Solving Kernel Ridge Regression with Gradient Descent for a Non-Constant Kernel (after an introduction to kernels)

Oskar Allerbo

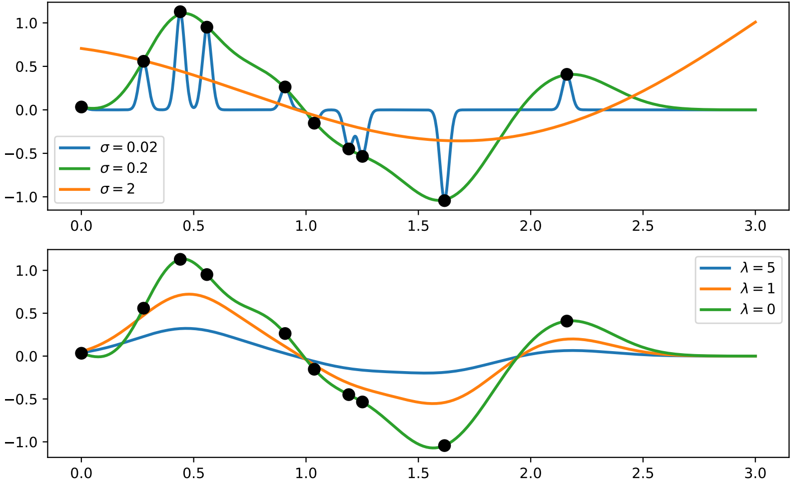
KTH Royal Institute of Technology

2024-03-11

https://github.com/allerbo/hemavan24/pres.pdf
https://github.com/allerbo/hemavan24/slides.pdf

Kernel Ridge Regression

Kernel Ridge Regression



Kernel Ridge Regression

Kernel ridge regression (KRR) is a generalization of linear ridge regression that is

Kernel Ridge Regression

Kernel ridge regression (KRR) is a generalization of linear ridge regression that is

- non-linear in the data,

Kernel Ridge Regression

Kernel ridge regression (KRR) is a generalization of linear ridge regression that is

- non-linear in the data,
- but linear in the parameters.

Kernel Ridge Regression

Kernel ridge regression (KRR) is a generalization of linear ridge regression that is

- non-linear in the data,
- but linear in the parameters.
- a convex problem, with a closed-form solution.

Outline

1. Introduction to Kernels
2. Kernel Principal Component Analysis (KPCA)
3. Kernel Ridge Regression (KRR)
4. Kernel Gradient Descent for Non-constant Kernels (KGD)
5. Generalization to Neural Networks (NNs)
6. Conclusions

What is a Kernel?

A kernel function:

What is a Kernel?

A kernel function:

- Takes two arguments and outputs a scalar: $k(\boldsymbol{x}, \boldsymbol{x'}) \in \mathbb{R}$. $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^p$.

What is a Kernel?

A kernel function:

- Takes two arguments and outputs a scalar:
  $k(\boldsymbol{x}, \boldsymbol{x'}) \in \mathbb{R}$. $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^p$.
- Is symmetric:
  $k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$.

What is a Kernel?

A kernel function:

- Takes two arguments and outputs a scalar:
  $k(\boldsymbol{x}, \boldsymbol{x'}) \in \mathbb{R}$. $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^p$.
- Is symmetric:
  $k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$.
- Is positive semi-definite:
  $\sum_{i=1}^{n} \sum_{i=j}^{n} c_i c_j k(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq 0$. For all $\boldsymbol{x_i}, \boldsymbol{x_j} \in \mathbb{R}^p$, $c_i, c_j \in \mathbb{R}$.

What is a Kernel?

A kernel function:

- Takes two arguments and outputs a scalar:
  $k(\boldsymbol{x}, \boldsymbol{x'}) \in \mathbb{R}$. $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^p$.
- Is symmetric:
  $k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$.
- Is positive semi-definite:
  $\sum_{i=1}^{n} \sum_{i=j}^{n} c_i c_j k(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq 0$. For all $\boldsymbol{x_i}, \boldsymbol{x_j} \in \mathbb{R}^p$, $c_i, c_j \in \mathbb{R}$.
- Is the dot product of the feature expansions of $\boldsymbol{x}$ and $\boldsymbol{x'}$:
  $k(\boldsymbol{x}, \boldsymbol{x'}) = \varphi(\boldsymbol{x})^\top \varphi(\boldsymbol{x'})$. $\varphi(\boldsymbol{x}), \varphi(\boldsymbol{x'}) \in \mathbb{R}^q$.

What is a Kernel?

A kernel function:

- Takes two arguments and outputs a scalar:
  $k(\boldsymbol{x}, \boldsymbol{x'}) \in \mathbb{R}$. $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^p$.
- Is symmetric:
  $k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$.
- Is positive semi-definite:
  $\sum_{i=1}^n \sum_{i=j}^n c_i c_j k(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq 0$. For all $\boldsymbol{x_i}, \boldsymbol{x_j} \in \mathbb{R}^p$, $c_i, c_j \in \mathbb{R}$.
- Is the dot product of the feature expansions of $\boldsymbol{x}$ and $\boldsymbol{x'}$:
  $k(\boldsymbol{x}, \boldsymbol{x'}) = \varphi(\boldsymbol{x})^\top \varphi(\boldsymbol{x'})$. $\varphi(\boldsymbol{x}), \varphi(\boldsymbol{x'}) \in \mathbb{R}^q$.
  $q = \infty$ is a possibility!

What is a Kernel?

A kernel function:

- Takes two arguments and outputs a scalar:
  $k(\boldsymbol{x}, \boldsymbol{x'}) \in \mathbb{R}$. $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{R}^p$.
- Is symmetric:
  $k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$.
- Is positive semi-definite:
  $\sum_{i=1}^n \sum_{i=j}^n c_i c_j k(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq 0$. For all $\boldsymbol{x_i}, \boldsymbol{x_j} \in \mathbb{R}^p$, $c_i, c_j \in \mathbb{R}$.
- Is the dot product of the feature expansions of $\boldsymbol{x}$ and $\boldsymbol{x'}$:
  $k(\boldsymbol{x}, \boldsymbol{x'}) = \varphi(\boldsymbol{x})^\top \varphi(\boldsymbol{x'})$. $\varphi(\boldsymbol{x}), \varphi(\boldsymbol{x'}) \in \mathbb{R}^q$.
  $q = \infty$ is a possibility!
- (Is associated to a Reproducing Kernel Hilbert Space, RKHS.)

Examples of Kernels

Examples of Kernels

- Linear: $k(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{x}^\top \boldsymbol{x'} + c$, $c \in \mathbb{R}$

Examples of Kernels

- Linear: $k(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{x}^\top \boldsymbol{x'} + c$, $c \in \mathbb{R}$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x}^\top \boldsymbol{x'} + c)^q$, $c \in \mathbb{R}$, $q \in \mathbb{N}$

Examples of Kernels

- Linear: $k(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{x}^\top \boldsymbol{x'} + c$, $c \in \mathbb{R}$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x}^\top \boldsymbol{x'} + c)^q$, $c \in \mathbb{R}$, $q \in \mathbb{N}$
- Gaussian: $k(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2^2}{2\sigma^2}}$, $\sigma > 0$

Examples of Kernels

- Linear: $k(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{x}^\top \boldsymbol{x'} + c$, $c \in \mathbb{R}$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x}^\top \boldsymbol{x'} + c)^q$, $c \in \mathbb{R}$, $q \in \mathbb{N}$
- Gaussian: $k(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2^2}{2\sigma^2}}$, $\sigma > 0$
- Laplace: $k(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}}$, $\sigma > 0$

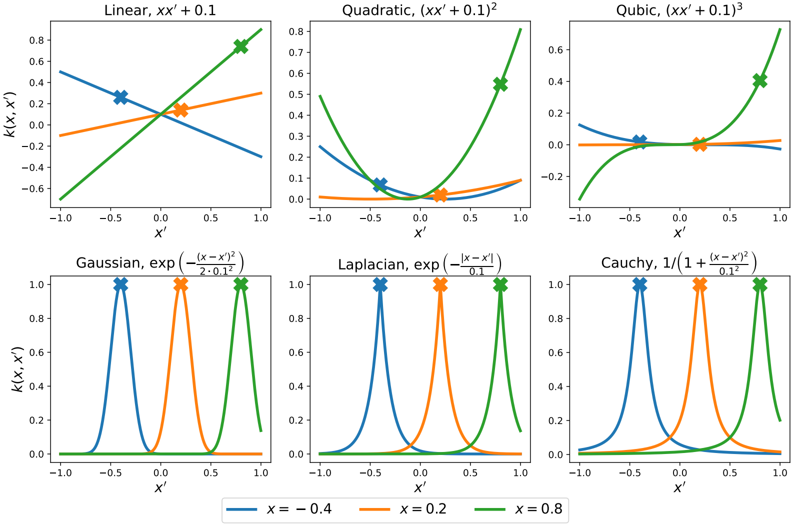
Examples of Kernels

- Linear: $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}' + c$, $c \in \mathbb{R}$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^\top \boldsymbol{x}' + c)^q$, $c \in \mathbb{R}$, $q \in \mathbb{N}$
- Gaussian: $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2}}$, $\sigma > 0$
- Laplace: $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2}{\sigma}}$, $\sigma > 0$
- Cauchy: $k(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{1+\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{\sigma^2}}$, $\sigma > 0$

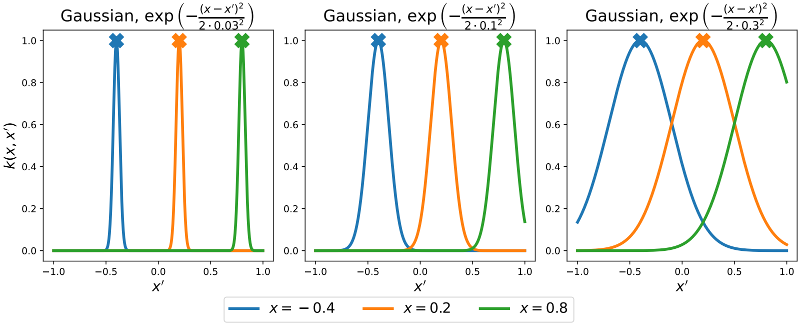
Examples of Kernels

- Linear: $k(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{x}^\top \boldsymbol{x'} + c$, $c \in \mathbb{R}$
- Polynomial: $k(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x}^\top \boldsymbol{x'} + c)^q$, $c \in \mathbb{R}$, $q \in \mathbb{N}$
- Gaussian: $k(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2^2}{2\sigma^2}}$, $\sigma > 0$
- Laplace: $k(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}}$, $\sigma > 0$
- Cauchy: $k(\boldsymbol{x}, \boldsymbol{x'}) = \frac{1}{1+\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2^2}{\sigma^2}}$, $\sigma > 0$
- Matérn: $k(\boldsymbol{x}, \boldsymbol{x'}) = \frac{2^{1-\nu}}{\Gamma(\nu)} \cdot \left(\sqrt{2\nu} \cdot \frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}\right)^\nu \cdot K_\nu \left(\sqrt{2\nu} \cdot \frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}\right)$

