Probabilistic Modelling and Reasoning: A Machine Learning Approach

Gaussian Processes

Edwin V. Bonilla

Principal Research Scientist, CSIRO's Data61
Associate Professor (Hon.), Australian National University

December 15th, 2021

This Lecture: Outline

1. The Bayesian Linear Model Revisited
2. Gaussian Processes: Function-Space View
   - Gaussian Process Regression
   - Model Selection
3. Challenges
4. Model Approximations

The Bayesian Linear Model Revisited

Limitations of the Bayesian Linear Model

- Linear models require specifying a set of basis functions
  - Polynomials, Trigonometric, . . . ??

Limitations of the Bayesian Linear Model

- Linear models require specifying a set of basis functions
  - Polynomials, Trigonometric, ...??
- Gaussian Processes work implicitly with a possibly infinite set of basis functions!

Bayesian Linear Model: Rewriting the Predictive Distribution

- Consider the predictive distribution of the noiseless targets:

$$p(f_* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}_*, \sigma^2) = \mathcal{N}(f_*; \sigma^{-2} \boldsymbol{\varphi}_*^\top \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \mathbf{y}, \boldsymbol{\varphi}_*^\top \boldsymbol{\Sigma} \boldsymbol{\varphi}_*),$$

where, as before, $\boldsymbol{\Sigma} = \left(\frac{1}{\sigma^2} \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \mathbf{S}^{-1} \right)^{-1}$ and $\boldsymbol{\varphi}_* \stackrel{\text{def}}{=} \boldsymbol{\varphi}(\mathbf{x}_*)$

Bayesian Linear Model: Rewriting the Predictive Distribution

- Consider the predictive distribution of the noiseless targets:

$$p(f_* \,|\, \mathbf{X}, \mathbf{y}, \mathbf{x}_*, \sigma^2) = \mathcal{N}(f_*; \sigma^{-2}\boldsymbol{\varphi}_*^\top \Sigma \Phi^\top \mathbf{y}, \boldsymbol{\varphi}_*^\top \Sigma \boldsymbol{\varphi}_*),$$

  where, as before, $\Sigma = \left(\frac{1}{\sigma^2}\Phi^\top\Phi + \mathbf{S}^{-1}\right)^{-1}$ and $\boldsymbol{\varphi}_* \stackrel{\text{def}}{=} \boldsymbol{\varphi}(\mathbf{x}_*)$
  - Need to invert a D-dimensional matrix.

Bayesian Linear Model: Rewriting the Predictive Distribution

- Consider the predictive distribution of the noiseless targets:

$$p(f_* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}_*, \sigma^2) = \mathcal{N}(f_*; \sigma^{-2}\boldsymbol{\varphi}_*^\top \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \mathbf{y}, \boldsymbol{\varphi}_*^\top \boldsymbol{\Sigma} \boldsymbol{\varphi}_*),$$

  where, as before, $\boldsymbol{\Sigma} = \left(\frac{1}{\sigma^2} \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \mathbf{S}^{-1} \right)^{-1}$ and $\boldsymbol{\varphi}_* \stackrel{\text{def}}{=} \boldsymbol{\varphi}(\mathbf{x}_*)$

  - Need to invert a D-dimensional matrix.

- We can re-write this predictive distribution as:

$$p(f_* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}_*, \sigma^2) = \mathcal{N}(f_*; \mathbf{k}_*^\top \mathbf{K}_{\mathbf{y}}^{-1} \mathbf{y}, k_{**} - \mathbf{k}_*^\top \mathbf{K}_{\mathbf{y}}^{-1} \mathbf{k}_*)$$

  where $\mathbf{k}_* = \boldsymbol{\Phi} \mathbf{S} \boldsymbol{\varphi}_*$, $k_{**} = \boldsymbol{\varphi}_*^\top \mathbf{S} \boldsymbol{\varphi}_*$, and $\mathbf{K}_{\mathbf{y}} = \boldsymbol{\Phi} \mathbf{S} \boldsymbol{\Phi}^\top + \sigma^2 \mathbf{I}$

Bayesian Linear Model: Rewriting the Predictive Distribution

- Consider the predictive distribution of the noiseless targets:

$$p(f_* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}_*, \sigma^2) = \mathcal{N}(f_*; \sigma^{-2} \boldsymbol{\varphi}_*^\top \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \mathbf{y}, \boldsymbol{\varphi}_*^\top \boldsymbol{\Sigma} \boldsymbol{\varphi}_*),$$

  where, as before, $\boldsymbol{\Sigma} = \left(\frac{1}{\sigma^2} \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \mathbf{S}^{-1} \right)^{-1}$ and $\boldsymbol{\varphi}_* \stackrel{\text{def}}{=} \boldsymbol{\varphi}(\mathbf{x}_*)$
  - Need to invert a D-dimensional matrix.
- We can re-write this predictive distribution as:

$$p(f_* \mid \mathbf{X}, \mathbf{y}, \mathbf{x}_*, \sigma^2) = \mathcal{N}(f_*; \mathbf{k}_*^\top \mathbf{K}_{\mathbf{y}}^{-1} \mathbf{y}, k_{**} - \mathbf{k}_*^\top \mathbf{K}_{\mathbf{y}}^{-1} \mathbf{k}_*)$$

  where $\mathbf{k}_* = \boldsymbol{\Phi} \mathbf{S} \boldsymbol{\varphi}_*$, $k_{**} = \boldsymbol{\varphi}_*^\top \mathbf{S} \boldsymbol{\varphi}_*$, and $\mathbf{K}_{\mathbf{y}} = \boldsymbol{\Phi} \mathbf{S} \boldsymbol{\Phi}^\top + \sigma^2 \mathbf{I}$
  - Need to invert an N-dimensional matrix

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \boldsymbol{\Phi}\mathbf{S}\boldsymbol{\varphi}_*,\ k_{**} = \boldsymbol{\varphi}_*^\top\mathbf{S}\boldsymbol{\varphi}_* \text{ and } \mathbf{K}_\mathbf{y} = \boldsymbol{\Phi}\mathbf{S}\boldsymbol{\Phi}^\top + \sigma^2\mathbf{I}$$

  the features always enter in the form $\boldsymbol{\varphi}(\mathbf{x})^\top\mathbf{S}\boldsymbol{\varphi}(\mathbf{x}')$

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \boldsymbol{\Phi}\mathbf{S}\boldsymbol{\varphi}_*,\ k_{**} = \boldsymbol{\varphi}_*^\top\mathbf{S}\boldsymbol{\varphi}_* \text{ and } \mathbf{K}_\mathbf{y} = \boldsymbol{\Phi}\mathbf{S}\boldsymbol{\Phi}^\top + \sigma^2\mathbf{I}$$

  the features always enter in the form $\boldsymbol{\varphi}(\mathbf{x})^\top\mathbf{S}\boldsymbol{\varphi}(\mathbf{x}')$
- This is an inner product wrt $\mathbf{S}$

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \Phi \mathbf{S} \boldsymbol{\varphi}_*,\ k_{**} = \boldsymbol{\varphi}_*^\top \mathbf{S} \boldsymbol{\varphi}_* \text{ and } \mathbf{K}_\mathbf{y} = \Phi \mathbf{S} \Phi^\top + \sigma^2 \mathbf{I}$$

  the features always enter in the form $\varphi(\mathbf{x})^\top \mathbf{S} \varphi(\mathbf{x}')$
- This is an inner product wrt $\mathbf{S}$
- As $\mathbf{S}$ is PD we can rewrite:

$$\begin{aligned} \varphi(\mathbf{x})^T \mathbf{S} \varphi(\mathbf{x}') &= \varphi(\mathbf{x})^T \mathbf{S}^{1/2} \mathbf{S}^{1/2} \varphi(\mathbf{x}') \\ &= (\underbrace{\mathbf{S}^{1/2} \varphi(\mathbf{x})}_{\psi(\mathbf{x})})^T (\underbrace{\mathbf{S}^{1/2} \varphi(\mathbf{x}')}_{\psi(\mathbf{x}')}) \end{aligned}$$

$$\kappa(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x}) \cdot \psi(\mathbf{x}')$$

