Chapter 2: Kernel Methods

Kernel Regression and Support Vector Machines

Machine Learning – Foundations and Algorithms

WS 2021/22

Prof. Gerhard Neumann

KIT, Institut für Anthrophomatik und Robotik

Change in plans - Lecture Content

Chapter 1: Classical Supervised Learning

- Lecture 1: Linear Regression, Ridge Regression
- Lecture 2: Linear Classification
- Lecture 3: Model Selection
- Lecture 4: k-Nearest Neighbors, Trees and Forests

Chapter 2: Classical Unsupervised Learning

- Lecture 5: Dimensionality Reduction and Clustering
- Lecture 6: Density Estimation and Mixture Models

Chapter 3: Kernel Methods

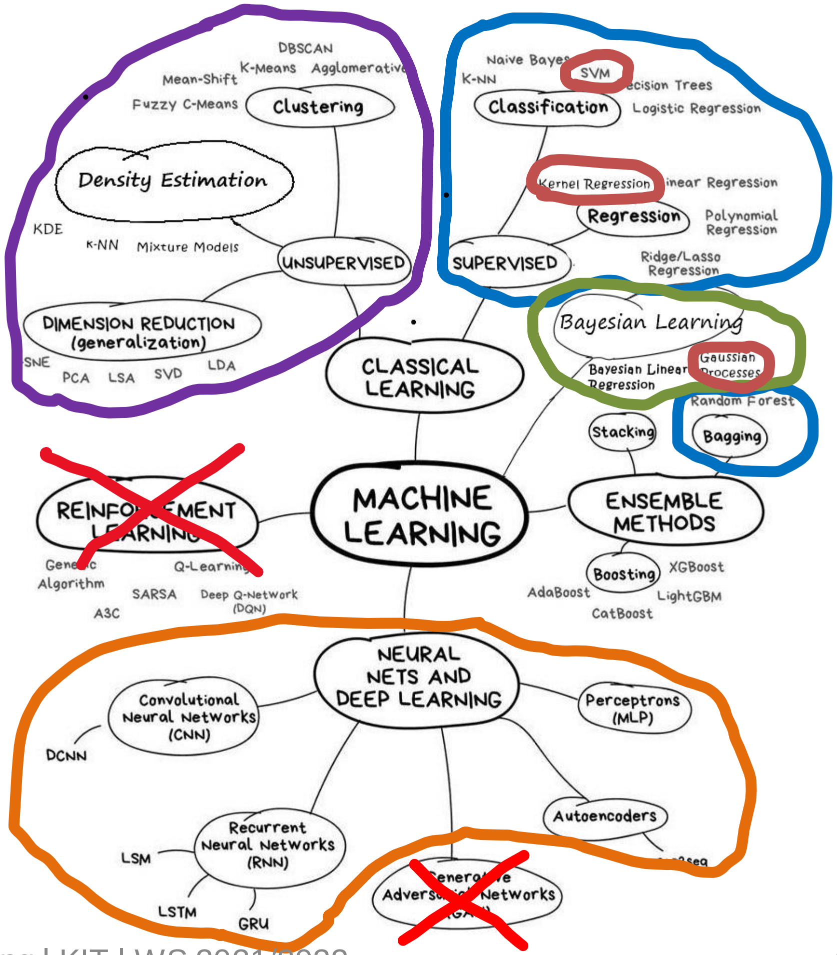
- Lecture 7: Kernel-Regression
- Lecture 8: Support Vector Machines

Chapter 4: Bayesian Learning

- Lecture 9: Bayesian Linear Regression and Gaussian Processes

Chapter 5: Neural Networks

- Lecture 10: Neural Networks and Backpropagation
- Lecture 11: CNNs and LSTMs
- Lecture 12: Variational Auto-Encoders (?)