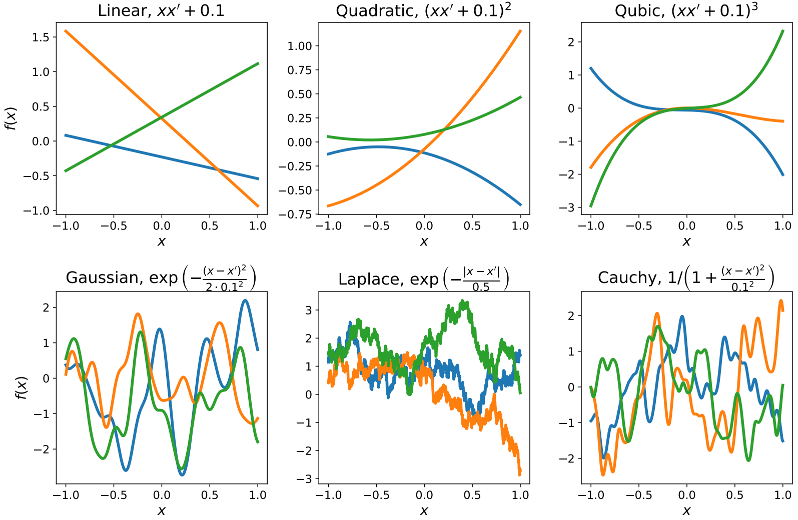
Examples of Kernels (1D Case)



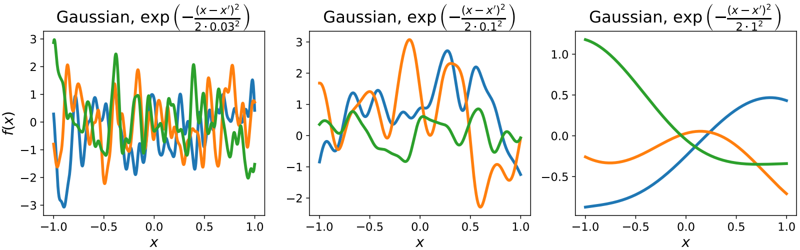
Examples of Kernels (1D Case)



Examples of Functions (1D Case)



Examples of Functions (1D Case)



Examples of Feature Expansions (1D Case)

Examples of Feature Expansions (1D Case)

- Linear, $k(x, x') = x \cdot x' + c$
  $\varphi(x) = \begin{bmatrix} x & \sqrt{c} \end{bmatrix}^\top$

Examples of Feature Expansions (1D Case)

- Linear, $k(x, x') = x \cdot x' + c$
  $\varphi(x) = \begin{bmatrix} x & \sqrt{c} \end{bmatrix}^\top$
  $\varphi(x)^\top \varphi(x') = x \cdot x' + \sqrt{c} \cdot \sqrt{c}$

Examples of Feature Expansions (1D Case)

- Linear, $k(x, x') = x \cdot x' + c$
  $\varphi(x) = \begin{bmatrix} x & \sqrt{c} \end{bmatrix}^\top$
  $\varphi(x)^\top \varphi(x') = x \cdot x' + \sqrt{c} \cdot \sqrt{c}$
- Quadratic, $k(x, x') = (x \cdot x' + c)^2$
  $\varphi(x) = \begin{bmatrix} x^2 & \sqrt{2c}x & c \end{bmatrix}^\top$

Examples of Feature Expansions (1D Case)

- Linear, $k(x, x') = x \cdot x' + c$
  $\varphi(x) = \begin{bmatrix} x & \sqrt{c} \end{bmatrix}^\top$
  $\varphi(x)^\top \varphi(x') = x \cdot x' + \sqrt{c} \cdot \sqrt{c}$
- Quadratic, $k(x, x') = (x \cdot x' + c)^2$
  $\varphi(x) = \begin{bmatrix} x^2 & \sqrt{2c}x & c \end{bmatrix}^\top$
  $\varphi(x)^\top \varphi(x') = x^2 \cdot x'^2 + 2c \cdot x \cdot x' + c \cdot c$

Examples of Feature Expansions (1D Case)

- Linear, $k(x, x') = x \cdot x' + c$
  $\varphi(x) = \begin{bmatrix} x & \sqrt{c} \end{bmatrix}^\top$
  $\varphi(x)^\top \varphi(x') = x \cdot x' + \sqrt{c} \cdot \sqrt{c}$
- Quadratic, $k(x, x') = (x \cdot x' + c)^2$
  $\varphi(x) = \begin{bmatrix} x^2 & \sqrt{2c}x & c \end{bmatrix}^\top$
  $\varphi(x)^\top \varphi(x') = x^2 \cdot x'^2 + 2c \cdot x \cdot x' + c \cdot c$
- Gaussian, $k(x, x') = e^{-\frac{(x-x')^2}{2\sigma^2}}$
  $\varphi(x) = e^{-\frac{x^2}{2\sigma^2}} \begin{bmatrix} 1 & \frac{x^1}{\sigma^1\sqrt{1!}} & \dots & \frac{x^k}{\sigma^k\sqrt{k!}} \dots \end{bmatrix}^\top$

Examples of Feature Expansions (1D Case)

- Linear, $k(x, x') = x \cdot x' + c$
  $\varphi(x) = \begin{bmatrix} x & \sqrt{c} \end{bmatrix}^\top$
  $\varphi(x)^\top \varphi(x') = x \cdot x' + \sqrt{c} \cdot \sqrt{c}$
- Quadratic, $k(x, x') = (x \cdot x' + c)^2$
  $\varphi(x) = \begin{bmatrix} x^2 & \sqrt{2c}x & c \end{bmatrix}^\top$
  $\varphi(x)^\top \varphi(x') = x^2 \cdot x'^2 + 2c \cdot x \cdot x' + c \cdot c$
- Gaussian, $k(x, x') = e^{-\frac{(x-x')^2}{2\sigma^2}}$
  $\varphi(x) = e^{-\frac{x^2}{2\sigma^2}} \begin{bmatrix} 1 & \frac{x^1}{\sigma^1\sqrt{1!}} & \dots & \frac{x^k}{\sigma^k\sqrt{k!}} \dots \end{bmatrix}^\top$
  $$\varphi(x)^\top \varphi(x') = e^{-\frac{1}{2\sigma^2}(x^2+x'^2)} \cdot \underbrace{\sum_{k=0}^{\infty} \frac{(x \cdot x'/\sigma^2)^k}{k!}}_{=e^{2\cdot\frac{x\cdot x'}{2\sigma^2}}}$$

Notation

Notation

Training Data: $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, $\boldsymbol{y} \in \mathbb{R}^{n}$.
New Observations: $\boldsymbol{X}^* \in \mathbb{R}^{n^* \times p}$.
Predictions: $\hat{\boldsymbol{f}} \in \mathbb{R}^{n}$, $\hat{\boldsymbol{f}}^* \in \mathbb{R}^{n^*}$.

Notation

Training Data: $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, $\boldsymbol{y} \in \mathbb{R}^{n}$.
New Observations: $\boldsymbol{X}^* \in \mathbb{R}^{n^* \times p}$.
Predictions: $\hat{\boldsymbol{f}} \in \mathbb{R}^{n}$, $\hat{\boldsymbol{f}}^* \in \mathbb{R}^{n^*}$.

Feature Expansion Matrices:
$\boldsymbol{\Phi} = \boldsymbol{\Phi}(\boldsymbol{X}) \in \mathbb{R}^{n \times q}$, $\boldsymbol{\Phi}^* = \boldsymbol{\Phi}^*(\boldsymbol{X}^*) \in \mathbb{R}^{n^* \times q}$.

Notation

Training Data: $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, $\boldsymbol{y} \in \mathbb{R}^{n}$.
New Observations: $\boldsymbol{X}^* \in \mathbb{R}^{n^* \times p}$.
Predictions: $\hat{\boldsymbol{f}} \in \mathbb{R}^{n}$, $\hat{\boldsymbol{f}}^* \in \mathbb{R}^{n^*}$.

Feature Expansion Matrices:
$\boldsymbol{\Phi} = \boldsymbol{\Phi}(\boldsymbol{X}) \in \mathbb{R}^{n \times q}$, $\boldsymbol{\Phi}^* = \boldsymbol{\Phi}^*(\boldsymbol{X}^*) \in \mathbb{R}^{n^* \times q}$.

Kernel Matrices:
$\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{X}) \in \mathbb{R}^{n \times n}$, $\boldsymbol{K}^* = \boldsymbol{K}(\boldsymbol{X}^*, \boldsymbol{X}) \in \mathbb{R}^{n^* \times n}$.
$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$, $\boldsymbol{K}^* = \boldsymbol{\Phi}^*\boldsymbol{\Phi}^\top$.
$k(\boldsymbol{x}, \boldsymbol{x}') = \varphi(\boldsymbol{x})^\top \varphi(\boldsymbol{x}')$.

Notation

$$\boldsymbol{X} = \begin{bmatrix} — & \boldsymbol{x}_1^\top & — \\ — & \boldsymbol{x}_2^\top & — \\ \vdots & \vdots & \vdots \\ — & \boldsymbol{x}_n^\top & — \end{bmatrix} \quad \boldsymbol{X}^* = \begin{bmatrix} — & \boldsymbol{x}_1^{*\top} & — \\ — & \boldsymbol{x}_2^{*\top} & — \\ \vdots & \vdots & \vdots \\ — & \boldsymbol{x}_{n^*}^{*\top} & — \end{bmatrix}$$

$$\boldsymbol{\Phi} = \begin{bmatrix} — & \varphi(\boldsymbol{x}_1)^\top & — \\ — & \varphi(\boldsymbol{x}_2)^\top & — \\ \vdots & \vdots & \vdots \\ — & \varphi(\boldsymbol{x}_n)^\top & — \end{bmatrix} \quad \boldsymbol{\Phi}^* = \begin{bmatrix} — & \varphi(\boldsymbol{x}_1^*)^\top & — \\ — & \varphi(\boldsymbol{x}_2^*)^\top & — \\ \vdots & \vdots & \vdots \\ — & \varphi(\boldsymbol{x}_{n^*}^*)^\top & — \end{bmatrix}$$

$$\boldsymbol{\Phi}^*\boldsymbol{\Phi}^\top = \boldsymbol{K}^* = \begin{bmatrix} k(\boldsymbol{x}_1^*, \boldsymbol{x}_1) & k(\boldsymbol{x}_1^*, \boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_1^*, \boldsymbol{x}_n) \\ k(\boldsymbol{x}_2^*, \boldsymbol{x}_1) & k(\boldsymbol{x}_2^*, \boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_2^*, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_{n^*}^*, \boldsymbol{x}_1) & k(\boldsymbol{x}_{n^*}^*, \boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_{n^*}^*, \boldsymbol{x}_n) \end{bmatrix}$$

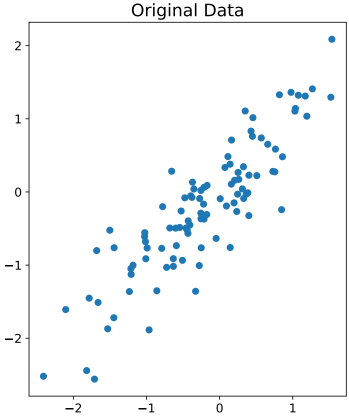
Kernel Principal Component Analysis

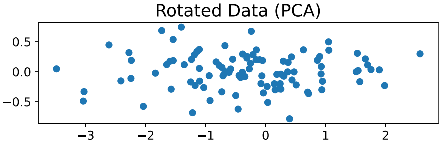
Kernel Principal Component Analysis

Principal Component Analysis (PCA):
Rotate data to find directions with maximum variance.