The Kernel Trick

- Note that in:

$$\mathbf{k}_* = \boldsymbol{\Phi}\mathbf{S}\boldsymbol{\varphi}_*,\ k_{**} = \boldsymbol{\varphi}_*^\top \mathbf{S}\boldsymbol{\varphi}_* \text{ and } \mathbf{K}_\mathbf{y} = \boldsymbol{\Phi}\mathbf{S}\boldsymbol{\Phi}^\top + \sigma^2\mathbf{I}$$

  the features always enter in the form $\boldsymbol{\varphi}(\mathbf{x})^\top \mathbf{S}\boldsymbol{\varphi}(\mathbf{x}')$
- This is an inner product wrt $\mathbf{S}$
- As $\mathbf{S}$ is PD we can rewrite:

$$\begin{aligned}\boldsymbol{\varphi}(\mathbf{x})^T \mathbf{S}\boldsymbol{\varphi}(\mathbf{x}') &= \boldsymbol{\varphi}(\mathbf{x})^T \mathbf{S}^{1/2}\mathbf{S}^{1/2}\boldsymbol{\varphi}(\mathbf{x}') \\ &= (\underbrace{\mathbf{S}^{1/2}\boldsymbol{\varphi}(\mathbf{x})}_{\boldsymbol{\psi}(\mathbf{x})})^T (\underbrace{\mathbf{S}^{1/2}\boldsymbol{\varphi}(\mathbf{x}')}_{\boldsymbol{\psi}(\mathbf{x}')})\end{aligned}$$

$$\kappa(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}(\mathbf{x}) \cdot \boldsymbol{\psi}(\mathbf{x}')$$

- $\kappa(\cdot, \cdot)$ is called a kernel or covariance function

Bayesian Linear Regression as a Kernel Machine

- Predictions can be expressed exclusively in terms of scalar products as follows

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\psi}(\mathbf{x}_i)^\top \boldsymbol{\psi}(\mathbf{x}_j)$$

Bayesian Linear Regression as a Kernel Machine

- Predictions can be expressed exclusively in terms of scalar products as follows

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\psi}(\mathbf{x}_i)^\top \boldsymbol{\psi}(\mathbf{x}_j)$$

- This allows us to work with either $k(\cdot, \cdot)$ or $\boldsymbol{\psi}(\cdot)$

Bayesian Linear Regression as a Kernel Machine

- Predictions can be expressed exclusively in terms of scalar products as follows

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\psi}(\mathbf{x}_i)^\top \boldsymbol{\psi}(\mathbf{x}_j)$$

- This allows us to work with either $k(\cdot, \cdot)$ or $\boldsymbol{\psi}(\cdot)$
- Why is this useful??

Bayesian Linear Regression as a Kernel Machine

- Predictions can be expressed exclusively in terms of scalar products as follows

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\psi}(\mathbf{x}_i)^\top \boldsymbol{\psi}(\mathbf{x}_j)$$

- This allows us to work with either $k(\cdot, \cdot)$ or $\boldsymbol{\psi}(\cdot)$
- Why is this useful??
- We can replace all occurrences of inner products by $\kappa(\cdot, \cdot)$

Bayesian Linear Regression as a Kernel Machine

- Predictions can be expressed exclusively in terms of scalar products as follows

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\psi}(\mathbf{x}_i)^\top \boldsymbol{\psi}(\mathbf{x}_j)$$

- This allows us to work with either $k(\cdot, \cdot)$ or $\boldsymbol{\psi}(\cdot)$
- Why is this useful??
- We can replace all occurrences of inner products by $\kappa(\cdot, \cdot)$
- Working with $\boldsymbol{\psi}(\cdot)$ costs $O(D^2)$ memory, $O(D^3)$ time

Bayesian Linear Regression as a Kernel Machine

- Predictions can be expressed exclusively in terms of scalar products as follows

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\psi}(\mathbf{x}_i)^\top \boldsymbol{\psi}(\mathbf{x}_j)$$

- This allows us to work with either $k(\cdot, \cdot)$ or $\boldsymbol{\psi}(\cdot)$
- Why is this useful??
- We can replace all occurrences of inner products by $\kappa(\cdot, \cdot)$
- Working with $\boldsymbol{\psi}(\cdot)$ costs $O(D^2)$ memory, $O(D^3)$ time
- Working with $k(\cdot, \cdot)$ costs $O(N^2)$ memory, $O(N^3)$ time

Bayesian Linear Regression as a Kernel Machine

- Predictions can be expressed exclusively in terms of scalar products as follows

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\psi}(\mathbf{x}_i)^\top \boldsymbol{\psi}(\mathbf{x}_j)$$

- This allows us to work with either $k(\cdot, \cdot)$ or $\boldsymbol{\psi}(\cdot)$
- Why is this useful??
- We can replace all occurrences of inner products by $\kappa(\cdot, \cdot)$
- Working with $\boldsymbol{\psi}(\cdot)$ costs $O(D^2)$ memory, $O(D^3)$ time
- Working with $k(\cdot, \cdot)$ costs $O(N^2)$ memory, $O(N^3)$ time
- We do not need to compute the feature vectors explicitly

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- Then $f(\mathbf{x})$ at a particular point is a random variable:

$$f(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w} | \mathbf{0}, \mathbf{S})$$

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- Then $f(\mathbf{x})$ at a particular point is a random variable:

$$f(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w} | \mathbf{0}, \mathbf{S})$$

- A collection $f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(N)})$, define a stochastic process

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- Then $f(\mathbf{x})$ at a particular point is a random variable:

$$f(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{S})$$

- A collection $f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(N)})$, define a stochastic process
- With mean and the covariance function

$$\mathbb{E}_{\mathbf{w}}[f(\mathbf{x})] = 0$$
$$\mathbb{E}_{\mathbf{w}}[f(\mathbf{x})f(\mathbf{x}')] = \boldsymbol{\varphi}^\top(\mathbf{x})\mathbf{S}\boldsymbol{\varphi}(\mathbf{x}')$$

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- Then $f(\mathbf{x})$ at a particular point is a random variable:

$$f(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{S})$$

- A collection $f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(N)})$, define a stochastic process
- With mean and the covariance function

$$\mathbb{E}_{\mathbf{w}}[f(\mathbf{x})] = 0$$
$$\mathbb{E}_{\mathbf{w}}[f(\mathbf{x})f(\mathbf{x}')] = \boldsymbol{\varphi}^\top(\mathbf{x})\mathbf{S}\boldsymbol{\varphi}(\mathbf{x}')$$

- The Bayesian linear model is a Gaussian process

From a Prior over Weights to a Prior over Functions

- Consider the kinds of functions that can be generated from a set of basis functions with random weights.
- Then $f(\mathbf{x})$ at a particular point is a random variable:

$$f(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \text{ with } \mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{S})$$

- A collection $f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(N)})$, define a stochastic process
- With mean and the covariance function

$$\mathbb{E}_{\mathbf{w}}[f(\mathbf{x})] = 0$$
$$\mathbb{E}_{\mathbf{w}}[f(\mathbf{x})f(\mathbf{x}')] = \boldsymbol{\varphi}^\top(\mathbf{x})\mathbf{S}\boldsymbol{\varphi}(\mathbf{x}')$$

- The Bayesian linear model is a Gaussian process
  - The Function values have a joint Gaussian distribution

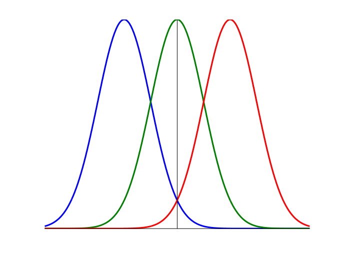
Sample Functions from the Linear Model

1. Define $\varphi_j(x) = \exp(-\frac{1}{2}(x - \mu_j)^2)$, for $j = 1, 2, 3$

Sample Functions from the Linear Model

1. Define $\varphi_j(x) = \exp(-\frac{1}{2}(x - \mu_j)^2)$, for $j = 1, 2, 3$
2. Construct $\Phi(i, j) = \phi_j(x^{(i)})$, for $j = 1, 2, 3$

