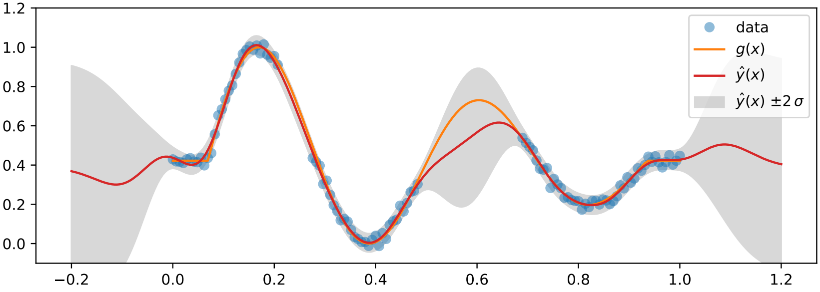
Introduction to Gaussian processes

Steve Schmerler

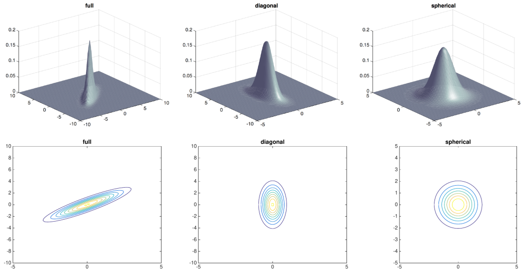
Helmholtz AI @HZDR

Motivation: Why GPs?



- interpolation or regression for low-dimensional problems ("smoothing device")
- **predictive uncertainty**
- building block for Bayesian optimization
- Bayesian stats and Gaussian process (GP) theory: understand uncertainty quantification (UQ) methods for neural networks (NNs)
- infinite width limits of NNs: neural network Gaussian process (NNGP) and the neural tangent kernel (NTK)
- two derivations: weight space, function space

Preliminaries: multivariate normal distribution



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

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \mathrm{cov}[x_1, x_2] \\ \mathrm{cov}[x_1, x_2] & \sigma_2^2 \end{bmatrix}$$

K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

Preliminaries: linear models

Linear model (parametric: $\dim \boldsymbol{w} = D \neq N$, data set content "compressed" into $\boldsymbol{w}$)

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} = w_1\, x_1 + w_2\, x_2 + \cdots$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top [1, \boldsymbol{x}] = w_0 + w_1\, x_1 + w_2\, x_2 + \cdots$$

Only regression models of the form

$$f : \mathbb{R}^D \to \mathbb{R}$$

Preliminaries: linear models

Linear model (parametric: $\dim \boldsymbol{w} = D \neq N$, data set content "compressed" into $\boldsymbol{w}$)

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} = w_1\, x_1 + w_2\, x_2 + \cdots$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top [1, \boldsymbol{x}] = w_0 + w_1\, x_1 + w_2\, x_2 + \cdots$$

Only regression models of the form

$$f : \mathbb{R}^D \to \mathbb{R}$$

Data set

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N = (\mathbf{X}, \boldsymbol{y})$$
$$\boldsymbol{x}_i \in \mathcal{X} = \mathbb{R}^D$$
$$y_i \in \mathcal{Y} = \mathbb{R}$$
$$\mathbf{X} \in \mathbb{R}^{N \times D}$$

Design matrix

$$\mathbf{X} = \overbrace{\begin{bmatrix} - & \boldsymbol{x}_1^\top & - \\ - & \boldsymbol{x}_2^\top & - \\ & \vdots & \\ - & \boldsymbol{x}_N^\top & - \end{bmatrix}}^{D} \in \mathbb{R}^{N \times D}$$

Preliminaries: linear models

Linear model (parametric: $\dim \boldsymbol{w} = D \neq N$, data set content "compressed" into $\boldsymbol{w}$)

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} = w_1\, x_1 + w_2\, x_2 + \cdots$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top [1, \boldsymbol{x}] = w_0 + w_1\, x_1 + w_2\, x_2 + \cdots$$

Only regression models of the form

$$f : \mathbb{R}^D \to \mathbb{R}$$

Notation

(noisy) data/target/label	y
model output (train)	$f = \boldsymbol{w}^\top \boldsymbol{x}$, $\boldsymbol{f} = \mathbf{X}\,\boldsymbol{w}$
model output (test)	$f_* = \boldsymbol{w}^\top \boldsymbol{x}_*$, $\boldsymbol{f}_* = \mathbf{X}_*\,\boldsymbol{w}$

Data set

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N = (\mathbf{X}, \boldsymbol{y})$$

$$\boldsymbol{x}_i \in \mathcal{X} = \mathbb{R}^D$$

$$y_i \in \mathcal{Y} = \mathbb{R}$$

$$\mathbf{X} \in \mathbb{R}^{N \times D}$$

Design matrix

$$\mathbf{X} = \overbrace{\begin{bmatrix} - & \boldsymbol{x}_1^\top & - \\ - & \boldsymbol{x}_2^\top & - \\ & \vdots & \\ - & \boldsymbol{x}_N^\top & - \end{bmatrix}}^{D} \in \mathbb{R}^{N \times D}$$

Basis functions

Feature space mapping

$$\boldsymbol{\phi} : \mathcal{X} \to \mathcal{F}$$
$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})$$

$f(\boldsymbol{x})$ is nonlinear in $\boldsymbol{x}$ but still linear in $\boldsymbol{w}$

$$\mathbf{X} = \begin{bmatrix} - \boldsymbol{x}_1^\top - \\ - \boldsymbol{x}_2^\top - \\ \vdots \\ - \boldsymbol{x}_N^\top - \end{bmatrix} \to \boldsymbol{\Phi} = \begin{bmatrix} - \boldsymbol{\phi}(\boldsymbol{x}_1)^\top - \\ - \boldsymbol{\phi}(\boldsymbol{x}_2)^\top - \\ \vdots \\ - \boldsymbol{\phi}(\boldsymbol{x}_N)^\top - \end{bmatrix}$$

Basis functions

Feature space mapping

$$\boldsymbol{\phi} : \mathcal{X} \to \mathcal{F}$$
$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})$$

$f(\boldsymbol{x})$ is nonlinear in $\boldsymbol{x}$ but still linear in $\boldsymbol{w}$

$$\mathbf{X} = \begin{bmatrix} \text{—}\ \boldsymbol{x}_1^\top\ \text{—} \\ \text{—}\ \boldsymbol{x}_2^\top\ \text{—} \\ \vdots \\ \text{—}\ \boldsymbol{x}_N^\top\ \text{—} \end{bmatrix} \to \boldsymbol{\Phi} = \begin{bmatrix} \text{—}\ \boldsymbol{\phi}(\boldsymbol{x}_1)^\top\ \text{—} \\ \text{—}\ \boldsymbol{\phi}(\boldsymbol{x}_2)^\top\ \text{—} \\ \vdots \\ \text{—}\ \boldsymbol{\phi}(\boldsymbol{x}_N)^\top\ \text{—} \end{bmatrix}$$

Example: polynomial basis: $\boldsymbol{x} \in \mathbb{R}^2$, $\mathcal{F} = \mathbb{R}^5$, $\boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^5$

$$\boldsymbol{\phi}(\boldsymbol{x}) = [1, x_1, x_2, x_1^2, x_2^2]$$
$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) = w_0 + w_1\, x_1 + w_2\, x_2 + w_3\, x_1^2 + w_4\, x_2^2$$