Kernel Principal Component Analysis

Principal Component Analysis (PCA):
Rotate data to find directions with maximum variance.





Kernel Principal Component Analysis

Principal Component Analysis (PCA):
Rotate data to find directions with maximum variance.

Kernel Principal Component Analysis

Principal Component Analysis (PCA):
Rotate data to find directions with maximum variance.
$\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top$, $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{P}$.

Kernel Principal Component Analysis

Principal Component Analysis (PCA):
Rotate data to find directions with maximum variance.
$\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top$, $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{P}$.

Equivalent, dual formulation:
$\boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^\top$, $\boldsymbol{Z} = \boldsymbol{U}\sqrt{\boldsymbol{D}}$.

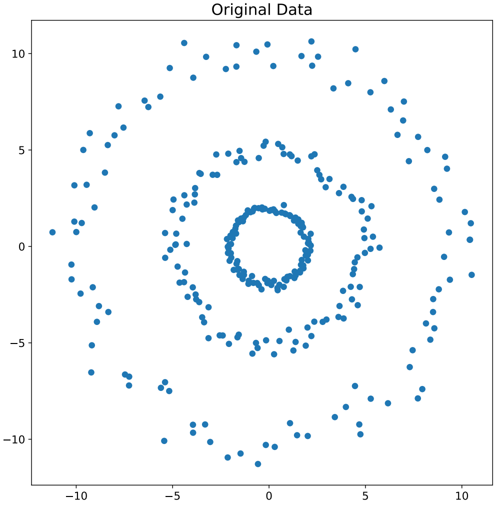
Kernel Principal Component Analysis

Principal Component Analysis (PCA):
Rotate data to find directions with maximum variance.
$\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{PDP}^\top$, $\boldsymbol{Z} = \boldsymbol{XP}$.

Equivalent, dual formulation:
$\boldsymbol{XX}^\top = \boldsymbol{UDU}^\top$, $\boldsymbol{Z} = \boldsymbol{U}\sqrt{\boldsymbol{D}}$.

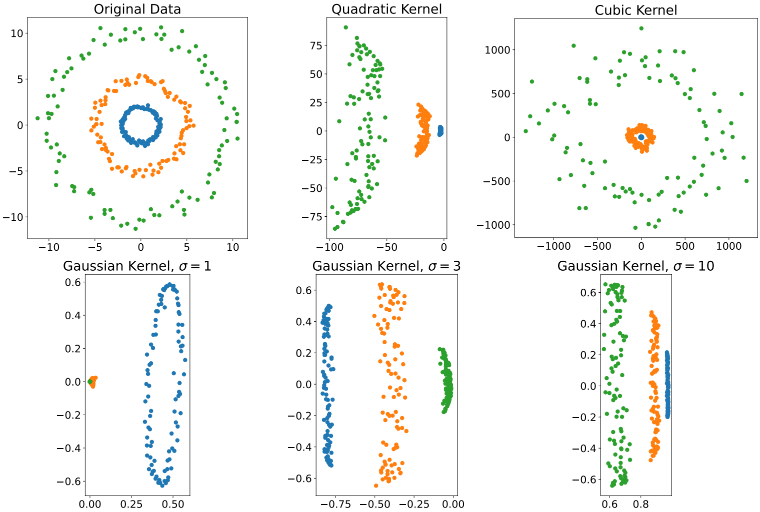
Kernel PCA:
$\boldsymbol{\Phi\Phi}^\top = \boldsymbol{K} = \boldsymbol{U_K D_K U_K^\top}$, $\boldsymbol{Z_K} = \boldsymbol{U_K}\sqrt{\boldsymbol{D_K}}$.

Kernel Principal Component Analysis

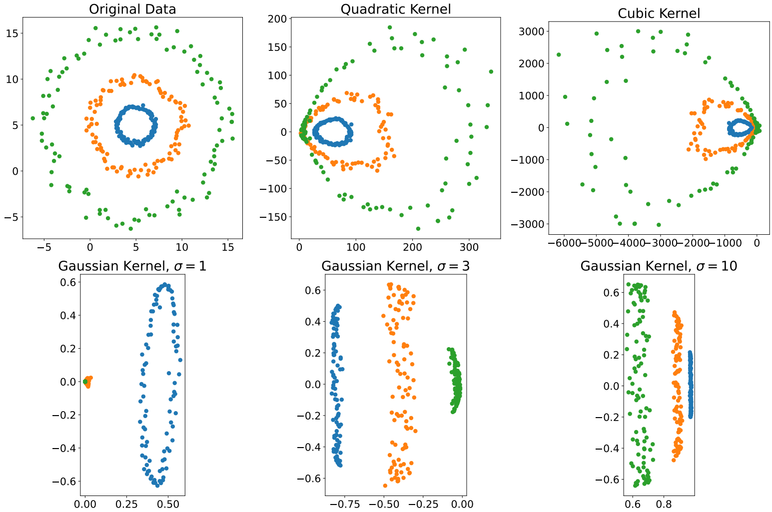


What about this?

Kernel Principal Component Analysis



Kernel Principal Component Analysis



Kernel Ridge Regression

Kernel Ridge Regression

- Linear Ridge Regression

Kernel Ridge Regression

- Linear Ridge Regression
- Dual Formulation of Linear Ridge Regression

Kernel Ridge Regression

- Linear Ridge Regression
- Dual Formulation of Linear Ridge Regression
- Ridge Regression in Feature Space

Kernel Ridge Regression

- Linear Ridge Regression
- Dual Formulation of Linear Ridge Regression
- Ridge Regression in Feature Space
- Dual Formulation of Ridge Regression in Feature Space

Kernel Ridge Regression

- Linear Ridge Regression
- Dual Formulation of Linear Ridge Regression
- Ridge Regression in Feature Space
- Dual Formulation of Ridge Regression in Feature Space (=Kernel Ridge Regression)

Linear Ridge Regression

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2} \underbrace{\|\boldsymbol{\beta}\|_2^2}_{=\boldsymbol{\beta}^\top \boldsymbol{\beta}}$$

Linear Ridge Regression

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2} \underbrace{\|\boldsymbol{\beta}\|_2^2}_{=\boldsymbol{\beta}^\top \boldsymbol{\beta}}$$

$$= \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I}_{p \times p}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

Linear Ridge Regression

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2}\underbrace{\|\boldsymbol{\beta}\|_2^2}_{=\boldsymbol{\beta}^\top\boldsymbol{\beta}}$$

$$= \left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

Predictions are given by

$$\begin{bmatrix}\hat{\boldsymbol{f}}\\ \hat{\boldsymbol{f}}^*\end{bmatrix} = \begin{bmatrix}\boldsymbol{X}\hat{\boldsymbol{\beta}}\\ \boldsymbol{X}^*\hat{\boldsymbol{\beta}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}\\ \boldsymbol{X}^*\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}\end{bmatrix}.$$

Dual Formulation of Linear Ridge Regression

Dual formulation for $\boldsymbol{\beta} = \boldsymbol{X}^\top \boldsymbol{\alpha}$:

Dual Formulation of Linear Ridge Regression

Dual formulation for $\boldsymbol{\beta} = \boldsymbol{X}^\top \boldsymbol{\alpha}$:

$$\hat{\boldsymbol{\alpha}} = \operatorname*{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{2} \left\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{\alpha} \right\|_2^2 + \frac{\lambda}{2} \underbrace{\boldsymbol{\alpha}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{\alpha}}_{= \|\boldsymbol{\alpha}\|^2_{\boldsymbol{X}\boldsymbol{X}^\top}}$$

Dual Formulation of Linear Ridge Regression

Dual formulation for $\boldsymbol{\beta} = \boldsymbol{X}^\top \boldsymbol{\alpha}$:

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \left\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{\alpha} \right\|_2^2 + \frac{\lambda}{2} \underbrace{\boldsymbol{\alpha}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{\alpha}}_{= \|\boldsymbol{\alpha}\|^2_{\boldsymbol{X}\boldsymbol{X}^\top}}$$

$$= \left(\boldsymbol{X}\boldsymbol{X}^\top + \lambda \boldsymbol{I}_{n \times n} \right)^{-1} \boldsymbol{y}$$

Dual Formulation of Linear Ridge Regression

Dual formulation for $\boldsymbol{\beta} = \boldsymbol{X}^\top \boldsymbol{\alpha}$:

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \left\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{\alpha} \right\|_2^2 + \frac{\lambda}{2} \underbrace{\boldsymbol{\alpha}^\top \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{\alpha}}_{= \|\boldsymbol{\alpha}\|^2_{\boldsymbol{X}\boldsymbol{X}^\top}}$$

$$= \left(\boldsymbol{X}\boldsymbol{X}^\top + \lambda \boldsymbol{I}_{n \times n} \right)^{-1} \boldsymbol{y}$$

Predictions are given by

$$\begin{bmatrix} \hat{\boldsymbol{f}} \\ \hat{\boldsymbol{f}}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{X} \cdot \boldsymbol{X}^\top \hat{\boldsymbol{\alpha}} \\ \boldsymbol{X}^* \cdot \boldsymbol{X}^\top \hat{\boldsymbol{\alpha}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}\boldsymbol{X}^\top \left(\boldsymbol{X}\boldsymbol{X}^\top + \lambda \boldsymbol{I}_{n \times n} \right)^{-1} \boldsymbol{y} \\ \boldsymbol{X}^* \boldsymbol{X}^\top \left(\boldsymbol{X}\boldsymbol{X}^\top + \lambda \boldsymbol{I}_{n \times n} \right)^{-1} \boldsymbol{y} \end{bmatrix}.$$

Linear Ridge Regression

Predictions given by

$$\begin{bmatrix} \boldsymbol{X}\hat{\beta} \\ \boldsymbol{X}^*\hat{\beta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y} \\ \boldsymbol{X}^*\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y} \end{bmatrix}$$

and

$$\begin{bmatrix} \boldsymbol{X}\boldsymbol{X}^\top\hat{\alpha} \\ \boldsymbol{X}^*\boldsymbol{X}^\top\hat{\alpha} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}\boldsymbol{X}^\top\left(\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I}_{n\times n}\right)^{-1}\boldsymbol{y} \\ \boldsymbol{X}^*\boldsymbol{X}^\top\left(\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I}_{n\times n}\right)^{-1}\boldsymbol{y} \end{bmatrix}.$$

Linear Ridge Regression

Predictions given by

$$\begin{bmatrix} \boldsymbol{X}\hat{\beta} \\ \boldsymbol{X}^*\hat{\beta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y} \\ \boldsymbol{X}^*\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y} \end{bmatrix}$$

and

$$\begin{bmatrix} \boldsymbol{X}\boldsymbol{X}^\top\hat{\alpha} \\ \boldsymbol{X}^*\boldsymbol{X}^\top\hat{\alpha} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}\boldsymbol{X}^\top\left(\boldsymbol{X}\boldsymbol{X}^\top + \lambda \boldsymbol{I}_{n\times n}\right)^{-1}\boldsymbol{y} \\ \boldsymbol{X}^*\boldsymbol{X}^\top\left(\boldsymbol{X}\boldsymbol{X}^\top + \lambda \boldsymbol{I}_{n\times n}\right)^{-1}\boldsymbol{y} \end{bmatrix}.$$

However,

$$\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda \boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{X}^\top = \boldsymbol{X}^\top\left(\boldsymbol{X}\boldsymbol{X}^\top + \lambda \boldsymbol{I}_{n\times n}\right)^{-1}.$$

Ridge Regression in Feature Space

$$\boldsymbol{x} \in \mathbb{R}^p \mapsto \varphi(\boldsymbol{x}) \in \mathbb{R}^q.$$

Ridge Regression in Feature Space

$\boldsymbol{x} \in \mathbb{R}^p \mapsto \varphi(\boldsymbol{x}) \in \mathbb{R}^q$.
E.g. polynomial regression, $x \mapsto [1,\ x,\ x^2,\ \ldots,\ x^{q\text{-}1}]$.