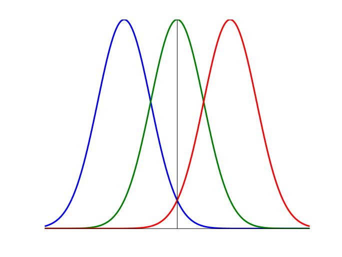
Φ



Sample Functions from the Linear Model

1. Define $\varphi_j(x) = \exp(-\frac{1}{2}(x - \mu_j)^2)$, for $j = 1, 2, 3$
2. Construct $\Phi(i, j) = \phi_j(x^{(i)})$, for $j = 1, 2, 3$
3. Draw $\mathbf{w} \sim \mathcal{N}(\mathbf{w}; \mathbf{0}, \mathbf{I})$

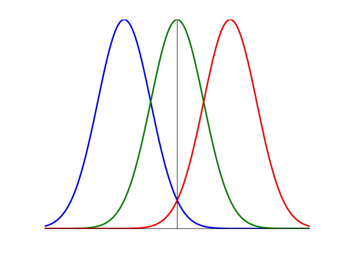
Φ



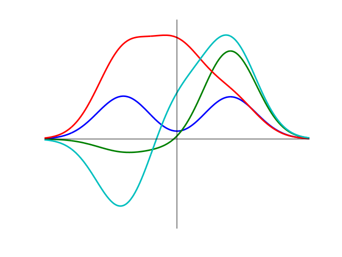
Sample Functions from the Linear Model

1. Define $\varphi_j(x) = \exp(-\frac{1}{2}(x - \mu_j)^2)$, for $j = 1, 2, 3$
2. Construct $\Phi(i, j) = \phi_j(x^{(i)})$, for $j = 1, 2, 3$
3. Draw $\mathbf{w} \sim \mathcal{N}(\mathbf{w}; \mathbf{0}, \mathbf{I})$
4. Draw $\mathbf{f} = \Phi\mathbf{w}$

Φ



$\mathbf{f}$



Gaussian Processes: Function-Space View

Definition

$f(\mathbf{x})$ is distributed according to a Gaussian process iff for any subset $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ the function values $f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(N)})$ follow a Gaussian distribution.

$$f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})\right)$$

$$\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

$$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$$

- $\mu(\mathbf{x})$: mean function, usually we make $\mu(\mathbf{x}) \equiv 0$

Definition
$f(\mathbf{x})$ is distributed according to a Gaussian process iff for any subset $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ the function values $f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(N)})$ follow a Gaussian distribution.

$$f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})\right)$$
$$\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$
$$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$$

- $\mu(\mathbf{x})$: mean function, usually we make $\mu(\mathbf{x}) \equiv 0$
- $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$: parameterized covariance function

Gaussian Processes: Function-Space View

Definition
$f(\mathbf{x})$ is distributed according to a Gaussian process iff for any subset $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ the function values $f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(N)})$ follow a Gaussian distribution.

$$f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})\right)$$
$$\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$
$$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$$

- $\mu(\mathbf{x})$: mean function, usually we make $\mu(\mathbf{x}) \equiv 0$
- $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta})$: parameterized covariance function
- Making $\mathbf{f} \equiv (f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(N)}))^\top$ then $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$

Definition
$f(\mathsf{x})$ is distributed according to a Gaussian process iff for any subset $\{\mathsf{x}^{(1)}, \ldots, \mathsf{x}^{(N)}\}$ the function values $f(\mathsf{x}^{(1)}), \ldots, f(\mathsf{x}^{(N)})$ follow a Gaussian distribution.

$$f(\mathsf{x}) \sim \mathcal{GP}\left(\mu(\mathsf{x}), \kappa(\mathsf{x}, \mathsf{x}'; \boldsymbol{\theta})\right)$$
$$\mu(\mathsf{x}) = \mathbb{E}[f(\mathsf{x})]$$
$$\kappa(\mathsf{x}, \mathsf{x}'; \boldsymbol{\theta}) = \mathbb{E}[(f(\mathsf{x}) - \mu(\mathsf{x}))(f(\mathsf{x}') - \mu(\mathsf{x}'))]$$

- $\mu(\mathsf{x})$: mean function, usually we make $\mu(\mathsf{x}) \equiv 0$
- $\kappa(\mathsf{x}, \mathsf{x}'; \boldsymbol{\theta})$: parameterized covariance function
- Making $\mathbf{f} \equiv (f(\mathsf{x}^{(1)}), \ldots, f(\mathsf{x}^{(N)}))^\top$ then $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathsf{K})$
- Consistency: $(\mathbf{f}_1, \mathbf{f}_2) \sim \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}, \mathsf{K}) \rightarrow \mathbf{f}_1 \sim \mathcal{N}(\mathbf{f}_1; \boldsymbol{\mu}_1, \mathsf{K}_{11})$

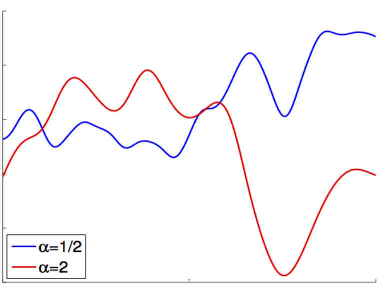
The Covariance Function (Kernel)

- It specifies the covariance between pairs of random variables:

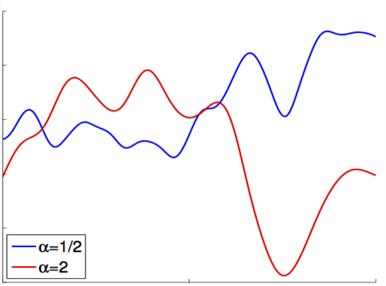
$$\mathbb{C}\text{ov}(f(\mathbf{x}^{(p)}), f(\mathbf{x}^{(q)})) = \kappa(\mathbf{x}^{(p)}, \mathbf{x}^{(q)}; \boldsymbol{\theta})$$

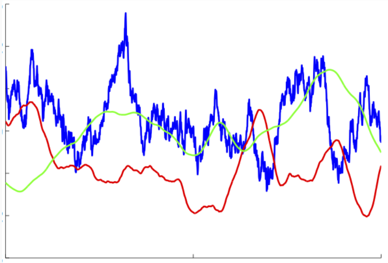
- Notion of similarity
- Let $\mathbf{K}$ be the covariance or Gram matrix, i.e., $K_{i,j} = \kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$
- It must generate a positive semidefinite (PSD) matrix at any subset of points, i.e. $\mathbf{b}^\top \mathbf{K} \mathbf{b} \geq 0, \forall \mathbf{b} \in \mathbb{R}^N$
- Stationary: $\vartheta(\mathbf{x} - \mathbf{x}')$–translation invariant
- Isotropic: $\vartheta(\|\mathbf{x} - \mathbf{x}'\|)$