`sklearn.preprocessing.PolynomialFeatures`

Kernels

Kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as similarity measure

- symmetric: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\boldsymbol{x}_j, \boldsymbol{x}_i)$
- positive: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$

Kernels

Kernel function $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ as similarity measure

- symmetric: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\boldsymbol{x}_j, \boldsymbol{x}_i)$
- positive: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$

Gram matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$

$$K_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\mathbf{K} := \kappa(\mathbf{X}, \mathbf{X})$$

Kernels

Kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as similarity measure

- symmetric: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\boldsymbol{x}_j, \boldsymbol{x}_i)$
- positive: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$

Gram matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$

$$K_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$\mathbf{K} := \kappa(\mathbf{X}, \mathbf{X})$$

"Kernel trick"

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left\langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \right\rangle \equiv \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j)$$

Rich theory (Reproducing kernel Hilbert space, Mercer's theorem, ...): no need to define $\boldsymbol{\phi}$ explicitly, sufficient to define $\kappa(\cdot, \cdot)$, for certain κ we have $f(\boldsymbol{x}) = \sum_{i=1}^{\infty} w_i \, \phi_i(\boldsymbol{x})$

Kernels

Kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as similarity measure

- symmetric: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\boldsymbol{x}_j, \boldsymbol{x}_i)$
- positive: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$

Gram matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$

$$K_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$
$$\mathbf{K} := \kappa(\mathbf{X}, \mathbf{X})$$

"Kernel trick"

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left\langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \right\rangle \equiv \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j)$$

Rich theory (Reproducing kernel Hilbert space, Mercer's theorem, ...): no need to define $\boldsymbol{\phi}$ explicitly, sufficient to define $\kappa(\cdot, \cdot)$, for certain κ we have $f(\boldsymbol{x}) = \sum_{i=1}^{\infty} w_i \, \phi_i(\boldsymbol{x})$

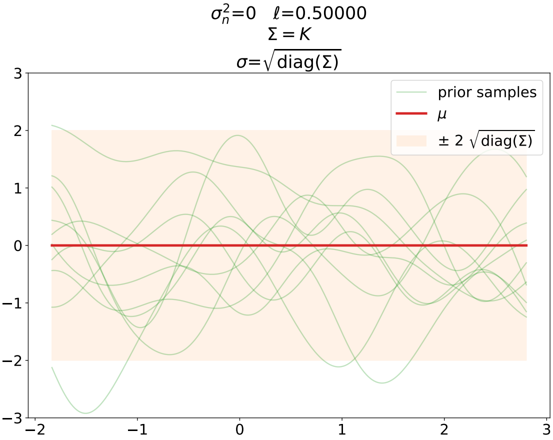
$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j$$

Linear / dot product kernel

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\,\ell^2} \right) = \begin{cases} 1 & \boldsymbol{x}_i = \boldsymbol{x}_j \\ < 1 & \text{else} \end{cases}$$

Gaussian/RBF/"squared exponential" kernel, characteristic length scale ℓ

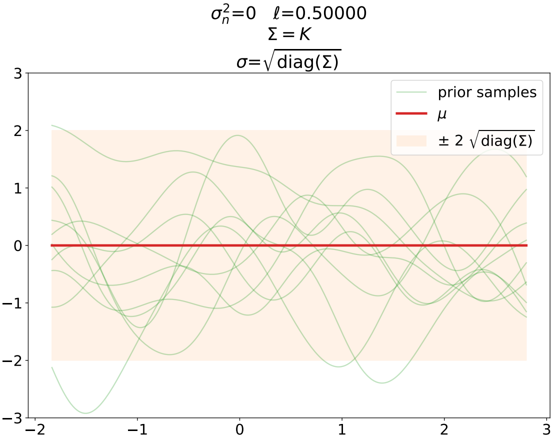
The GP prior for the RBF kernel, fixed ℓ



$1D$ example where

$$\boldsymbol{x} = x \in \mathbb{R}, \mathbf{X} \in \mathbb{R}^{N \times 1}$$

The GP prior for the RBF kernel, fixed ℓ



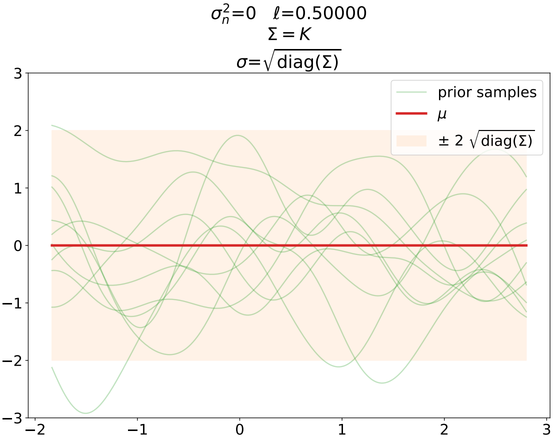
$1D$ example where

$$\boldsymbol{x} = x \in \mathbb{R}, \mathbf{X} \in \mathbb{R}^{N\times 1}$$

GP prior for model $f = f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})$

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{w}}) \quad \text{weight prior}$$

The GP prior for the RBF kernel, fixed ℓ



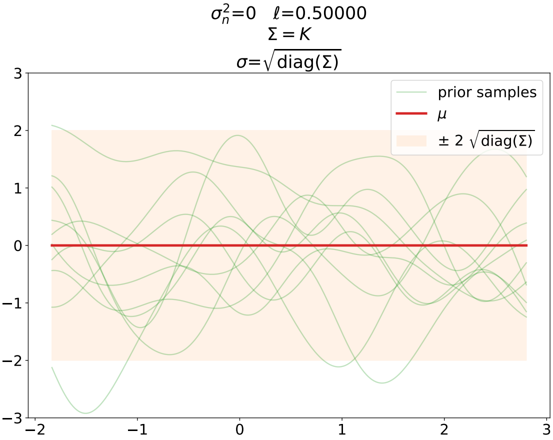
$1D$ example where

$$\boldsymbol{x} = x \in \mathbb{R}, \mathbf{X} \in \mathbb{R}^{N\times 1}$$

GP prior for model $f = f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})$

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{w}}) \quad \text{weight prior}$$
$$p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{0}, \mathbf{K})$$

The GP prior for the RBF kernel, fixed ℓ



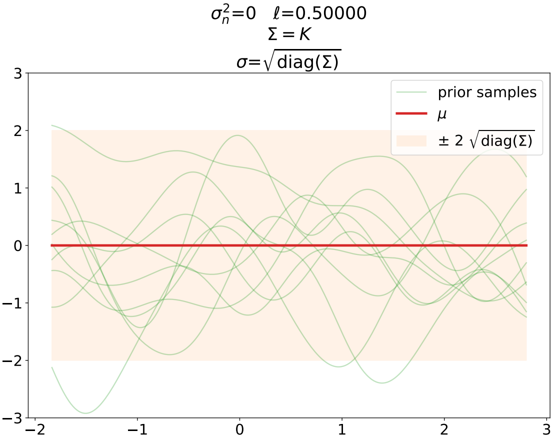
$1D$ example where

$$\boldsymbol{x} = x \in \mathbb{R}, \mathbf{X} \in \mathbb{R}^{N\times 1}$$

GP prior for model $f = f(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{w}}) \quad \text{weight prior}$$
$$p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{0}, \mathbf{K})$$
$$\mathbb{E}[\boldsymbol{f}] = \mathbb{E}[\boldsymbol{\Phi}\, \boldsymbol{w}] = \boldsymbol{\Phi}\, \mathbb{E}[\boldsymbol{w}] = \boldsymbol{0}$$

