02471 Machine Learning for Signal Processing

Kernel methods and kernel ridge regression

Tommy Sonne Alstrøm

Cognitive Systems section

DTU Compute
Department of Applied Mathematics and Computer Science

Outline

- Last week review
- Non-linear modeling
  - Anomaly detection and de-noising
  - Using general linear models
- Kernel machines
- Examples of kernels
- Kernel algorithms
  - Kernel ridge regression
- Next week

Material: 11.5–11.7 (skip the proof for Theorem 11.2, 11.6.1–11.6.2).

Feedback

- Please remember to fill in the end-of-course evaluation: `https://evaluering.dtu.dk`
- There is a lot of great feedback so far, thank you very much!

Course outline

What you have learned so far:

- Parameter estimation [L2 regularization, biased estimation, mean squared error minimization]. L1 regularization, Bayesian parameter estimation.
- Filtering signals [Stochastic processes, correlation functions, Wiener filter, linear prediction, adaptive filtering using stochastic gradient decent (LMS, APA/NLMS), adaptive filtering using regularization (RLS)
- Signal representations [Time frequency analysis with STFT], Sparsity aware sensing (lasso, sparse priors), factor models [Independent component analysis, Non-negative matrix factorization, k-SVD],
- Bayesian parameter estimation and probabilistic graphical models, Kalman filtering. Inference and EM.

Next two weeks:

- Kernel methods: Today: non-linear models, kernels, kernel Ridge regression, support vector regression.

Learning objectives

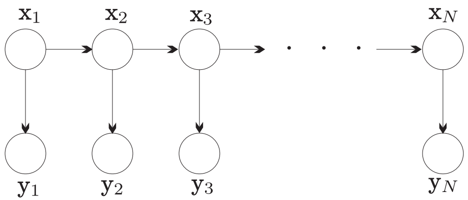
Learning objectives

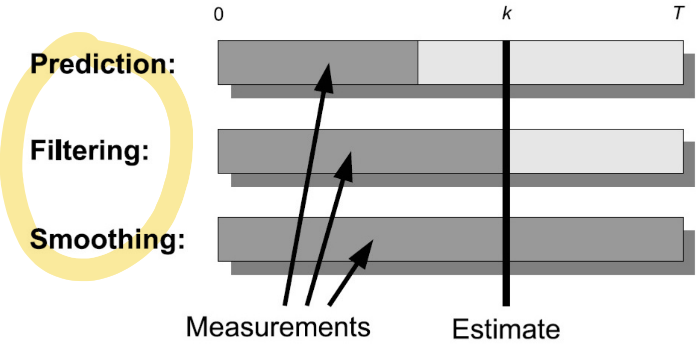
A student who has met the objectives of the course will be able to:

- Explain, apply and analyze properties of discrete time signal processing systems
- Apply the short time Fourier transform to compute the spectrogram of a signal and analyze the signal content
- Explain compressed sensing and determine the relevant parameters in specific applications
- Deduce and determine how to apply factor models such as non-negative matrix factorization (NMF), independent component analysis (ICA) and sparse coding
- Deduce and apply correlation functions for various signal classes, in particular for stochastic signals
- Analyze filtering problems and demonstrate the application of least squares filter components such as the Wiener filter
- Describe, apply and derive non-linear signal processing methods based such as kernel methods and reproducing kernel Hilbert space for applications such as denoising
- Derive maximum likelihood estimates and apply the EM algorithm to learn model parameters
- Describe, apply and derive state-space models such as Kalman filters and Hidden Markov models
- Solve and interpret the result of signal processing systems by use of a programming language
- Design simple signal processing systems based on an analysis of involved signal characteristics, the objective of the processing system, and utility of methods presented in the course
- Describe a number of signal processing applications and interpret the results

Last week review

State-space model and types of operations





Kalman filtering

$f(x_{n-1})$

Linear dynamical system

$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \quad \text{State equation}$$
$$\mathbf{y}_n = H_n \mathbf{x}_n + \mathbf{v}_n, \quad \text{Observation equation}$$

$H(x_n)$

Kalman filter has two stages; prediction, and update (or correction). For prediction, we seek estimation formulas for:

- $\hat{\mathbf{x}}_{n|n-1}$ (called prior estimator)
- $P_{n|n-1}$ (called prior covariance matrix)

For update (correction), we seek estimation formulas for

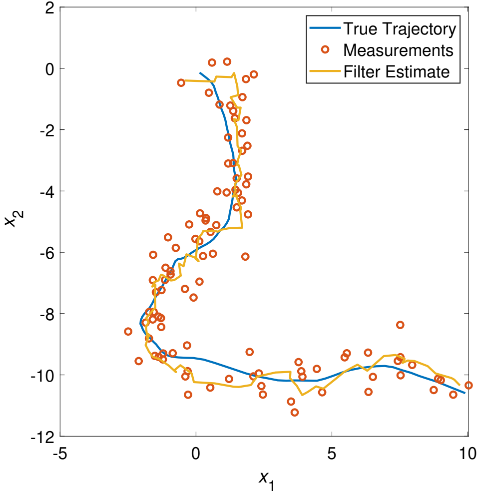
- $\hat{\mathbf{x}}_{n|n}$ (called posterior estimator)
- $P_{n|n}$ (called posterior covariance matrix)

Last week review

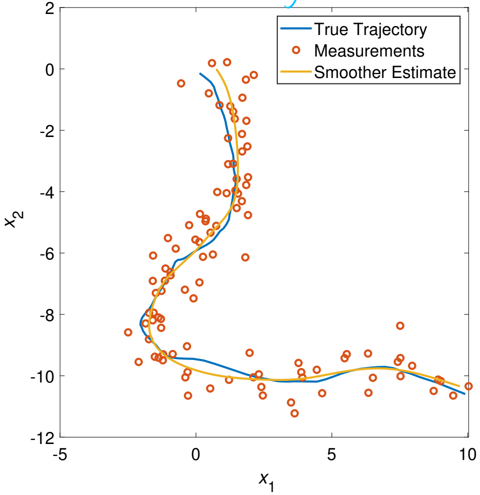
Filtering and smoothing

ex 11.3

Filter



Smoothing

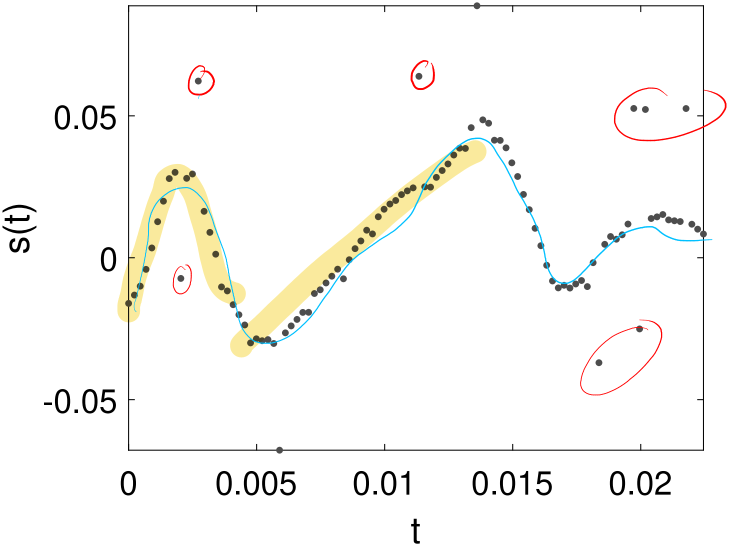


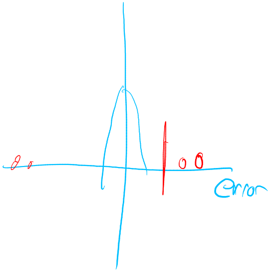
Summary

- For linear dynamical systems, Kalman filtering is the prediction/update formulas.
- Kalman filtering requires specification of model parameters.
- Is used heavily in e.g. object tracking, where the "location" is sensed using noisy sensor readouts.

