Imperial College London

From Parametric Models to Gaussian Processes

Yingzhen Li

yingzhen.li@imperial.ac.uk

Online resources for learning about GPs

Part of this talk is adapted from my GPSS 2019 lecture on GPs & Bayesian neural networks.

Tour overview

Today's agenda:

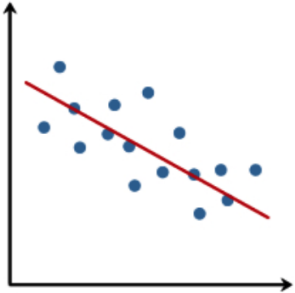
1. Linear regression $\rightarrow$ kernel methods for regression
2. (Shallow) neural networks $\rightarrow$ Gaussian processes
3. Gaussian processes $\rightarrow$ neural networks

Linear regression

Fitting linear regression models:

- Dataset: $\mathcal{D} = \{\mathbf{X}, \boldsymbol{y}\}$,
  $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times d}$,
  $\boldsymbol{y} = [y_1, ..., y_N]^\top \in \mathbb{R}^{N \times 1}$
- Goal: find $\boldsymbol{\theta} \in \mathbb{R}^{d \times 1}$ such that

$$\boldsymbol{y} \approx \mathbf{X}\boldsymbol{\theta}$$



Linear regression

A typical linear regression model:

- $\boldsymbol{x} \in \mathbb{R}^{d \times 1}$: input features; $y \in \mathbb{R}$: output value
- Model assumes noisy output with Gaussian noise:

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

$$\Rightarrow \quad p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \boldsymbol{x}^\top \boldsymbol{\theta}, \sigma^2)$$

- $\boldsymbol{\theta} \in \mathbb{R}^{d \times 1}$: model parameter; σ^2: output variance
- Prior distribution: $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$

Linear regression

Maximum a posteriori (MAP) fitting:[1]

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

[1] also called *ridge regression* for Gaussian prior case

Linear regression

Maximum a posteriori (MAP) fitting:[1]

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

- (conditional) i.i.d. assumption on data:

$$\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\theta}) = \log \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$$

[1] also called *ridge regression* for Gaussian prior case

Linear regression

Maximum a posteriori (MAP) fitting:[1]

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

- (conditional) i.i.d. assumption on data:

$$\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\theta}) = \log \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$$

$$= -\sum_{n=1}^{N} \frac{1}{2\sigma^2}(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 + const. = -\frac{1}{2\sigma^2}||\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 + const.$$

[1] also called *ridge regression* for Gaussian prior case

Linear regression

Maximum a posteriori (MAP) fitting:[1]

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

- (conditional) i.i.d. assumption on data:

$$\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\theta}) = \log \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$$

$$= -\sum_{n=1}^{N} \frac{1}{2\sigma^2}(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 + const. = -\frac{1}{2\sigma^2}||\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 + const.$$

- Gaussian prior: $\log p(\boldsymbol{\theta}) = -\frac{\lambda}{2}||\boldsymbol{\theta}||_2^2 + const.$

[1] also called *ridge regression* for Gaussian prior case

Linear regression

Maximum a posteriori (MAP) fitting:[1]

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} \frac{1}{\sigma^2}||\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 + \lambda||\boldsymbol{\theta}||_2^2$$

- (conditional) i.i.d. assumption on data:

$$\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\theta}) = \log \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$$

$$= -\sum_{n=1}^{N} \frac{1}{2\sigma^2}(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 + const. = -\frac{1}{2\sigma^2}||\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 + const.$$

- Gaussian prior: $\log p(\boldsymbol{\theta}) = -\frac{\lambda}{2}||\boldsymbol{\theta}||_2^2 + const.$

[1] also called *ridge regression* for Gaussian prior case

Linear regression

MAP solution of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} \frac{1}{\sigma^2}||\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 + \lambda||\boldsymbol{\theta}||_2^2 := \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{L}(\boldsymbol{\theta})$$

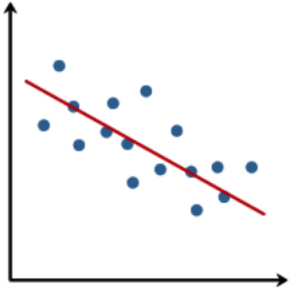
- Setting $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = 0$:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) \propto -\frac{1}{\sigma^2}\mathbf{X}^\top(\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda\boldsymbol{\theta} = 0$$

$$\Rightarrow (\lambda\mathbf{I} + \frac{1}{\sigma^2}\mathbf{X}^\top\mathbf{X})\boldsymbol{\theta}^* = \frac{1}{\sigma^2}\mathbf{X}\boldsymbol{y}$$

$$\Rightarrow \boldsymbol{\theta}^* = (\sigma^2\lambda\mathbf{I} + \mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y}$$

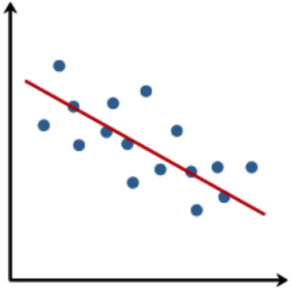
Regression with non-linear features



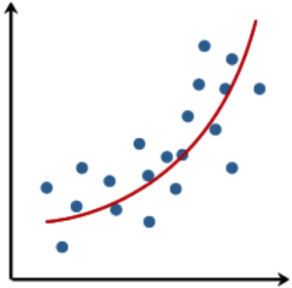
Linear regression

$$y = \boldsymbol{x}^\top \boldsymbol{\theta}$$

Regression with non-linear features



$\Rightarrow$



Linear regression

$$y = \boldsymbol{x}^\top \boldsymbol{\theta}$$

Non-linear regression

$$y = f(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \boldsymbol{\theta}$$

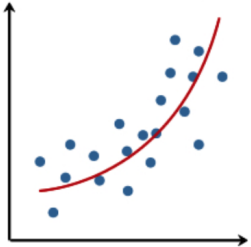
Regression with non-linear features

Extending to non-linear regression:

- Key idea: using a non-linear feature mapping: $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^p$
- The non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

- Recover linear regression when $\phi(\boldsymbol{x}) = \boldsymbol{x}$



example: $\phi(x) = [x^2, x, 1]$

Regression with non-linear features

Maximum a posteriori (MAP) fitting:

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

- Write $\Phi = [\phi(\boldsymbol{x}_1), ..., \phi(\boldsymbol{x}_N)]^\top \in \mathbb{R}^{N \times p}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \Theta} \frac{1}{\sigma^2} ||\boldsymbol{y} - \Phi\boldsymbol{\theta}||_2^2 + \lambda ||\boldsymbol{\theta}||_2^2$$

- MAP solution for $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^* = (\sigma^2 \lambda \mathbf{I} + \Phi^\top \Phi)^{-1} \Phi^\top \boldsymbol{y}$$

The kernel trick

Let's take one step back to look at the non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

The kernel trick

Let's take one step back to look at the non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

- Equivalent to linear regression with input space defined by $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^p$

The kernel trick

Let's take one step back to look at the non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

- Equivalent to linear regression with input space defined by $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^p$
- Requires a definition of inner product $\phi(\boldsymbol{x})^\top \boldsymbol{\theta}$

The kernel trick

Let's take one step back to look at the non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

- Equivalent to linear regression with input space defined by $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^p$
- Requires a definition of inner product $\phi(\boldsymbol{x})^\top \boldsymbol{\theta}$
  - what if $p = \infty$?
  - what if the output domain of $\phi(\cdot)$ is not even $\mathbb{R}^p$?

The kernel trick

Let's take one step back to look at the non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

- Equivalent to linear regression with input space defined by $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^p$
- Requires a definition of inner product $\phi(\boldsymbol{x})^\top \boldsymbol{\theta}$
  - what if $p = \infty$?
  - what if the output domain of $\phi(\cdot)$ is not even $\mathbb{R}^p$?

Solution: define inner product directly and use it through out!