Samples from a Gaussian Process

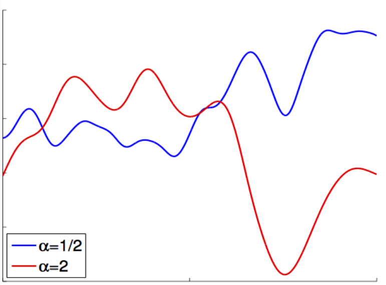


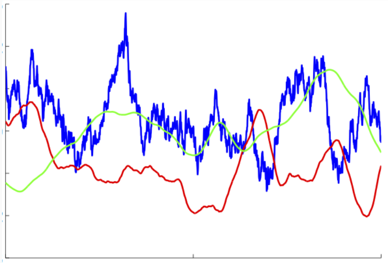
Samples from a Gaussian Process

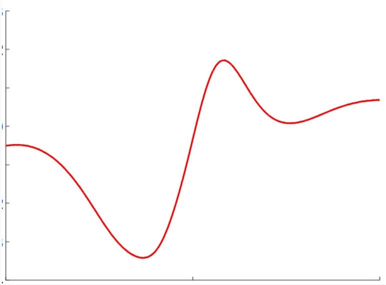




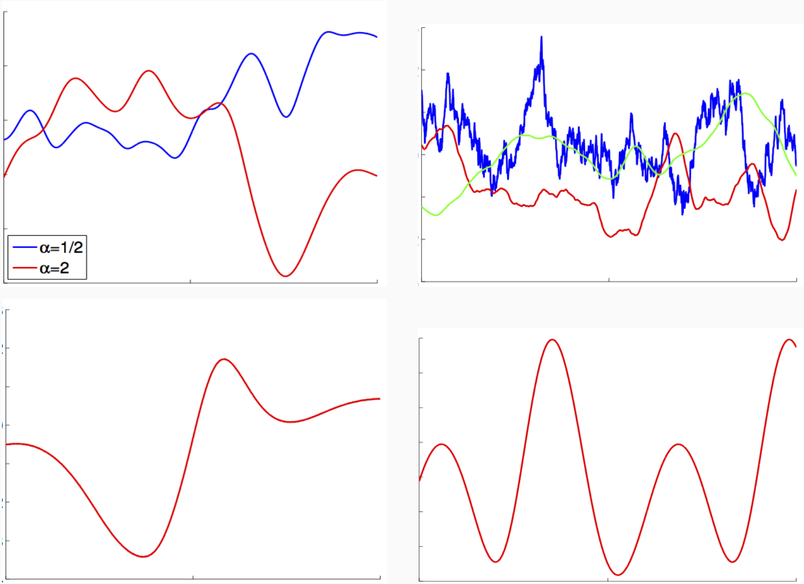
Samples from a Gaussian Process





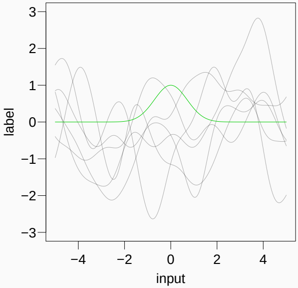


Samples from a Gaussian Process

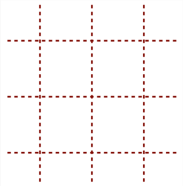


Computing with Infinite Vectors

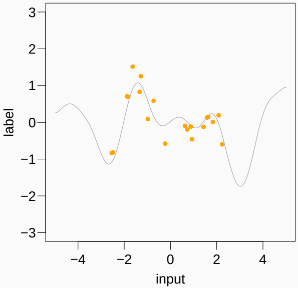
GP prior



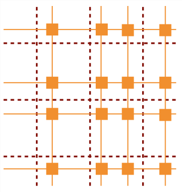
$$K_\infty =$$



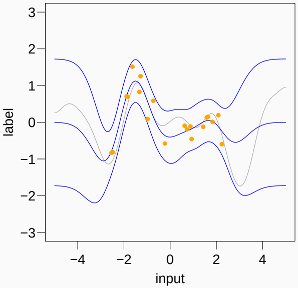
GP regression example



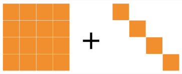
$$K_\infty =$$



Inference result



$$K_y =$$



The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C} (\mathbf{x} - \mathbf{x}')\right)$$

- σ_s^2 is the signal variance

The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C}(\mathbf{x} - \mathbf{x}')\right)$$

- σ_s^2 is the signal variance
- $\mathbf{C}$ is a symmetric matrix that can have different parameterizations

The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C} (\mathbf{x} - \mathbf{x}')\right)$$

- σ_s^2 is the signal variance
- $\mathbf{C}$ is a symmetric matrix that can have different parameterizations
- $\mathbf{C} = \ell^{-2}\mathbf{I}$: isotropic SE

The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C}(\mathbf{x} - \mathbf{x}')\right)$$

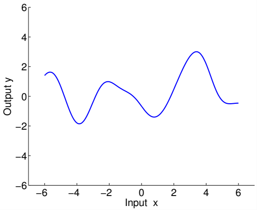
- σ_s^2 is the signal variance
- $\mathbf{C}$ is a symmetric matrix that can have different parameterizations
- $\mathbf{C} = \ell^{-2}\mathbf{I}$: isotropic SE
- $\mathbf{C} = \mathrm{diag}(\boldsymbol{\ell})^{-2}$ with $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_D)$: Automatic Relevance Determination (ARD)

The Squared Exponential (SE) Covariance Function

$$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C} (\mathbf{x} - \mathbf{x}')\right)$$

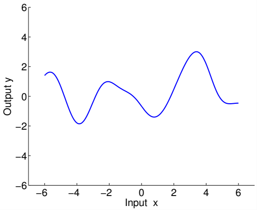
- σ_s^2 is the signal variance
- $\mathbf{C}$ is a symmetric matrix that can have different parameterizations
- $\mathbf{C} = \ell^{-2}\mathbf{I}$: isotropic SE
- $\mathbf{C} = \mathrm{diag}(\boldsymbol{\ell})^{-2}$ with $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_D)$: Automatic Relevance Determination (ARD)
- Each ℓ_j is known as the characteristic length-scale: distance for which the function values are expected to vary significantly

Samples from a GP with a SE Covariance Function

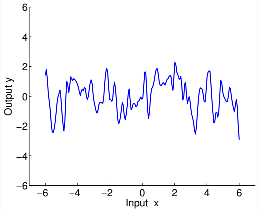


$$\ell = 1, \sigma_s^2 = 1$$

Samples from a GP with a SE Covariance Function

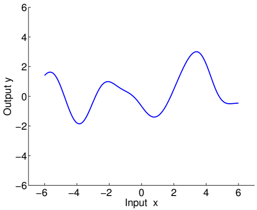


$\ell = 1, \sigma_s^2 = 1$

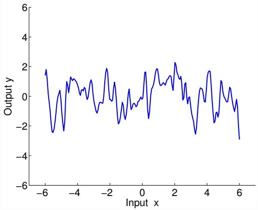


$\ell = 0.1, \sigma_s^2 = 1$

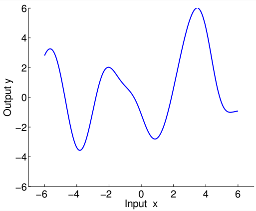
Samples from a GP with a SE Covariance Function



$\ell = 1, \sigma_s^2 = 1$

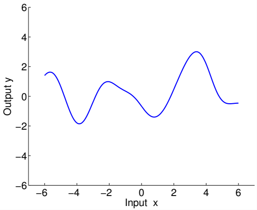


$\ell = 0.1, \sigma_s^2 = 1$

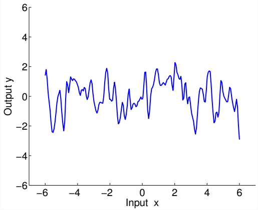


$\ell = 1, \sigma_s^2 = 4$

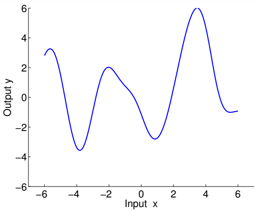
Samples from a GP with a SE Covariance Function



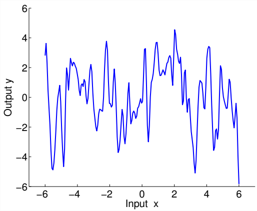
$\ell = 1, \sigma_s^2 = 1$



$\ell = 0.1, \sigma_s^2 = 1$



$\ell = 1, \sigma_s^2 = 4$



$\ell = 0.1, \sigma_s^2 = 4$

Gaussian Processes for Regression

- Data: $\mathcal{D} = \{\mathbf{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$, $\mathbf{x}^{(n)} \in \mathbb{R}^{D_x}$, $y^{(n)} \in \mathbb{R}$
- Inputs : $\mathbf{X} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)})^{\top}$
- Labels : $\mathbf{y} = (y^{(1)}, \ldots, y^{(N)})^{\top}$
- Goal: : $\mathbf{x} \stackrel{f(\mathbf{x})}{\rightarrow} y$
- Prior over latent variables: $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$

Gaussian Processes for Regression

- Data: $\mathcal{D} = \{\mathbf{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$, $\mathbf{x}^{(n)} \in \mathbb{R}^{D_x}$, $y^{(n)} \in \mathbb{R}$
- Inputs : $\mathbf{X} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)})^{\top}$
- Labels : $\mathbf{y} = (y^{(1)}, \ldots, y^{(N)})^{\top}$
- Goal: : $\mathbf{x} \stackrel{f(\mathbf{x})}{\rightarrow} y$

- Prior over latent variables: $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$
- Conditional Likelihood : $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I})$

Gaussian Processes for Regression

- Data: $\mathcal{D} = \{\mathbf{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$, $\mathbf{x}^{(n)} \in \mathbb{R}^{D_x}$, $y^{(n)} \in \mathbb{R}$
- Inputs : $\mathbf{X} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)})^{\top}$
- Labels : $\mathbf{y} = (y^{(1)}, \ldots, y^{(N)})^{\top}$
- Goal: : $\mathbf{x} \stackrel{f(\mathbf{x})}{\rightarrow} y$

