)$
Back to Regression: Distribution over Functions

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

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

where $\Sigma_{ff} \in \mathbb{R}^{k \times k}$ and $\Sigma_{f\tilde{f}} \in \mathbb{R}^{N \times k}$, $k \to \infty$. 
Back to Regression: Distribution over Functions

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

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

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

- Again, the marginal remains finite

$$p(f) = \int p(f, \tilde{f})d\tilde{f} = \mathcal{N}(\mu_f, \Sigma_{ff})$$
Marginal and Conditional over Functions

Define $f_* := f_{\text{test}}$, $f := f_{\text{train}}$. 
Marginal and Conditional over Functions

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

- **Marginal**

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

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

- **Marginal**

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

- **Conditional** (predictive distribution)

$$p(f_\star | f) = \mathcal{N}(m, S)$$

$$m = \mu_\star + \Sigma_{*f} \Sigma_{ff}^{-1} (f - \mu)$$

$$S = \Sigma_{**} - \Sigma_{*f} \Sigma_{ff}^{-1} \Sigma_{f*}$$

We need to compute (cross-)covariances between unknown function values
Marginal and Conditional over Functions

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

- **Marginal**

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

- **Conditional** (predictive distribution)

  $$p(f_\ast | f) = \mathcal{N} (m, S)$$

  $$m = \mu_\ast + \Sigma_{f\ast} \Sigma_{ff}^{-1} (f - \mu)$$

  $$S = \Sigma_{\ast\ast} - \Sigma_{f\ast} \Sigma_{ff}^{-1} \Sigma_{f\ast}$$

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

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

\[
\begin{align*}
p(f_* | f) &= \mathcal{N} (m, S) \\
m &= \mu_* + \Sigma_{*f} \Sigma_{ff}^{-1} (f - \mu) \\
S &= \Sigma_{**} - \Sigma_{*f} \Sigma_{ff}^{-1} \Sigma_{f*}
\end{align*}
\]
Kernelization

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

- A kernel function \( k \) is symmetric and positive definite.

\[ p(f_*|f) = \mathcal{N}(m, S) \]

\[ m = \mu_* + \Sigma_* f \Sigma_{ff}^{-1} (f - \mu) \]

\[ S = \Sigma_{**} - \Sigma_* f \Sigma_{ff}^{-1} \Sigma_{f*} \]
Kernelization

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

\[ p(f_\ast | f) = \mathcal{N}(m, S) \]

\[ m = \mu_\ast + \Sigma_{\ast f} \Sigma_{ff}^{-1} (f - \mu) \]

\[ S = \Sigma_{\ast\ast} - \Sigma_{\ast f} \Sigma_{ff}^{-1} \Sigma_{f\ast} \]

- A kernel function \( k \) is symmetric and positive definite.
- Kernel computes covariances between unknown function values \( f(x_i) \) and \( f(x_j) \) by just looking at the corresponding inputs \( x_i, x_j \)

\[ \Sigma_{ff}^{(i,j)} = \text{Cov}[f(x_i), f(x_j)] = k(x_i, x_j) \]
Kernelization

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

\[ p(f_* | f) = \mathcal{N}(m, S) \]

\[ m = \mu_* + \Sigma_{f*} \Sigma_{ff}^{-1} (f - \mu) \]

\[ S = \Sigma_{**} - \Sigma_{f*} \Sigma_{ff}^{-1} \Sigma_{f*} \]

- A **kernel function** \( k \) is symmetric and positive definite.

- Kernel computes covariances between unknown function values \( f(x_i) \) and \( f(x_j) \) by just looking at the corresponding inputs \( x_i, x_j \)

\[ \Sigma^{(i,j)}_{ff} = \text{Cov}[f(x_i), f(x_j)] = k(x_i, x_j) \]

- This yields the predictive distribution

\[ p(f_* | f, X, X_*) = \mathcal{N}(m, S) \]

\[ m = \mu_* + k(X_*, X)k(X, X)^{-1}(f - \mu) \]

\[ S = K_{**} - k(X_*, X)k(X, X)^{-1}k(X, X_*) \]
GP Regression as a Bayesian Inference Problem

Objective

For a set of observations $y_i = f(x_i) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2_\varepsilon)$, find a distribution over functions $p(f)$ that explains the data.
GP Regression as a Bayesian Inference Problem

**Objective**

For a set of observations \( y_i = f(x_i) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2_{\varepsilon}) \), find a distribution over functions \( p(f) \) that explains the data.