Change in plans - Lecture Content

Chapter 1: Classical Supervised Learning

- Lecture 1: Linear Regression, Ridge Regression
- Lecture 2: Linear Classification
- Lecture 3: Model Selection
- Lecture 4: k-Nearest Neighbors, Trees and Forests

Chapter 2: Kernel Methods

- Lecture 5: Kernel-Regression
- Lecture 6: Support Vector Machines

Chapter 3: Bayesian Learning

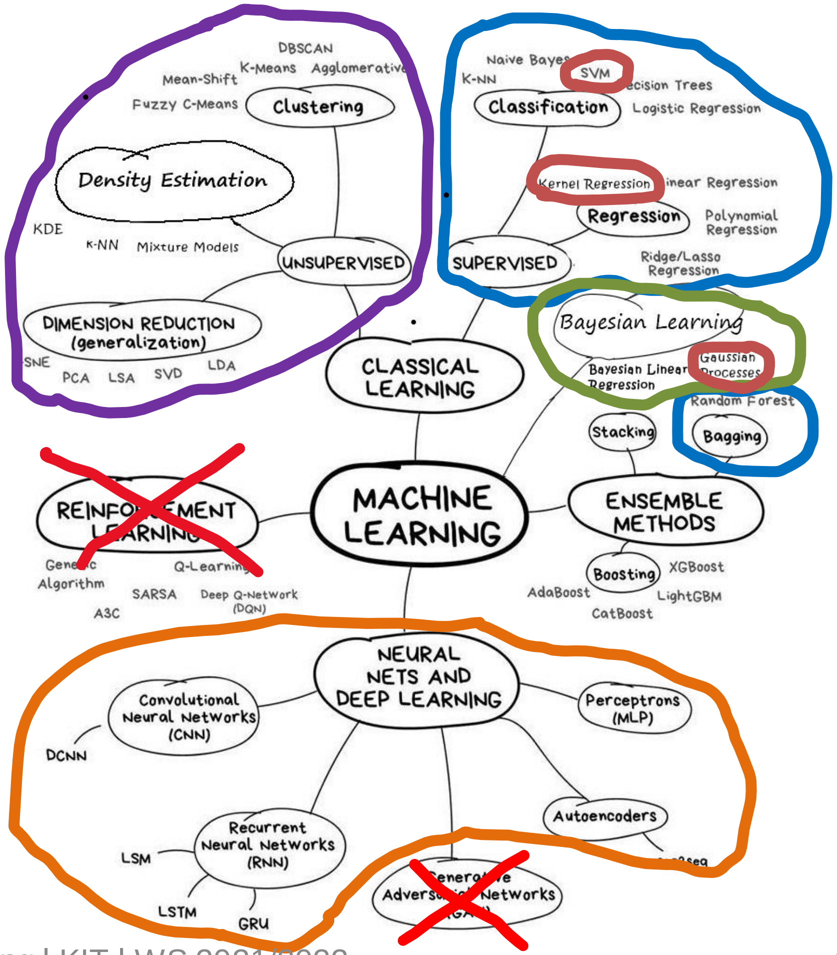
- Lecture 7: Bayesian Linear Regression and Gaussian Processes

Chapter 4: Neural Networks

- Lecture 8: Neural Networks and Backpropagation
- Lecture 9: CNNs and LSTMs

Chapter 5: Unsupervised Learning

- Lecture 10: Dimensionality Reduction and Clustering
- Lecture 11: Density Estimation and Mixture Models
- Lecture 12: Variational Auto-Encoders (?)



Learning Outcomes

- What are kernels and how are they useful?
- What do we mean by the “Kernel trick”?
- How to use kernels in regression (using Kernel Regression)?
- How to use kernels in classification (using SVMs)?
- Understand how to obtain dual optimization problems from the primal
- … and its relation to kernel methods

Today‘s Agenda!