The kernel trick

Constructing predictions for new inputs $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_M^*$ with features $\hat{\Phi} = [\phi(\boldsymbol{x}_1^*), \ldots, \phi(\boldsymbol{x}_M^*)]^\top$:

$$\hat{\boldsymbol{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}$$

The kernel trick

Constructing predictions for new inputs $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_M^*$ with features $\hat{\Phi} = [\phi(\boldsymbol{x}_1^*), \ldots, \phi(\boldsymbol{x}_M^*)]^\top$:

$$\hat{\boldsymbol{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}$$

- Notice the following identities:

$$(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)\Phi^\top = \sigma^2\lambda\Phi^\top + \Phi^\top\Phi\Phi^\top = \Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)$$

The kernel trick

Constructing predictions for new inputs $\boldsymbol{x}_1^*, ..., \boldsymbol{x}_M^*$ with features $\hat{\Phi} = [\phi(\boldsymbol{x}_1^*), ..., \phi(\boldsymbol{x}_M^*)]^\top$:

$$\hat{\boldsymbol{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}$$

- Notice the following identities:

$$(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)\Phi^\top = \sigma^2\lambda\Phi^\top + \Phi^\top\Phi\Phi^\top = \Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)$$

$$\Rightarrow \Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)^{-1} = (\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top$$

The kernel trick

Constructing predictions for new inputs $\boldsymbol{x}_1^*, ..., \boldsymbol{x}_M^*$ with features $\hat{\Phi} = [\phi(\boldsymbol{x}_1^*), ..., \phi(\boldsymbol{x}_M^*)]^\top$:

$$\hat{\boldsymbol{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}$$

- Notice the following identities:

$$(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)\Phi^\top = \sigma^2\lambda\Phi^\top + \Phi^\top\Phi\Phi^\top = \Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)$$

$$\Rightarrow \Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)^{-1} = (\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top$$

$$\Rightarrow \hat{\Phi}\Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)^{-1}\boldsymbol{y} = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y} = \hat{\Phi}\boldsymbol{\theta}^*$$

The kernel trick

Constructing predictions for new inputs $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_M^*$ with features $\hat{\Phi} = [\phi(\boldsymbol{x}_1^*), \ldots, \phi(\boldsymbol{x}_M^*)]^\top$:

$$\hat{\boldsymbol{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}$$

- Notice the following identities:

$$(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)\Phi^\top = \sigma^2\lambda\Phi^\top + \Phi^\top\Phi\Phi^\top = \Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)$$

$$\Rightarrow \Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)^{-1} = (\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top$$

$$\Rightarrow \hat{\Phi}\Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)^{-1}\boldsymbol{y} = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y} = \hat{\Phi}\boldsymbol{\theta}^*$$

also $\quad \hat{\Phi}\Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)^{-1}\boldsymbol{y} = \lambda^{-1}\hat{\Phi}\Phi^\top(\sigma^2\mathbf{I} + \lambda^{-1}\Phi\Phi^\top)^{-1}\boldsymbol{y}$

The kernel trick

Equivalent views on constructing predictions for new inputs $\hat{X}$ with features $\hat{\Phi}$:

$$\text{Parametric regression:} \quad \hat{\boldsymbol{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}$$

$$\Leftrightarrow \quad \text{Kernel regression:} \quad \hat{\boldsymbol{y}} = \lambda^{-1}\hat{\Phi}\Phi^\top(\sigma^2\mathbf{I} + \lambda^{-1}\Phi\Phi^\top)^{-1}\boldsymbol{y}$$

The kernel trick

Equivalent views on constructing predictions for new inputs $\hat{X}$ with features $\hat{\Phi}$:

$$\text{Parametric regression:} \quad \hat{\boldsymbol{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}$$

$$\Leftrightarrow \quad \text{Kernel regression:} \quad \hat{\boldsymbol{y}} = \lambda^{-1}\hat{\Phi}\Phi^\top(\sigma^2\mathbf{I} + \lambda^{-1}\Phi\Phi^\top)^{-1}\boldsymbol{y}$$

Advantages of the kernel regression view:

- Efficient in $N << p$ scheme:
  - $\Phi^\top\Phi \in \mathbb{R}^{p\times p} \Rightarrow$ matrix inversion needs $\mathcal{O}(p^3)$ time
  - $\Phi\Phi^\top \in \mathbb{R}^{N\times N} \Rightarrow$ matrix inversion needs $\mathcal{O}(N^3)$ time

The kernel trick

Equivalent views on constructing predictions for new inputs $\hat{X}$ with features $\hat{\Phi}$:

$$\text{Parametric regression:} \quad \hat{\boldsymbol{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}$$

$$\Leftrightarrow \quad \text{Kernel regression:} \quad \hat{\boldsymbol{y}} = \lambda^{-1}\hat{\Phi}\Phi^\top(\sigma^2\mathbf{I} + \lambda^{-1}\Phi\Phi^\top)^{-1}\boldsymbol{y}$$

Advantages of the kernel regression view:

- No need to explicitly define $\phi(\cdot)$:
  - The entries of $\mathbf{K_{XX}} := \lambda^{-1}\Phi\Phi^\top$: $\mathbf{K_{XX}}[i,j] := \lambda^{-1}\phi(\boldsymbol{x}_i)^\top\phi(\boldsymbol{x}_j)$
  - The entries of $\mathbf{K_{\hat{X}X}} := \lambda^{-1}\hat{\Phi}\Phi^\top$: $\mathbf{K_{\hat{X}X}}[k,l] := \lambda^{-1}\phi(\boldsymbol{x}_k^*)^\top\phi(\boldsymbol{x}_l)$
  - Enables the usage of $p = \infty$-dim features (or even non-$\mathbb{R}^p$ features)

The kernel trick

Equivalent views on constructing predictions for new inputs $\hat{X}$ with features $\hat{\Phi}$:

$$\text{Parametric regression:} \quad \hat{\boldsymbol{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}$$

$$\Leftrightarrow \quad \text{Kernel regression:} \quad \hat{\boldsymbol{y}} = \lambda^{-1}\hat{\Phi}\Phi^\top(\sigma^2\mathbf{I} + \lambda^{-1}\Phi\Phi^\top)^{-1}\boldsymbol{y}$$

Advantages of the parametric regression view:

- Easy to design $\phi(\cdot)$ with prior knowledge
- Better interpretability for non-linear features $\phi(\cdot)$
- Interpretability relates to function-space view

A function-space view of non-linear regression

Recall the parametric non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

A function-space view of non-linear regression

Recall the parametric non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

Parameter-space prior $\rightarrow$ function-space prior: recall $f(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \boldsymbol{\theta}$

- For any $N > 0$ and $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times d}$,
  we have $\boldsymbol{f} = [f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)]^\top = \Phi\boldsymbol{\theta} \in \mathbb{R}^{N \times 1}$

A function-space view of non-linear regression

Recall the parametric non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

Parameter-space prior $\rightarrow$ function-space prior: recall $f(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \boldsymbol{\theta}$

- For any $N > 0$ and $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times d}$,
  we have $\boldsymbol{f} = [f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)]^\top = \Phi\boldsymbol{\theta} \in \mathbb{R}^{N \times 1}$

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I}) \quad \Rightarrow \quad p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{f}; \mathbf{0}, \mathbf{K_{XX}}), \quad \mathbf{K_{XX}} = \lambda^{-1}\Phi\Phi^\top$$

$\boldsymbol{f} = \Phi\boldsymbol{\theta}$ makes $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ correlated

$\mathbf{K_{XX}}[i, j] = Cov[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)]$

A function-space view of non-linear regression

Recall the parametric non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

Parameter-space prior $\rightarrow$ function-space prior: recall $f(\boldsymbol{x}) = \phi(\boldsymbol{x})^\top \boldsymbol{\theta}$

- For any $N > 0$ and $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times d}$,
  we have $\boldsymbol{f} = [f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)]^\top = \Phi\boldsymbol{\theta} \in \mathbb{R}^{N \times 1}$

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I}) \quad \Rightarrow \quad p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{f}; \mathbf{0}, \mathbf{K_{XX}}), \quad \mathbf{K_{XX}} = \lambda^{-1}\Phi\Phi^\top$$

$\boldsymbol{f} = \Phi\boldsymbol{\theta}$ makes $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ correlated

$\mathbf{K_{XX}}[i, j] = Cov[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)]$

$$\prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}(y_n; \phi(\boldsymbol{x}_n)^\top \boldsymbol{\theta}, \sigma^2) \quad \Rightarrow \quad p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2\mathbf{I})$$

A function-space view of non-linear regression

A function-space view of the non-linear regression model:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2\mathbf{I}), \quad p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{f}; \mathbf{0}, \mathbf{K}_{\mathbf{XX}}), \quad \mathbf{K}_{\mathbf{XX}} = \lambda^{-1}\Phi\Phi^\top$$

Notice this function-space view applies to any $N > 0$.