Training data: \( X, y \). Bayes’ theorem yields

\[
p(f|X,y) = \frac{p(y|f,X) \, p(f)}{p(y|X)}
\]
GP Regression as a Bayesian Inference Problem

Objective

For a set of observations $y_i = f(x_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2_{\epsilon})$, find a distribution over functions $p(f)$ that explains the data.

Training data: $X, y$. Bayes’ theorem yields

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

Prior: $p(f) = \text{GP}(m, k)$ → Specify mean $m$ function and kernel $k$
Objective

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

Training data: $X, y$. Bayes' theorem yields

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

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

Likelihood (noise model): $p(y|f, X) = \mathcal{N}(f(X), \sigma^2_{\varepsilon} I)$
GP Regression as a Bayesian Inference Problem

**Objective**

For a set of observations \( y_i = f(x_i) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2_\varepsilon) \), find a distribution over functions \( p(f) \) that explains the data.

Training data: \( X, y \). Bayes’ theorem yields

\[
p(f|X, y) = \frac{p(y|f, X)p(f)}{p(y|X)}
\]

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

Likelihood (noise model): \( p(y|f, X) = \mathcal{N}(f(X), \sigma^2_\varepsilon I) \)

Marginal likelihood (evidence): \( p(y|X) = \int p(y|f, X)p(f)df \)
GP Regression as a Bayesian Inference Problem

Objective

For a set of observations $y_i = f(x_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2_\epsilon)$, find a distribution over functions $p(f)$ that explains the data.

Training data: $X, y$. Bayes’ theorem yields

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

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

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

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

Posterior: $p(f|y, X) = GP(m_{\text{post}}, k_{\text{post}})$
GP Regression as a Bayesian Inference Problem

Objective

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

Training data: \( X, y \). Bayes’ theorem yields

\[
p(f|X, y) = \frac{p(y|f, X) \ p(f)}{p(y|X)}
\]

Prior: \( p(f) = \text{GP}(m, k) \) ▶ Specify mean \( m \) function and kernel \( k \)

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

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

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

\[
m_{\text{post}}(x_i) = m(x_i) + k(X, x_i)^\top (K + \sigma_\epsilon^2 I)^{-1} (y - m(x_i))
\]

\[
k_{\text{post}}(x_i, x_j) = k(x_i, x_j) - k(X, x_i)^\top (K + \sigma_\epsilon^2 I)^{-1} k(X, x_j)
\]
GP Regression as a Bayesian Inference Problem (2)

Posterior Gaussian process:

\[
m_{\text{post}}(x_i) = m(x_i) + k(X, x_i)\top (K + \sigma_\varepsilon^2 \mathbf{I})^{-1}(y - m(x_i))
\]

\[
k_{\text{post}}(x_i, x_j) = k(x_i, x_j) - k(X, x_i)\top (K + \sigma_\varepsilon^2 \mathbf{I})^{-1}k(X, x_j)
\]
GP Regression as a Bayesian Inference Problem (2)

Posterior Gaussian process:

\[
\begin{align*}
    m_{\text{post}}(x_i) &= m(x_i) + k(X, x_i)^\top (K + \sigma^2 I)^{-1} (y - m(x_i)) \\
    k_{\text{post}}(x_i, x_j) &= k(x_i, x_j) - k(X, x_i)^\top (K + \sigma^2 I)^{-1} k(X, x_j)
\end{align*}
\]

Predictive distribution \( p(f_* | X, y, x_*) \) at test inputs \( x_* \):

\[
\begin{align*}
    p(f_* | X, y, x_*) &= \mathcal{N}(\mathbb{E}[f_*], \mathbb{V}[f_*]) \\
    \mathbb{E}[f_* | X, y, x_*] &= m_{\text{post}}(x_*) = m(x_*) + k(X, x_*)^\top (K + \sigma^2 I)^{-1} (y - m(x_*)) \\
    \mathbb{V}[f_* | X, y, x_*] &= k_{\text{post}}(x_*, x_*) = k(x_*, x_*) - k(X, x_*)^\top (K + \sigma^2 I)^{-1} k(X, x_*)
\end{align*}
\]
GP Regression as a Bayesian Inference Problem (2)