Kernels:

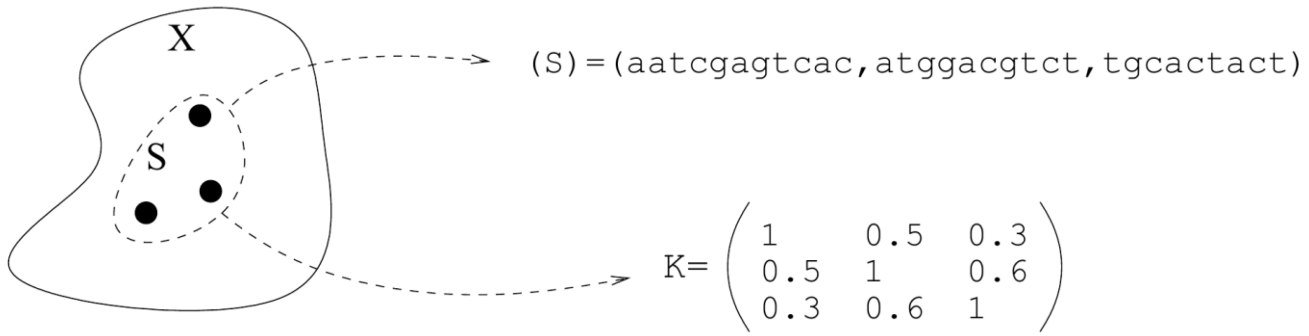
- Definition and properties
- Kernel trick

Kernel Regression:

- Kernel trick for Ridge Regression
- Analytical Solution

What is a kernel?

Representation by point-wise comparisons



$$K = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.6 \\ 0.3 & 0.6 & 1 \end{pmatrix}$$

- Define a "comparison function" $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
- Represent a set of points $\mathcal{S} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ by **the n x n matrix** $[\boldsymbol{K}]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Kernel Matrix

Properties:

- ***K*** is always an $n \times n$ matrix, whatever the nature of data: the same algorithm will work for any type of data (vectors, strings, ...).
- Total modularity between the choice of function k and the choice of the algorithm.
- Poor scalability with respect to the dataset size (n^2 to compute and store ***K***)...
- We will restrict ourselves to a particular class of pairwise comparison functions.

Positive definite kernels

A **positive definite kernel function** *k* is a function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ that is:

(i) Symmetric**:** $\forall \boldsymbol{x}, \boldsymbol{x}' : k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$

(ii) Similarity matrix is always positive definite

$$\boldsymbol{a}^T \boldsymbol{K} \boldsymbol{a} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \quad \forall \boldsymbol{a}, \forall S = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$$

Kernel methods are algorithms that take such matrices as input.

Example: Linear kernel

The linear kernel is the simplest kernel for vectors

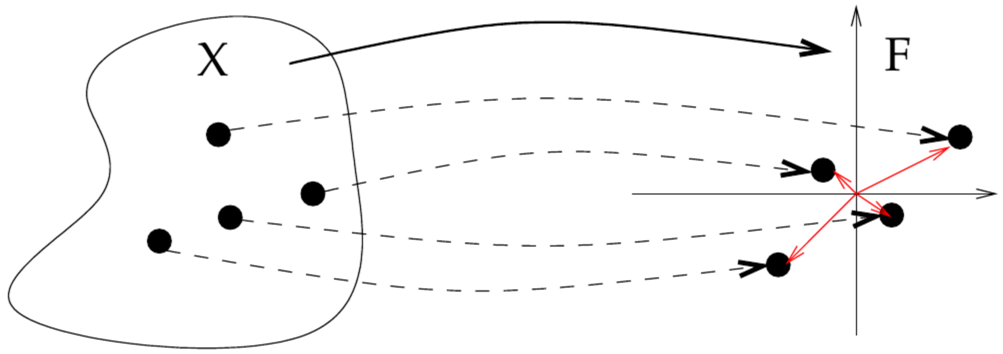
- Its defined by the scalar product:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle, \qquad \text{where } \langle \cdot, \cdot \rangle \text{ denotes the inner product}$$

- It is always positive definite:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle = \Big\| \sum_i a_i \boldsymbol{x}_i \Big\|^2 \geq 0$$

Kernels in Feature Spaces



Let $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathbb{R}^d$ be an arbitrary feature function, then $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle$ defines a positive definite kernel.