Ridge Regression in Feature Space

$\boldsymbol{x} \in \mathbb{R}^p \mapsto \varphi(\boldsymbol{x}) \in \mathbb{R}^q$.
E.g. polynomial regression, $x \mapsto [1,\ x,\ x^2,\ \ldots,\ x^{q\text{-}1}]$.
$\boldsymbol{X} \in \mathbb{R}^{n\times p} \mapsto \boldsymbol{\Phi} \in \mathbb{R}^{n\times q}$, $\boldsymbol{X}^* \in \mathbb{R}^{n^*\times p} \mapsto \boldsymbol{\Phi}^* \in \mathbb{R}^{n^*\times q}$

Ridge Regression in Feature Space

$\boldsymbol{x} \in \mathbb{R}^p \mapsto \varphi(\boldsymbol{x}) \in \mathbb{R}^q$.
E.g. polynomial regression, $x \mapsto [1,\ x,\ x^2,\ \ldots,\ x^{q\text{-}1}]$.
$\boldsymbol{X} \in \mathbb{R}^{n\times p} \mapsto \boldsymbol{\Phi} \in \mathbb{R}^{n\times q}$, $\boldsymbol{X}^* \in \mathbb{R}^{n^*\times p} \mapsto \boldsymbol{\Phi}^* \in \mathbb{R}^{n^*\times q}$

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^q}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2$$

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\alpha}\right\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\alpha}\|_{\boldsymbol{\Phi}\boldsymbol{\Phi}^\top}^2$$

Ridge Regression in Feature Space

$\boldsymbol{x} \in \mathbb{R}^p \mapsto \varphi(\boldsymbol{x}) \in \mathbb{R}^q$.
E.g. polynomial regression, $x \mapsto [1,\ x,\ x^2,\ \ldots,\ x^{q\text{-}1}]$.
$\boldsymbol{X} \in \mathbb{R}^{n\times p} \mapsto \boldsymbol{\Phi} \in \mathbb{R}^{n\times q}$, $\boldsymbol{X}^* \in \mathbb{R}^{n^*\times p} \mapsto \boldsymbol{\Phi}^* \in \mathbb{R}^{n^*\times q}$

$$\hat{\beta} = \underset{\boldsymbol{\beta}\in\mathbb{R}^q}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi}\beta\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2$$

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\alpha}\right\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\alpha}\|_{\boldsymbol{\Phi}\boldsymbol{\Phi}^\top}^2$$

Predictions are given by

$$\begin{bmatrix}\hat{\boldsymbol{f}}\\ \hat{\boldsymbol{f}}^*\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y}\\ \boldsymbol{\Phi}^*\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{I}_{p\times p}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda\boldsymbol{I}_{n\times n}\right)^{-1}\boldsymbol{y}\\ \boldsymbol{\Phi}^*\boldsymbol{\Phi}^\top\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda\boldsymbol{I}_{n\times n}\right)^{-1}\boldsymbol{y}\end{bmatrix}$$

Kernel Ridge Regression

For $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \in \mathbb{R}^{n \times n}$, $\boldsymbol{K}^* = \boldsymbol{\Phi}^*\boldsymbol{\Phi}^\top \in \mathbb{R}^{n^* \times n}$,

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\alpha}\|_{\boldsymbol{K}}^2$$

$$\begin{bmatrix} \hat{\boldsymbol{f}} \\ \hat{\boldsymbol{f}}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{K}^* \end{bmatrix} \hat{\boldsymbol{\alpha}} = \begin{bmatrix} \boldsymbol{K}\left(\boldsymbol{K} + \lambda \boldsymbol{I}_{n \times n}\right)^{-1} \boldsymbol{y} \\ \boldsymbol{K}^*\left(\boldsymbol{K} + \lambda \boldsymbol{I}_{n \times n}\right)^{-1} \boldsymbol{y} \end{bmatrix}$$

Kernel Ridge Regression

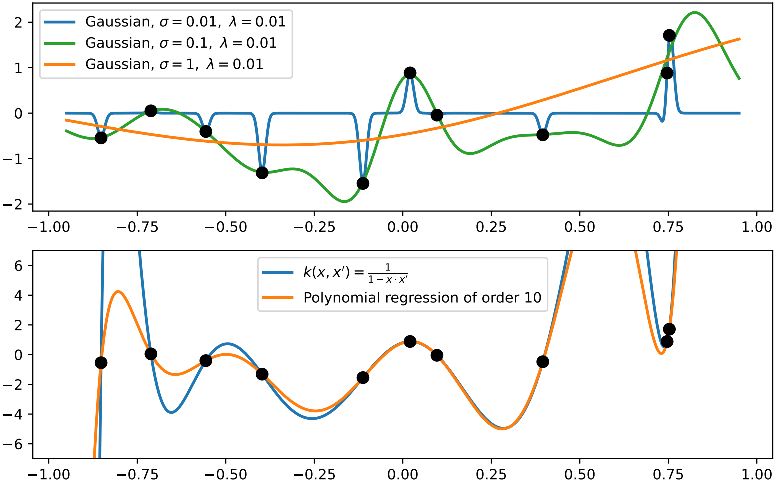
For $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \in \mathbb{R}^{n \times n}$, $\boldsymbol{K}^* = \boldsymbol{\Phi}^*\boldsymbol{\Phi}^\top \in \mathbb{R}^{n^* \times n}$,

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\alpha}\|_{\boldsymbol{K}}^2$$

$$\begin{bmatrix} \hat{\boldsymbol{f}} \\ \hat{\boldsymbol{f}}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{K}^* \end{bmatrix} \hat{\boldsymbol{\alpha}} = \begin{bmatrix} \boldsymbol{K} \left(\boldsymbol{K} + \lambda \boldsymbol{I}_{n \times n}\right)^{-1} \boldsymbol{y} \\ \boldsymbol{K}^* \left(\boldsymbol{K} + \lambda \boldsymbol{I}_{n \times n}\right)^{-1} \boldsymbol{y} \end{bmatrix}$$

$q = \infty$ is OK, since $\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}^*$ are never explicitly calculated.

Kernel Ridge Regression



$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \dots, \ |x| < 1$$

Kernel Gradient Descent for Non-Constant Kernels

Kernel Gradient Descent for Non-Constant Kernels

Kernel gradient descent in function/prediction space:
(with regularization through early stopping)

$$\begin{bmatrix} \hat{\boldsymbol{f}}_{t+1} \\ \hat{\boldsymbol{f}}^*_{t+1} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}_{t} \\ \hat{\boldsymbol{f}}^*_{t} \end{bmatrix} - \eta \cdot \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{K}^* \end{bmatrix} (\hat{\boldsymbol{f}}_t - \boldsymbol{y})$$

$$\left(\text{where } \begin{bmatrix} \hat{\boldsymbol{f}} \\ \hat{\boldsymbol{f}}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{K}^* \end{bmatrix} \hat{\boldsymbol{\alpha}}\right)$$

Kernel Gradient Descent for Non-Constant Kernels

Kernel gradient descent in function/prediction space:
(with regularization through early stopping)

$$\begin{bmatrix} \hat{\boldsymbol{f}}_{t+1} \\ \hat{\boldsymbol{f}}^*_{t+1} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}_t \\ \hat{\boldsymbol{f}}^*_t \end{bmatrix} - \eta \cdot \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{K}^* \end{bmatrix} (\hat{\boldsymbol{f}}_t - \boldsymbol{y})$$

$$\left(\text{where } \begin{bmatrix} \hat{\boldsymbol{f}} \\ \hat{\boldsymbol{f}}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{K}^* \end{bmatrix} \hat{\alpha}\right)$$

Time dependent kernels:

$$\begin{bmatrix} \hat{\boldsymbol{f}}_{t+1} \\ \hat{\boldsymbol{f}}^*_{t+1} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}_t \\ \hat{\boldsymbol{f}}^*_t \end{bmatrix} - \eta \cdot \begin{bmatrix} \boldsymbol{K}_t \\ \boldsymbol{K}^*_t \end{bmatrix} (\hat{\boldsymbol{f}}_t - \boldsymbol{y})$$

Kernel Regression with Gradient Descent and Non-Constant Kernels

Proposition

For a translational invariant kernel with bandwidth σ, $k(\boldsymbol{x},\boldsymbol{x'},\sigma) = k\left(\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}\right)$, and for constant training time, t,

$$\left\|\nabla_{\boldsymbol{x}^*}\hat{f}(\boldsymbol{x}^*,t)\right\|_2 \leq \frac{1}{\sigma}\cdot t\cdot C(k(\cdot),\boldsymbol{X},\boldsymbol{y})$$

i.e. a larger bandwidth results in a simpler model, and a smaller bandwidth in a more complex model.

Kernel Regression with Gradient Descent and Non-Constant Kernels

Proposition

For a translational invariant kernel with bandwidth σ*,* $k(\boldsymbol{x}, \boldsymbol{x'}, \sigma) = k\left(\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}\right)$*, and for constant training time,* t*,*

$$\left\|\nabla_{\boldsymbol{x}^*} \hat{f}(\boldsymbol{x}^*, t)\right\|_2 \leq \frac{1}{\sigma} \cdot t \cdot C(k(\cdot), \boldsymbol{X}, \boldsymbol{y})$$

i.e. a larger bandwidth results in a simpler model, and a smaller bandwidth in a more complex model.
We use $1/\sigma$ as a proxy for complexity.