Posterior Gaussian process:

\[
m_{\text{post}}(x_i) = m(x_i) + k(X, x_i)^\top (K + \sigma_\epsilon^2 I)^{-1}(y - m(x_i))
\]
\[
k_{\text{post}}(x_i, x_j) = k(x_i, x_j) - k(X, x_i)^\top (K + \sigma_\epsilon^2 I)^{-1}k(X, x_j)
\]

Predictive distribution \(p(f_*|X, y, x_*)\) at test inputs \(x_*\):

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

From now: Set prior mean function \(m \equiv 0\)
Illustration

Prior belief about the function

Predictive (marginal) mean and variance:

\[
E[f(x_*) | x_*, \emptyset] = m(x_*) = 0 \\
\text{Var}[f(x_*) | x_*, \emptyset] = \sigma^2(x_*) = k(x_*, x_*)
\]
Prior belief about the function

Predictive (marginal) mean and variance:

\[
\begin{align*}
\mathbb{E}[f(x_*)|x_*, \varnothing] &= m(x_*) = 0 \\
\mathbb{V}[f(x_*)|x_*, \varnothing] &= \sigma^2(x_*) = k(x_*, x_*)
\end{align*}
\]
Posterior belief about the function

Predictive (marginal) mean and variance:

\[
\mathbb{E}[f(x_*)|x_*, X, y] = m(x_*) = k(X, x_*)^\top (K + \sigma^2 \mathcal{I})^{-1} y
\]

\[
\mathbb{V}[f(x_*)|x_*, X, y] = \sigma^2(x_*) = k(x_*, x_*) - k(X, x_*)^\top (K + \sigma^2 \mathcal{I})^{-1} k(X, x_*)
\]
Illustration

Posterior belief about the function

Predictive (marginal) mean and variance:

\[
\mathbb{E}[f(x_*)|x_*, X, y] = m(x_*) = k(X, x_*)^\top(K + \sigma^2 I)^{-1}y
\]

\[
\mathbb{V}[f(x_*)|x_*, X, y] = \sigma^2(x_*) = k(x_*, x_*) - k(X, x_*)^\top(K + \sigma^2 I)^{-1}k(X, x_*)
\]
Illustration

Posterior belief about the function

Predictive (marginal) mean and variance:

\[
E[f(x_*)|x_*, X, y] = m(x_*) = k(X, x_*)^\top (K + \sigma^2 I)^{-1} y \\
\text{Var}[f(x_*)|x_*, X, y] = \sigma^2(x_*) = k(x_*, x_*) - k(X, x_*)^\top (K + \sigma^2 I)^{-1} k(X, x_*)
\]
Illustration

Posterior belief about the function

Predictive (marginal) mean and variance:

\[ \mathbb{E}[f(x_*)|x_*, X, y] = m(x_*) = k(X, x_*)^\top (K + \sigma^2\epsilon I)^{-1}y \]
\[ \mathbb{V}[f(x_*)|x_*, X, y] = \sigma^2(x_*) = k(x_*, x_*) - k(X, x_*)^\top (K + \sigma^2\epsilon I)^{-1}k(X, x_*) \]
Posterior belief about the function

Predictive (marginal) mean and variance:

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

\[
\mathbb{V}[f(x_*)|x_*, X, y] = \sigma^2(x_*) = k(x_*, x_*) - k(X, x_*)^\top (K + \sigma^2 \mathbf{I})^{-1} k(X, x_*)
\]
Predictive (marginal) mean and variance:

\[
E[f(x_*)|x_*, X, y] = m(x_*) = k(X, x_*)^\top (K + \sigma^2 I)^{-1} y
\]

\[
\text{Var}[f(x_*)|x_*, X, y] = \sigma^2(x_*) = k(x_*, x_*) - k(X, x_*)^\top (K + \sigma^2 I)^{-1} k(X, x_*)
\]
Posterior belief about the function