Proof: $$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle = \Big\| \sum_i a_i \boldsymbol{\phi}(\boldsymbol{x}_i) \Big\|^2 \geq 0$$

Kernels as inner products

Theorem (Aransjan 1950):

k is a positive definite kernel on the set $\mathcal{X}$ if and only if there exists a feature space $\mathcal{H}$ and a feature mapping

$$\phi : \mathcal{X} \rightarrow \mathcal{H}$$

such that for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$$

- ➢ **Every p.d. kernel comes with an associated feature space!**

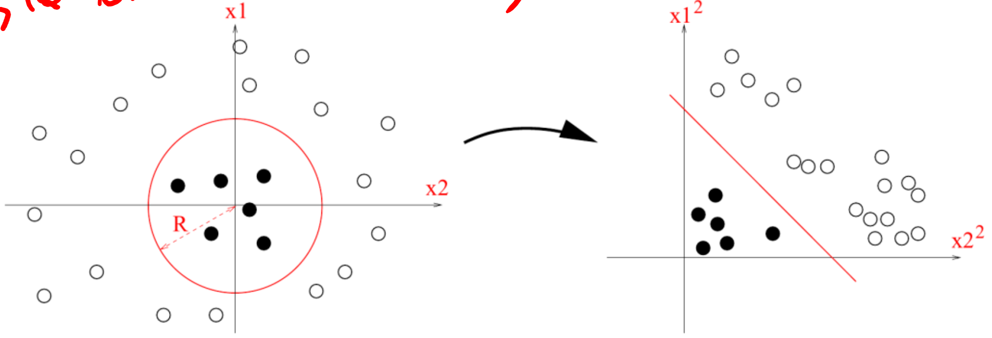
Example: polynomial kernel

For $\boldsymbol{x} = [x_1, x_2]^T$, let $\underline{\boldsymbol{\phi}(\boldsymbol{x}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]}$

$(a+b)^2 = (a^2 + 2ab + b^2)$

The kernel is defined by:

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{x}') &= x_1^2 x_1'^2 + \underline{2x_1x_2x_1'x_2' + x_2^2 x_2'^2} \\ &= \underline{(x_1x_1' + x_2x_2')^2} \\ &= \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2 \end{aligned}$$



Kernel for polynomials of degree d:

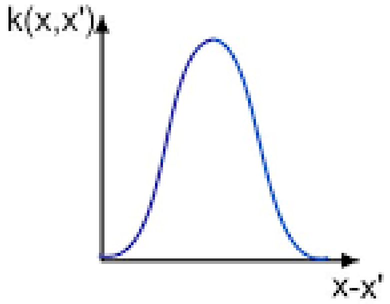
$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^d$$

Example: Gaussian Kernel

The Gaussian kernel is defined by:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$$

- where σ is the bandwidth parameter



Often also called:

- Radial basis function kernel (RBF)
- Squared exponential kernel

It is the most used kernel for kernel methods

Is the Gaussian kernel a valid p.d. kernel?

Remember: If we can show that the kernel is a valid product of feature vectors, then it is p.d.

- Consider the following feature function:

$$\phi_{\boldsymbol{\mu}}(\boldsymbol{x}) = 1/Z \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{\mu}\|^2}{\sigma^2}\right), \quad \forall \boldsymbol{\mu} \in \mathbb{R}^d$$

- I.e. we have an infinite amount of features (for every possible center $\boldsymbol{\mu}$)
- Z is a normalization constant (which we will ignore)

Inner product:

- Inner product becomes an integral due to infinite amount of dimensions

$$\langle \phi_{\boldsymbol{\mu}}(\boldsymbol{x}), \phi_{\boldsymbol{\mu}}(\boldsymbol{y}) \rangle = \int \phi_{\boldsymbol{\mu}}(\boldsymbol{x}) \phi_{\boldsymbol{\mu}}(\boldsymbol{y}) d\boldsymbol{\mu}$$

Is the Gaussian kernel a valid p.d. kernel?