A function-space view of non-linear regression

A function-space view of the non-linear regression model:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2\mathbf{I}), \quad p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{f}; \mathbf{0}, \mathbf{K}_{\mathbf{XX}}), \quad \mathbf{K}_{\mathbf{XX}} = \lambda^{-1}\Phi\Phi^\top$$

Notice this function-space view applies to any $N > 0$.

Consider the regression function on both N training inputs $\mathbf{X}$ and M test inputs $\hat{\mathbf{X}}$:

$$\begin{bmatrix} \boldsymbol{f} \\ \hat{\boldsymbol{f}} \end{bmatrix} = \begin{bmatrix} \Phi \\ \hat{\Phi} \end{bmatrix} \boldsymbol{\theta} \in \mathbb{R}^{(N+M)\times 1}, \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

A function-space view of non-linear regression

A function-space view of the non-linear regression model:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2\mathbf{I}), \quad p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{f}; \mathbf{0}, \mathbf{K}_{\mathbf{XX}}), \quad \mathbf{K}_{\mathbf{XX}} = \lambda^{-1}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top$$

Notice this function-space view applies to any $N > 0$.

Consider the regression function on both N training inputs $\mathbf{X}$ and M test inputs $\hat{\mathbf{X}}$:

$$\begin{bmatrix} \boldsymbol{f} \\ \hat{\boldsymbol{f}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi} \\ \hat{\boldsymbol{\Phi}} \end{bmatrix} \boldsymbol{\theta} \in \mathbb{R}^{(N+M)\times 1}, \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

$$\Rightarrow \quad p(\boldsymbol{f}, \hat{\boldsymbol{f}}|\mathbf{X}, \hat{\mathbf{X}}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ \hat{\boldsymbol{f}} \end{bmatrix}; \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathbf{XX}} & \mathbf{K}_{\mathbf{X}\hat{\mathbf{X}}} \\ \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}} & \mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}} \end{bmatrix} \right),$$

$$\mathbf{K}_{\mathbf{XX}} = \lambda^{-1}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top \in \mathbb{R}^{N\times N}, \ \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}} = \lambda^{-1}\hat{\boldsymbol{\Phi}}\boldsymbol{\Phi}^\top \in \mathbb{R}^{M\times N}, \ \mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}} = \lambda^{-1}\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{\Phi}}^\top \in \mathbb{R}^{M\times M}$$

A function-space view of non-linear regression

A function-space view of the non-linear regression model:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2\mathbf{I}), \quad p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{f}; \mathbf{0}, \mathbf{K}_{\mathbf{XX}}), \quad \mathbf{K}_{\mathbf{XX}} = \lambda^{-1}\Phi\Phi^\top$$

Notice this function-space view applies to any $N > 0$.

Consider the regression function on both N training inputs $\mathbf{X}$ and M test inputs $\hat{\mathbf{X}}$:

Independent observation noise model for training outputs $\boldsymbol{y}$ and test outputs $\hat{\boldsymbol{y}}$:

$$\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\epsilon}, \quad \boldsymbol{f} = \Phi\boldsymbol{\theta}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \quad \Rightarrow \quad p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2\mathbf{I})$$

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{f}} + \hat{\boldsymbol{\epsilon}}, \quad \hat{\boldsymbol{f}} = \hat{\Phi}\boldsymbol{\theta}, \quad \hat{\boldsymbol{\epsilon}} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \quad \Rightarrow \quad p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}) = \mathcal{N}(\hat{\boldsymbol{y}}; \hat{\boldsymbol{f}}, \sigma^2\mathbf{I})$$

A function-space view of non-linear regression

A function-space view of the non-linear regression model:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2\mathbf{I}), \quad p(\boldsymbol{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{f}; \mathbf{0}, \mathbf{K}_{\mathbf{XX}}), \quad \mathbf{K}_{\mathbf{XX}} = \lambda^{-1}\Phi\Phi^\top$$

Notice this function-space view applies to any $N > 0$.

Consider the regression function on both N training inputs $\mathbf{X}$ and M test inputs $\hat{\mathbf{X}}$:

Independent observation noise model for training outputs $\boldsymbol{y}$ and test outputs $\hat{\boldsymbol{y}}$:

$$\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\epsilon}, \quad \boldsymbol{f} = \Phi\boldsymbol{\theta}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \quad \Rightarrow \quad p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2\mathbf{I})$$

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{f}} + \hat{\boldsymbol{\epsilon}}, \quad \hat{\boldsymbol{f}} = \hat{\Phi}\boldsymbol{\theta}, \quad \hat{\boldsymbol{\epsilon}} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \quad \Rightarrow \quad p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}) = \mathcal{N}(\hat{\boldsymbol{y}}; \hat{\boldsymbol{f}}, \sigma^2\mathbf{I})$$

$$\boldsymbol{\epsilon} \perp \hat{\boldsymbol{\epsilon}} \quad \Rightarrow \quad p(\boldsymbol{y}, \hat{\boldsymbol{y}}|\boldsymbol{f}, \hat{\boldsymbol{f}}) = p(\boldsymbol{y}|\boldsymbol{f})p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}})$$

A function-space view of non-linear regression

A function-space view of the non-linear regression model (incl. train & test):

$$p(\boldsymbol{y}, \hat{\boldsymbol{y}} | \boldsymbol{f}, \hat{\boldsymbol{f}}) = p(\boldsymbol{y}|\boldsymbol{f}) p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}), \quad p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2 \mathbf{I}), \quad p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}) = \mathcal{N}(\hat{\boldsymbol{y}}; \hat{\boldsymbol{f}}, \sigma^2 \mathbf{I})$$

$$p(\boldsymbol{f}, \hat{\boldsymbol{f}} | \mathbf{X}, \hat{\mathbf{X}}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ \hat{\boldsymbol{f}} \end{bmatrix}; \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathbf{X}\mathbf{X}} & \mathbf{K}_{\mathbf{X}\hat{\mathbf{X}}} \\ \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}} & \mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}} \end{bmatrix} \right)$$

A function-space view of non-linear regression

A function-space view of the non-linear regression model (incl. train & test):

$$p(\boldsymbol{y}, \hat{\boldsymbol{y}} | \boldsymbol{f}, \hat{\boldsymbol{f}}) = p(\boldsymbol{y}|\boldsymbol{f}) p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}), \quad p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2 \mathbf{I}), \quad p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}) = \mathcal{N}(\hat{\boldsymbol{y}}; \hat{\boldsymbol{f}}, \sigma^2 \mathbf{I})$$

$$p(\boldsymbol{f}, \hat{\boldsymbol{f}} | \mathbf{X}, \hat{\mathbf{X}}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ \hat{\boldsymbol{f}} \end{bmatrix}; \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathbf{X}\mathbf{X}} & \mathbf{K}_{\mathbf{X}\hat{\mathbf{X}}} \\ \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}} & \mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}} \end{bmatrix} \right)$$

Prediction for $\hat{\boldsymbol{f}}$ given $\mathbf{X}, \boldsymbol{y}$, $\hat{\mathbf{X}}$: notice that $p(\boldsymbol{f}, \hat{\boldsymbol{f}} | \mathbf{X}, \hat{\mathbf{X}}) = p(\hat{\boldsymbol{f}} | \boldsymbol{f}, \mathbf{X}, \hat{\mathbf{X}}) p(\boldsymbol{f} | \mathbf{X})$

Conditionals of a Gaussian is still Gaussian!