Non-linear modeling

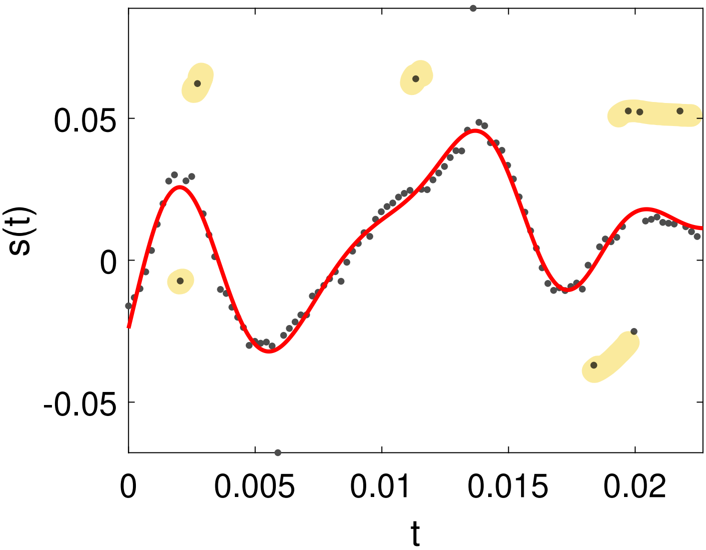
Example: outliers present





Example: one solution

SVR



Revisit – the general linear model

Linear

General linear models

$$\mathrm{y} = f(\mathbf{x}, \boldsymbol{\theta}) := \theta_0 + \sum_{k=1}^{K} \theta_k \phi_k(\mathbf{x})$$

Feature maps

$\phi_k(\mathbf{x})$ is any function that maps $\boldsymbol{x} \in \mathbb{R}^l$, $\phi_k : \mathbb{R}^l \to \mathbb{R}$

Example

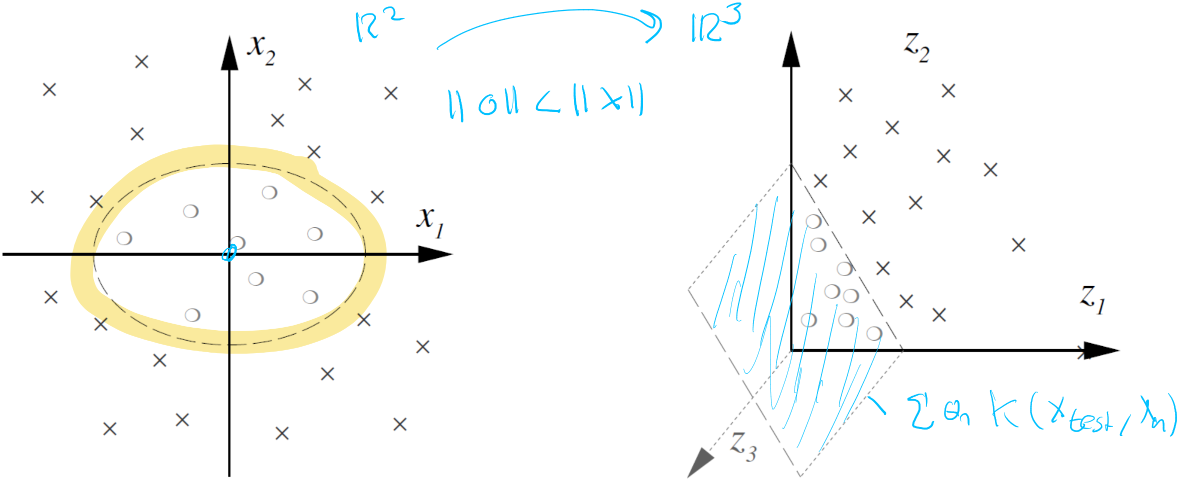
input space → Feature space

$$\boldsymbol{\phi}(\boldsymbol{x}) : \mathbb{R}^2 \to \mathbb{R}^3$$

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^T$$

What kind of data is this useful for?

This particular mapping leads to linear estimation



Source: Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond — 2002, by Schölkopf, Bernhard; Smola, Alexander J.