Inner product:

$$
\begin{aligned}
\langle \phi_{\boldsymbol{\mu}}(\boldsymbol{x}), \phi_{\boldsymbol{\mu}}(\boldsymbol{y}) \rangle &= \int \phi_{\boldsymbol{\mu}}(\boldsymbol{x}) \phi_{\boldsymbol{\mu}}(\boldsymbol{y}) d\boldsymbol{\mu} \\
&\propto \int \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{\mu}\|^2}{\sigma^2}\right) \exp\left(-\frac{\|\boldsymbol{y}-\boldsymbol{\mu}\|^2}{\sigma^2}\right) d\boldsymbol{\mu} \text{ ... ignore normalization constants} \\
&\propto \int \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{x}, \sigma^2/2\boldsymbol{I}) \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{y}, \sigma^2/2\boldsymbol{I}) d\boldsymbol{\mu} \text{ ... product of 2 Gaussians (see matrix cookbook)} \\
&= \mathcal{N}(\boldsymbol{x}|\boldsymbol{y}, \sigma^2\boldsymbol{I}) \underbrace{\int \mathcal{N}(\boldsymbol{\mu}|..., ...) d\boldsymbol{\mu}}_{=1} \\
&\propto \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right) = k(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

Product of 2 Gaussians stays a Gaussian

$$
\begin{aligned}
&\mathcal{N}(\boldsymbol{x}|\boldsymbol{a}, \boldsymbol{A}) \mathcal{N}(\boldsymbol{x}|\boldsymbol{b}, \boldsymbol{B}) \\
&= \mathcal{N}(\boldsymbol{a}|\boldsymbol{b}, \boldsymbol{A}+\boldsymbol{B}) \mathcal{N}(\boldsymbol{x}|\boldsymbol{BFa}+\boldsymbol{AFb}, \boldsymbol{AFB})
\end{aligned}
$$

with $\boldsymbol{F} = (\boldsymbol{A}+\boldsymbol{B})^{-1}$

I.e. the Gaussian kernel is the inner product of 2 infinite dimensional feature vectors!

Kernel Trick

So why do we do this?

- Kernels can be used for all feature based algorithms that can be rewritten such that they contain inner products of feature vectors
  - This is true for almost all feature based algorithms (Linear regression, Support Vector Machines, …)
  - This is called the Kernel Trick
- Kernels can be used to map the data x in an infinite dimensional feature space (i.e., a function space)
  - The feature vector never has to be represented explicitly
  - As long as we can evaluate the inner product of two feature vectors
- Hence, we obtain a more powerful representation than standard linear feature models

A few kernel identities

Let $\boldsymbol{\Phi}_X = \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^T \end{bmatrix} \in \mathbb{R}^{N \times d}$ then the following identities hold:

- **Kernel matrix:** $\boldsymbol{K} = \boldsymbol{\Phi}_X \boldsymbol{\Phi}_X^T$
  - Check: $[\boldsymbol{K}]_{ij} = \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- **Kernel vector:** $\boldsymbol{k}(\boldsymbol{x}^*) = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}^*) \\ \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}^*) \end{bmatrix} = \begin{bmatrix} \boxed{\boldsymbol{\phi}(\boldsymbol{x}_1)^T \boldsymbol{\phi}(x^*)} \\ \vdots \\ \boxed{\boldsymbol{\phi}(\boldsymbol{x}_N)^T \boldsymbol{\phi}(x^*)} \end{bmatrix} = \boldsymbol{\Phi}_X \boldsymbol{\phi}(\boldsymbol{x}^*)$

Today‘s Agenda!