A function-space view of non-linear regression

A function-space view of the non-linear regression model (incl. train & test):

$$p(\boldsymbol{y}, \hat{\boldsymbol{y}} | \boldsymbol{f}, \hat{\boldsymbol{f}}) = p(\boldsymbol{y}|\boldsymbol{f}) p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}), \quad p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2 \mathbf{I}), \quad p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}) = \mathcal{N}(\hat{\boldsymbol{y}}; \hat{\boldsymbol{f}}, \sigma^2 \mathbf{I})$$

$$p(\boldsymbol{f}, \hat{\boldsymbol{f}} | \mathbf{X}, \hat{\mathbf{X}}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ \hat{\boldsymbol{f}} \end{bmatrix}; \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathbf{X}\mathbf{X}} & \mathbf{K}_{\mathbf{X}\hat{\mathbf{X}}} \\ \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}} & \mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}} \end{bmatrix} \right)$$

Prediction for $\hat{\boldsymbol{f}}$ given $\mathbf{X}, \boldsymbol{y}$, $\hat{\mathbf{X}}$: notice that $p(\boldsymbol{f}, \hat{\boldsymbol{f}} | \mathbf{X}, \hat{\mathbf{X}}) = p(\hat{\boldsymbol{f}} | \boldsymbol{f}, \mathbf{X}, \hat{\mathbf{X}}) p(\boldsymbol{f} | \mathbf{X})$

Conditionals of a Gaussian is still Gaussian!

- "MAP" solution: $\boldsymbol{f}^* = \arg\max_{\boldsymbol{f}} \log p(\boldsymbol{y}|\boldsymbol{f}) + \log p(\boldsymbol{f}|\mathbf{X})$, $\hat{\boldsymbol{f}} \sim p(\hat{\boldsymbol{f}} | \boldsymbol{f}^*, \mathbf{X}, \hat{\mathbf{X}})$

A function-space view of non-linear regression

A function-space view of the non-linear regression model (incl. train & test):

$$p(\boldsymbol{y}, \hat{\boldsymbol{y}}|\boldsymbol{f}, \hat{\boldsymbol{f}}) = p(\boldsymbol{y}|\boldsymbol{f})p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}), \quad p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma^2\mathbf{I}), \quad p(\hat{\boldsymbol{y}}|\hat{\boldsymbol{f}}) = \mathcal{N}(\hat{\boldsymbol{y}}; \hat{\boldsymbol{f}}, \sigma^2\mathbf{I})$$

$$p(\boldsymbol{f}, \hat{\boldsymbol{f}}|\mathbf{X}, \hat{\mathbf{X}}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{f}\\ \hat{\boldsymbol{f}}\end{bmatrix}; \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{\mathbf{X}\mathbf{X}} & \mathbf{K}_{\mathbf{X}\hat{\mathbf{X}}}\\ \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}} & \mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}}\end{bmatrix}\right)$$

Prediction for $\hat{\boldsymbol{f}}$ given $\mathbf{X}, \boldsymbol{y}$, $\hat{\mathbf{X}}$: notice that $p(\boldsymbol{f}, \hat{\boldsymbol{f}}|\mathbf{X}, \hat{\mathbf{X}}) = p(\hat{\boldsymbol{f}}|\boldsymbol{f}, \mathbf{X}, \hat{\mathbf{X}})p(\boldsymbol{f}|\mathbf{X})$

Conditionals of a Gaussian is still Gaussian!

- "MAP" solution: $\boldsymbol{f}^* = \arg\max_{\boldsymbol{f}} \log p(\boldsymbol{y}|\boldsymbol{f}) + \log p(\boldsymbol{f}|\mathbf{X})$, $\hat{\boldsymbol{f}} \sim p(\hat{\boldsymbol{f}}|\boldsymbol{f}^*, \mathbf{X}, \hat{\mathbf{X}})$
- Full Bayesian treatment ($\Rightarrow$ Gaussian process regression):

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

$$p(\boldsymbol{f}|\boldsymbol{y}, \mathbf{X}) \propto p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\mathbf{X})$$

Gaussian process regression

A Gaussian process (GP) is a collection of random variables $\{f(\boldsymbol{x}_n)\}_{n=1}^{\infty}$ indexed by $\{\boldsymbol{x}_n\}_{n=1}^{\infty}$:

- Jointly Gaussian for any N output random variables $\boldsymbol{f} = [f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)]^\top$

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

- The variance of the Gaussian is computed using a kernel function:

$$\mathbf{K}_{\mathbf{XX}}[i, j] = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

This allows us to write a GP as $f(\cdot) \sim \mathcal{GP}(0, \mathcal{K}(\cdot, \cdot))$

Gaussian process regression

Gaussian process regression model in 2 components:

- Specify GP prior: $f(\cdot) \sim \mathcal{GP}(0, \mathcal{K}(\cdot, \cdot))$
- An observation model: $p(y|f(\cdot), \boldsymbol{x}) = p(y|f(\boldsymbol{x})) = \mathcal{N}(y; f(x), \sigma^2)$

Prediction:

- Training data/observations: $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\top, \boldsymbol{y} = [y_1, ..., y_N]^\top$
- For test input $\hat{\mathbf{X}}$, compute

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

$$p(\boldsymbol{f}|\boldsymbol{y}, \mathbf{X}) \propto p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f}|\mathbf{X})$$

Gaussian process regression

How to plot function samples from a GP?

- define $\hat{\mathbf{X}} = [\boldsymbol{x}_1^*, ..., \boldsymbol{x}_M^*]^\top$ using a grid
- compute the covariance matrix $\mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}}$ with elements $\mathcal{K}(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*)$
- Prior sample $\hat{\boldsymbol{f}} = [f(\boldsymbol{x}_1^*), ..., f(\boldsymbol{x}_N^*)]$ as $\hat{\boldsymbol{f}} \sim N(0, \mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}})$
- Posterior sample $\hat{\boldsymbol{f}} = [f(\boldsymbol{x}_1^*), ..., f(\boldsymbol{x}_N^*)]$ as $\hat{\boldsymbol{f}} \sim p(\hat{\boldsymbol{f}}|\boldsymbol{y}, \mathbf{X}, \hat{\mathbf{X}})$

Gaussian process regression

How to plot function samples from a GP?

- define $\hat{\mathbf{X}} = [\boldsymbol{x}_1^*, ..., \boldsymbol{x}_M^*]^\top$ using a grid
- compute the covariance matrix $\mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}}$ with elements $\mathcal{K}(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*)$
- Prior sample $\hat{\boldsymbol{f}} = [f(\boldsymbol{x}_1^*), ..., f(\boldsymbol{x}_N^*)]$ as $\hat{\boldsymbol{f}} \sim N(0, \mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}})$
- Posterior sample $\hat{\boldsymbol{f}} = [f(\boldsymbol{x}_1^*), ..., f(\boldsymbol{x}_N^*)]$ as $\hat{\boldsymbol{f}} \sim p(\hat{\boldsymbol{f}}|\boldsymbol{y}, \mathbf{X}, \hat{\mathbf{X}})$

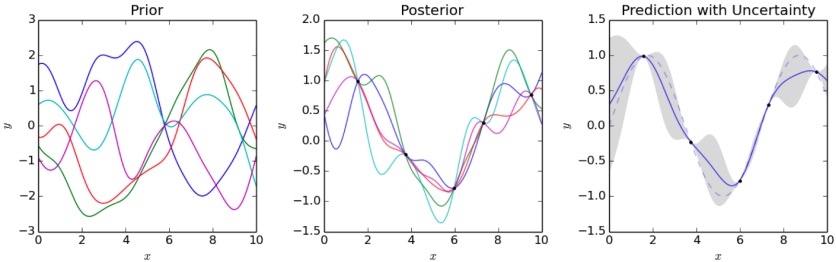


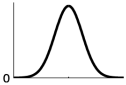
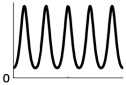
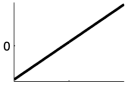
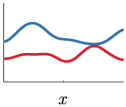
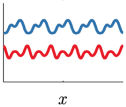
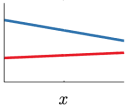
Figure from Richard Wilkinson's GPSS 2019 lecture

Constructing kernels

What is a kernel function?