Kernel Regression with Gradient Descent and Non-Constant Kernels

Proposition

For a translational invariant kernel with bandwidth σ, $k(\boldsymbol{x},\boldsymbol{x'},\sigma)=k\left(\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}\right)$, and for constant training time, t,

$$\left\|\nabla_{\boldsymbol{x}^*}\hat{f}(\boldsymbol{x}^*,t)\right\|_2 \leq \frac{1}{\sigma}\cdot t\cdot C(k(\cdot),\boldsymbol{X},\boldsymbol{y})$$

i.e. a larger bandwidth results in a simpler model, and a smaller bandwidth in a more complex model.
We use $1/\sigma$ as a proxy for complexity.

$$\begin{bmatrix}\hat{\boldsymbol{f}}_{t+1}\\ \hat{\boldsymbol{f}}^*_{t+1}\end{bmatrix}=\begin{bmatrix}\hat{\boldsymbol{f}}_{t}\\ \hat{\boldsymbol{f}}^*_{t}\end{bmatrix}-\eta\cdot\begin{bmatrix}\boldsymbol{K}(\sigma_t)\\ \boldsymbol{K}^*(\sigma_t)\end{bmatrix}(\hat{\boldsymbol{f}}_t-\boldsymbol{y})$$

Kernel Regression with Gradient Descent and Non-Constant Kernels

Proposition

For a translational invariant kernel with bandwidth σ, $k(\boldsymbol{x}, \boldsymbol{x'}, \sigma) = k\left(\frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}\right)$, *and for constant training time, t,*

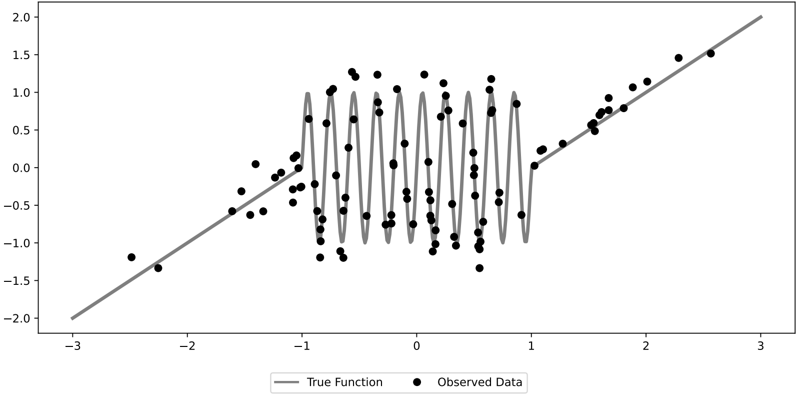
$$\left\|\nabla_{\boldsymbol{x}^*} \hat{f}(\boldsymbol{x}^*, t)\right\|_2 \leq \frac{1}{\sigma} \cdot t \cdot C(k(\cdot), \boldsymbol{X}, \boldsymbol{y})$$

i.e. a larger bandwidth results in a simpler model, and a smaller bandwidth in a more complex model.
We use $1/\sigma$ as a proxy for complexity.

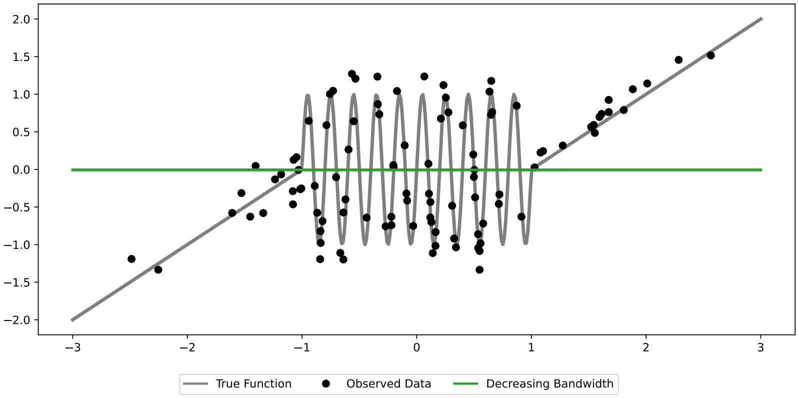
$$\begin{bmatrix} \hat{\boldsymbol{f}}_{t+1} \\ \hat{\boldsymbol{f}}^*_{t+1} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}_t \\ \hat{\boldsymbol{f}}^*_t \end{bmatrix} - \eta \cdot \begin{bmatrix} \boldsymbol{K}(\sigma_t) \\ \boldsymbol{K}^*(\sigma_t) \end{bmatrix} (\hat{\boldsymbol{f}}_t - \boldsymbol{y})$$

Idea: Start with a kernel with large bandwidth (a simple model).
Gradually decrease the bandwidth towards zero during training.

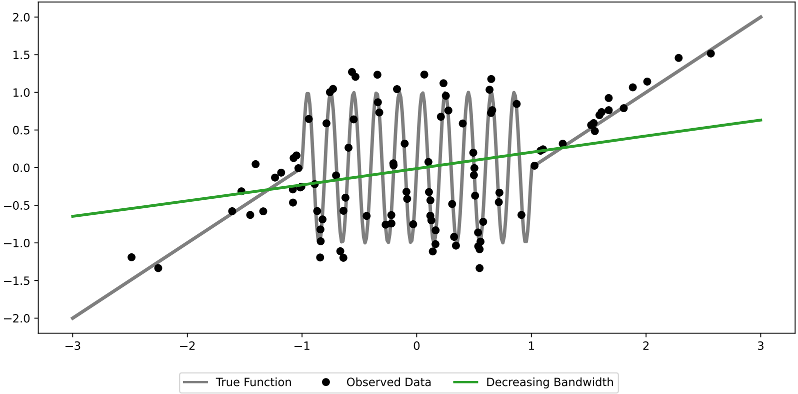
Kernel Regression with Gradient Descent and Non-Constant Kernels



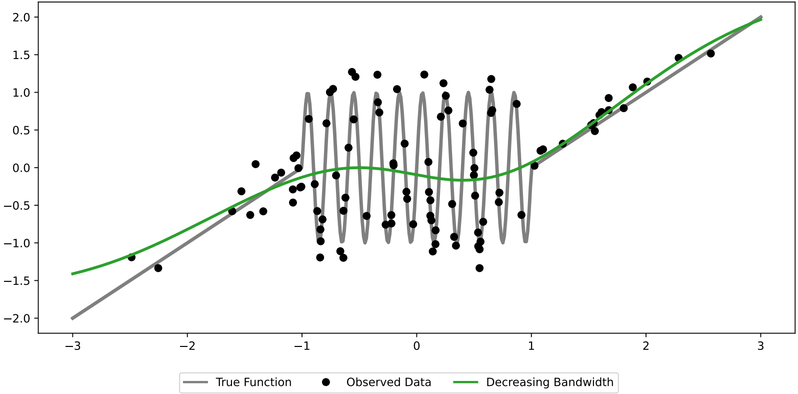
Kernel Regression with Gradient Descent and Non-Constant Kernels



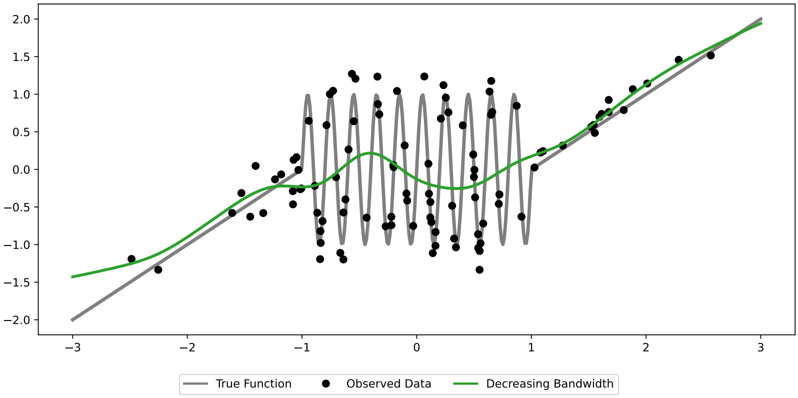
Kernel Regression with Gradient Descent and Non-Constant Kernels



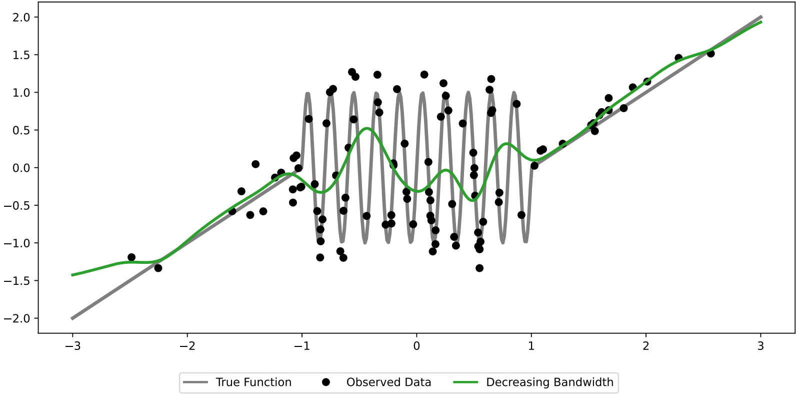
Kernel Regression with Gradient Descent and Non-Constant Kernels



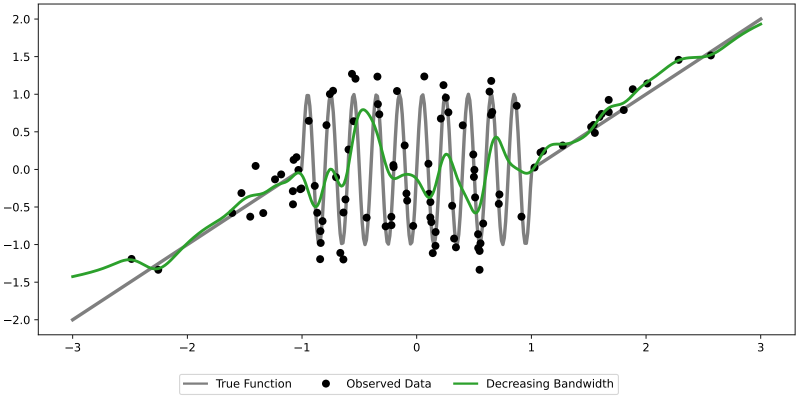
Kernel Regression with Gradient Descent and Non-Constant Kernels



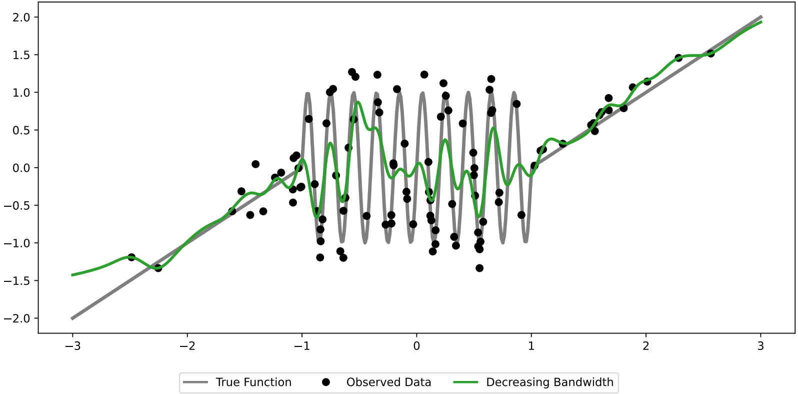
Kernel Regression with Gradient Descent and Non-Constant Kernels