The GP prior for the RBF kernel, fixed ℓ



$1D$ example where

$$\boldsymbol{x} = x \in \mathbb{R}, \mathbf{X} \in \mathbb{R}^{N\times 1}$$

GP prior for model $f = f(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$

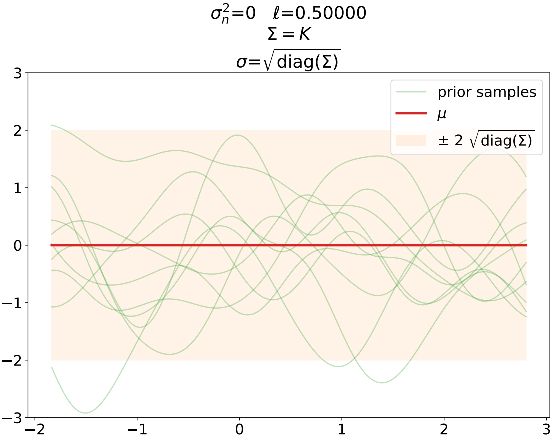
$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{w}}) \quad \text{weight prior}$$

$$p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{0}, \mathbf{K})$$

$$\mathbb{E}[\boldsymbol{f}] = \mathbb{E}[\boldsymbol{\Phi}\, \boldsymbol{w}] = \boldsymbol{\Phi}\, \mathbb{E}[\boldsymbol{w}] = \boldsymbol{0}$$

$$\mathrm{cov}[\boldsymbol{f}] = \mathbb{E}[(\boldsymbol{f} - \mathbb{E}[\boldsymbol{f}])\,(\boldsymbol{f} - \mathbb{E}[\boldsymbol{f}])^\top] = \boldsymbol{\Phi}\, \boldsymbol{\Sigma}_{\boldsymbol{w}}\, \boldsymbol{\Phi}^\top =: \mathbf{K}$$

The GP prior for the RBF kernel, fixed ℓ



$1D$ example where

$$\boldsymbol{x} = x \in \mathbb{R}, \mathbf{X} \in \mathbb{R}^{N\times 1}$$

GP prior for model $f = f(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{w}}) \quad \text{weight prior}$$

$$p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{0}, \mathbf{K})$$

$$\mathbb{E}[\boldsymbol{f}] = \mathbb{E}[\boldsymbol{\Phi}\, \boldsymbol{w}] = \boldsymbol{\Phi}\, \mathbb{E}[\boldsymbol{w}] = \boldsymbol{0}$$

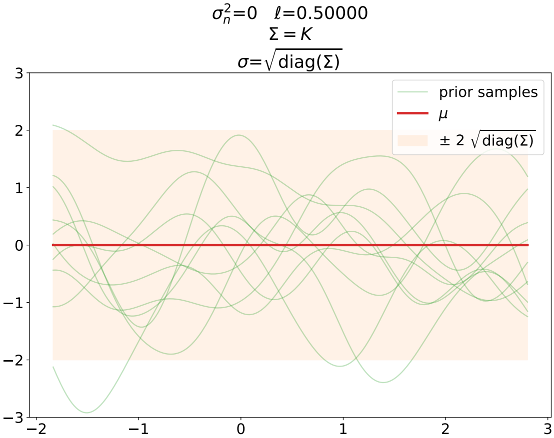
$$\mathrm{cov}[\boldsymbol{f}] = \mathbb{E}[(\boldsymbol{f} - \mathbb{E}[\boldsymbol{f}])\, (\boldsymbol{f} - \mathbb{E}[\boldsymbol{f}])^\top] = \boldsymbol{\Phi}\, \boldsymbol{\Sigma}_{\boldsymbol{w}}\, \boldsymbol{\Phi}^\top =: \mathbf{K}$$

Covariance (kernel) function $\kappa(\cdot, \cdot)$

$$K_{ij} = \phi(\boldsymbol{x}_i)^\top\, \boldsymbol{\Sigma}_{\boldsymbol{w}}\, \phi(\boldsymbol{x}_j) =: \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

e.g. $\boldsymbol{\Sigma}_{\boldsymbol{w}} = \tau^2\, \mathbf{I}_D \rightarrow$ scaling factor in κ

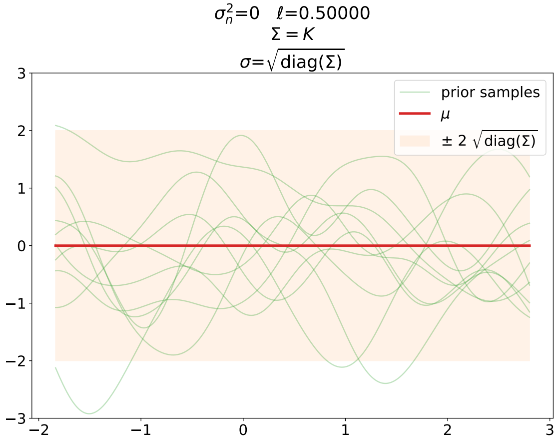
Function space view: the GP prior



The GP as a distribution over *functions* f

$$f \sim \mathcal{GP}(m(\cdot), \kappa(\cdot, \cdot))$$

Function space view: the GP prior



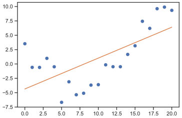
The GP as a distribution over *functions* f

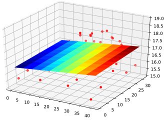
$$f \sim \mathcal{GP}(m(\cdot), \kappa(\cdot, \cdot))$$

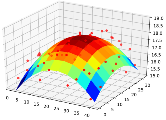
$$p(f_i|\boldsymbol{x}_i) = \mathcal{N}(m(\boldsymbol{x}_i), \kappa(\boldsymbol{x}_i, \boldsymbol{x}_i))$$
$$p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{m}(\mathbf{X}), \mathbf{K})$$
$$\mathbb{E}[f_i] = m(\boldsymbol{x}_i)$$
$$\mathbb{E}[\boldsymbol{f}] = \boldsymbol{m}(\mathbf{X})$$

$$\mathrm{cov}[f_i, f_j] = \mathbb{E}[(f_i - m(\boldsymbol{x}_i))\,(f_j - m(\boldsymbol{x}_j))]$$
$$=: \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$
$$\mathrm{cov}[\boldsymbol{f}] = \mathbf{K}$$

Likelihood







Model noise σ_n^2 in data y.
$p(y|\boldsymbol{x}, \boldsymbol{w})$ interpretation:

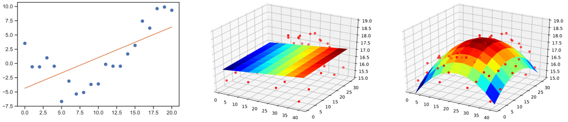
- distribution $p(y|\ldots)$ over y
- *function* of $\boldsymbol{w}$

"The likelihood function reflects the data we expect to see for each setting of the parameters $\boldsymbol{w}$."