ML Algorithms

Kernels:

- Definition and properties
- Kernel trick

Kernel Regression:

- Kernel trick for Ridge Regression
- Analytical Solution

Kernel ridge Regression

Recap: Ridge Regression

- Squared error function + L2 regularization
- Linear feature space
- Not directly applicable in infinite dimensional feature spaces

Objective:

$$L_{\text{ridge}} = \underbrace{(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w})}_{\text{sum of squared errors}} + \lambda \underbrace{\boldsymbol{w}^T\boldsymbol{w}}_{L_2 \text{ regularization}}$$

Solution:

$$\boldsymbol{w}^*_{\text{ridge}} = \underbrace{(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}}_{d\times d \text{ matrix inversion}} \boldsymbol{\Phi}^T\boldsymbol{y}$$

Matrix inversion infeasible in infinite dimensions

Kernel Ridge regression

We can apply the "kernel trick":

- Rewrite solution as inner products of the feature space!
- We can do this by using the following matrix identity

$$(\boldsymbol{I} + \underset{\Phi^T}{\underline{\boldsymbol{A}\boldsymbol{B}}})^{-1}\overset{\Phi}{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{I} + \underline{\boldsymbol{B}\boldsymbol{A}})^{-1}$$

  – "Searle set of identities", The Matrix Cookbook

$$\boldsymbol{w}^* = \underbrace{(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}}_{d\times d \text{ matrix inversion}} \boldsymbol{\Phi}^T\boldsymbol{y} = \boldsymbol{\Phi}^T \underbrace{(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\boldsymbol{I})^{-1}}_{N\times N \text{ matrix inversion}} \boldsymbol{y}$$

  – With $\boldsymbol{A} = \boldsymbol{\Phi}^T$ and $\boldsymbol{B} = \boldsymbol{\Phi}$

Kernel ridge regression

The “kernelized” solution is given by:

$$\boldsymbol{w}^* = \boldsymbol{\Phi}^T \underbrace{(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda \boldsymbol{I})^{-1}}_{N \times N \text{ matrix inversion}} \boldsymbol{y} = \boldsymbol{\Phi}^T \boxed{\underbrace{(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}}_{\boldsymbol{\alpha}}} = \boldsymbol{\Phi}^T \boldsymbol{\alpha} \quad \rightarrow N \times 1$$

- Instead of inverting a d x d matrix, we can now invert an N x N matrix
- Is beneficial for d >> N (e.g., infinite)
- Still, $\boldsymbol{w}^* \in \mathbb{R}^d$ is potentially infinite dimensional and can not be represented

Yet, we can evaluate the function f that is specified by $\boldsymbol{w}^*$:

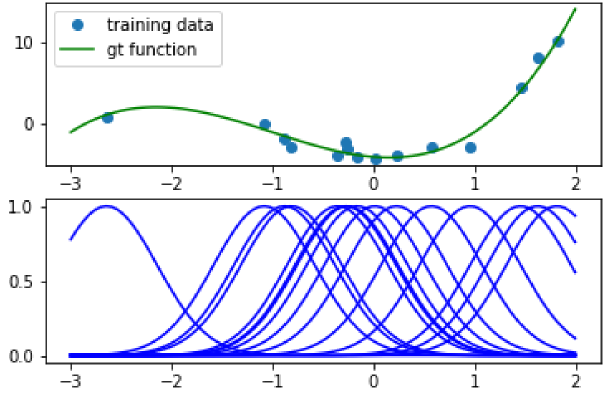
$$f(\boldsymbol{x}) = \underline{\boldsymbol{\phi}(\boldsymbol{x})}^T \boldsymbol{w}^* = \boxed{\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\Phi}^T} \boldsymbol{\alpha} = \boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{\alpha} = \sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})$$