- Prior over latent variables: $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$
- Conditional Likelihood : $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I})$
- Marginal likelihood: $p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I})$

Gaussian Processes for Regression

- Data: $\mathcal{D} = \{\mathbf{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$, $\mathbf{x}^{(n)} \in \mathbb{R}^{D_x}$, $y^{(n)} \in \mathbb{R}$
- Inputs : $\mathbf{X} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)})^{\top}$
- Labels : $\mathbf{y} = (y^{(1)}, \ldots, y^{(N)})^{\top}$
- Goal: : $\mathbf{x} \stackrel{f(\mathbf{x})}{\rightarrow} y$

- Prior over latent variables: $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$
- Conditional Likelihood : $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I})$
- Marginal likelihood: $p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I})$
- Predictive distribution: can use Bayes' rule but easily obtained by realizing that the joint over $\mathbf{y}$ and f_* is a Gaussian

GP Regression: Predictive Distribution

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n \mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_* \mathbf{x}_*) \end{matrix}\right)$$

GP Regression: Predictive Distribution

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n \mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_* \mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\mathbf{K}_\mathbf{y} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I}$

GP Regression: Predictive Distribution

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n \mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_*\mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\mathbf{K}_\mathbf{y} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I}$ then:

$$\begin{aligned} p(f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*) &= \mathcal{N}(f_*; \mathbb{E}[f_*], \mathbb{V}[f_*]), \\ \mathbb{E}[f_*] &= \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1}\mathbf{y}, \\ \mathbb{V}[f_*] &= \kappa(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1}\mathbf{k}_*. \end{aligned}$$

GP Regression: Predictive Distribution

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n \mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_* \mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\mathbf{K}_\mathbf{y} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma^2 \mathbf{I}$ then:

$$\begin{aligned} p(f_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*) &= \mathcal{N}(f_*; \mathbb{E}[f_*], \mathbb{V}[f_*]), \\ \mathbb{E}[f_*] &= \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1} \mathbf{y}, \\ \mathbb{V}[f_*] &= \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1} \mathbf{k}_*. \end{aligned}$$

- $\mathbb{E}[f_*]$: Linear combination of N observations, i.e. linear predictor

GP Regression: Predictive Distribution

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n \mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_* \mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\mathbf{K}_\mathbf{y} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma^2 \mathbf{I}$ then:

$$\begin{aligned} p(f_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*) &= \mathcal{N}(f_*; \mathbb{E}[f_*], \mathbb{V}[f_*]), \\ \mathbb{E}[f_*] &= \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1} \mathbf{y}, \\ \mathbb{V}[f_*] &= \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1} \mathbf{k}_*. \end{aligned}$$

- $\mathbb{E}[f_*]$: Linear combination of N observations, i.e. linear predictor
- Say $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$ then $\mathbb{E}[f_*] = \sum_{n=1}^{N} \alpha_i \kappa(\mathbf{x}^{(n)}, \mathbf{x}_*)$

GP Regression: Predictive Distribution

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n \mathbf{I} & \mathbf{k}(\mathbf{X}, \mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*, \mathbf{X}) & \kappa(\mathbf{x}_* \mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X}, \mathbf{x}_*)$ and $\mathbf{K}_\mathbf{y} = \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}$ then:

$$\begin{aligned} p(f_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*) &= \mathcal{N}(f_*; \mathbb{E}[f_*], \mathbb{V}[f_*]), \\ \mathbb{E}[f_*] &= \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1} \mathbf{y}, \\ \mathbb{V}[f_*] &= \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1} \mathbf{k}_*. \end{aligned}$$

- $\mathbb{E}[f_*]$: Linear combination of N observations, i.e. linear predictor
- Say $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$ then $\mathbb{E}[f_*] = \sum_{n=1}^{N} \alpha_i \kappa(\mathbf{x}^{(n)}, \mathbf{x}_*)$
- $\mathbb{V}[f_*]$ does not depend on $\mathbf{y}$

GP Regression: Predictive Distribution

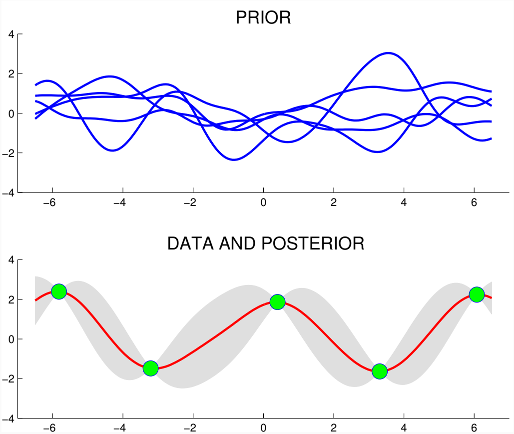
$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n \mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*,\mathbf{X}) & \kappa(\mathbf{x}_* \mathbf{x}_*) \end{matrix}\right)$$

Denoting $\mathbf{k}_* = \mathbf{K}(\mathbf{X},\mathbf{x}_*)$ and $\mathbf{K}_\mathbf{y} = \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma^2 \mathbf{I}$ then:

$$\begin{aligned} p(f_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*) &= \mathcal{N}(f_*; \mathbb{E}[f_*], \mathbb{V}[f_*]), \\ \mathbb{E}[f_*] &= \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1} \mathbf{y}, \\ \mathbb{V}[f_*] &= \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T \mathbf{K}_\mathbf{y}^{-1} \mathbf{k}_*. \end{aligned}$$

- $\mathbb{E}[f_*]$: Linear combination of N observations, i.e. linear predictor
- Say $\boldsymbol{\alpha} = (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}\mathbf{y}$ then $\mathbb{E}[f_*] = \sum_{n=1}^N \alpha_i \kappa(\mathbf{x}^{(n)}, \mathbf{x}_*)$
- $\mathbb{V}[f_*]$ does not depend on $\mathbf{y}$
- In fact we have a Gaussian posterior process

Demo



- Smooth functions
- Closeness in input space $\rightarrow$ closeness in output space

Model Selection

- Covariance function and its parameters (hyper-parameters)

Model Selection

- Covariance function and its parameters (hyper-parameters)
- Likelihood parameters

Model Selection

- Covariance function and its parameters (hyper-parameters)
- Likelihood parameters
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C} (\mathbf{x} - \mathbf{x}')^T\right)$

Model Selection

- Covariance function and its parameters (hyper-parameters)
- Likelihood parameters
- E.g. for the SE: $\kappa(\mathsf{x}, \mathsf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathsf{x} - \mathsf{x}')^T \mathsf{C} (\mathsf{x} - \mathsf{x}')^T\right)$
- Define $\boldsymbol{\theta} = \{\sigma_s^2, \mathsf{C}, \sigma^2\}$

Model Selection

- Covariance function and its parameters (hyper-parameters)
- Likelihood parameters
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C} (\mathbf{x} - \mathbf{x}')^T\right)$
- Define $\boldsymbol{\theta} = \{\sigma_s^2, \mathbf{C}, \sigma^2\}$
- We can do cross-validation (potential problems?)

Model Selection

- Covariance function and its parameters (hyper-parameters)
- Likelihood parameters
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C} (\mathbf{x} - \mathbf{x}')^T\right)$
- Define $\boldsymbol{\theta} = \{\sigma_s^2, \mathbf{C}, \sigma^2\}$
- We can do cross-validation (potential problems?)
- We focus here on the so-called type II maximum likelihood,

Model Selection

- Covariance function and its parameters (hyper-parameters)
- Likelihood parameters
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C}(\mathbf{x} - \mathbf{x}')^T\right)$
- Define $\boldsymbol{\theta} = \{\sigma_s^2, \mathbf{C}, \sigma^2\}$
- We can do cross-validation (potential problems?)
- We focus here on the so-called type II maximum likelihood,
- Integrate out the "parameters" of the GP: (which parameters?)

Model Selection

- Covariance function and its parameters (hyper-parameters)
- Likelihood parameters
- E.g. for the SE: $\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma_s^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{C} (\mathbf{x} - \mathbf{x}')^T\right)$
- Define $\boldsymbol{\theta} = \{\sigma_s^2, \mathbf{C}, \sigma^2\}$
- We can do cross-validation (potential problems?)
- We focus here on the so-called type II maximum likelihood,
- Integrate out the "parameters" of the GP: (which parameters?)