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}), \sigma_n^2)$$

K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

Likelihood



Model noise σ_n^2 in data y.
$p(y|\boldsymbol{x}, \boldsymbol{w})$ interpretation:

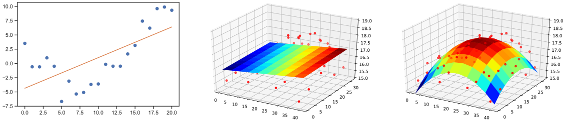
- distribution $p(y| \ldots)$ over y
- *function* of $\boldsymbol{w}$

"The likelihood function reflects the data we expect to see for each setting of the parameters $\boldsymbol{w}$."

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}), \sigma_n^2)$$
$$f = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})$$
$$y = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + \epsilon$$
$$\epsilon \sim \mathcal{N}(0, \sigma_n^2)$$
$$\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{\xi})$$
$$\boldsymbol{\xi} = (\sigma_n^2) \quad \text{hyper parameters}$$

K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

Likelihood



Model noise σ_n^2 in data y.
$p(y|\boldsymbol{x}, \boldsymbol{w})$ interpretation:

- distribution $p(y| \ldots)$ over y
- *function* of $\boldsymbol{w}$

"The likelihood function reflects the data we expect to see for each setting of the parameters $\boldsymbol{w}$."

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}), \sigma_n^2)$$
$$f = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})$$
$$y = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + \epsilon$$
$$\epsilon \sim \mathcal{N}(0, \sigma_n^2)$$
$$\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{\xi})$$
$$\boldsymbol{\xi} = (\sigma_n^2) \quad \text{hyper parameters}$$

$$\begin{aligned} p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) &= \mathcal{N}(\boldsymbol{\Phi}\, \boldsymbol{w}, \sigma_n^2\, \mathbf{I}_N) \\ &= \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) \\ &= \prod_{i=1}^{N} \frac{1}{\sqrt{2\,\pi}\,\sigma_n} \exp\left(-\frac{(y_i - f_i)^2}{2\,\sigma_n^2}\right) \\ &= \frac{1}{\sqrt{(2\,\pi)^N}\,\sigma_n} \exp\left(-\frac{\boldsymbol{\epsilon}^\top \boldsymbol{\epsilon}}{2\,\sigma_n^2}\right) \end{aligned}$$

K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

Bayes' rule

Bayesian inference: infer *posterior* distribution over weights (i.e. models) $p(\boldsymbol{w}|\mathbf{X},\boldsymbol{y})$ by using training data $(\mathbf{X},\boldsymbol{y})$

$$\overbrace{p(\boldsymbol{w}|\mathbf{X},\boldsymbol{y})}^{\text{weight posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\mathbf{X},\boldsymbol{w})}^{\text{likelihood}} \quad \overbrace{p(\boldsymbol{w})}^{\text{weight prior}}}{\underbrace{p(\boldsymbol{y}|\mathbf{X})}_{\text{marginal likelihood or evidence}}} = \frac{p(\boldsymbol{y}|\mathbf{X},\boldsymbol{w})\,p(\boldsymbol{w})}{\int p(\boldsymbol{y}|\mathbf{X},\boldsymbol{w})\,p(\boldsymbol{w})\,\mathrm{d}\boldsymbol{w}}$$

Bayes' rule

Bayesian inference: infer *posterior* distribution over weights (i.e. models) $p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})$ by using training data $(\mathbf{X}, \boldsymbol{y})$

$$\overbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}^{\text{weight posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})}^{\text{likelihood}} \quad \overbrace{p(\boldsymbol{w})}^{\text{weight prior}}}{\underbrace{p(\boldsymbol{y}|\mathbf{X})}_{\text{marginal likelihood or evidence}}} = \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})}{\int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}}$$

More compact notation

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})\, p(\boldsymbol{w})}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\boldsymbol{w})\, p(\boldsymbol{w})}{\int p(\mathcal{D}|\boldsymbol{w})\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}}$$

In simple cases, inference can be performed analytically, e.g. for a Gaussian likelihood.

Posterior predictive distribution

Bayesian model averaging (BMA)

$$\langle w \rangle = \int w\, p(w)\, \mathrm{d}w$$

$$\langle f(w) \rangle = \int f(w)\, p(w)\, \mathrm{d}w$$

Posterior predictive distribution

Bayesian model averaging (BMA)

$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \int \underbrace{p(\boldsymbol{f}_*|\mathbf{X}_*, \boldsymbol{w})}_{\text{likelihood}} \; \underbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}_{\text{weight posterior}} \; \mathrm{d}\boldsymbol{w} = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

$$\langle w \rangle = \int w \, p(w) \, \mathrm{d}w$$

$$\langle f(w) \rangle = \int f(w) \, p(w) \, \mathrm{d}w$$

Posterior predictive distribution

Bayesian model averaging (BMA)

$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \int \underbrace{p(\boldsymbol{f}_*|\mathbf{X}_*, \boldsymbol{w})}_{\text{likelihood}} \; \underbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}_{\text{weight posterior}} \; \mathrm{d}\boldsymbol{w} = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

$$\langle w \rangle = \int w \, p(w) \, \mathrm{d}w$$

$$\langle f(w) \rangle = \int f(w) \, p(w) \, \mathrm{d}w$$

Predictive mean $\boldsymbol{\mu}_*$ and cov. $\boldsymbol{\Sigma}_*$

$$\begin{aligned} \boldsymbol{\mu}_* &= \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2 \, \mathbf{I}_N\right)^{-1} \boldsymbol{y} \\ &= \mathbf{K}_* \, \boldsymbol{\alpha} \end{aligned}$$

$$\mathbf{K}_* = \kappa(\mathbf{X}_*, \mathbf{X})$$
$$\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$$

Posterior predictive distribution

Bayesian model averaging (BMA)

$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \int \underbrace{p(\boldsymbol{f}_*|\mathbf{X}_*, \boldsymbol{w})}_{\text{likelihood}} \underbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}_{\text{weight posterior}} \, \mathrm{d}\boldsymbol{w} = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

$$\langle w \rangle = \int w \, p(w) \, \mathrm{d}w$$

$$\langle f(w) \rangle = \int f(w) \, p(w) \, \mathrm{d}w$$

Predictive mean $\boldsymbol{\mu}_*$ and cov. $\boldsymbol{\Sigma}_*$

$$\begin{aligned} \boldsymbol{\mu}_* &= \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2 \, \mathbf{I}_N\right)^{-1} \boldsymbol{y} \\ &= \mathbf{K}_* \, \boldsymbol{\alpha} \end{aligned}$$

$$\mu_* = \sum_{j=1}^{N} \alpha_j \, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_j)$$

$$\mathbf{K}_* = \kappa(\mathbf{X}_*, \mathbf{X})$$

$$\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$$

Posterior predictive distribution

Bayesian model averaging (BMA)