Predictive (marginal) mean and variance:

\[
\mathbb{E}[f(x_*)|x_*, X, y] = m(x_*) = k(X, x_*)^\top (K + \sigma^2 \delta I)^{-1} y \\
\mathbb{V}[f(x_*)|x_*, X, y] = \sigma^2(x_*) = k(x_*, x_*) - k(X, x_*)^\top (K + \sigma^2 \delta I)^{-1} k(X, x_*)
\]
Posterior belief about the function

Predictive (marginal) mean and variance:

\[
E[f(x_*)|x_*, X, y] = m(x_*) = k(X, x_*)^\top (K + \sigma^2 \mathbb{I})^{-1} y
\]

\[
V[f(x_*)|x_*, X, y] = \sigma^2(x_*) = k(x_*, x_*) - k(X, x_*)^\top (K + \sigma^2 \mathbb{I})^{-1} k(X, x_*)
\]
Illustration

Posterior belief about the function

Predictive (marginal) mean and variance:

\[ \mathbb{E}[f(x_*)| x_*, X, y] = m(x_*) = k(X, x_*)^\top (K + \sigma^2 I)^{-1} y \]

\[ \text{Var}[f(x_*)| x_*, X, y] = \sigma^2(x_*) = k(x_*, x_*) - k(X, x_*)^\top (K + \sigma^2 I)^{-1} k(X, x_*) \]
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

\[
k_{\text{Gauss}}(x_i, x_j) = \theta_1^2 \exp \left( -\frac{(x_i - x_j)^T(x_i - x_j)}{2\theta_2^2} \right)
\]

- \(\theta_1\): Amplitude of the latent function
- \(\theta_2\): Length scale. How far do we have to move in input space before the function value changes significantly
  - Smoothness parameter
Matérn Covariance Function

\[ k_{\text{Mat}, 3/2}(x_i, x_j) = \theta_1^2 \left( 1 + \frac{\sqrt{3}(x_i - x_j)}{\theta_2} \right) \exp \left( - \frac{\sqrt{3}(x_i - x_j)}{\theta_2} \right) \]
Periodic Covariance Function

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

\[ = k_{Gauss}(\mathbf{u}(x_i), \mathbf{u}(x_j)), \quad \mathbf{u}(x) = \begin{bmatrix} \cos(x) \\ \sin(x) \end{bmatrix} \]

\( \theta_3 \): Periodicity parameter
Model Selection

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

- How do we find good parameters $\theta$?
- 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
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 $\theta$ (kernel and mean function parameters)
Model Selection: Hyper-Parameters

GP Training

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

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

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

$$p(y|X, \theta) = \int p(y|f(X))p(f|\theta)df$$
Model Selection: Hyper-Parameters

GP Training

Find good GP hyper-parameters \( \theta \) (kernel and mean function parameters)

- Assign a prior \( p(\theta) \) over hyper-parameters
- Posterior over hyper-parameters:

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

- Choose MAP hyper-parameters \( \theta^* \), such that

\[
\theta^* \in \arg \max_{\theta} \log p(\theta) + \log p(y|X, \theta)
\]
Model Selection: Hyper-Parameters

GP Training

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

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

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

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

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

Maximize marginal likelihood if $p(\theta) = \mathcal{U}$ (uniform prior)
Training via Marginal Likelihood Maximization

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

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

\[ \log p(y|X, \theta) = -\frac{1}{2} y^T K_\theta^{-1} y - \frac{1}{2} \log |K_\theta| + \text{const} \]
Training via Marginal Likelihood Maximization

GP Training

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

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

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

- 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)

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

\[
\log p(y|X, \theta) = -\frac{1}{2} y^\top K^{-1} y - \frac{1}{2} \log |K_\theta| + \text{const}
\]

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

\[
\frac{\partial \log p(y|X, \theta)}{\partial \theta} = \frac{1}{2} y^\top K^{-1} \frac{\partial K}{\partial \theta} K^{-1} y - \frac{1}{2} \text{tr}(K^{-1} \frac{\partial K}{\partial \theta})
\]
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)
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?
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:**

\[
p(M_i|X, y) = \frac{p(M_i)p(y|X, M_i)}{p(y|X)}
\]
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:**

\[
p(M_i|X,y) = \frac{p(M_i)p(y|X,M_i)}{p(y|X)}
\]