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

Log Marginal Likelihood

$$\mathcal{L} = \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \underbrace{-\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}}_{\text{data-fit}} \underbrace{-\frac{1}{2}\log|\mathbf{K} + \sigma^2\mathbf{I}|}_{\text{complexity}} - \underbrace{\frac{N}{2}\log 2\pi}_{\textit{normaliz.}}$$

Log Marginal Likelihood

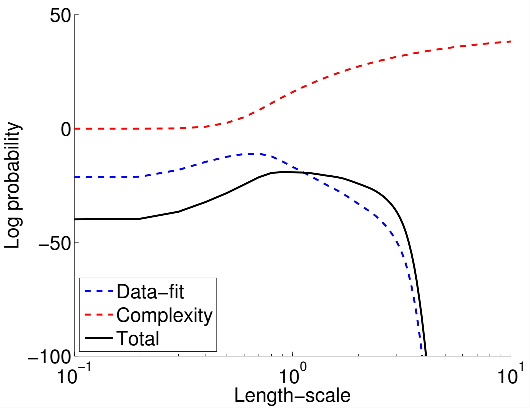
$$\mathcal{L} = \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \underbrace{-\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}}_{\text{data-fit}} \underbrace{-\frac{1}{2}\log|\mathbf{K} + \sigma^2\mathbf{I}|}_{\text{complexity}} - \underbrace{\frac{N}{2}\log 2\pi}_{\textit{normaliz.}}$$

- Isotropic SE
- $\sigma_s^2 = 1, \sigma^2 = 0.01$
- $\ell = 1$
- $N = 20$

Log Marginal Likelihood

$$\mathcal{L} = \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \underbrace{-\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}}_{\text{data-fit}} \underbrace{-\frac{1}{2}\log|\mathbf{K} + \sigma^2\mathbf{I}|}_{\text{complexity}} - \underbrace{\frac{N}{2}\log 2\pi}_{\textit{normaliz.}}$$

- Isotropic SE
- $\sigma_s^2 = 1, \sigma^2 = 0.01$
- $\ell = 1$
- $N = 20$



Hyper-parameter Learning

Let $\mathsf{K_y} = \mathsf{K} + \sigma^2 \mathsf{I}$:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta_i} &= \frac{1}{2} \mathbf{y}^T \mathsf{K_y}^{-1} \frac{\partial \mathsf{K_y}}{\partial \theta_i} \mathsf{K_y}^{-1} \mathbf{y} - \frac{1}{2} \operatorname{tr} \left(\mathsf{K_y}^{-1} \frac{\partial \mathsf{K_y}}{\partial \theta_i} \right) \\
&= \frac{1}{2} \operatorname{tr} \left((\boldsymbol{\alpha} \boldsymbol{\alpha}^T - \mathsf{K_y}^{-1}) \frac{\partial \mathsf{K_y}}{\partial \theta_i} \right)
\end{aligned}
$$

where $\boldsymbol{\alpha} = \mathsf{K_y}^{-1} \mathbf{y}$.

- Can use gradient-based optimization

Hyper-parameter Learning

Let $\mathsf{K_y} = \mathsf{K} + \sigma^2 \mathsf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^T \mathsf{K_y}^{-1} \frac{\partial \mathsf{K_y}}{\partial \theta_i} \mathsf{K_y}^{-1}\mathbf{y} - \frac{1}{2}\,\mathrm{tr}\,\left(\mathsf{K_y}^{-1}\frac{\partial \mathsf{K_y}}{\partial \theta_i}\right)$$

$$= \frac{1}{2}\,\mathrm{tr}\,\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \mathsf{K_y}^{-1})\frac{\partial \mathsf{K_y}}{\partial \theta_i}\right)$$

where $\boldsymbol{\alpha} = \mathsf{K_y}^{-1}\mathbf{y}$.

- Can use gradient-based optimization
- General approach and only needs derivatives of the covariance

Hyper-parameter Learning

Let $\mathsf{K_y} = \mathsf{K} + \sigma^2 \mathsf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^T \mathsf{K}_\mathbf{y}^{-1} \frac{\partial \mathsf{K}_\mathbf{y}}{\partial \theta_i} \mathsf{K}_\mathbf{y}^{-1}\mathbf{y} - \frac{1}{2}\,\mathrm{tr}\,\left(\mathsf{K}_\mathbf{y}^{-1}\frac{\partial \mathsf{K}_\mathbf{y}}{\partial \theta_i}\right)$$
$$= \frac{1}{2}\,\mathrm{tr}\,\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \mathsf{K}_\mathbf{y}^{-1})\frac{\partial \mathsf{K}_\mathbf{y}}{\partial \theta_i}\right)$$

where $\boldsymbol{\alpha} = \mathsf{K}_\mathbf{y}^{-1}\mathbf{y}$.

- Can use gradient-based optimization
- General approach and only needs derivatives of the covariance
- Non-convex optimization

Hyper-parameter Learning

Let $\mathsf{K_y} = \mathsf{K} + \sigma^2\mathsf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathbf{y}^T \mathsf{K_y}^{-1} \frac{\partial \mathsf{K_y}}{\partial \theta_i} \mathsf{K_y}^{-1}\mathbf{y} - \frac{1}{2}\,\mathrm{tr}\,\left(\mathsf{K_y}^{-1}\frac{\partial \mathsf{K_y}}{\partial \theta_i}\right)$$
$$= \frac{1}{2}\,\mathrm{tr}\,\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \mathsf{K_y}^{-1})\frac{\partial \mathsf{K_y}}{\partial \theta_i}\right)$$

where $\boldsymbol{\alpha} = \mathsf{K_y}^{-1}\mathbf{y}$.

- Can use gradient-based optimization
- General approach and only needs derivatives of the covariance
- Non-convex optimization
- Multiple local optima $\rightarrow$ different explanations of the data

Hyper-parameter Learning

Let $\mathsf{K_y} = \mathsf{K} + \sigma^2 \mathsf{I}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2} \mathbf{y}^T \mathsf{K}_\mathbf{y}^{-1} \frac{\partial \mathsf{K}_\mathbf{y}}{\partial \theta_i} \mathsf{K}_\mathbf{y}^{-1} \mathbf{y} - \frac{1}{2} \operatorname{tr} \left(\mathsf{K}_\mathbf{y}^{-1} \frac{\partial \mathsf{K}_\mathbf{y}}{\partial \theta_i} \right)$$
$$= \frac{1}{2} \operatorname{tr} \left((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \mathsf{K}_\mathbf{y}^{-1}) \frac{\partial \mathsf{K}_\mathbf{y}}{\partial \theta_i} \right)$$

where $\boldsymbol{\alpha} = \mathsf{K}_\mathbf{y}^{-1} \mathbf{y}$.

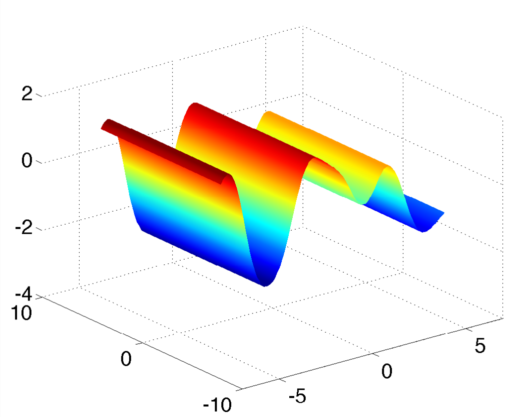
- Can use gradient-based optimization
- General approach and only needs derivatives of the covariance
- Non-convex optimization
- Multiple local optima $\rightarrow$ different explanations of the data
- Computational cost?

Automatic Relevance Determination (ARD)

- Inverse of the length-scale $\rightarrow$ relevance of the dimension.

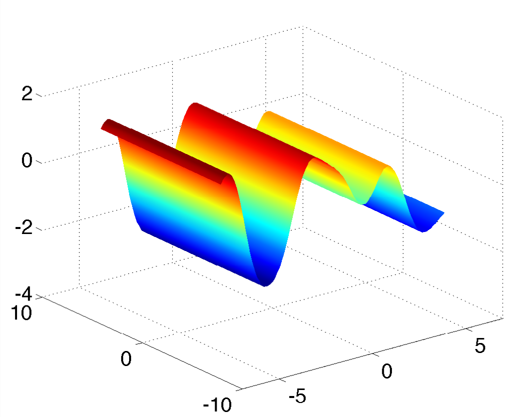
Automatic Relevance Determination (ARD)

- Inverse of the length-scale → relevance of the dimension.