$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \int \underbrace{p(\boldsymbol{f}_*|\mathbf{X}_*, \boldsymbol{w})}_{\text{likelihood}} \underbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}_{\text{weight posterior}} \,\mathrm{d}\boldsymbol{w} = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

$$\langle w \rangle = \int w\, p(w)\, \mathrm{d}w$$

$$\langle f(w) \rangle = \int f(w)\, p(w)\, \mathrm{d}w$$

Predictive mean $\boldsymbol{\mu}_*$ and cov. $\boldsymbol{\Sigma}_*$

$$\begin{aligned}
\boldsymbol{\mu}_* &= \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2\, \mathbf{I}_N\right)^{-1} \boldsymbol{y} \\
&= \mathbf{K}_* \,\boldsymbol{\alpha}
\end{aligned}$$

$$\mu_* = \sum_{j=1}^{N} \alpha_j\, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_j)$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2\, \mathbf{I}_N\right)^{-1} \mathbf{K}_*^\top$$

$$\mathbf{K}_* = \kappa(\mathbf{X}_*, \mathbf{X})$$

$$\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$$

Posterior predictive distribution

Bayesian model averaging (BMA)

$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \int \underbrace{p(\boldsymbol{f}_*|\mathbf{X}_*, \boldsymbol{w})}_{\text{likelihood}} \; \underbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}_{\text{weight posterior}} \; \mathrm{d}\boldsymbol{w} = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

$$\langle w \rangle = \int w \, p(w) \, \mathrm{d}w$$

$$\langle f(w) \rangle = \int f(w) \, p(w) \, \mathrm{d}w$$

Predictive mean $\boldsymbol{\mu}_*$ and cov. $\boldsymbol{\Sigma}_*$

$$\begin{aligned} \boldsymbol{\mu}_* &= \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2 \, \mathbf{I}_N\right)^{-1} \boldsymbol{y} \\ &= \mathbf{K}_* \, \boldsymbol{\alpha} \end{aligned}$$

$$\mu_* = \sum_{j=1}^{N} \alpha_j \, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_j)$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2 \, \mathbf{I}_N\right)^{-1} \mathbf{K}_*^\top$$

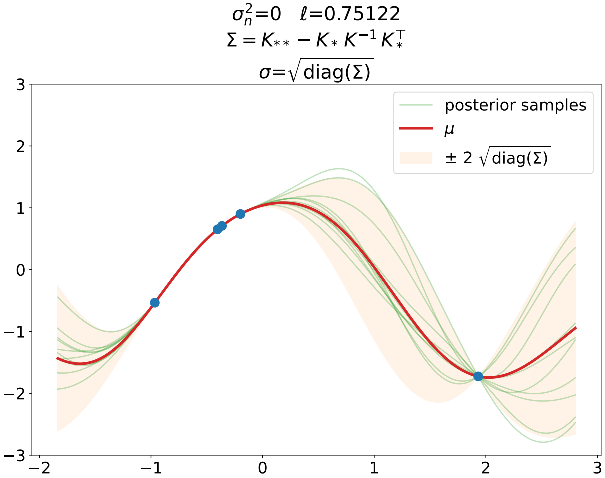
$$\mathbf{K}_* = \kappa(\mathbf{X}_*, \mathbf{X})$$

$$\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$$

Non-parametric model: $\boldsymbol{\mu} = \mathbf{K} \, \boldsymbol{\alpha}$

- $K_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- $\mathbf{K} \in \mathbb{R}^{N \times N}$ contains info about whole training inputs $\mathbf{X} \in \mathbb{R}^{N \times D}$
- weights $\boldsymbol{\alpha} \in \mathbb{R}^N$ contain info about $(\mathbf{X}, \boldsymbol{y})$
- large data sets (large N) make vanilla GPs costly

Posterior predictive with $\sigma_n^2 = 0$



$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

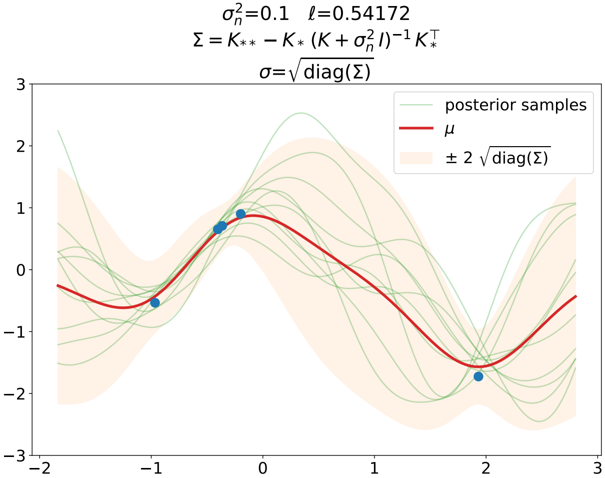
$$\boldsymbol{\mu}_* = \mathbf{K}_* \mathbf{K}^{-1} \boldsymbol{y}$$

$$= \mathbf{K}_* \boldsymbol{\alpha}$$

$$\mu_* = \sum_j \alpha_j \, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_j)$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_* \mathbf{K}^{-1} \mathbf{K}_*^\top$$

Posterior predictive with $\sigma_n^2 > 0$



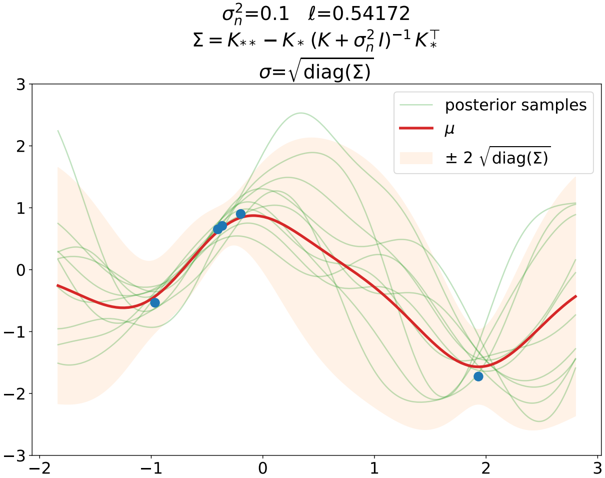
$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

$$\begin{aligned}\boldsymbol{\mu}_* &= \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2 \, \mathbf{I}_N\right)^{-1} \boldsymbol{y} \\ &= \mathbf{K}_* \, \boldsymbol{\alpha}\end{aligned}$$

$$\mu_* = \sum_j \alpha_j \, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_j)$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2 \, \mathbf{I}_N\right)^{-1} \mathbf{K}_*^\top$$

Posterior predictive with $\sigma_n^2 > 0$



$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

$$\boldsymbol{\mu}_* = \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2\, \mathbf{I}_N\right)^{-1} \boldsymbol{y} = \mathbf{K}_* \, \boldsymbol{\alpha}$$

$$\mu_* = \sum_j \alpha_j \, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_j)$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2\, \mathbf{I}_N\right)^{-1} \mathbf{K}_*^\top$$

Data noise σ_n^2: transform interpolation $\rightarrow$ regression, same effect as a regularization term in NN training

Function space view of GPs: the joint

We rewrite the prior: Divide data into "train" $\boldsymbol{f}$ and "test/prediction" $\boldsymbol{f}_*$

$$\begin{aligned} M &= N + N_* \\ \mathbf{X} \in \mathbb{R}^{M \times D} &\rightarrow \left(\mathbf{X} \in \mathbb{R}^{N \times D}, \mathbf{X}_* \in \mathbb{R}^{N_* \times D}\right) \\ \boldsymbol{f} \in \mathbb{R}^{M} &\rightarrow \left(\boldsymbol{f} \in \mathbb{R}^{N}, \boldsymbol{f}_* \in \mathbb{R}^{N_*}\right) \end{aligned}$$