- Choose MAP model $M_\star$, such that

\[
M_\star \in \arg \max_M \log p(M) + \log p(y|X,M)
\]

Gaussian Processes
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:

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

- Choose MAP model $M_*$, such that

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

➤ 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:

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

- Choose MAP model $M_*$, such that

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

- Compare marginal likelihoods if $p(M_i) = 1/|M|$
- 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 \( m = 0 \))
- 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
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 $O(N^3)$
- Prediction (variances) scales in $O(N^2)$
- Memory requirement: $O(ND + N^2)$

Practical limit $N \approx 10,000$
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>Gaussian Distribution</td>
</tr>
<tr>
<td>Gaussian Processes</td>
</tr>
<tr>
<td>Definition and Derivation</td>
</tr>
<tr>
<td>Inference</td>
</tr>
<tr>
<td>Covariance Functions</td>
</tr>
<tr>
<td>Model Selection</td>
</tr>
<tr>
<td>Limitations</td>
</tr>
<tr>
<td>Scaling Gaussian Processes to Large Data Sets</td>
</tr>
<tr>
<td>Sparse Gaussian Processes</td>
</tr>
<tr>
<td>Distributed Gaussian Processes</td>
</tr>
</tbody>
</table>
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

\[
p(f, f_\ast) = \mathcal{N}
\begin{bmatrix}
0 \\
0
\end{bmatrix},
\begin{bmatrix}
K_{ff} & K_{f\ast} \\
K_{f\ast} & K_{\ast\ast}
\end{bmatrix}
\]
Inducing Variables

- Introduce **inducing function values** \( f_u \)
  - “Hypothetical” function values

\[
\begin{align*}
\text{Inducing function values} & : f(u_1), \ldots, f(u_M) \\
\text{Training data} & : f(x_1), \ldots, f(x_N) \\
\text{Test data} & : f^*_1, \ldots, f^*_L
\end{align*}
\]
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: $f \perp f_* | f_u$
Central Approximation Scheme

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

\[
q(f, f_*) = \int p(f \mid f_u) p(f_* \mid f_u) p(f_u) df_u
\]

\[
= \mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{ff} & Q_{f*} \\ Q_{*f} & K_{**} \end{bmatrix} \right) , \quad Q_{ab} := K_{afu} K_{fufu}^{-1} K_{fub}
\]
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

\[ q(f, f^*) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} Q_{ff} - \text{diag}(Q_{ff} - K_{ff}) & Q_{f*} \\ Q_{*f} & K_{**} \end{bmatrix}\right) \]

- \(Q_{**} - \text{diag}(Q_{**} - K_{**})\) can be used instead of \(K_{**}\) ➤ 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

\[
q(f, f^*) = \mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} Q_{ff} - \text{diag}(Q_{ff} - K_{ff}) & Q_{f^*} \\ Q_{f^*}^T & K_{**} \end{bmatrix} \right)
\]

• \(Q_{**} - \text{diag}(Q_{**} - K_{**})\) can be used instead of \(K_{**}\) ➤ FIC

• Training: \(O(NM^2)\), Prediction: \(O(M^2)\)
Inducing Inputs

- FI(T)C sparse approximation relies on inducing function values $f(u_i)$, where $u_i$ are the corresponding inputs

Intuitively: The marginal likelihood is not only parameterized by the hyper-parameters $\theta$, but also by the inducing inputs $u_1:M$.

End up with a high-dimensional non-convex optimization problem with MD additional parameters.

Gaussian Processes

Marc Deisenroth

@MLSS, 14 April 2015
Inducing Inputs

- FI(T)C sparse approximation relies on inducing function values \( f(u_i) \), where \( u_i \) 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 $f(u_i)$, where $u_i$ 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):

$$u_{1:M}^* \in \arg\max_{u_{1:M}} q_{FITC}(y|X, u_{1:M}, \theta)$$
Inducing Inputs