Examples and comparison to RBF regression

For a Gaussian kernel, the prediction corresponds to

$$f(\boldsymbol{x}) = \sum_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) = \sum_i \alpha_i \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}_i||^2}{2\sigma^2}\right)$$

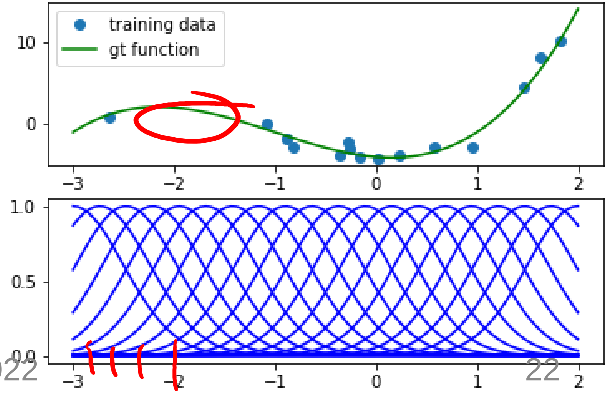
- The kernel allows setting the centres adaptively to the available data!
- One centre per data-point



Comparison: Linear regression with radial basis function (RBF) features

$$f(\boldsymbol{x}) = \sum_i w_i \phi_i(\boldsymbol{x}) = \sum_i w_i \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{\mu}_i||^2}{2\sigma^2}\right)$$

$$\boldsymbol{\mu}_i \ldots \ i^{\text{th}} \text{ center location (fixed)}$$



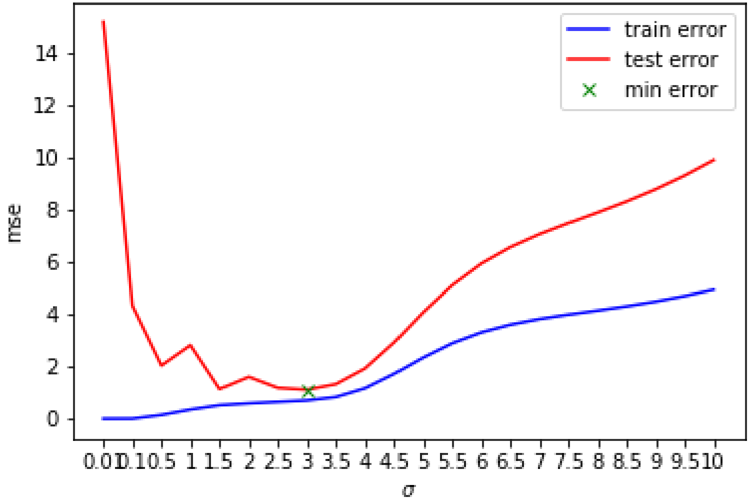
Selecting the hyper-parameters

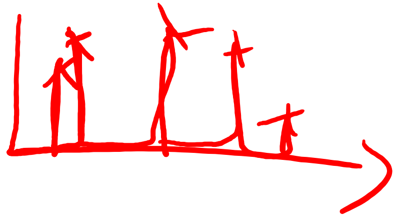
- The parameters of the kernel, e.g., sigma in

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$$

  are called hyper-parameters.

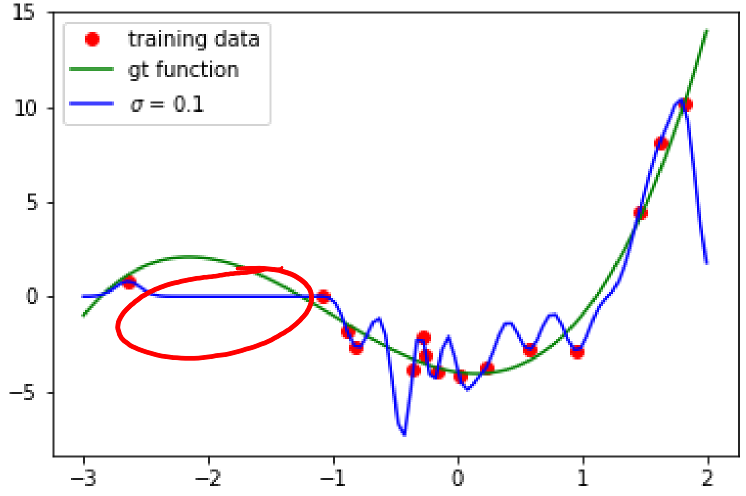
- Choosing them is again a model-selection problem that can be solved via cross-validation.