Function space view of GPs: the joint

We rewrite the prior: Divide data into "train" $\boldsymbol{f}$ and "test/prediction" $\boldsymbol{f}_*$

$$\begin{aligned} M &= N + N_* \\ \mathbf{X} \in \mathbb{R}^{M \times D} &\rightarrow \left(\mathbf{X} \in \mathbb{R}^{N \times D}, \mathbf{X}_* \in \mathbb{R}^{N_* \times D}\right) \\ \boldsymbol{f} \in \mathbb{R}^{M} &\rightarrow \left(\boldsymbol{f} \in \mathbb{R}^{N}, \boldsymbol{f}_* \in \mathbb{R}^{N_*}\right) \end{aligned}$$

write the prior as joint over concat. $(\boldsymbol{f}, \boldsymbol{f}_*)$

$$\begin{aligned} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_* \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}(\mathbf{X}) \\ \boldsymbol{m}(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathrm{cov}[\boldsymbol{f}] & \mathbf{K}_*^\top \\ \mathbf{K}_* & \mathrm{cov}[\boldsymbol{f}_*] \end{bmatrix} \right) \\ &\sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}(\mathbf{X}) \\ \boldsymbol{m}(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_*^\top \\ \mathbf{K}_* & \mathbf{K}_{**} \end{bmatrix} \right) \\ &\sim p(\boldsymbol{f}, \boldsymbol{f}_* | \mathbf{X}, \mathbf{X}_*) \end{aligned}$$

Function space view of GPs: the joint

We rewrite the prior: Divide data into "train" $\boldsymbol{f}$ and "test/prediction" $\boldsymbol{f}_*$

$$
\begin{aligned}
M &= N + N_* \\
\mathbf{X} \in \mathbb{R}^{M \times D} &\rightarrow (\mathbf{X} \in \mathbb{R}^{N \times D}, \mathbf{X}_* \in \mathbb{R}^{N_* \times D}) \\
\boldsymbol{f} \in \mathbb{R}^{M} &\rightarrow (\boldsymbol{f} \in \mathbb{R}^{N}, \boldsymbol{f}_* \in \mathbb{R}^{N_*})
\end{aligned}
$$

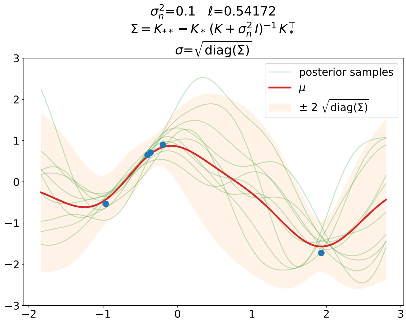
write the prior as joint over concat. $(\boldsymbol{f}, \boldsymbol{f}_*)$

$$
\begin{aligned}
\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_* \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}(\mathbf{X}) \\ \boldsymbol{m}(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathrm{cov}[\boldsymbol{f}] & \mathbf{K}_*^\top \\ \mathbf{K}_* & \mathrm{cov}[\boldsymbol{f}_*] \end{bmatrix} \right) \\
&\sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}(\mathbf{X}) \\ \boldsymbol{m}(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_*^\top \\ \mathbf{K}_* & \mathbf{K}_{**} \end{bmatrix} \right) \\
&\sim p(\boldsymbol{f}, \boldsymbol{f}_* | \mathbf{X}, \mathbf{X}_*)
\end{aligned}
$$

For noisy $y = f + \epsilon$, we have

$$
\begin{aligned}
\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}(\mathbf{X}) \\ \boldsymbol{m}(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathrm{cov}[\boldsymbol{y}] & \mathbf{K}_*^\top \\ \mathbf{K}_* & \mathrm{cov}[\boldsymbol{f}_*] \end{bmatrix} \right) \\
&\sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}(\mathbf{X}) \\ \boldsymbol{m}(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I}_N & \mathbf{K}_*^\top \\ \mathbf{K}_* & \mathbf{K}_{**} \end{bmatrix} \right) \\
&\sim p(\boldsymbol{y}, \boldsymbol{f}_* | \mathbf{X}, \mathbf{X}_*)
\end{aligned}
$$

Function space view of GPs: posterior predictive



Transform the joint $p(\boldsymbol{y}, \boldsymbol{f}_* | \mathbf{X}, \mathbf{X}_*)$ into the posterior predictive $p(\boldsymbol{f}_* | \mathbf{X}_*, \mathbf{X}, \boldsymbol{y})$ by conditioning on $(\mathbf{X}, \boldsymbol{y})$ ("training data").

$$p(\boldsymbol{f}_* | \mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

$$\boldsymbol{\mu}_* = \boldsymbol{m}(\mathbf{X}_*) + \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2 \mathbf{I}_N\right)^{-1} (\boldsymbol{y} - \boldsymbol{m}(\mathbf{X}))$$

$$\boldsymbol{\Sigma}_* = \text{cov}[\boldsymbol{f}_*] = \mathbf{K}_{**} - \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2 \mathbf{I}_N\right)^{-1} \mathbf{K}_*^\top$$

Same result as the posterior predictive obtained from Bayes' rule + model averaging. Here we also have a mean function $m(\cdot) \neq 0$.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

GP hyperparameter optimization

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

Bayes' rule

$$\overbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}^{\text{weight posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})}^{\text{likelihood}}\ \overbrace{p(\boldsymbol{w})}^{\text{weight prior}}}{\underbrace{p(\boldsymbol{y}|\mathbf{X})}_{\text{marginal likelihood or evidence}}}$$

$$= \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})}{\int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}}$$

GP hyperparameter optimization

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

Marginal likelihood as function of hyperparameters $\boldsymbol{\xi} = (\ell, \sigma_n^2)$.

Bayes' rule

$$\overbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}^{\text{weight posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})}^{\text{likelihood}}\ \overbrace{p(\boldsymbol{w})}^{\text{weight prior}}}{\underbrace{p(\boldsymbol{y}|\mathbf{X})}_{\text{marginal likelihood or evidence}}}$$

$$= \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})}{\int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}}$$

GP hyperparameter optimization

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

Marginal likelihood as function of hyperparameters $\boldsymbol{\xi} = (\ell, \sigma_n^2)$. Because of $\boldsymbol{f} = \boldsymbol{\Phi}\, \boldsymbol{w}$, $\int \cdots \mathrm{d}\boldsymbol{w} \to \int \cdots \mathrm{d}\boldsymbol{f}$

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

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

Bayes' rule

$$\overbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}^{\text{weight posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})}^{\text{likelihood}}\ \overbrace{p(\boldsymbol{w})}^{\text{weight prior}}}{\underbrace{p(\boldsymbol{y}|\mathbf{X})}_{\text{marginal likelihood or evidence}}}$$

$$= \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})}{\int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}}$$

GP hyperparameter optimization

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

Marginal likelihood as function of hyperparameters $\boldsymbol{\xi} = (\ell, \sigma_n^2)$. Because of $\boldsymbol{f} = \boldsymbol{\Phi}\,\boldsymbol{w}$, $\int \cdots \mathrm{d}\boldsymbol{w} \rightarrow \int \cdots \mathrm{d}\boldsymbol{f}$

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

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

Bayes' rule

$$\overbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}^{\text{weight posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})}^{\text{likelihood}}\ \overbrace{p(\boldsymbol{w})}^{\text{weight prior}}}{\underbrace{p(\boldsymbol{y}|\mathbf{X})}_{\text{marginal likelihood or evidence}}} = \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})}{\int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}}$$

