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

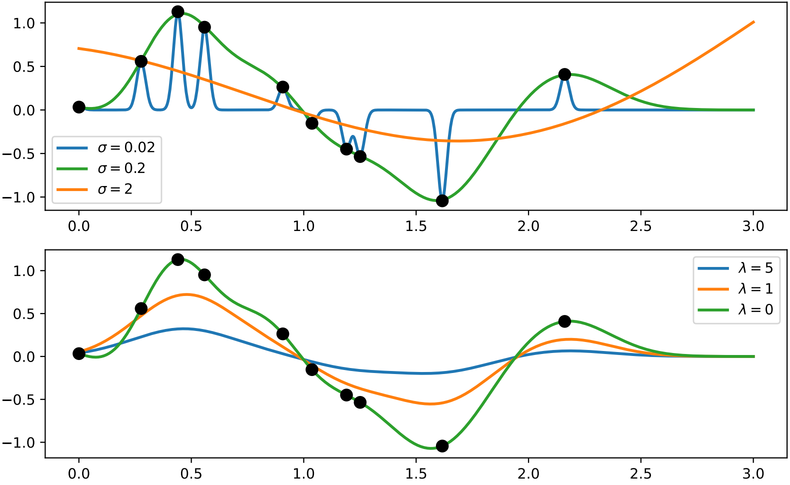
Oskar Allerbo

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 (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

What is a Kernel?

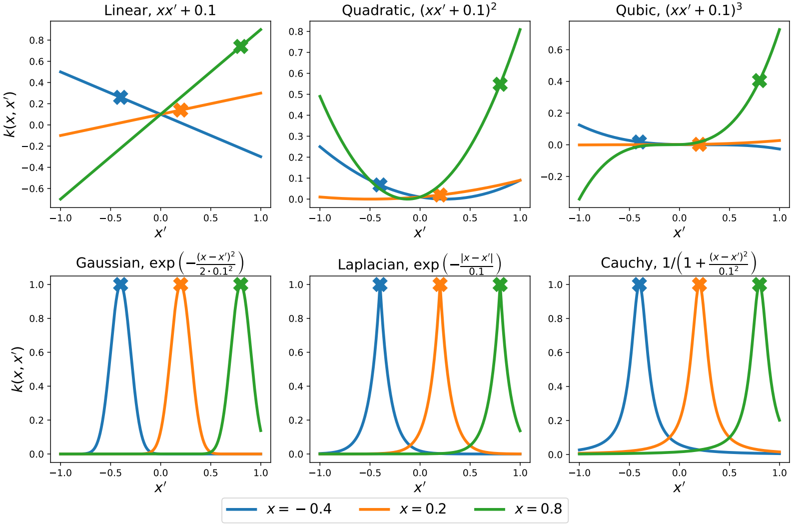
A kernel function:

- 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.)

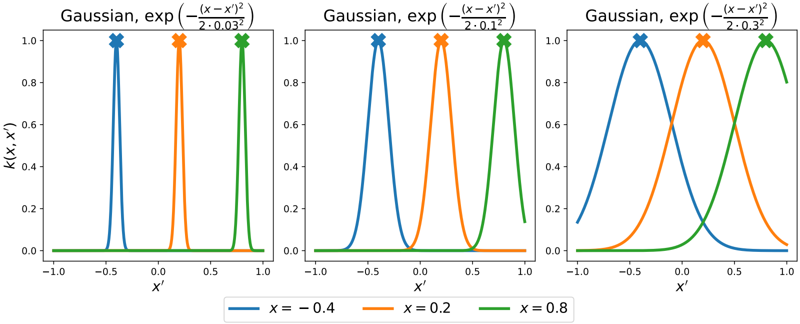
Examples of Kernels

- 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: $k(\boldsymbol{x}, \boldsymbol{x'}) = \frac{2^{1-\nu}}{\Gamma(\nu)} \cdot \left(\sqrt{2\nu} \cdot \frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}\right)^{\nu} \cdot K_\nu \left(\sqrt{2\nu} \cdot \frac{\|\boldsymbol{x}-\boldsymbol{x'}\|_2}{\sigma}\right)$