- FI(T)C sparse approximation relies on inducing function values $f(u_i)$, where $u_i$ are the corresponding inputs.
- These inputs are unknown a priori $\Rightarrow$ Find “optimal” ones.
- Find them by maximizing the FI(T)C marginal likelihood with respect to the inducing inputs (and the standard hyper-parameters):

$$u^*_1:M \in \arg\max_{u_1:M} q_{\text{FITC}}(y|X, u_1:M, \theta)$$

- Intuitively: The marginal likelihood is not only parameterized by the hyper-parameters $\theta$, but also by the inducing inputs $u_1:M$. 
Inducing Inputs

- FI(T)C sparse approximation relies on inducing function values \( f(u_i) \), where \( u_i \) 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):

\[
u_{1:M}^* \in \arg \max_{u_{1:M}} q_{\text{FITC}}(y|X,u_{1:M},\theta)
\]

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

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

Efficient compression of the original data set

Figure from Ed Snelson
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: $O(pM^3q)$ or $O(pNM^2q)$ for training.

Practical limit $M \lesssim 10^4$. Often: $M \approx O(p10^2q)$ in the case of inducing variables.

If we set $M = \frac{1}{100}$, i.e., each inducing function value summarizes 100 real function values, our practical limit is $N \approx O(p10^6q)$.
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)
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 \( \mathcal{O}(NM^2) \) for training
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 $\mathcal{O}(NM^2)$ for training.
- **Practical limit** $M \leq 10^4$. 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 \( \mathcal{O}(NM^2) \) for training
- Practical limit \( M \leq 10^4 \). 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 \( \mathcal{O}(NM^2) \) for training
- Practical limit \( M \leq 10^4 \). 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

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 $K$

Combine independent computations to an overall result

Gaussian Processes

Marc Deisenroth

@MLSS, 14 April 2015
An Orthogonal Approximation: Distributed GPs

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

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

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

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

- 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:

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

- Distributed optimization and training straightforward
- Computational complexity: $O(MP^3)$ [instead of $O(N^3)$]
  But distributed over many machines
- Memory footprint: $O(MP^2 + ND)$ [instead of $O(N^2 + ND)$]
• NLML is proportional to training time
Empirical Training Time

- NLML is proportional to training time
- Full GP (16K training points) \(\approx\) sparse GP (50K training points) 
  \(\approx\) 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: $\approx 30$ min
- Training* with $N = 5 \times 10^6, D = 8$ on a workstation: $\approx 4$ hours
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***
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 $O(N^3)$)
Predictions with the Distributed GP

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

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

**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

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

Figure: Two computational graphs
Objectives

- **Scale** to large data sets ✓
- **Good approximation** of full GP (“ground truth”)
- Predictions **independent of computational graph**
  - Runs on heterogeneous computing infrastructures (laptop, cluster, ...)

Figure: Two computational graphs
Objectives

- 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

Figure: Two computational graphs
Investigate various product-of-experts models
Same training procedure, but different mechanisms for predictions
Product of GP Experts

- Prediction model (independent predictors):

\[
p(f_* | x_*, D) = \prod_{k=1}^{M} p_k(f_* | x_*, D^{(k)}) ,
\]

\[
p_k(f_* | x_*, D^{(k)}) = \mathcal{N}(f_* | \mu_k(x_*), \sigma_k^2(x_*))
\]
Product of GP Experts

- Prediction model (independent predictors):

\[ p(f_* | x_*, D) = \prod_{k=1}^{M} p_k(f_* | x_*, D^{(k)}) , \]

\[ p_k(f_* | x_*, D^{(k)}) = \mathcal{N}(f_* | \mu_k(x_*), \sigma^2_k(x_*)) \]

- Predictive precision (inverse variance) and mean:

\[ (\sigma_{*\text{poe}})^{-2} = \sum_k \sigma_k^{-2}(x_*) \]

\[ \mu_{*\text{poe}} = (\sigma_{*\text{poe}})^2 \sum_k \sigma_k^{-2}(x_*) \mu_k(x_*) \]
Product of GP Experts

- Prediction model (independent predictors):
  \[
  p(f_\ast | x_\ast, D) = \prod_{k=1}^{M} p_k(f_\ast | x_\ast, D^{(k)}) ,
  \]
  \[
  p_k(f_\ast | x_\ast, D^{(k)}) = \mathcal{N}(f_\ast | \mu_k(x_\ast), \sigma_k^2(x_\ast))
  \]