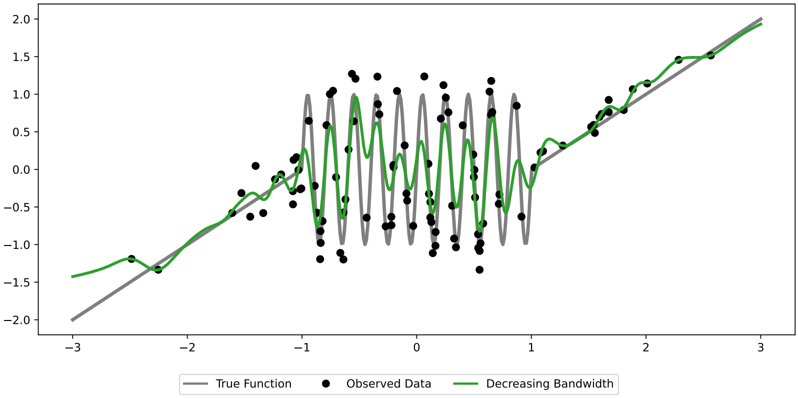
Kernel Regression with Gradient Descent and Non-Constant Kernels



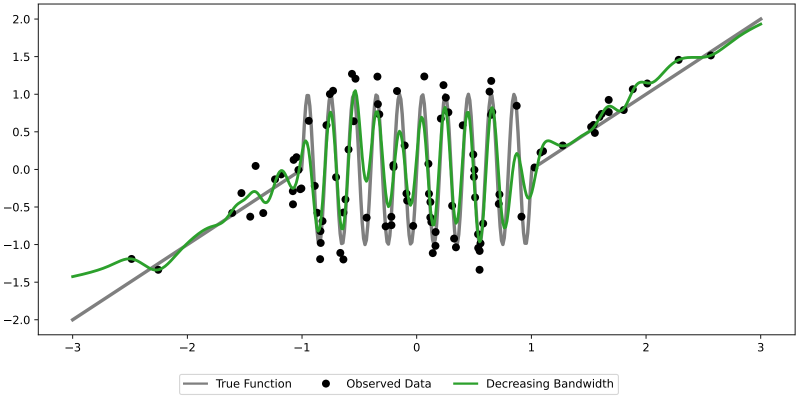
Kernel Regression with Gradient Descent and Non-Constant Kernels



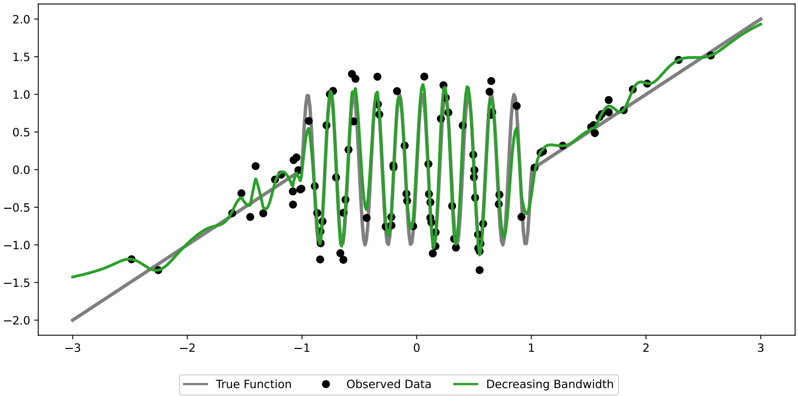
Kernel Regression with Gradient Descent and Non-Constant Kernels



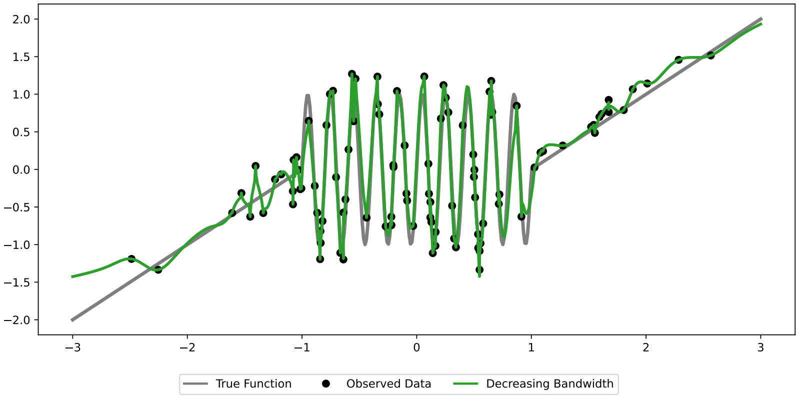
Kernel Regression with Gradient Descent and Non-Constant Kernels



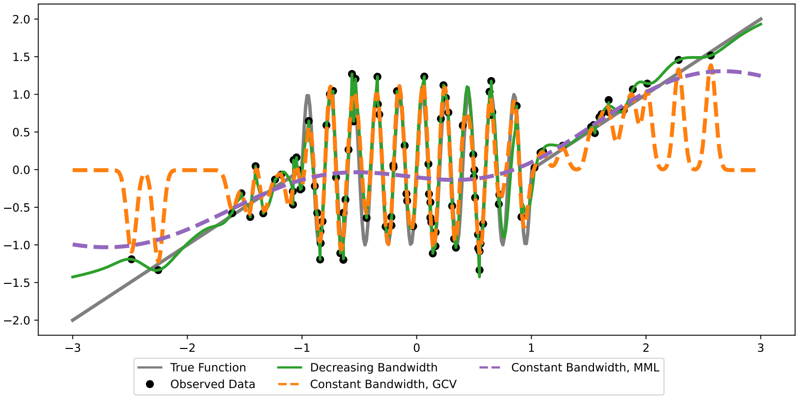
Kernel Regression with Gradient Descent and Non-Constant Kernels



Kernel Regression with Gradient Descent and Non-Constant Kernels



Kernel Regression with Gradient Descent and Non-Constant Kernels



Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Generalization as function of model complexity:

Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Generalization as function of model complexity:

A too simple model generalizes poorly.

Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Generalization as function of model complexity:

A too simple model generalizes poorly.
A model of appropriate complexity generalizes well.

Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Generalization as function of model complexity:

- A too simple model generalizes poorly.
- A model of appropriate complexity generalizes well.
- A too complex model generalizes poorly (overfitting).

Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Generalization as function of model complexity:

$$\left.\begin{array}{l}\text{A too simple model generalizes poorly.}\\ \text{A model of appropriate complexity generalizes well.}\\ \text{A too complex model generalizes poorly (overfitting).}\end{array}\right\} \begin{array}{l}\text{Classical}\\ \text{statistical}\\ \text{knowledge}\end{array}$$

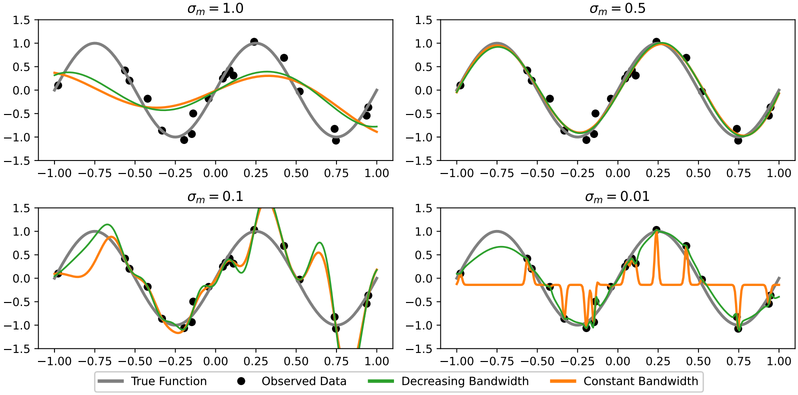
Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Generalization as function of model complexity:

$$\left.\begin{array}{l}\text{A too simple model generalizes poorly.}\\ \text{A model of appropriate complexity generalizes well.}\\ \text{A too complex model generalizes poorly (overfitting).}\end{array}\right\} \begin{array}{l}\text{Classical}\\ \text{statistical}\\ \text{knowledge}\end{array}$$

An extremely complex model generalizes well (double descent).

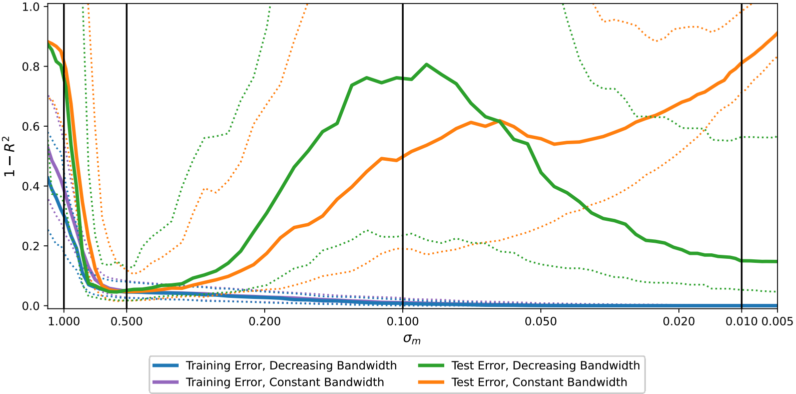
Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent



σ_m: Minimum allowed bandwidth when decreasing the bandwidth.

Employed bandwidth when using a constant bandwidth.

Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent



Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Proposition (Simplified)

$$|\hat{f}(\boldsymbol{x}^*, t, \sigma_m)| \leq \min\left(\overline{\sigma^{-1}}(\sigma_m),\ \overline{k_2^*}(\sigma_m)\right) \cdot t \cdot C(k(\cdot), \boldsymbol{X}, \boldsymbol{y})$$

where $\overline{\sigma^{-1}}(\sigma_m)$ *increases with model complexity and* $\overline{k_2^*}(\sigma_m)$ *decreases with model complexity.*

Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Proposition (Simplified)

$$|\hat{f}(\boldsymbol{x}^*, t, \sigma_m)| \leq \min\left(\overline{\sigma^{-1}}(\sigma_m),\ \overline{k_2^*}(\sigma_m)\right) \cdot t \cdot C(k(\cdot), \boldsymbol{X}, \boldsymbol{y})$$

where $\overline{\sigma^{-1}}(\sigma_m)$ *increases with model complexity and* $\overline{k_2^*}(\sigma_m)$ *decreases with model complexity.*

Low complexity: $\overline{\sigma^{-1}}(\sigma_m)$ is small $|\hat{f}|$ is too small.

Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Proposition (Simplified)

$$|\hat{f}(\boldsymbol{x}^*, t, \sigma_m)| \leq \min\left(\overline{\sigma^{-1}}(\sigma_m),\ \overline{k_2^*}(\sigma_m)\right) \cdot t \cdot C(k(\cdot), \boldsymbol{X}, \boldsymbol{y})$$

where $\overline{\sigma^{-1}}(\sigma_m)$ increases with model complexity and $\overline{k_2^*}(\sigma_m)$ decreases with model complexity.

Low complexity:	$\overline{\sigma^{-1}}(\sigma_m)$ is small	$\lvert\hat{f}\rvert$ is too small.
Moderate complexity:	$\overline{\sigma^{-1}}(\sigma_m)$ is moderate	$\lvert\hat{f}\rvert$ is appropriate.

Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Proposition (Simplified)

$$|\hat{f}(\boldsymbol{x}^*, t, \sigma_m)| \leq \min\left(\overline{\sigma^{-1}}(\sigma_m),\ \overline{k_2^*}(\sigma_m)\right) \cdot t \cdot C(k(\cdot), \boldsymbol{X}, \boldsymbol{y})$$

where $\overline{\sigma^{-1}}(\sigma_m)$ *increases with model complexity and* $\overline{k_2^*}(\sigma_m)$ *decreases with model complexity.*

Low complexity:	$\overline{\sigma^{-1}}(\sigma_m)$ is small	$\lvert\hat{f}\rvert$ is too small.
Moderate complexity:	$\overline{\sigma^{-1}}(\sigma_m)$ is moderate	$\lvert\hat{f}\rvert$ is appropriate.
High complexity:	$\overline{\sigma^{-1}}(\sigma_m)$ is large	$\lvert\hat{f}\rvert$ is too large.

Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

Proposition (Simplified)

$$|\hat{f}(\boldsymbol{x}^*, t, \sigma_m)| \leq \min\left(\overline{\sigma^{-1}}(\sigma_m),\ \overline{k_2^*}(\sigma_m)\right) \cdot t \cdot C(k(\cdot), \boldsymbol{X}, \boldsymbol{y})$$

where $\overline{\sigma^{-1}}(\sigma_m)$ increases with model complexity and $\overline{k_2^*}(\sigma_m)$ decreases with model complexity.

Low complexity:	$\overline{\sigma^{-1}}(\sigma_m)$ is small	$\lvert\hat{f}\rvert$ is too small.
Moderate complexity:	$\overline{\sigma^{-1}}(\sigma_m)$ is moderate	$\lvert\hat{f}\rvert$ is appropriate.
High complexity:	$\overline{\sigma^{-1}}(\sigma_m)$ is large	$\lvert\hat{f}\rvert$ is too large.
Very high complexity:	$\overline{k_2^*}(\sigma_m)$ is moderate	$\lvert\hat{f}\rvert$ is appropriate.

Neural Tangent Kernel Gradient Flow

Neural Tangent Kernel Gradient Flow

$$\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix} - \Delta t \cdot \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

Neural Tangent Kernel Gradient Flow

$$\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix} - \Delta t \cdot \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\iff \frac{\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix}}{\Delta t} = -\begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

Neural Tangent Kernel Gradient Flow

$$\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix} - \Delta t \cdot \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\iff \frac{\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix}}{\Delta t} = - \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial t} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial t} \end{bmatrix}$$

Neural Tangent Kernel Gradient Flow

$$\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix} - \Delta t \cdot \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\Longleftrightarrow \quad \frac{\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix}}{\Delta t} = - \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial t} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial t} \end{bmatrix} \overset{\text{C.R.}}{=} \begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \end{bmatrix} \cdot \frac{\partial \hat{\boldsymbol{\theta}}(t)}{\partial t}$$

Neural Tangent Kernel Gradient Flow

$$\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix} - \Delta t \cdot \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\iff \frac{\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix}}{\Delta t} = - \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial t} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial t} \end{bmatrix} \overset{\text{C.R.}}{=} \begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \end{bmatrix} \cdot \frac{\partial \hat{\boldsymbol{\theta}}(t)}{\partial t} \overset{\text{G.D.}}{=} - \begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \end{bmatrix} \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{\theta}}(t)}$$

Neural Tangent Kernel Gradient Flow

$$\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix} - \Delta t \cdot \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\iff \frac{\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix}}{\Delta t} = - \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial t} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial t} \end{bmatrix} \overset{\text{C.R.}}{=} \begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \end{bmatrix} \cdot \frac{\partial \hat{\boldsymbol{\theta}}(t)}{\partial t} \overset{\text{G.D.}}{=} - \begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \end{bmatrix} \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{\theta}}(t)}$$

$$\overset{\text{C.R.}}{=} - \begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \end{bmatrix} \cdot \left(\frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \right)^\top \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{f}}(t)}$$

Neural Tangent Kernel Gradient Flow

$$\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix} - \Delta t \cdot \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\iff \frac{\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}^*(t+\Delta t) \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}^*(t) \end{bmatrix}}{\Delta t} = - \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} (\hat{\boldsymbol{f}}(t) - \boldsymbol{y})$$

$$\begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial t} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial t} \end{bmatrix} \overset{\text{C.R.}}{=} \begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \end{bmatrix} \cdot \frac{\partial \hat{\boldsymbol{\theta}}(t)}{\partial t} \overset{\text{G.D.}}{=} - \begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \end{bmatrix} \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{\theta}}(t)}$$

$$\overset{\text{C.R.}}{=} - \begin{bmatrix} \frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \end{bmatrix} \cdot \left(\frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)} \right)^\top \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{f}}(t)}$$

$$\overset{\text{def}}{=} - \begin{bmatrix} \boldsymbol{\Phi}(t)\boldsymbol{\Phi}(t)^\top \\ \boldsymbol{\Phi}^*(t)\boldsymbol{\Phi}(t)^\top \end{bmatrix} \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{f}}(t)}$$

Neural Tangent Kernel Gradient Flow

$$\begin{bmatrix}\hat{\boldsymbol{f}}(t+\Delta t)\\ \hat{\boldsymbol{f}}^*(t+\Delta t)\end{bmatrix} = \begin{bmatrix}\hat{\boldsymbol{f}}(t)\\ \hat{\boldsymbol{f}}^*(t)\end{bmatrix} - \Delta t \cdot \begin{bmatrix}\boldsymbol{K}(t)\\ \boldsymbol{K}^*(t)\end{bmatrix}(\hat{\boldsymbol{f}}(t)-\boldsymbol{y})$$

$$\iff \frac{\begin{bmatrix}\hat{\boldsymbol{f}}(t+\Delta t)\\ \hat{\boldsymbol{f}}^*(t+\Delta t)\end{bmatrix} - \begin{bmatrix}\hat{\boldsymbol{f}}(t)\\ \hat{\boldsymbol{f}}^*(t)\end{bmatrix}}{\Delta t} = -\begin{bmatrix}\boldsymbol{K}(t)\\ \boldsymbol{K}^*(t)\end{bmatrix}(\hat{\boldsymbol{f}}(t)-\boldsymbol{y})$$

$$\begin{bmatrix}\frac{\partial \hat{\boldsymbol{f}}(t)}{\partial t}\\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial t}\end{bmatrix} \overset{\text{C.R.}}{=} \begin{bmatrix}\frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)}\\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)}\end{bmatrix} \cdot \frac{\partial \hat{\boldsymbol{\theta}}(t)}{\partial t} \overset{\text{G.D.}}{=} -\begin{bmatrix}\frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)}\\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)}\end{bmatrix} \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{\theta}}(t)}$$

$$\overset{\text{C.R.}}{=} -\begin{bmatrix}\frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)}\\ \frac{\partial \hat{\boldsymbol{f}}^*(t)}{\partial \hat{\boldsymbol{\theta}}(t)}\end{bmatrix} \cdot \left(\frac{\partial \hat{\boldsymbol{f}}(t)}{\partial \hat{\boldsymbol{\theta}}(t)}\right)^\top \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{f}}(t)}$$

$$\overset{\text{def}}{=} -\begin{bmatrix}\boldsymbol{\Phi}(t)\boldsymbol{\Phi}(t)^\top\\ \boldsymbol{\Phi}^*(t)\boldsymbol{\Phi}(t)^\top\end{bmatrix} \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{f}}(t)} = -\begin{bmatrix}\boldsymbol{K}(t)\\ \boldsymbol{K}^*(t)\end{bmatrix} \cdot \frac{\partial L(\hat{\boldsymbol{f}}(t))}{\partial \hat{\boldsymbol{f}}(t)}.$$

Improved Generalization by Neural Tangent Control

- Kernel gradient descent performs best when the bandwidth decreases toward zero,

Improved Generalization by Neural Tangent Control

- Kernel gradient descent performs best when the bandwidth decreases toward zero,
- that is, when $\begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{K}^* \end{bmatrix}$ goes to $\begin{bmatrix} \boldsymbol{I}_{n\times n} \\ \mathbf{0}_{n^*\times n} \end{bmatrix}$.

Improved Generalization by Neural Tangent Control

- Kernel gradient descent performs best when the bandwidth decreases toward zero,
- that is, when $\begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{K}^* \end{bmatrix}$ goes to $\begin{bmatrix} \boldsymbol{I}_{n\times n} \\ \boldsymbol{0}_{n^*\times n} \end{bmatrix}$.
- Can this be behaviour be enforced for the neural tangent kernel?

Improved Generalization by Neural Tangent Control

- Kernel gradient descent performs best when the bandwidth decreases toward zero,
- that is, when $\begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{K}^* \end{bmatrix}$ goes to $\begin{bmatrix} \boldsymbol{I}_{n \times n} \\ \boldsymbol{0}_{n^* \times n} \end{bmatrix}$.
- Can this be behaviour be enforced for the neural tangent kernel?
- Work in progress, but it seems so...

Conclusions

Kernels models

Conclusions

Kernels models

- are non-linear in data.

Conclusions

Kernels models

- are non-linear in data.
- are linear in parameters (and thus convex with closed-form solutions).

Conclusions

Kernels models

- are non-linear in data.
- are linear in parameters (and thus convex with closed-form solutions).
- correspond to a (possibly infinite dimensional) feature expansion.

Conclusions

Kernels models

- are non-linear in data.
- are linear in parameters (and thus convex with closed-form solutions).
- correspond to a (possibly infinite dimensional) feature expansion.

Examples:

Conclusions

Kernels models

- are non-linear in data.
- are linear in parameters (and thus convex with closed-form solutions).
- correspond to a (possibly infinite dimensional) feature expansion.

Examples:

- Kernel Principal Component Analysis.

Conclusions

Kernels models

- are non-linear in data.
- are linear in parameters (and thus convex with closed-form solutions).
- correspond to a (possibly infinite dimensional) feature expansion.

Examples:

- Kernel Principal Component Analysis.
- Kernel Ridge Regression.

Conclusions

Kernels models

- are non-linear in data.
- are linear in parameters (and thus convex with closed-form solutions).
- correspond to a (possibly infinite dimensional) feature expansion.

Examples:

- Kernel Principal Component Analysis.
- Kernel Ridge Regression.

Kernel regression with gradually increased model complexity

Conclusions

Kernels models

- are non-linear in data.
- are linear in parameters (and thus convex with closed-form solutions).
- correspond to a (possibly infinite dimensional) feature expansion.

Examples:

- Kernel Principal Component Analysis.
- Kernel Ridge Regression.

Kernel regression with gradually increased model complexity

- reduces the need for hyper parameter selection.

Conclusions

Kernels models

- are non-linear in data.
- are linear in parameters (and thus convex with closed-form solutions).
- correspond to a (possibly infinite dimensional) feature expansion.

Examples:

- Kernel Principal Component Analysis.
- Kernel Ridge Regression.

Kernel regression with gradually increased model complexity

- reduces the need for hyper parameter selection.
- exhibits a double descent behavior.

Conclusions

Kernels models

- are non-linear in data.
- are linear in parameters (and thus convex with closed-form solutions).
- correspond to a (possibly infinite dimensional) feature expansion.

Examples:

- Kernel Principal Component Analysis.
- Kernel Ridge Regression.

Kernel regression with gradually increased model complexity

- reduces the need for hyper parameter selection.
- exhibits a double descent behavior.
- is generalizable to any parametric model trained with gradient descent.

The End

Thank you!