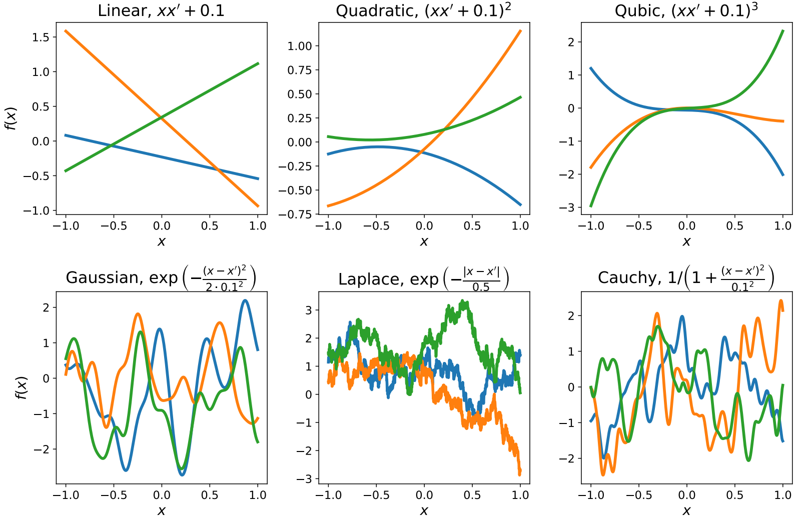
Examples of Kernels (1D Case)



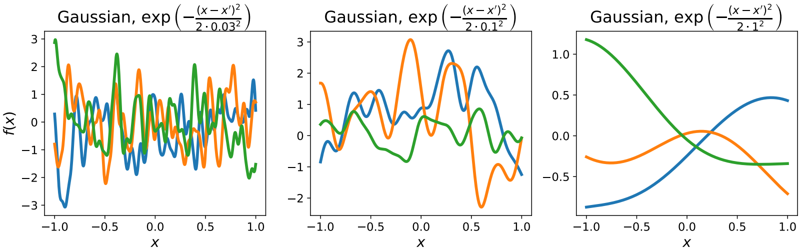
Examples of Kernels (1D Case)



Examples of Functions (1D Case)



Examples of Functions (1D Case)



Examples of Feature Expansions (1D Case)

- 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

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

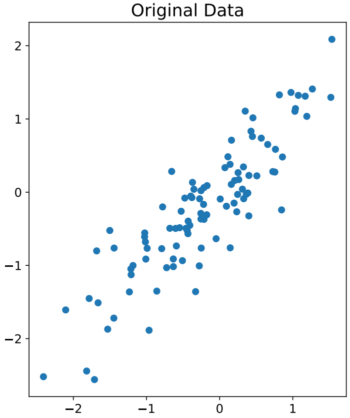
$$
\boldsymbol{X} = \begin{bmatrix} — & \boldsymbol{x}_1^\top & — \\ — & \boldsymbol{x}_2^\top & — \\ \vdots & \vdots & \vdots \\ — & \boldsymbol{x}_n^\top & — \end{bmatrix} \quad \boldsymbol{X}^* = \begin{bmatrix} — & \boldsymbol{x}_1^{*\top} & — \\ — & \boldsymbol{x}_2^{*\top} & — \\ \vdots & \vdots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

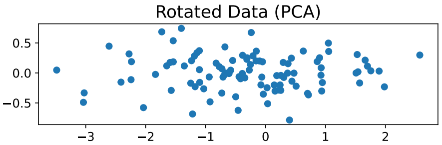
$$
\boldsymbol{\Phi} = \begin{bmatrix} — & \varphi(\boldsymbol{x}_1)^\top & — \\ — & \varphi(\boldsymbol{x}_2)^\top & — \\ \vdots & \vdots & \vdots \\ — & \varphi(\boldsymbol{x}_n)^\top & — \end{bmatrix} \quad \boldsymbol{\Phi}^* = \begin{bmatrix} — & \varphi(\boldsymbol{x}_1^*)^\top & — \\ — & \varphi(\boldsymbol{x}_2^*)^\top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