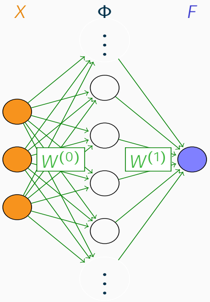
Automatic Relevance Determination (ARD)

- Inverse of the length-scale $\rightarrow$ relevance of the dimension.



Learned lengh-scale for irrelevant dimension: 1.0557×10^5

Gaussian Processes as Infinitely-Wide Shallow Neural Nets

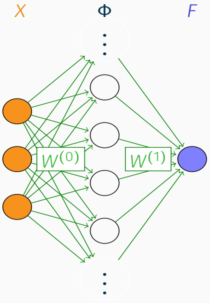


- Take $W^{(i)} \sim \mathcal{N}(\mathbf{0}, \alpha_i I)$
- Central Limit Theorem implies that **f** is Gaussian

- **f** has zero-mean
- $\mathrm{cov}(\mathbf{f}) = \mathbb{E}_{p(W^{(0)}, W^{(1)})}[\Phi(XW^{(0)})W^{(1)}W^{(1)\top}\Phi(XW^{(0)})^\top]$

Neal, *LNS*, 1996

Gaussian Processes as Infinitely-Wide Shallow Neural Nets



- Take $W^{(i)} \sim \mathcal{N}(\mathbf{0}, \alpha_i I)$
- Central Limit Theorem implies that $\mathbf{f}$ is Gaussian
- $\mathbf{f}$ has zero-mean
- $\mathrm{cov}(\mathbf{f}) = \alpha_1 \mathbb{E}_{p(W^{(0)})}[\Phi(XW^{(0)})\Phi(XW^{(0)})^\top]$
- Some choices of Φ lead to analytic expression of known kernels (RBF, Matérn, arc-cosine, Brownian motion, ...)

Neal, *LNS*, 1996

Challenges

Challenges

- Non-Gaussian Likelihoods?
- Scalability?
- Kernel design?

Marginal likelihood of GP models: non-Gaussian case

- Marginal likelihood

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

  can be computed analytically if $p(\mathbf{y}|\mathbf{f})$ is Gaussian
- What if $p(\mathbf{y}|\mathbf{f})$ is **not** Gaussian?

Scalability

- Marginal likelihood

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

  can be computed analytically if $p(\mathbf{y}|\mathbf{X}, \mathbf{f})$ is Gaussian

- … even then

$$\log[p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})] = -\frac{1}{2}\log|\mathbf{K_y}| - \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K_y}^{-1}\mathbf{y} + \mathrm{const.}$$

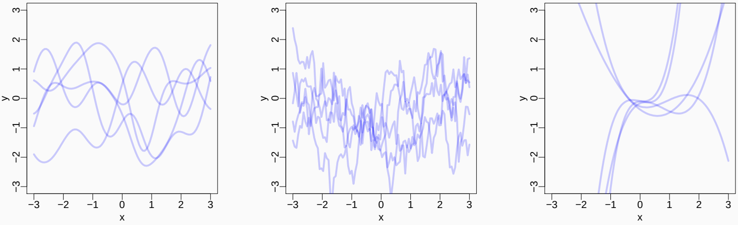
  where $\mathbf{K_y} = \mathbf{K}(\mathbf{X}, \boldsymbol{\theta})$ is a $N \times N$ dense matrix!

- Complexity of exact method is $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ space!

Kernel Design

- The choice of a kernel is critical for good performance
- This encodes any assumptions on the prior over functions

RBF Matérn Polynomial



Model Approximations

Bochner's theorem

- Continuous shift-invariant covariance function

$$k(\mathbf{x}_i - \mathbf{x}_j|\boldsymbol{\theta}) = \sigma^2 \int p(\boldsymbol{\omega}|\boldsymbol{\theta}) \exp\left(\iota(\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\omega}\right) d\boldsymbol{\omega}$$

Bochner's theorem

- Continuous shift-invariant covariance function

$$k(\mathbf{x}_i - \mathbf{x}_j|\boldsymbol{\theta}) = \sigma^2 \int p(\boldsymbol{\omega}|\boldsymbol{\theta}) \exp\left(\iota(\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\omega}\right) d\boldsymbol{\omega}$$

- Monte Carlo estimate

$$k(\mathbf{x}_i - \mathbf{x}_j|\boldsymbol{\theta}) \approx \frac{\sigma^2}{N_{\mathrm{RF}}} \sum_{r=1}^{N_{\mathrm{RF}}} \mathbf{z}(\mathbf{x}_i|\tilde{\boldsymbol{\omega}}_r)^\top \mathbf{z}(\mathbf{x}_j|\tilde{\boldsymbol{\omega}}_r)$$

with

$$\tilde{\boldsymbol{\omega}}_r \sim p(\boldsymbol{\omega}|\boldsymbol{\theta})$$

$$\mathbf{z}(\mathbf{x}|\boldsymbol{\omega}) = [\cos(\mathbf{x}^\top \boldsymbol{\omega}), \sin(\mathbf{x}^\top \boldsymbol{\omega})]^\top$$

Rahimi and Recht, *NIPS*, 2008 - Lázaro-Gredilla et al., *JMLR*, 2010

GPs with Random Fourier Features

- Define

$$\Phi = \sqrt{\frac{\sigma^2}{N_{\mathrm{RF}}}} [\cos(X\Omega), \sin(X\Omega)]$$

  and

$$f = \Phi w$$

- GPs become Bayesian linear models with

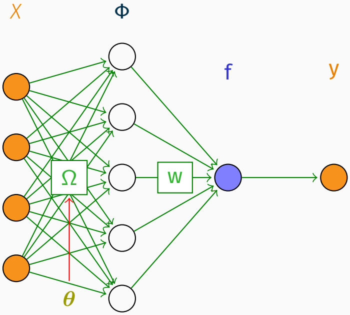
$$p(w) = \mathcal{N}(0, I)$$

- Low-rank approximation of K

$$\mathrm{cov}(f) = \mathbb{E}[\Phi w w^\top \Phi^\top] = \Phi\Phi^\top \approx K$$

GPs with Random Features become Bayesian Linear Models

- Neural Network-like diagram



Low-Rank Approximations

- Marginal likelihood GP regression:

$$-\frac{1}{2}\log|\mathsf{K}_\mathsf{y}| - \frac{1}{2}\mathsf{y}^\top \mathsf{K}_\mathsf{y}^{-1}\mathsf{y} + \text{const.}$$

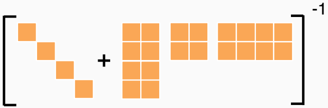
- Most GP approximations aim to form a low-rank approximation to the covariance matrix

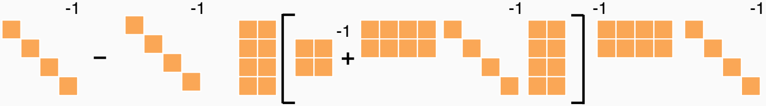
$$\mathsf{K}_\mathsf{y} = \mathsf{K} + \sigma^2 \mathsf{I} \approx \mathsf{UCV} + \sigma^2 \mathsf{I}$$

Low-Rank Approximations

- Woodbury identity for the inverse

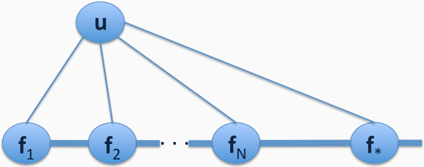
$$(\mathsf{A}+\mathsf{UCV})^{-1}=\mathsf{A}^{-1}-\mathsf{A}^{-1}\mathsf{U}(\mathsf{C}^{-1}+\mathsf{VA}^{-1}\mathsf{U})^{-1}\mathsf{VA}^{-1}$$





- Similar for the log-determinant
- This reduces complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(M^3)+\mathcal{O}(NM^2)$ with $M \ll N$

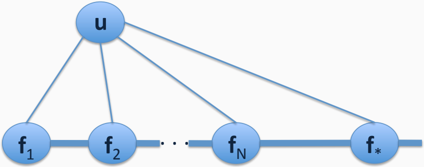
GP Approximations: A Unifying Framework (1)



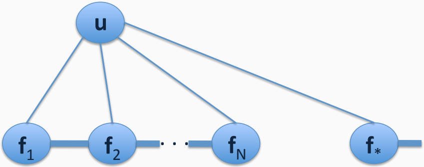
Exact GP. All latent functions are fully connected.