Kernel machines

Example: Gaussian kernel

$\exp(-\gamma\|x-y\|^2)$

$\exp(-\text{big}) \to 0$

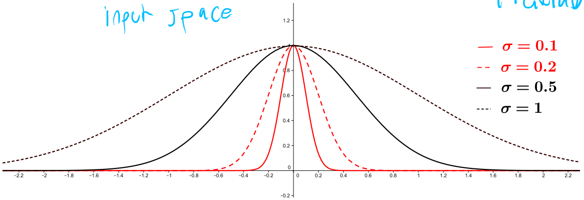
$\exp(\sim 0) \sim 1$

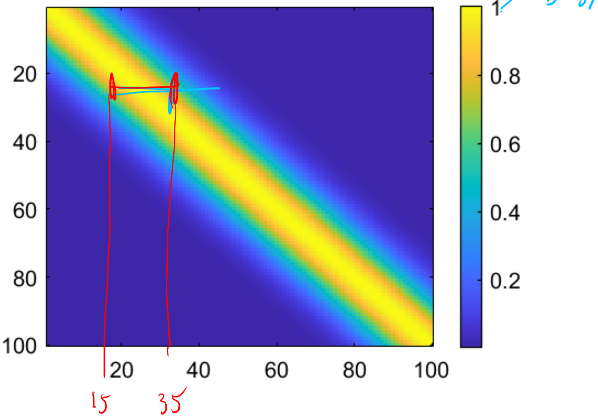
~ 0

$$\boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{y}\|^2\right) \to [0, 1]$$

↖ Matlab

input space





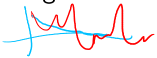
Reproducing kernel Hilbert space (RKHS)

6 weeks

A space with the following properties   Function

✓ 1 A well defined norm, $\|\boldsymbol{x}\|$, that satisfies the usual norm properties (sec 9.2) (normed vector space).

✓ 2 Is complete – which loosely speaking, behaves nicely and all elements exist, that is, every convergent series has smaller and smaller elements (Banach space).



✓ 3 An inner product, denoted as $\langle \cdot, \cdot \rangle$, e.g. in $\mathbb{R}^N$, we have $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T \boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{y}$. Additionally, the norm satisfies $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$, e.g. in $\mathbb{R}^N$ we have $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T \boldsymbol{x}}$ (Hilbert space, denoted $\mathbb{H}$). $\mathbb{R}^D$

4 Has a special function, called the kernel, $\kappa(\cdot, \boldsymbol{x}) \in \mathbb{H}$, with the reproducing property, which essentially means $\kappa(\cdot, \boldsymbol{x})$ is bounded for bounded input (RKHS).

Criteria 1–3 are treated extensively in:

- 01125 Fundamental topological concepts and metric spaces.
- 01325 Mathematics 4: Analysis - a Toolbox in Physics and Engineering

Which is not part of the listed prerequisites.

Some nice properties of functions in RKHS

Kernel properties, assuming $\kappa(\cdot, x) \in \mathbb{H}$, and has the reproducing property

$$\langle \kappa(\cdot, \boldsymbol{x}), \kappa(\cdot, \boldsymbol{y}) \rangle = \kappa(\boldsymbol{x}, \boldsymbol{y}) = \kappa(\boldsymbol{y}, \boldsymbol{x})$$ – symmetric

Or written differently, if $\boldsymbol{\phi}(\boldsymbol{x}) := \kappa(\cdot, \boldsymbol{x})$, then $\phi(x)$ feature map.

$$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{y}) \rangle = \kappa(\boldsymbol{x}, \boldsymbol{y}), \quad \text{Kernel Trick ?}$$

Kernel matrix
$$\mathcal{K} = \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$
$N \times N$ = "Covariance matrix"

$$\mathcal{K} = \mathcal{K}^T$$

$$\boldsymbol{a}\mathcal{K}\boldsymbol{a} \geq 0, \quad \boldsymbol{a} \in \mathbb{R}^l, \quad \mathcal{K} \text{ is positive semi-definite}$$

Expectations from you: can use the properties, but not show the properties!