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

Kernel Principal Component Analysis

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





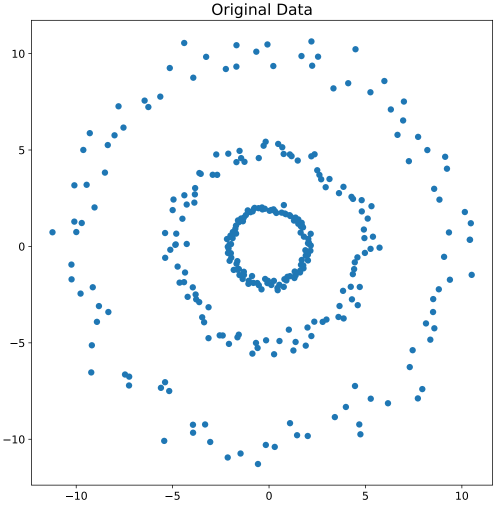
Kernel Principal Component Analysis

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

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

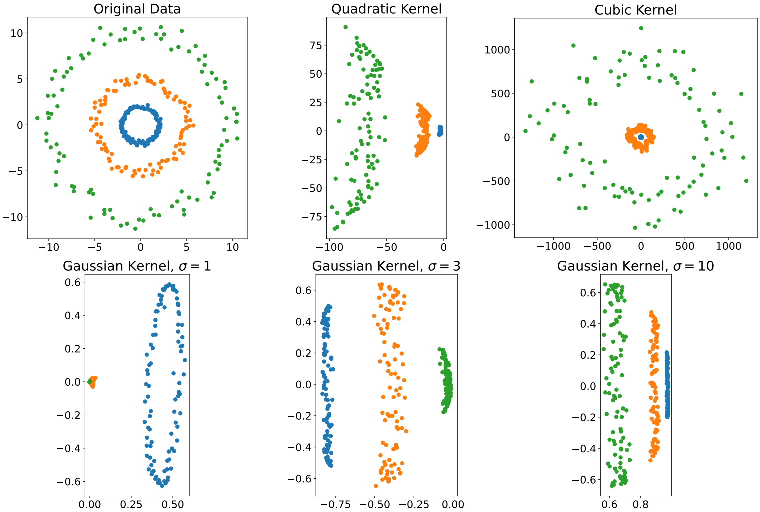
Kernel PCA:
$\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \boldsymbol{K} = \boldsymbol{U_K}\boldsymbol{D_K}\boldsymbol{U_K}^\top$, $\boldsymbol{Z_K} = \boldsymbol{U_K}\sqrt{\boldsymbol{D_K}}$.

Kernel Principal Component Analysis

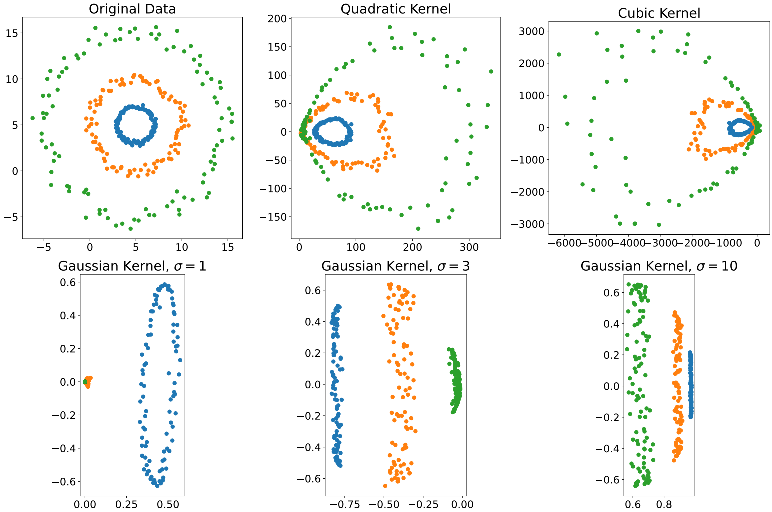


What about this?

Kernel Principal Component Analysis



Kernel Principal Component Analysis



Kernel Ridge Regression

- 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

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

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

Predictions are given by

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

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

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

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

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

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

Kernel Ridge Regression

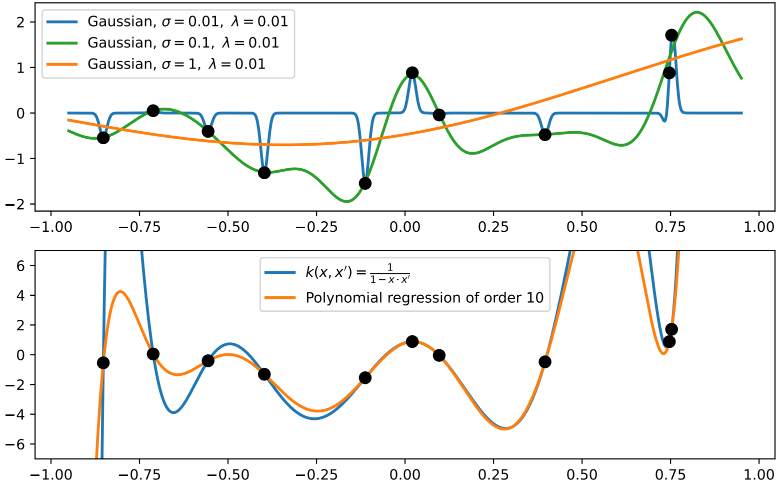
For $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \in \mathbb{R}^{n\times n}$, $\boldsymbol{K}^* = \boldsymbol{\Phi}^*\boldsymbol{\Phi}^\top \in \mathbb{R}^{n^*\times n}$,