- Predictive precision (inverse variance) and mean:
  \[
  (\sigma^\text{poe}_\ast)^{-2} = \sum_k \sigma_k^{-2}(x_\ast)
  \]
  \[
  \mu^\text{poe}_\ast = (\sigma^\text{poe}_\ast)^2 \sum_k \sigma_k^{-2}(x_\ast) \mu_k(x_\ast)
  \]

- Independent of the computational graph ✓
Product of GP Experts

- Unreasonable variances for $M > 1$:
Product of GP Experts

- Unreasonable variances for $M > 1$:

$$\left(\sigma_{\text{poe}}^*\right)^{-2} = \sum_k \sigma_{k}^{-2}(x_*)$$

- The more experts the more certain the prediction, even if every expert itself is very uncertain $\times$ ➤ Cannot fall back to the prior
Generalized Product of GP Experts

- Weight the responsibility of each expert in PoE with $\beta_k$
Generalized Product of GP Experts

- Weight the responsibility of each expert in PoE with $\beta_k$
- Prediction model (independent predictors):

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

$$p_k(f_*|x_*, D^{(k)}) = \mathcal{N}(f_* | \mu_k(x_*), \sigma_k^2(x_*))$$
Generalized Product of GP Experts

- Weight the responsibility of each expert in PoE with $\beta_k$
- Prediction model (independent predictors):

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

$$p_k(f_*|x_*, D^{(k)}) = \mathcal{N}(f_* | \mu_k(x_*), \sigma_k^2(x_*))$$

- Predictive precision and mean:

$$\left(\sigma_{*}^{gpoe}\right)^{-2} = \sum_k \beta_k \sigma_k^{-2}(x_*)$$

$$\mu_{*}^{gpoe} = \left(\sigma_{*}^{gpoe}\right)^2 \sum_k \beta_k \sigma_k^{-2}(x_*) \mu_k(x_*)$$
Generalized Product of GP Experts

- Weight the responsibility of each expert in PoE with $\beta_k$
- Prediction model (independent predictors):
  
  $$p(f_* | x_*, D) = \prod_{k=1}^{M} p_k^{\beta_k}(f_* | x_*, D^{(k)})$$

  $$p_k(f_* | x_*, D^{(k)}) = \mathcal{N}(f_* | \mu_k(x_*), \sigma_k^2(x_*))$$

- Predictive precision and mean:
  
  $$\left(\sigma_{gpoe}^*\right)^{-2} = \sum_k \beta_k \sigma_k^{-2}(x_*)$$

  $$\mu_{gpoe}^* = \left(\sigma_{gpoe}^*\right)^2 \sum_k \beta_k \sigma_k^{-2}(x_*) \mu_k(x_*)$$

- 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 responsibility of each expert in PoE with $\beta_k$
- Prediction model (independent predictors):

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

$$p_k(f_*|x_*, D^{(k)}) = \mathcal{N}(f_* | \mu_k(x_*), \sigma_k^2(x_*))$$

- Predictive precision and mean:

$$\left(\sigma_{*}^{\text{gpoe}}\right)^{-2} = \sum_k \beta_k \sigma_k^{-2}(x_*)$$

$$\mu_{*}^{\text{gpoe}} = \left(\sigma_{*}^{\text{gpoe}}\right)^2 \sum_k \beta_k \sigma_k^{-2}(x_*) \mu_k(x_*)$$

- 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 \((\mathcal{D}^{(j)} \perp \mathcal{D}^{(k)} | f_\ast)\):

\[
p(f_\ast | x_\ast, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_\ast | x_\ast, \mathcal{D}^{(k)})}{p^{M-1}(f_\ast)}
\]
Bayesian Committee Machine

- **Apply Bayes’ theorem when combining predictions** (and not only for computing predictions)
- Prediction model \( \mathcal{D}^{(j)} \perp \mathcal{D}^{(k)} | f_* \):
  \[
p(f_* | x_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_* | x_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}
\]
- Predictive precision and mean:
  \[
  (\sigma^{bcm}_*)^{-2} = \sum_{k=1}^{M} \sigma^{-2}_k(x_*) - (M - 1)\sigma^{-2}_{**}
  