Example of a mapping

ex 12.1.3

We have this mapping:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^T$$ ⇒ modeling a circle

The corresponding kernel function is

inner prod. $$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{y}) \rangle = \boldsymbol{\phi}^T(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{y})$$

Input space $\mathbb{R}^2$

$$= \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix}$$

$$= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2$$

$$= (x_1y_1 + x_2y_2)^2$$ $= x^Ty^2$

$k(x, y)$ $$= (\boldsymbol{x}^T\boldsymbol{y})^2$$

for $\phi(x)$

"1998" – "2012"

Representer theorem

Representer theorem

Let

$$\Omega : [0, \infty) \to \mathbb{R}$$

Regularizer

be an arbitrary strictly monotonic increasing function. Let also

$$\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \cup \{\infty\}$$

Loss func. $L(\cdot,\cdot) \to \mathbb{R}$

be an arbitrary loss function. Then each minimizer, $f \in \mathbb{H}$, of the regularized minimization task

$\hat{y}_n$

$$\hat{f}(\cdot) = \arg\min_{f \in \mathbb{H}} J(f) := \sum_{n=1}^{N} \mathcal{L}\left(y_n, f(\boldsymbol{x}_n)\right) + \lambda\Omega\left(\|f\|^2\right)$$

data

admits a representation of the form,

kernel func.

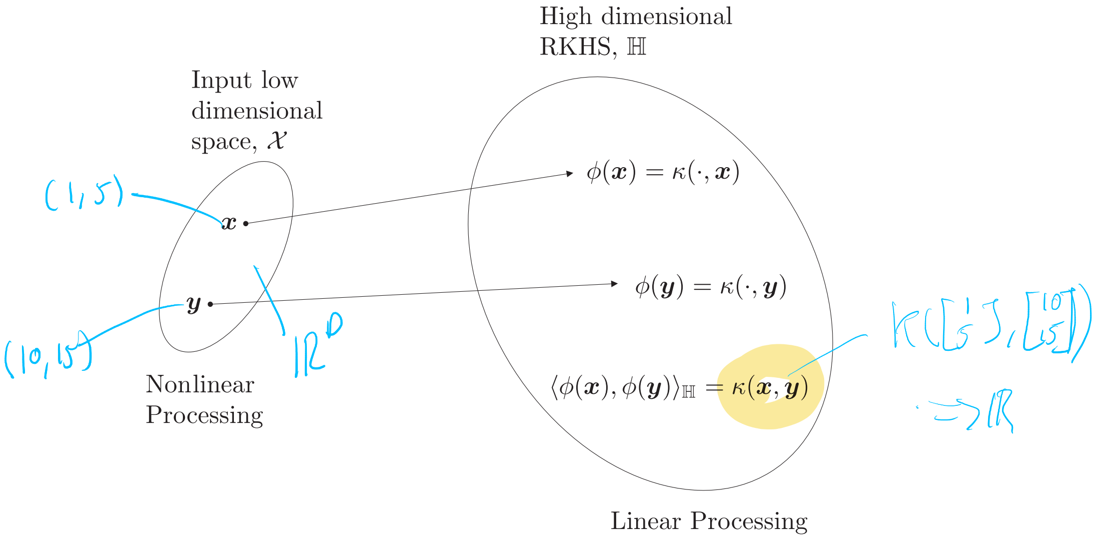
$$\hat{f}(\cdot) = \sum_{n=1}^{N} \theta_n \kappa(\cdot, \boldsymbol{x}_n)$$

linear in param.

SVM
SVR

where $\theta_n \in \mathbb{R}, n = 1, 2, \cdots, N$

Consequences – linear processing



Consequences – algorithms can generally be kernelized

DESIGN

1. Map (implicitly) the input training data to an RKHS. — Choose $K(x, y) = \mathbb{R}_+$
2. Solve the linear estimation task in $\mathbb{H}$.
3. Cast the algorithm in terms of inner products $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ $\Rightarrow$ $K(x, y)$ ($\mathbb{R}^l$ is a Hilbert space).
4. Replace each inner product by a kernel evaluation, that is, $\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{y}) \rangle = \kappa(\boldsymbol{x}, \boldsymbol{y})$.

Example (exercise 12.1.3) :

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix}^T$$

The corresponding kernel would be

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T\boldsymbol{y})^2, \quad \text{Polynomial kernel}$$

Summary

- Reproducing kernel Hilbert space enables linear processing while obtaining nonlinear decision boundaries.
- If you limit yourself to already proven reproducing kernels, you do not as such need to understand the theory behind RKHS but can readily apply it.
- Choice of kernel and kernel parameters is critical for performance.