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

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

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

Kernel Ridge Regression



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

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,

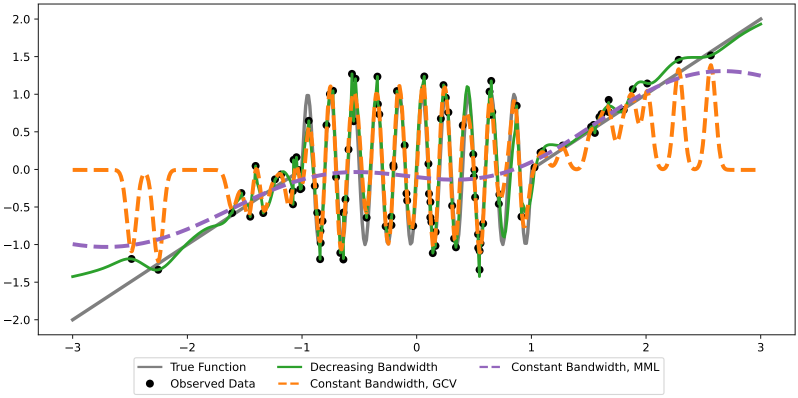
$$\left\|\nabla_{\boldsymbol{x}^*}\hat{f}(\boldsymbol{x}^*,t)\right\|_2\leq\frac{1}{\sigma}\cdot t\cdot C(k(\cdot),\boldsymbol{X},\boldsymbol{y})$$

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



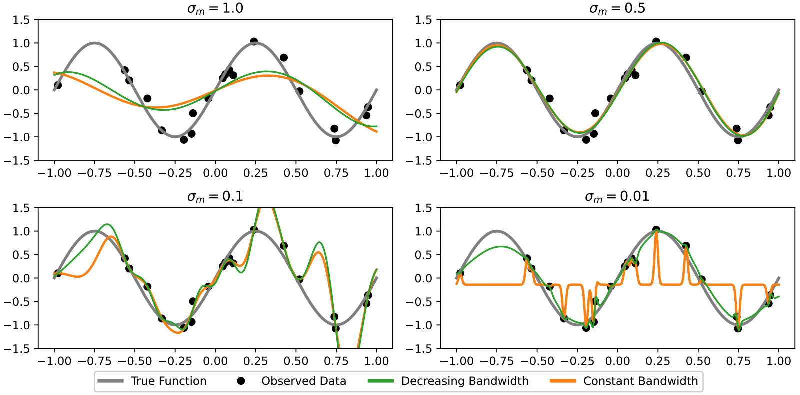
Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent

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).

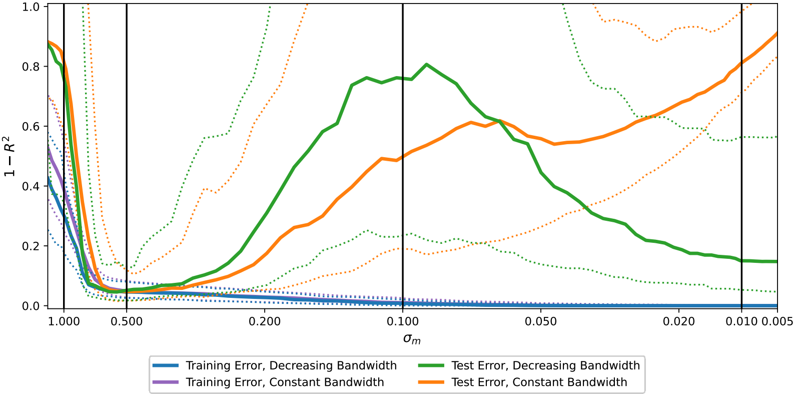
Kernel Regression with Gradient Descent and Non-Constant Kernels, Double Descent



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.

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

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

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

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

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

- 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!