- In previous non-linear regression example: $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \lambda^{-1}\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)$
- Can also directly define $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ without explicitly define $\phi(\boldsymbol{x})$:
  - Non-negative: $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}) \geq 0$ for any $\boldsymbol{x}$
  - Symmetric: $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x}', \boldsymbol{x})$
  - Positive semi-definite (PSD): for any $N > 0$, $\mathbf{X} \in \mathbb{R}^{N \times d}$, $\mathbf{K}_{\mathbf{XX}}$ is a PSD matrix.

Constructing kernels

Kernel name:	Squared-exp (SE)	Periodic (Per)	Linear (Lin)
$k(x, x') =$	$\sigma_f^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$	$\sigma_f^2 \exp\left(-\frac{2}{\ell^2}\sin^2\left(\pi\frac{x-x'}{p}\right)\right)$	$\sigma_f^2(x-c)(x'-c)$
Plot of $k(x, x')$:			
Functions $f(x)$ sampled from GP prior:			
Type of structure:	local variation	repeating structure	linear functions

See David Duvenaud's PhD thesis, Chapter 2

Constructing kernels

Selecting length-scale ℓ for squared-exponential (SE) kernel: $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$

- Correlation between $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$:

$$corr[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = \frac{Cov[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)]}{\sqrt{\mathbb{V}[f(\boldsymbol{x}_i)]\mathbb{V}[f(\boldsymbol{x}_j)]}}$$

$$Cov[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \mathbb{V}[f(\boldsymbol{x}_i)] = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_i) = \sigma_f^2$$

Constructing kernels

Selecting length-scale ℓ for squared-exponential (SE) kernel: $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$

- Correlation between $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$:

$$corr[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = \exp(-\frac{1}{2\ell^2}||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2)$$

Constructing kernels

Selecting length-scale ℓ for squared-exponential (SE) kernel: $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$

- Correlation between $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$:

$$corr[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = \exp(-\frac{1}{2\ell^2}||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2)$$

  - $||\boldsymbol{x}_i - \boldsymbol{x}_j||_2/\ell$ is small $\Rightarrow f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ are highly and positively correlated
  - increasing $\ell \Rightarrow f(\cdot)$ becomes smoother

Constructing kernels

$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$

$\sigma_f = 1,\ \ell = 0.25$

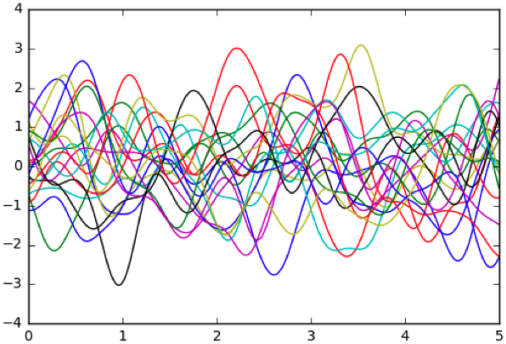


Figure from Richard Wilkinson's GPSS 2019 lecture

Constructing kernels

$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$

$\sigma_f = 1,\ \ell = 4$

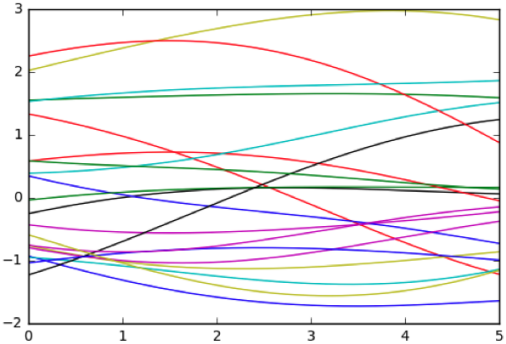


Figure from Richard Wilkinson's GPSS 2019 lecture

Constructing kernels

$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$
$\sigma_f = 1,\ \ell = 1$

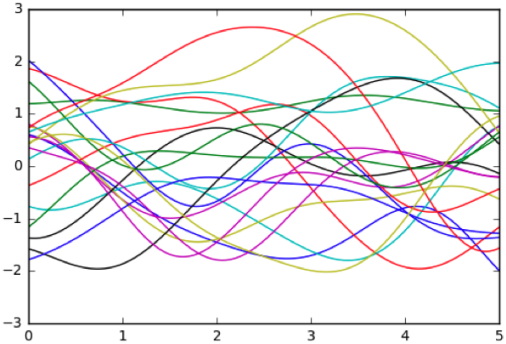


Figure from Richard Wilkinson's GPSS 2019 lecture

Constructing kernels

$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$

$\sigma_f = 10,\ \ell = 1$

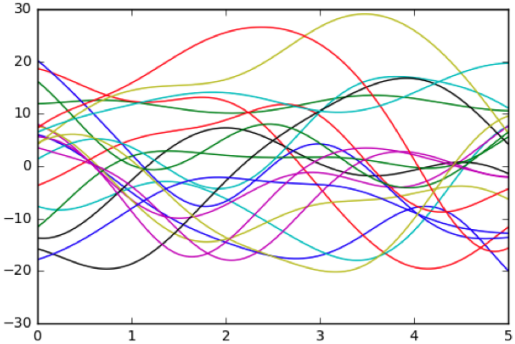


Figure from Richard Wilkinson's GPSS 2019 lecture

Neural networks 101

Recall the parametric non-linear regression model:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; \phi(\boldsymbol{x})^\top \boldsymbol{\theta}, \sigma^2), \quad p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \lambda^{-1}\mathbf{I})$$

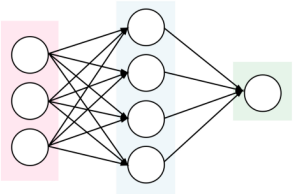
- Requires a pre-defined non-linear feature map $\phi(\cdot)$
- However, real-world data are very complex

Deep learning solution: learn the feature mapping $\phi(\cdot)$

Neural networks 101

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$

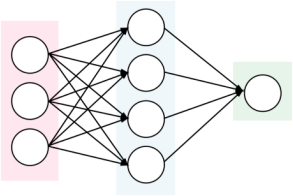


- Parameters: $\boldsymbol{\theta} = \{\boldsymbol{w}_m, b_m, v_m\}_{m=1}^{M}$
- $\psi(\cdot)$ is a non-linear activation function (e.g. ReLU)
- learning $\boldsymbol{w}_m, b_m \Rightarrow$ learning non-linear features

Neural networks 101

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$



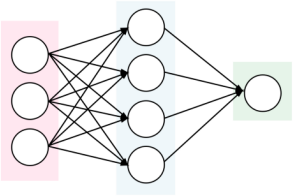
Initialising a neural network: sample $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$

- Example: He Initialisation:
  $w_{ij} \sim \mathcal{N}(0, \sigma^2/d)$

Neural networks 101

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$



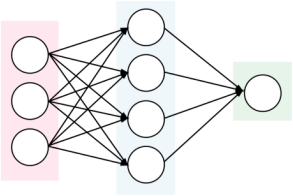
Initialising a neural network: sample $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$

- Example: He Initialisation:
  $w_{ij} \sim \mathcal{N}(0, \sigma^2/d)$
- In most cases $p(\boldsymbol{\theta})$ is factorised:
  $p(\boldsymbol{\theta}) = \prod_{m=1}^{M} p(\boldsymbol{w}_m) p(b_m) p(v_m)$

Neural networks 101

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$



Initialising a neural network: sample $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$

- Example: He Initialisation:
  $w_{ij} \sim \mathcal{N}(0, \sigma^2/d)$
- In most cases $p(\boldsymbol{\theta})$ is factorised:
  $p(\boldsymbol{\theta}) = \prod_{m=1}^{M} p(\boldsymbol{w}_m) p(b_m) p(v_m)$

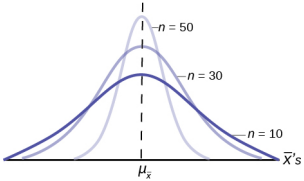
How does $f(\cdot)$ look like for a randomly initialised network?