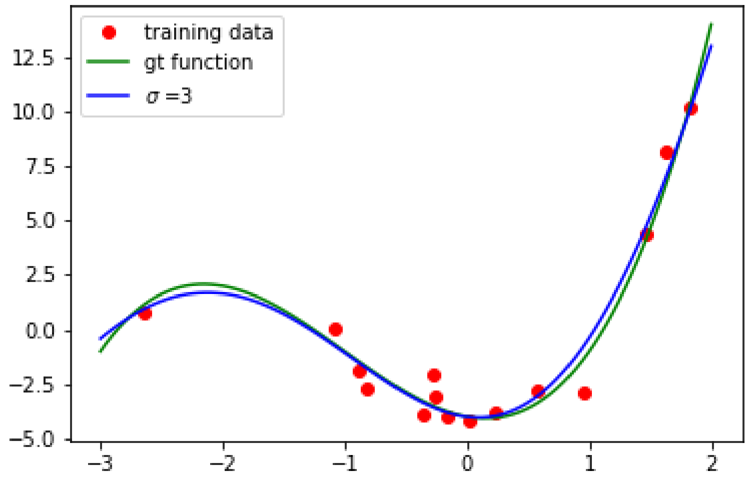


Different bandwidth factors

Overfitting



Good fit

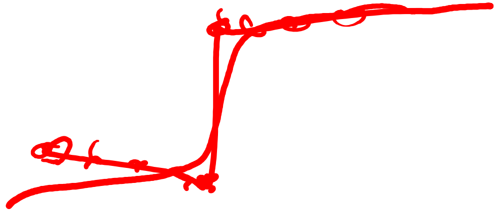


Summary: Kernel ridge regression

The solution for kernel ridge regression is given by

$$f^*(\boldsymbol{x}) = \boldsymbol{k}(\boldsymbol{x})^T(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

- No evaluations of the feature vectors needed
- Only pair-wise scalar products (evaluated by the kernel)
- Need to invert a N x N matrix (can be costly



Note:

- We have to store all samples in kernel-based methods (they also belong to the instance-based or non-parametric methods)
  - Computationally expensive (matrix inverse is $O(n^{2.376})$) !
- Hyper-parameters of the method are given by the kernel-parameters
  - Can be optimized on validation-set
- Very flexible function representation, only few hyper-parameters

Takeaway messages

What have we learned today?

- Kernels estimate the similarity between samples
- They represent an inner product in a feature space
  - Allows to use potentially infinite dimensional
  - That’s ok due to the kernel trick and regularization
- Many standard ML algorithms can be “kernelized”
  - I.e. rewritten in terms of inner products
  - **Regression:** Kernel Ridge regression, Gaussian Processes (to be covered), Support Vector Regression (not covered)
  - **Classification:** SVMs, Kernel Logistic Regression (not covered)

✓ Very flexible representation that adapts to the complexity of the data

✓ Works well with small data sets

× Hard to scale to more complex problems

Self-test questions

You should know now:

- What is the definition of a kernel and its relation to an underlying feature space?
- Why are kernels more powerful than traditional feature-based methods?
- What do we mean by the kernel trick?
- How do we apply the kernel trick to ridge regression?
- How do we compute with infinite dimensional vectors?
- What are hyper-parameters of a kernel and how can we optimize them?