(negative) log marginal likelihood

$$\ln p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\xi}) = -\frac{1}{2}\left[\underbrace{\boldsymbol{y}^\top \left(\mathbf{K} + \sigma_n^2\, \mathbf{I}_N\right)^{-1} \boldsymbol{y}}_{\text{model fit}} + \underbrace{\ln\left|\mathbf{K} + \sigma_n^2\, \mathbf{I}_N\right|}_{\text{model complexity}} + N\, \ln(2\,\pi)\right]$$

GP hyperparameter optimization

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

Marginal likelihood as function of hyperparameters $\boldsymbol{\xi} = (\ell, \sigma_n^2)$. Because of $\boldsymbol{f} = \boldsymbol{\Phi}\, \boldsymbol{w}$, $\int \cdots \mathrm{d}\boldsymbol{w} \to \int \cdots \mathrm{d}\boldsymbol{f}$

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

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

Bayes' rule

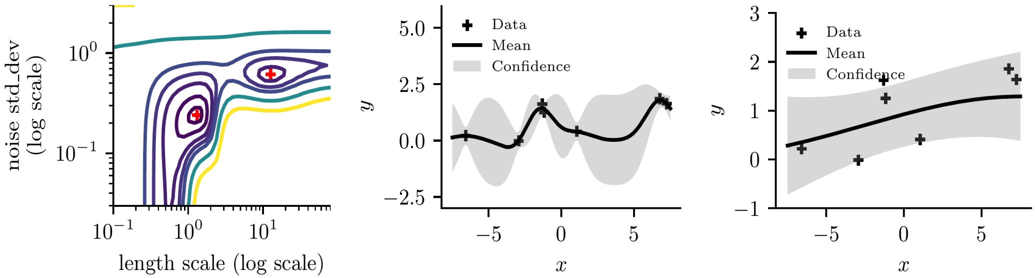
$$\overbrace{p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})}^{\text{weight posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})}^{\text{likelihood}}\ \overbrace{p(\boldsymbol{w})}^{\text{weight prior}}}{\underbrace{p(\boldsymbol{y}|\mathbf{X})}_{\text{marginal likelihood or evidence}}} = \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})}{\int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}}$$

(negative) log marginal likelihood

$$\ln p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\xi}) = -\frac{1}{2}\left[\underbrace{\boldsymbol{y}^\top \left(\mathbf{K} + \sigma_n^2\, \mathbf{I}_N\right)^{-1} \boldsymbol{y}}_{\text{model fit}} + \underbrace{\ln\left|\mathbf{K} + \sigma_n^2\, \mathbf{I}_N\right|}_{\text{model complexity}} + N \ln(2\,\pi)\right]$$

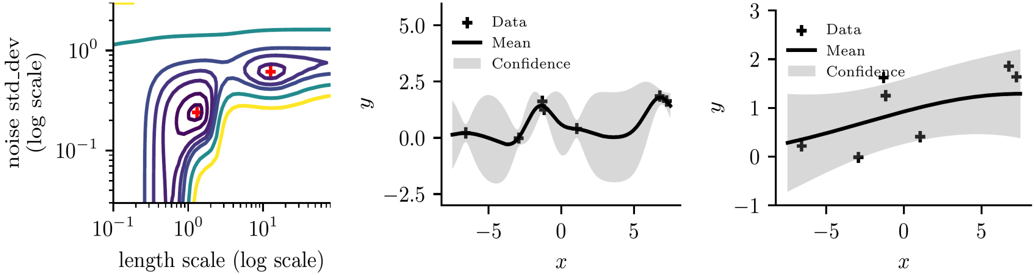
$$\boldsymbol{\xi}^* = \arg\max_{\boldsymbol{\xi}} \ln p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\xi}) = \arg\min_{\boldsymbol{\xi}} (-\ln p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\xi}))$$

Multiple minima of negative log marginal likelihood



K. P. Murphy. *Probabilistic Machine Learning: Advanced Topics (draft version)*. MIT Press, 2022.

Multiple minima of negative log marginal likelihood



Multiple minima: explain data in different ways

- small length scale ℓ, flexible model, low variance σ_n^2 $\rightarrow$ good model fit but complex model
- large length scale ℓ, "stiff"/low curvature model, high variance σ_n^2 $\rightarrow$ worse model fit but low model complexity

---

K. P. Murphy. *Probabilistic Machine Learning: Advanced Topics (draft version)*. MIT Press, 2022.

Relation to uncertainty quantification

Different kinds of uncertainty:

- epistemic / model uncertainty: weight posterior $p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})$ and $\operatorname{cov}[\boldsymbol{f}_*] = \boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2 \, \mathbf{I}_N\right)^{-1} \mathbf{K}_*^\top$
- aleatoric / data uncertainty: σ_n^2

Relation to uncertainty quantification

Different kinds of uncertainty:

- epistemic / model uncertainty: weight posterior $p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})$ and $\mathrm{cov}[\boldsymbol{f}_*] = \boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_* \left(\mathbf{K} + \sigma_n^2\, \mathbf{I}_N\right)^{-1} \mathbf{K}_*^\top$
- aleatoric / data uncertainty: σ_n^2

Distinction between "noise-free/noiseless prediction" of $\boldsymbol{f}_*$ when using $\boldsymbol{\Sigma}_*$

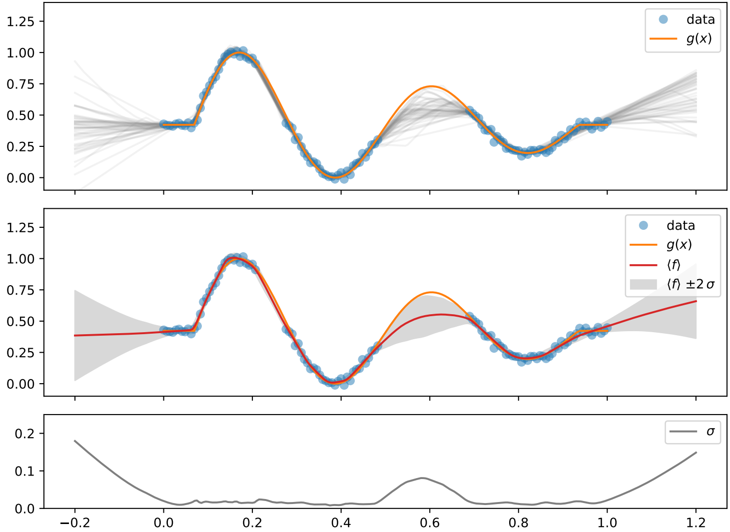
$$p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$$

and "noisy predictions" of $\boldsymbol{y}_*$ where we use $\boldsymbol{\Sigma}_* + \sigma_n^2\, \mathbf{I}_N$

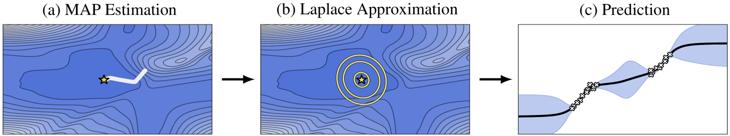
$$p(\boldsymbol{y}_*|\mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_* + \sigma_n^2\, \mathbf{I}_N)$$