Randomly initialised neural networks $\rightarrow$ Gaussian process function priors

Quick refresher: Central limit theorem

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ *be i.i.d. samples from* $p(\boldsymbol{x})$ *and* $p(\boldsymbol{x})$ *has mean* μ *and covariance* Σ*, then*

$$\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \xrightarrow{d} \mathcal{N}\left(\mu, \frac{1}{N}\Sigma\right), \quad N \rightarrow +\infty$$



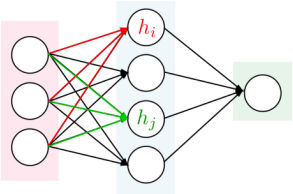
Randomly initialised neural networks $\rightarrow$ Gaussian process function priors

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$

Param. init. independently from the same prior (i.i.d.):

$$p(\boldsymbol{w}_i) = p(\boldsymbol{w}_j), \quad p(b_i) = p(b_j), \quad \forall i, j$$



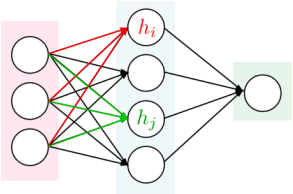
Randomly initialised neural networks $\rightarrow$ Gaussian process function priors

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$

Param. init. independently from the same prior (i.i.d.):

$$p(\boldsymbol{w}_i) = p(\boldsymbol{w}_j), \quad p(b_i) = p(b_j), \quad \forall i, j$$



$\Rightarrow$ Given $\boldsymbol{x}$, $h_1(\boldsymbol{x}), \ldots, h_M(\boldsymbol{x})$ are i.i.d. samples from an implicitly defined distribution:

$$h_i(\boldsymbol{x}) \perp h_j(\boldsymbol{x}), \quad h_i(\boldsymbol{x}) \stackrel{d}{=} h_j(\boldsymbol{x}), \quad h_i(\boldsymbol{x}) = \psi(\boldsymbol{w}_i^T \boldsymbol{x} + b_i)$$

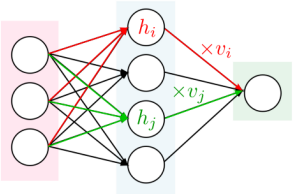
Randomly initialised neural networks $\rightarrow$ Gaussian process function priors

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$

2nd layer weights v_i i.i.d. sampled,
independent from the 1st layer:

$$v_i \perp \{\boldsymbol{w}_m, b_m\}, \qquad p(v_i) = p(v_j), \quad \forall i, j$$



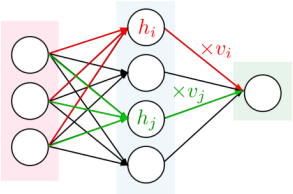
Randomly initialised neural networks $\rightarrow$ Gaussian process function priors

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$

2nd layer weights v_i i.i.d. sampled,
independent from the 1st layer:

$$v_i \perp \{\boldsymbol{w}_m, b_m\}, \qquad p(v_i) = p(v_j), \quad \forall i, j$$



$\Rightarrow$ Given $\boldsymbol{x}$, contributions to the output are also i.i.d:

$$v_i h_i(\boldsymbol{x}) \perp v_j h_j(\boldsymbol{x}), \quad v_i h_i(\boldsymbol{x}) \stackrel{d}{=} v_j h_j(\boldsymbol{x})$$

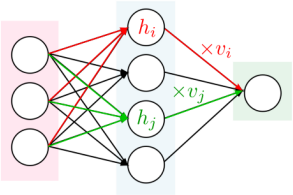
$\Rightarrow$ $f(\boldsymbol{x})$ is a sum of i.i.d. random variables

Randomly initialised neural networks $\rightarrow$ Gaussian process function prior

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$

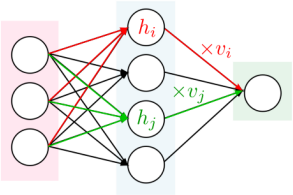
Assume $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2 / M$:



Randomly initialised neural networks $\rightarrow$ Gaussian process function prior

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$

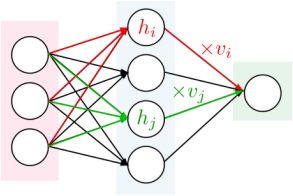


Assume $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2/M$:
$\mathbb{E}[f(\boldsymbol{x})] = \sum_{m=1}^{M} \mathbb{E}[v_m]\mathbb{E}[h_m(\boldsymbol{x})] = 0$

Randomly initialised neural networks $\rightarrow$ Gaussian process function prior

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$



Assume $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2/M$:
$\mathbb{E}[f(\boldsymbol{x})] = \sum_{m=1}^{M} \mathbb{E}[v_m]\mathbb{E}[h_m(\boldsymbol{x})] = 0$

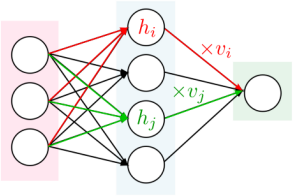
$$\mathbb{V}[f(\boldsymbol{x})] = \sum_{m=1}^{M} \mathbb{V}[v_m h_m(\boldsymbol{x})] = \frac{1}{M}\sum_{m=1}^{M} \sigma_v^2 \mathbb{E}[h_m(\boldsymbol{x})^2] \rightarrow \sigma_v^2 \mathbb{E}[h(\boldsymbol{x})^2]$$

($v_i h_i(\boldsymbol{x}) \perp v_j h_j(\boldsymbol{x})$, $v_m \perp \{\boldsymbol{w}_m, b_m\} \Rightarrow v_m \perp h_m(\boldsymbol{x})$, $M \rightarrow +\infty, h_i(\boldsymbol{x}) \stackrel{d}{=} h_j(\boldsymbol{x})$)

Randomly initialised neural networks $\rightarrow$ Gaussian process function prior

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \psi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$



Assume $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2/M$:
$\mathbb{E}[f(\boldsymbol{x})] = \sum_{m=1}^{M} \mathbb{E}[v_m]\mathbb{E}[h_m(\boldsymbol{x})] = 0$

$$\mathbb{V}[f(\boldsymbol{x})] = \sum_{m=1}^{M} \mathbb{V}[v_m h_m(\boldsymbol{x})] = \frac{1}{M}\sum_{m=1}^{M} \sigma_v^2 \mathbb{E}[h_m(\boldsymbol{x})^2] \rightarrow \sigma_v^2 \mathbb{E}[h(\boldsymbol{x})^2]$$

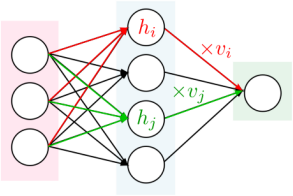
($v_i h_i(\boldsymbol{x}) \perp v_j h_j(\boldsymbol{x})$, $v_m \perp \{\boldsymbol{w}_m, b_m\} \Rightarrow v_m \perp h_m(\boldsymbol{x})$, $M \rightarrow +\infty$, $h_i(\boldsymbol{x}) \stackrel{d}{=} h_j(\boldsymbol{x})$)