DESIGN

Examples of kernels

Example of kernels

The Gaussian (or exponential, or rbf) kernel (be aware that sometimes $\gamma = \frac{1}{2\sigma^2}$):

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{y}\|^2\right)$$

The inhomogeneous polynomial kernel:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T\boldsymbol{y} + c)^r$$

The Laplacian kernel:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp(-t\|\boldsymbol{x} - \boldsymbol{y}\|_1)$$

Creating new kernels

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= c k_1(\mathbf{x}, \mathbf{x}') && (6.13)\\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') && (6.14)\\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.15)\\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.16)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') && (6.17)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') && (6.18)\\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) && (6.19)\\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x}' && (6.20)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.21)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.22)
\end{aligned}
$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Source: Pattern Recognition and Machine Learning, 2006, C. Bishop

Kernel algorithms

Example – Kernel ridge regression

ex 12.2

Kernel Ridge regression (without bias)

Assume the regression task (η_n is white noise)

$$y_n = g(\boldsymbol{x}_n) + \eta_n, \quad n = 1, 2, \cdots, N$$

Assume the solution (according to representer theorem)

$$f(\cdot) = \sum_{m=1}^{N} \theta_m \kappa(\cdot, \boldsymbol{x}_m)$$

The model, where $C \in \mathbb{R}$ is the regularization parameter, is then:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$\hat{y}_n$

LOSS

$$J(\boldsymbol{\theta}) := \sum_{n=1}^{N} \left(y_n - \boxed{\sum_{m=1}^{N} \theta_m \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)} \right)^2 + C\langle f, f \rangle$$

$$\hat{\boldsymbol{\theta}} = (\mathcal{K} + CI)^{-1} \boldsymbol{y}$$

$\| \quad \hat{\theta}_{RR} = (X^T X + CI)^{-1} X^T y$

Ridge regression vs kernel ridge regression

Recall from section 3.8 that the ridge regression (without bias) minimizes the following function:

$$J_{RR}(\boldsymbol{\theta}) := \sum_{n=1}^{N}\left(y_n - \sum_{i=1}^{l}\theta_i x_{ni}\right)^2 + \lambda \sum_{i=1}^{l}|\theta_i|^2$$

The kernel ridge regression (without bias) instead minimizes

$$J_{KRR}(\boldsymbol{\theta}) := \sum_{n=1}^{N}\left(y_n - \sum_{m=1}^{N}\theta_m \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)\right)^2 + C\langle f, f\rangle$$

where $C \in \mathbb{R}$ is a regularization parameter.

Derivation of kernel ridge regression

From the representer theorem

$$\hat{f}(\cdot) = \arg\min_{f \in \mathbb{H}} \; J(f) := \sum_{n=1}^{N} \mathcal{L}\left(y_n, f(\boldsymbol{x}_n)\right) + \lambda \Omega\left(\|f\|^2\right)$$

$$\hat{f}(\cdot) = \sum_{n=1}^{N} \theta_n \kappa(\cdot, \boldsymbol{x}_n)$$

We can with $f(\boldsymbol{x}) = \sum_{m=1}^{N} \theta_n \kappa(\boldsymbol{x}, \boldsymbol{x}_m)$, and a squared loss, arrive at the kernel ridge regression cost function:

$$J(\boldsymbol{\theta}) := \sum_{n=1}^{N} \left(y_n - \sum_{m=1}^{N} \theta_m \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) \right)^2 + C\langle f, f \rangle$$

Lecture summary

- Reproducing kernel Hilbert space enables linear processing while obtaining nonlinear signal processing.
- If you limit yourself to apply already proven reproducing kernels, you do not need to understand the theory behind RKHS to apply kernel methods. Use list of kernels.
- Choice of kernel and kernel parameters is critical for performance.
- No restrictions on $\boldsymbol{x}$, only on the kernels.
- Requires tuning of parameters, but is not “that” sensitive.
- The kernel trick is widely used, also for e.g. kernel LMS, kernel-k-means etc.

Next week

2022: 6.1
PS3 6.2

Material: 11.8.

- Exam preparation and general Q/A (maximum 45 minutes)
- Lecture (will start latest at 14.00, but if talk about exam is shorter we may start earlier):
  - Anomaly detection using Huber loss
  - Support vector regression.