  \mu^{bcm}_* = (\sigma^{bcm})^2 \sum_{k=1}^{M} \sigma^{-2}_k(x_*) \mu_k(x_*)
  \]
Bayesian Committee Machine

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

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

- Predictive precision and mean:

$$\left(\sigma^\text{bcm}\right)^{-2} = \sum_{k=1}^{M} \sigma^{-2}_{k}(x_*) - (M - 1)\sigma^{-2}_{**}$$

$$\mu^\text{bcm} = \left(\sigma^\text{bcm}\right)^2 \sum_{k=1}^{M} \sigma^{-2}_{k}(x_*) \mu_k(x_*)$$

- 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** \((\mathcal{D}^{(j)} \perp \mathcal{D}^{(k)} | f_*)\):

\[
p(f_* | x_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_* | x_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}
\]

- **Predictive precision and mean**:

\[
(\sigma_{*}^{\text{bcm}})^{-2} = \sum_{k=1}^{M} \sigma_{k}^{-2}(x_*) - (M - 1)\sigma_{**}^{-2}
\]

\[
\mu_{*}^{\text{bcm}} = (\sigma_{*}^{\text{bcm}})^2 \sum_{k=1}^{M} \sigma_{k}^{-2}(x_*) \mu_k(x_*)
\]

- **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 \( (\mathcal{D}^{(j)} \perp \mathcal{D}^{(k)} | f_*) \):

\[
p(f_*|x_*, \mathcal{D}) = \frac{\prod_{k=1}^{M} p_k(f_*|x_*, \mathcal{D}^{(k)})}{p^{M-1}(f_*)}
\]

- Predictive precision and mean:

\[
(\sigma_{*}^{bcm})^{-2} = \sum_{k=1}^{M} \sigma_k^{-2}(x_*) -(M - 1)\sigma_{**}^{-2}
\]

\[
\mu_{*}^{bcm} = (\sigma_{*}^{bcm})^2 \sum_{k=1}^{M} \sigma_k^{-2}(x_*) \mu_k(x_*)
\]

- 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

Gaussian Processes

Marc Deisenroth

@MLSS, 14 April 2015
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 $\mathcal{D}^{(j)} \perp \mathcal{D}^{(k)} | f_*$):

$$p(f_* | x_*, \mathcal{D}) = \frac{\prod_{k=1}^M \beta_k}{p \sum_k \beta_k^{-1}(f_*)} p_k (f_* | x_*, \mathcal{D}^{(k)})$$
Robust Bayesian Committee Machine

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

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

- Predictive precision and mean:

$$\sigma_{*}^{\text{rbcm}}^{-2} = \sum_{k=1}^{M} \beta_k \sigma_k^{-2}(x_*) + (1 - \sum_{k=1}^{M} \beta_k) \sigma_{**}^{-2},$$

$$\mu_*^{\text{rbcm}} = (\sigma_*^{\text{rbcm}})^2 \sum_k \beta_k \sigma_k^{-2}(x_*) \mu_k(x_*)$$
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 $\beta_k$) ✓
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

\[
 f(x) = \sum_{i \in \mathbb{Z}} \int_{i}^{i+1} \gamma(s) \exp \left( -\frac{(x - s)^2}{\lambda^2} \right) ds = \int_{-\infty}^{\infty} \gamma(s) \exp \left( -\frac{(x - s)^2}{\lambda^2} \right) ds
\]
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

\[
f(x) = \sum_{i \in \mathbb{Z}} \int_{i}^{i+1} \gamma(s) \exp \left( -\frac{(x - s)^2}{\lambda^2} \right) \, ds = \int_{-\infty}^{\infty} \gamma(s) \exp \left( -\frac{(x - s)^2}{\lambda^2} \right) \, ds
\]

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

\( \Rightarrow \) GP with mean 0 and Gaussian covariance function
Two-level inference

- **level-1 inference:**
  \[
  p(f|X, y, \theta) = \frac{p(y|X, f) p(f|\theta)}{p(y|X, \theta)},
  \]
  \[
  p(y|X, \theta) = \int p(y|X, f) p(f|\theta) dh
  \]

- **level-2 inference**
  \[
  p(\theta|X, y) = \frac{p(y|X, \theta) p(\theta)}{p(y|X)}
  \]
References I


References II