Applying CLT: $f(\boldsymbol{x}) \sim \mathcal{N}(0, \sigma_v^2 \mathbb{E}[h(\boldsymbol{x})^2])$, $\quad M \rightarrow +\infty$

Randomly initialised neural networks $\rightarrow$ Gaussian process function prior

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \phi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$



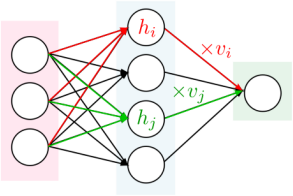
Assume $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2/M$:

$$\mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = \frac{1}{M} \sum_{m=1}^{M} \sigma_v^2 \mathbb{E}[h_m(\boldsymbol{x}) h_m(\boldsymbol{x}')] \rightarrow \sigma_v^2 \mathbb{E}[h(\boldsymbol{x}) h(\boldsymbol{x}')]$$

Randomly initialised neural networks $\rightarrow$ Gaussian process function prior

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \phi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$



Assume $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2/M$:

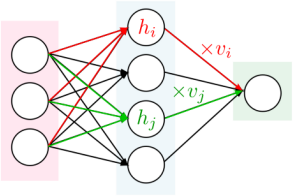
$$\mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = \frac{1}{M} \sum_{m=1}^{M} \sigma_v^2 \mathbb{E}[h_m(\boldsymbol{x}) h_m(\boldsymbol{x}')] \rightarrow \sigma_v^2 \mathbb{E}[h(\boldsymbol{x}) h(\boldsymbol{x}')]$$

Applying CLT: $(f(\boldsymbol{x}), f(\boldsymbol{x}')) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \mathbf{K}), \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_v^2 \mathbb{E}[h(\boldsymbol{x}) h(\boldsymbol{x}')]$

Randomly initialised neural networks $\rightarrow$ Gaussian process function prior

A one hidden layer neural network with M hidden units:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} v_m \phi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)$$



Assume $\mathbb{E}[v_m] = 0$ and $\mathbb{V}[v_m] = \sigma_v^2/M$:

$$\mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = \frac{1}{M} \sum_{m=1}^{M} \sigma_v^2 \mathbb{E}[h_m(\boldsymbol{x}) h_m(\boldsymbol{x}')] \rightarrow \sigma_v^2 \mathbb{E}[h(\boldsymbol{x}) h(\boldsymbol{x}')]$$

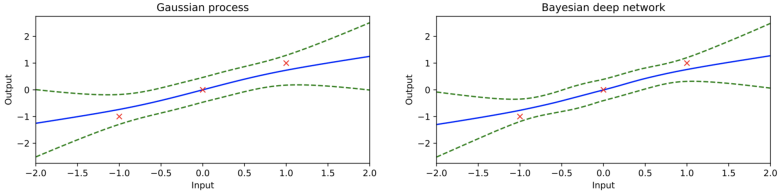
Applying CLT: $(f(\boldsymbol{x}), f(\boldsymbol{x}')) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \mathbf{K}), \quad \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_v^2 \mathbb{E}[h(\boldsymbol{x}) h(\boldsymbol{x}')]$

It holds for any $\boldsymbol{x}, \boldsymbol{x}' \Rightarrow f(\cdot) \sim \mathcal{GP}(0, \mathcal{K}(\cdot, \cdot))$

Bayesian neural networks → Gaussian processes

Recent extensions of Radford Neal's result (Neural networks → Gaussian processes):

- deep and wide BNNs have GP limits
  - mean-field prior over weights
  - the activation function satisfies $|\phi(x)| \leq c + A|x|$
  - hidden layer widths strictly increasing to infinity

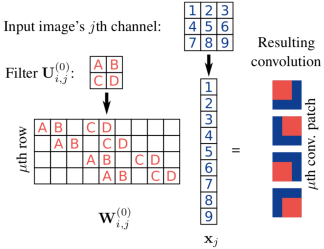


Matthews et al. 2018, Lee et al. 2018

Bayesian neural networks → Gaussian processes

Recent extensions of Radford Neal's result (Neural networks → Gaussian processes):

- Bayesian CNNs have GP limits
  - Convolution in CNN = fully connected layer applied to different locations in the image
  - # channels in CNN = # hidden units in fully connected NN



Garriga-Alonso et al. 2019, Novak et al. 2019

Gaussian process $\rightarrow$ Bayesian neural networks

Exact kernel regression & GP inference can be very expensive:

Predictive inference for GP regression:

$$p(\hat{\boldsymbol{f}}|\boldsymbol{y}, \mathbf{X}, \hat{\mathbf{X}}) = \mathcal{N}(\hat{\boldsymbol{f}}; \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}}(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{y}, \mathbf{K}_{\hat{\mathbf{X}}\hat{\mathbf{X}}} - \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}}(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{\mathbf{X}\hat{\mathbf{X}}})$$

$$\mathbf{K}_{\mathbf{X}\mathbf{X}}[i,j] = \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \mathbf{K}_{\mathbf{X}\mathbf{X}} \in \mathbb{R}^{N \times N}$$

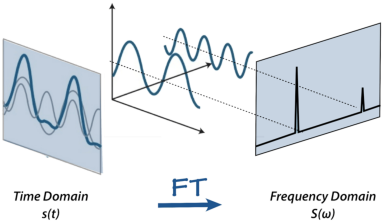
Inverting $\mathbf{K}_{nn} + \sigma^2\mathbf{I}$ has $\mathcal{O}(N^3)$ cost!

Gaussian process → Bayesian neural networks

Quick refresher: Fourier (inverse) transform

$$S(w) = \int s(t) e^{-itw} dt$$

$$s(t) = \int S(w) e^{itw} dw$$



Gaussian process $\rightarrow$ Bayesian neural networks

Bochner's theorem: (Fourier inverse transform)

A (properly scaled) translation invariant kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x} - \boldsymbol{x}')$ can be represented as

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{p(\boldsymbol{w})}\left[\sigma^2 e^{i\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{x}')}\right]$$

for some distribution $p(\boldsymbol{w})$.

Rahimi and Recht 2007

Gaussian process $\rightarrow$ Bayesian neural networks

Bochner's theorem: (Fourier inverse transform)

A (properly scaled) translation invariant kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x} - \boldsymbol{x}')$ can be represented as

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{p(\boldsymbol{w})}\left[\sigma^2 e^{i\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{x}')}\right]$$

for some distribution $p(\boldsymbol{w})$.

- Real value kernel $\Rightarrow \mathbb{E}_{p(\boldsymbol{w})}\left[\sigma^2 e^{i\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{x}')}\right] = \mathbb{E}_{p(\boldsymbol{w})}\left[\sigma^2 cos(\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}'))\right]$
- $cos(x - x') = 2\mathbb{E}_{p(b)}[cos(x + b)cos(x' + b)], \quad p(b) = \text{Uniform}[0, 2\pi]$

Rahimi and Recht 2007

Gaussian process $\rightarrow$ Bayesian neural networks

Bochner's theorem: (Fourier inverse transform)

A (properly scaled) translation invariant kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x} - \boldsymbol{x}')$ can be represented as

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{p(\boldsymbol{w})p(\boldsymbol{b})}\left[\sigma^2 cos(\boldsymbol{w}^T\boldsymbol{x} + b) cos(\boldsymbol{w}^T\boldsymbol{x}' + b)\right]$$

for some distribution $p(\boldsymbol{w})$ and $p(b) = \text{Uniform}[0, 2\pi]$.

Rahimi and Recht 2007

Gaussian process $\rightarrow$ Bayesian neural networks

Bochner's theorem: (Fourier inverse transform)

A (properly scaled) translation invariant kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x} - \boldsymbol{x}')$ can be represented as

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{p(\boldsymbol{w})p(\boldsymbol{b})}\left[\sigma^2 cos(\boldsymbol{w}^T\boldsymbol{x} + b) cos(\boldsymbol{w}^T\boldsymbol{x}' + b)\right]$$

for some distribution $p(\boldsymbol{w})$ and $p(b) = \text{Uniform}[0, 2\pi]$.