Alternative Interpretation

- In forward stagewise additive modelling, model m is used to fit the residuals of model $m-1$.

$$\hat{\boldsymbol{f}}'_m(\boldsymbol{X}) = \underset{f'_m \in \mathcal{F}_m}{\operatorname{argmin}} L\left(\boldsymbol{f}'_m(\boldsymbol{X}), \boldsymbol{y} - \hat{\boldsymbol{f}}_{m-1}(\boldsymbol{X})\right)$$

$$\hat{\boldsymbol{f}}_m(\boldsymbol{X}) = \hat{\boldsymbol{f}}_{m-1}(\boldsymbol{X}) + \hat{\boldsymbol{f}}'_m(\boldsymbol{X}),$$

- If we interpret every bandwidth update during training as a model change, then kernel gradient descent with decreasing bandwidth is forward stagewise additive modelling with increasing model complexity.

Algorithm

Algorithm 1 Kernel Gradient Descent with Decreasing Bandwidth

Input: Training data, $(\boldsymbol{X}, \boldsymbol{y})$. Prediction covariates, $\boldsymbol{X}^*$. Initial bandwidth, σ_0. Prior $\mu(\boldsymbol{x})$.
Minimum allowed bandwidth, σ_m. Step-size, Δt. Minimum R^2 speed, v_{R^2}. Maximum R^2, $R^2_{\max}$.

Output: Vector of predictions $[\hat{\boldsymbol{f}}(t)^\top, \hat{\boldsymbol{f}}(t)^\top]^\top$.

Initialize $[\hat{\boldsymbol{f}}(0)^\top, \hat{\boldsymbol{f}}^*(0)^\top]^\top = [\boldsymbol{\mu}(\boldsymbol{X})^\top, \boldsymbol{\mu}(\boldsymbol{X}^*)^\top]^\top$.
Initialize $\sigma(0) = \sigma_0$, $\boldsymbol{K} = \boldsymbol{K}(\sigma_0)$ and $\boldsymbol{K}^* = \boldsymbol{K}^*(\sigma_0)$.
repeat
    Calculate $R^2(t) = 1 - \frac{\|\boldsymbol{y}-\hat{\boldsymbol{f}}(t)\|_2^2}{\|\boldsymbol{y}-\bar{y}\|_2^2}$ and $\frac{\partial R^2(t)}{\partial t} = 1 - \frac{\|\boldsymbol{y}-\hat{\boldsymbol{f}}(t)\|_{\boldsymbol{K}(t)}^2}{\|\boldsymbol{y}-\bar{y}\|_2^2}$.
    if $\frac{\partial R^2(t)}{\partial t} < v_{R^2}$ **then**
        repeat
            Decrease $\sigma(t)$ and recalculate $\boldsymbol{K}(\sigma(t))$ and $\frac{\partial R^2(t)}{\partial t}$
        until $\frac{\partial R(t)^2}{\partial t} \geq v_{R^2}$ **or** $\sigma(t) \leq \sigma_m$.
        Recalculate $\boldsymbol{K}^*(\sigma(t))$.
    end if
    Update $[\hat{\boldsymbol{f}}(t)^\top, \hat{\boldsymbol{f}}(t)^\top]^\top$ according to

$$\begin{bmatrix} \hat{\boldsymbol{f}}(t+\Delta t) \\ \hat{\boldsymbol{f}}(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}(t) \\ \hat{\boldsymbol{f}}(t) \end{bmatrix} + \Delta t \begin{bmatrix} \boldsymbol{K}(t) \\ \boldsymbol{K}^*(t) \end{bmatrix} \left(\boldsymbol{y} - \hat{\boldsymbol{f}}(t)\right)$$

until $R^2(t) \geq R^2_{\max}$.

Two Propositions

Proposition 1.

Let $\overline{k_2^*}$ *denote the weighted average of* $\|\boldsymbol{k}^*(\tau)\|_2$ *during training, with weight function* $\left\|\boldsymbol{y} - \hat{\boldsymbol{f}}(\tau)\right\|_2$*, and let* $\overline{s_{\min}}$ *denote the time average of the smallest eigenvalue of* $\boldsymbol{K}(\tau)$ *during training, i.e.*

$$\overline{k_2^*} := \frac{\int_0^t \|\boldsymbol{k}^*(\tau)\|_2 \cdot \left\|\boldsymbol{y} - \hat{\boldsymbol{f}}\,(\tau)\right\|_2 d\tau}{\int_0^t \left\|\boldsymbol{y} - \hat{\boldsymbol{f}}\,(\tau)\right\|_2 d\tau} \quad \textit{and} \quad \overline{s_{\min}} := \frac{1}{t}\int_0^t s_{\min}(\boldsymbol{K}(\tau))d\tau.$$

Then, for the KGF estimate, $\hat{f}_\mu(\boldsymbol{x}^*, t)$*,*

$$|\hat{f}_\mu(\boldsymbol{x}^*, t)| \leq \overline{k_2^*} \cdot \min\left(t, \frac{1}{\overline{s_{\min}}}\right) \cdot \|\boldsymbol{y_\mu}\|_2. \tag{7}$$

Proposition 3.

Let $\overline{\sigma^{-1}}$ *denote the weighted average of the inverse bandwidth during training, with weight function* $\left\|\boldsymbol{y} - \hat{\boldsymbol{f}}(\tau)\right\|_2$*, i.e.,*

$$\overline{\sigma^{-1}} := \frac{\int_0^t \frac{1}{\sigma(\tau)} \cdot \left\|\boldsymbol{y} - \hat{\boldsymbol{f}}\,(\tau)\right\|_2 d\tau}{\int_0^t \left\|\boldsymbol{y} - \hat{\boldsymbol{f}}\,(\tau)\right\|_2 d\tau}.$$

Then, for a kernel $k(\boldsymbol{x}, \boldsymbol{x'}, \sigma(\tau)) = k\left(\frac{1}{\sigma(\tau)} \cdot \|\boldsymbol{x} - \boldsymbol{x'}\|_{\boldsymbol{\Theta}}\right)$*, with bounded derivative,* $|k'(u)| \leq k'_{\max}\ \forall u$*, for the KGF estimate,* $\hat{f}_\mu(\boldsymbol{x}^*, t)$*,*

$$\left\|\frac{\partial \hat{f}_\mu(\boldsymbol{x}^*, t)}{\partial \boldsymbol{x}^*}\right\|_2 \leq \overline{\sigma^{-1}} \cdot \min\left(t, \frac{1}{\overline{s_{\min}}}\right) \cdot \|\boldsymbol{y_\mu}\|_2 \cdot k'_{\max} \cdot \sqrt{n \cdot \|\boldsymbol{\Theta}\|_2}.$$

Assuming the data is centered, so that $\overline{y_\mu} = 0$*, we further obtain*

$$\left|\hat{f}_\mu(\boldsymbol{x}^*, t)\right| \leq \overline{\sigma^{-1}} \cdot \min\left(t, \frac{1}{\overline{s_{\min}}}\right) \cdot \|\boldsymbol{y_\mu}\|_2 \cdot k'_{\max} \cdot \sqrt{n \cdot \|\boldsymbol{\Theta}\|_2} \cdot \|\boldsymbol{x}^* - \boldsymbol{x_m}\|_2 \tag{10}$$

where $\boldsymbol{x_m} \in \boldsymbol{X}$ *is the observation furthest away from* $\boldsymbol{x}^*$*.*

Double Descent

Table 1: Conceptual sketch of how the bound on $|f_\mu|$ from Equation 15, and thus the generalization properties, changes with model complexity, σ_m. The active elements in the minimum functions are marked in bold. In the third and fourth rows, it is not obvious which element is smaller, but this uncertainty does not affect the bound. The constants C_1 and C_2 are omitted to improve readability. For lower model complexities, the bound on the deviation from the prior grows with model complexity, but for very complex models it starts to shrink again.

σ_m	Model Complexity	$\overline{\sigma^{-1}}$	$\overline{k_2^*}$	t	$1/\overline{s_{\min}}$	Bound on $\lvert\hat{f}_\mu\rvert$	Generalization
Large	Low	**Small**	Large	**Moderate**	Large	$\overline{\sigma^{-1}} \cdot t$ Small	Poor, due to basically predicting the prior
Moderate	Moderate	**Moderate**	Large	**Moderate**	Large	$\overline{\sigma^{-1}} \cdot t$ Moderate	Good, due to moderate deviations from the prior
Small	High	**Large**	Large	**Moderate**	Large	$\overline{\sigma^{-1}} \cdot t$ Large	Poor, due to extreme predictions
Very small	Very high	Very large	**Moderate**	Moderate	**Moderate**	$\overline{k_2^*}/\overline{s_{\min}}$ Moderate	Good, due to moderate deviations from the prior

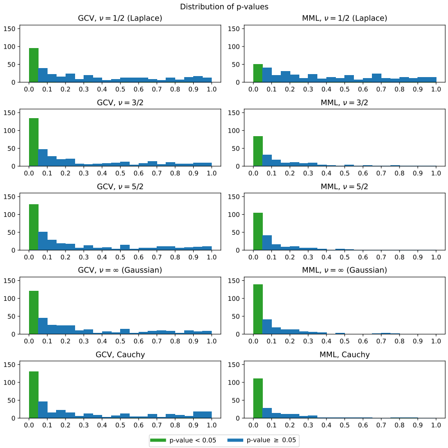
performs excellently on training data but tends to generalize poorly, something that is often referred to as overfitting. However, the wisdom from double descent is that if the model is made even more complex, it may generalize well despite excellent performance on training data.

If we constrain the complexity of the final model by introducing a minimum allowed bandwidth, σ_m, we may, for a constant, fairly long, training time, obtain a double descent behavior in the complexity (i.e. in σ_m). This can qualitatively be seen from the bound on $|\hat{f}_\mu(\boldsymbol{x}^*)|$ obtained by combining Equations 7 and 10:

$$|\hat{f}_\mu(\boldsymbol{x}^*, t, \sigma_m)| \leq \min\left(\overline{\sigma^{-1}}(\sigma_m) \cdot C_1,\ \overline{k_2^*}(\sigma_m)\right) \cdot \min\left(t,\ \frac{1}{\overline{s_{\min}}(\sigma_m)}\right) \cdot C_2, \tag{15}$$

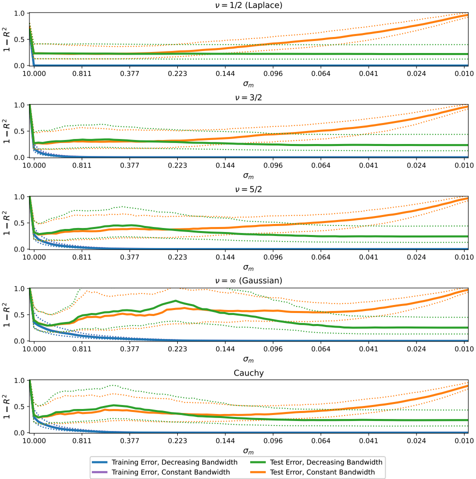
Additional Results

Daily mean temperature in the U.K. during one year, as function of location. 45000 observations, 366 days. Wilcoxon Signed-Rank Test, testing whether $R^2_{\text{test}}(\text{decr}) > R^2_{\text{test}}(\text{const})$.



Additional Results

U.K. temperature data



Additional Results

Synthetic data