In both cases we have the same $\boldsymbol{\mu}_*$

Approximate $p(\boldsymbol{w}|\mathcal{D})$: NN ensembles for UQ



Approximate $p(\boldsymbol{w}|\mathcal{D})$: Laplace approximation for UQ



Post-processing step after NN training (= MAP estimate): $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} - \ln p(\boldsymbol{w}|\mathcal{D})$

$$-\ln p(\boldsymbol{w}|\mathcal{D}) = -\ln\left(\frac{p(\mathcal{D}|\boldsymbol{w})\, p(\boldsymbol{w})}{p(\mathcal{D})}\right) = \overbrace{-\ln p(\mathcal{D}|\boldsymbol{w}) - \ln p(\boldsymbol{w})}^{\text{NN loss } L(\boldsymbol{w})=\text{NLL+regularizer}} + \ln p(\mathcal{D})$$

With gradient $\boldsymbol{g} = \nabla L|_{\boldsymbol{w}^*}$, Hessian $\mathbf{H} = \partial^2 L|_{\boldsymbol{w}^*}$ and $\boldsymbol{h} = \boldsymbol{w} - \boldsymbol{w}^*$, approximate loss to 2nd order

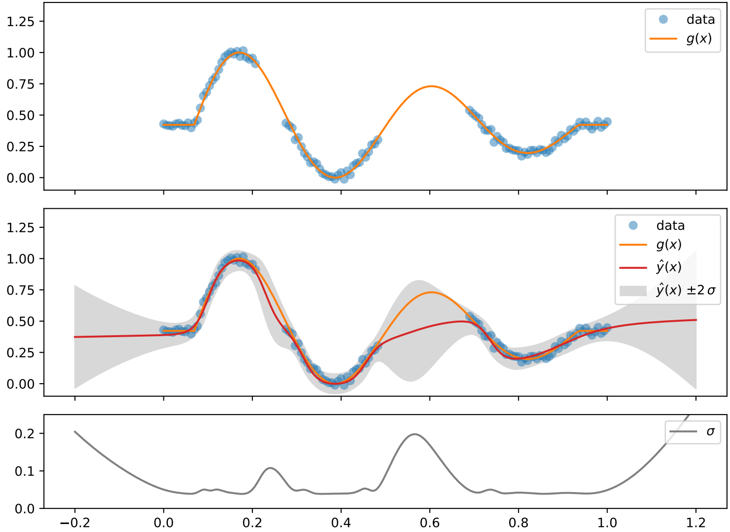
$$L(\boldsymbol{w}) \approx L(\boldsymbol{w}^*) + \underbrace{\boldsymbol{g}^\top}_{=\boldsymbol{0}} \boldsymbol{h} + \frac{1}{2}\boldsymbol{h}^\top \mathbf{H}\, \boldsymbol{h}$$

Approximate posterior probability distribution over $\boldsymbol{w}$ (i.e. over models)

$$p(\boldsymbol{w}|\mathcal{D}) \approx \mathcal{N}(\boldsymbol{w}^*, \boldsymbol{\Sigma}) \quad \text{where } \boldsymbol{\Sigma} = \mathbf{H}^{-1}$$

E. Daxberger et al. *Laplace Redux – Effortless Bayesian Deep Learning*. Version 3. 2022. URL: http://arxiv.org/abs/2106.14806 (visited on 02/05/2023).

Approximate $p(\boldsymbol{w}|\mathcal{D})$: Laplace approximation for UQ

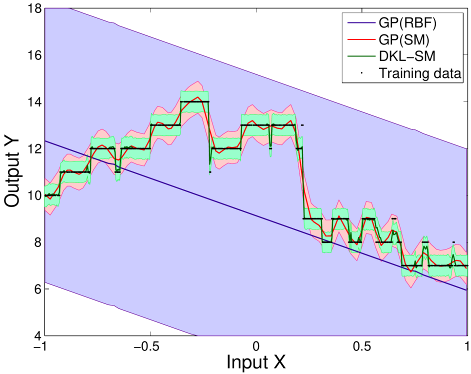


Kernel learning with NNs

(deep) kernel learning: more flexible kernels via NNs: use base kernel + NN features:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{h}_\gamma(\boldsymbol{x}_i) - \boldsymbol{h}_\gamma(\boldsymbol{x}_j)\|_2^2}{2\,\ell^2}\right)$$

with $\boldsymbol{h}_\gamma(\boldsymbol{x}_i)$ an NN embedding (”feature extractor”) and γ the NN parameters (weights, biases), optimize $\boldsymbol{\xi} = (\gamma, \ell, \sigma_n^2)$ jointly using log marginal likelihood



A. G. Wilson et al. “Deep Kernel Learning”. In: *Proc. 19th Int. Conf. Artif. Intell. Stat.* Artificial Intelligence and Statistics. PMLR, 2016, pp. 370–378.

Software

- `https://scikit-learn.org`, uses *numpy*
  - `sklearn.gaussian_process.GaussianProcessRegressor`
  - `sklearn.kernel_ridge.KernelRidge`
- `https://gpytorch.ai`: *PyTorch*-based, lots of advanced features, approximate methods for scaling GPs, API flexible but complex, GPU support via *PyTorch*
  - Define a mean function, since $f \sim \mathcal{GP}(m(\cdot), \kappa(\cdot, \cdot))$, so far we had $m(\cdot) = 0$
  - GPs are *non-parametric* models, $\mathbf{K} \in \mathbb{R}^{N \times N}$, $\dim \boldsymbol{\alpha} = N$, Cholesky decomposition for $\boldsymbol{\alpha} = \left(\mathbf{K} + \sigma_n^2 \, \mathbf{I}_N\right)^{-1} \boldsymbol{y}$ is $\mathcal{O}(N^3)$, **lots** of approximate methods, such as KISS-GP (a.k.a. SKI = structured kernel interpolation) for improved scaling
  - variational GPs for approximate inference of $p(\boldsymbol{w}|\mathcal{D})$, e.g. for non-Gaussian likelihoods
  - GP theory is for $f: \mathbb{R}^D \rightarrow \mathbb{R}$, *GPyTorch* supports multi-output GPs for $f: \mathbb{R}^D \rightarrow \mathbb{R}^M$
- `https://github.com/dfm/tinygp`: basic (educational) code, GPU support via *JAX*
- `https://github.com/JaxGaussianProcesses`, similar to *tinygp* but more features, GPU support via *JAX*
- `https://github.com/SheffieldML/GPy`, uses *numpy* + *Cython*
- Laplace approximation: `https://github.com/AlexImmer/Laplace`, *PyTorch*

Resources

- The Book: C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006 (http://gaussianprocess.org/gpml)
- K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022, K. P. Murphy. *Probabilistic Machine Learning: Advanced Topics (draft version)*. MIT Press, 2022 (https://probml.github.io/pml-book)
- M. Kanagawa et al. *Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences*. 2018. URL: http://arxiv.org/abs/1807.02582
- shameless plug: https://elcorto.github.io/gp_playground
- UQ in classifcation problems: P. Steinbach et al. "Machine Learning State-of-the-Art with Uncertainties". In: *ICLR* (2022)
- J. Gawlikowski et al. *A Survey of Uncertainty in Deep Neural Networks*. 2022. URL: http://arxiv.org/abs/2107.03342 (visited on 07/12/2022)