- Monte Carlo approximation:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \approx \tilde{\mathcal{K}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sigma^2}{M}\sum_{m=1}^{M} cos(\boldsymbol{w}_m^T\boldsymbol{x} + b_m) cos(\boldsymbol{w}_m^T\boldsymbol{x}' + b_m), \quad \boldsymbol{w}_m, b_m \sim p(\boldsymbol{w})p(b)$$

Rahimi and Recht 2007

Gaussian process $\rightarrow$ Bayesian neural networks

Bochner's theorem: (Fourier inverse transform)

A (properly scaled) translation invariant kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x} - \boldsymbol{x}')$ can be represented as

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{p(\boldsymbol{w})p(\boldsymbol{b})}\left[\sigma^2 cos(\boldsymbol{w}^T\boldsymbol{x} + b) cos(\boldsymbol{w}^T\boldsymbol{x}' + b)\right]$$

for some distribution $p(\boldsymbol{w})$ and $p(b) = \text{Uniform}[0, 2\pi]$.

- Monte Carlo approximation: Define
  $\boldsymbol{h}(\boldsymbol{x}) = [h_1(\boldsymbol{x}), ..., h_M(\boldsymbol{x})], \quad h_m(\boldsymbol{x}) = cos(\boldsymbol{w}_m^T\boldsymbol{x} + b_m), \quad \boldsymbol{w}_m \sim p(\boldsymbol{w}), b_m \sim p(b)$

$$\Rightarrow \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \approx \tilde{\mathcal{K}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sigma^2}{M}\boldsymbol{h}(\boldsymbol{x})^T\boldsymbol{h}(\boldsymbol{x}')$$

Rahimi and Recht 2007

Gaussian process $\rightarrow$ Bayesian neural networks

Approximating the GP kernel with random feature expansions:

$$f(\cdot) \sim \mathcal{GP}(0, \mathcal{K}(\cdot,\cdot)), \quad f(\cdot) \approx \tilde{f}(\cdot), \quad \tilde{f}(\cdot) \sim \mathcal{GP}(0, \tilde{\mathcal{K}}(\cdot,\cdot)), \quad \tilde{\mathcal{K}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sigma^2}{M} \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{h}(\boldsymbol{x}')$$

Gaussian process $\rightarrow$ Bayesian neural networks

Approximating the GP kernel with random feature expansions:

$$f(\cdot) \sim \mathcal{GP}(0, \mathcal{K}(\cdot,\cdot)), \quad f(\cdot) \approx \tilde{f}(\cdot), \quad \tilde{f}(\cdot) \sim \mathcal{GP}(0, \tilde{\mathcal{K}}(\cdot,\cdot)), \quad \tilde{\mathcal{K}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sigma^2}{M} \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{h}(\boldsymbol{x}')$$

Weight space view $\Rightarrow$ single hidden layer BNN:

$$\tilde{f}(\cdot) \sim \mathcal{GP}(0, \tilde{K}(\cdot,\cdot)) \quad \Leftrightarrow \quad \tilde{f}(\boldsymbol{x}) = \begin{cases} \boldsymbol{v}^T \boldsymbol{h}(\boldsymbol{x}), & \boldsymbol{v} \sim p(\boldsymbol{v}) = \mathcal{N}(\boldsymbol{0}, \frac{\sigma^2}{M}\mathbf{I}) \\ h_m(\boldsymbol{x}) = cos(\boldsymbol{w}_m^\top \boldsymbol{x} + b_m), & \boldsymbol{w}_m \sim p(\boldsymbol{w}), b_m \sim p(b) \end{cases}$$

Adding number of components (increase M) $\Rightarrow$ adding hidden units in NNs

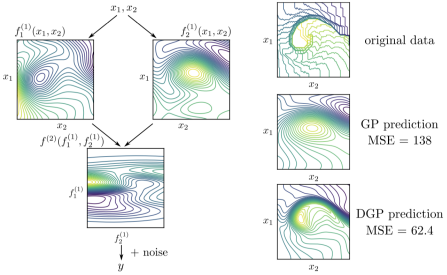
Gaussian process → Bayesian neural networks

Deep GPs → deep BNNs with bottleneck layers:

Deep Gaussian process:

$$f(\boldsymbol{x}) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(0)}(\boldsymbol{x}),$$

$$f^{(i)}(\cdot) \sim \mathcal{GP}(0, \mathcal{K}^{(i)}(\cdot,\cdot))$$



Bui et al. 2016

Recall weight space view: $\tilde{\mathcal{K}}(\boldsymbol{x}, \boldsymbol{x}') \approx \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$

$$\tilde{f}(\cdot) \sim \mathcal{GP}(0, \tilde{K}(\cdot,\cdot)) \quad \Leftrightarrow \quad \tilde{f}(\boldsymbol{x}) = \boldsymbol{v}^T cos(W\boldsymbol{x} + \boldsymbol{b}) \quad W, \boldsymbol{b}, \boldsymbol{v} \sim p(W)p(\boldsymbol{b})p(\boldsymbol{v})$$

Gaussian process → Bayesian neural networks

Deep GPs → deep BNNs with bottleneck layers:

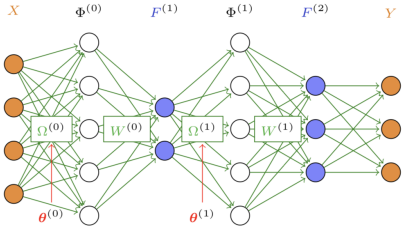
Deep BNN approximation to deep GP:

$$\tilde{f} \approx f, \quad \tilde{f}(\boldsymbol{x}) = \tilde{f}^{(L)} \circ \tilde{f}^{(L-1)} \circ \ldots \circ \tilde{f}^{(0)}(\boldsymbol{x}),$$

$$\tilde{f}^{(i)}(\boldsymbol{x}) = \boldsymbol{v}_i^T cos(W_i \boldsymbol{x} + \boldsymbol{b}_i),$$

$$W_i, \boldsymbol{b}_i, \boldsymbol{v}_i \sim p(W_i)p(\boldsymbol{b}_i)p(\boldsymbol{v}_i),$$

$$\prod_{n=1}^{N} p(\boldsymbol{y}_n | f(\boldsymbol{x}_n))p(\boldsymbol{f}|\mathbf{X}) \approx \prod_{n=1}^{N} p(\boldsymbol{y}_n | \boldsymbol{x}_n, W)p(W)$$



Cutajar et al. 2017

What we have covered today...

Connecting parametric regression models and GPs:

parameter-space view		function-space view
linear regression/ridge regression		GP regression with linear kernel
non-linear regression with pre-defined features		GP regression
randomly initialised neural networks	$\overset{CLT}{\rightarrow}$	GP function-space priors
randomly initialised neural networks	$\overset{random_features}{\approx}$	GP function-space priors

Hot topic of research: connections between NN & GP after training

References

Neal 1994. Bayesian Learning for Neural Networks. PhD thesis

Rahimi and Recht 2007. Random Features for Large-Scale Kernel Machines. NeurIPS 2007

Lázaro-Gredilla et al. 2010. Sparse spectrum Gaussian process regression. JMLR 2010

Duvenaud 2014. Automatic Model Construction with Gaussian Processes. PhD thesis

Bui et al. 2016. Deep Gaussian Processes for Regression using Approximate Expectation Propagation. ICML 2016

Cutajar et al. 2017. Random Feature Expansions for Deep Gaussian Processes. ICML 2017

Matthews et al. 2018. Gaussian Process Behaviour in Wide Deep Neural Networks. ICLR 2018

Lee et al. 2018. Deep Neural Networks as Gaussian Processes. ICLR 2018

Garriga-Alonso et al. 2019. Deep Convolutional Networks as shallow Gaussian Processes. ICLR 2019

Novak et al. 2019. Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes. ICLR 2019