Quiñonero-Candela and Rasmussen (JMLR 2005)

GP Approximations: A Unifying Framework (1)



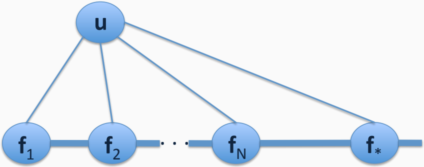
Exact GP. All latent functions are fully connected.



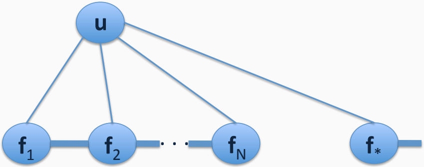
Training and test are cond. independent given **u**

Quiñonero-Candela and Rasmussen (JMLR 2005)

GP Approximations: A Unifying Framework (1)



Exact GP. All latent functions are fully connected.



Training and test are cond. independent given $\mathbf{u}$

- Joint prior augmented with inducing variables $\mathbf{u} = \{u_j\}_{j=1}^{M}$
- which are indexed by the inducing inputs $\mathbf{Z} = \{\mathbf{z}^{(j)}\}_{j=1}^{M}$
- Let $p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{ZZ}})$, where $\mathbf{K}_{\mathbf{ZZ}} = \kappa(\mathbf{Z}, \mathbf{Z}; \boldsymbol{\theta})$ then

$$p(\mathbf{f}_*, \mathbf{f}) = \int p(\mathbf{f}_*, \mathbf{f}|\mathbf{u})p(\mathbf{u})d\mathbf{u}$$

Quiñonero-Candela and Rasmussen (JMLR 2005)

Sparse GP Approximations: A Unifying Framework (2)

We now approximate:

†Snelson and Ghahramani, *NIPS*, 2005

Sparse GP Approximations: A Unifying Framework (2)

We now approximate:

$$p(\mathbf{f}_*, \mathbf{f}) \approx q(\mathbf{f}_*, \mathbf{f}) \stackrel{\text{def}}{=} \int q(\mathbf{f}_*|\mathbf{u}) q(\mathbf{f}|\mathbf{u}) p(\mathbf{u}) d\mathbf{u}$$

† Snelson and Ghahramani, *NIPS*, 2005

Sparse GP Approximations: A Unifying Framework (2)

We now approximate:

$$p(\mathbf{f}_*, \mathbf{f}) \approx q(\mathbf{f}_*, \mathbf{f}) \stackrel{\text{def}}{=} \int q(\mathbf{f}_*|\mathbf{u})q(\mathbf{f}|\mathbf{u})p(\mathbf{u})d\mathbf{u}$$

$q(\mathbf{f}|\mathbf{u})$ is the training conditional and $q(\mathbf{f}_*|\mathbf{u})$ is the test conditional.

† Snelson and Ghahramani, *NIPS*, 2005

Sparse GP Approximations: A Unifying Framework (2)

We now approximate:

$$p(\mathbf{f}_*, \mathbf{f}) \approx q(\mathbf{f}_*, \mathbf{f}) \stackrel{\text{def}}{=} \int q(\mathbf{f}_*|\mathbf{u}) q(\mathbf{f}|\mathbf{u}) p(\mathbf{u}) d\mathbf{u}$$

$q(\mathbf{f}|\mathbf{u})$ is the training conditional and $q(\mathbf{f}_*|\mathbf{u})$ is the test conditional.

Most approximation methods can be defined by:

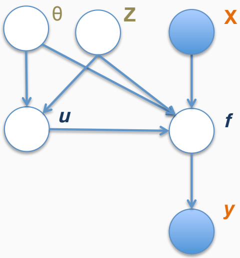
- Different specifications of these conditionals.
- Different $\mathbf{Z}$: Subset of training/test points, new $\mathbf{x}$ points
- Learn inducing inputs by (approx.) marginal likelihood optimization[†]

[†] Snelson and Ghahramani, *NIPS*, 2005

Sparse GPs and the Nyström Approximation

- Introduce M pseudo-inputs collected in Z ...
- ... and corresponding inducing variables $\mathbf{u}$
- Nyström approximation

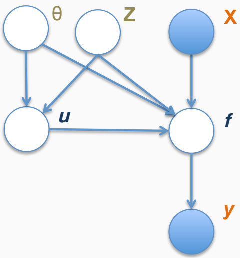
$$\mathrm{K} \approx \mathrm{K}_{XZ}\mathrm{K}_{ZZ}^{-1}\mathrm{K}_{ZX}$$



Sparse GPs and the Nyström Approximation

- Introduce M pseudo-inputs collected in $\mathbf{Z}$ ...
- ... and corresponding inducing variables $\mathbf{u}$
- Nyström approximations with diagonal correction

$$\mathbf{K} \approx \operatorname{diag}(\mathbf{K} - \mathbf{K}_{\mathbf{X}\mathbf{Z}}\mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z}\mathbf{X}}) + \mathbf{K}_{\mathbf{X}\mathbf{Z}}\mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1}\mathbf{K}_{\mathbf{Z}\mathbf{X}}$$



Structured Inputs

- Inputs lie on a regular 1D grid
- $\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \kappa(\mathbf{x}^{(i)} - \mathbf{x}^{(j)})$
- $\mathbf{K}$ is Toeplitz

$$\mathbf{K} = \begin{pmatrix} a & b & c & d \\ b & a & b & c \\ c & b & a & b \\ d & c & b & a \end{pmatrix}$$

- Solving $\mathbf{K}$ exactly costs $\mathcal{O}(N \log N)$ time!

Saatçi, *Ph.D. Thesis*, 2011

Structured Inputs

- Inputs lie on a regular 1D grid
- $\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \kappa(\mathbf{x}^{(i)} - \mathbf{x}^{(j)})$
- $\mathbf{K}$ can be decomposed is Toeplitz

$$\mathbf{K} = \mathbf{K}_1 \otimes \ldots \otimes \mathbf{K}_d$$

  where $\mathbf{K}_p$ has entries $\kappa(x_p^{(i)}, x_p^{(j)})$

- Algebraic operations for $\mathbf{K}$ are based on faster ones for each factor $\mathbf{K}_p$ in the Kronecker product

Saatçi, *Ph.D. Thesis*, 2011

Structured Inducing Points

- Consider a sparse GP:

$$\mathsf{K} \approx \mathsf{K}_{\mathsf{XZ}}\mathsf{K}_{\mathsf{ZZ}}^{-1}\mathsf{K}_{\mathsf{ZX}}$$

- Z on a grid makes the inverse fast (Toeplitz)!
- Can afford $M \gg N$
- Still expensive to deal with $\mathsf{K}_{\mathsf{ZX}} \ldots \mathcal{O}(NM^2)$

Structured Inducing Points

- Consider a sparse GP:

$$\mathbf{K} \approx \mathbf{K}_{XZ}\mathbf{K}_{ZZ}^{-1}\mathbf{K}_{ZX}$$

- Kernel Interpolation (KISS-GP)

$$\mathbf{K}_{XZ} \approx \mathbf{W}\mathbf{K}_{ZZ}$$

with $\mathbf{W}$ a sparse "interpolation" matrix, so that

$$\mathbf{K} \approx \mathbf{K}_{XZ}\mathbf{K}_{ZZ}^{-1}\mathbf{K}_{ZX} \approx \mathbf{W}\mathbf{K}_{ZZ}^{-1}\mathbf{W}^{\top}$$

- All products/inverses are fast even if $M \gg N$!

Wilson, *NIPS*, 2015

Conclusions

- Bayesian linear regression as a Gaussian process
- Gaussian processes as a prior over functions
- Predictions and hyper-parameter learning
- Challenges
  - Non-linear Non-Gaussian likelihoods
  - Scalability, $O(N^3)$
- Inducing variable approximations as a unifying framework
- Structured covariances