02471 Machine Learning for Signal Processing

Kernel methods – Support vector regression

Tommy Sonne Alstrøm

Cognitive Systems section

DTU Compute
Department of Applied Mathematics and Computer Science

Outline

- Course admin
- Last week review
- Anomaly detection and de-noising
- Support vector regression

Material: 11.8.

Course outline

So far:

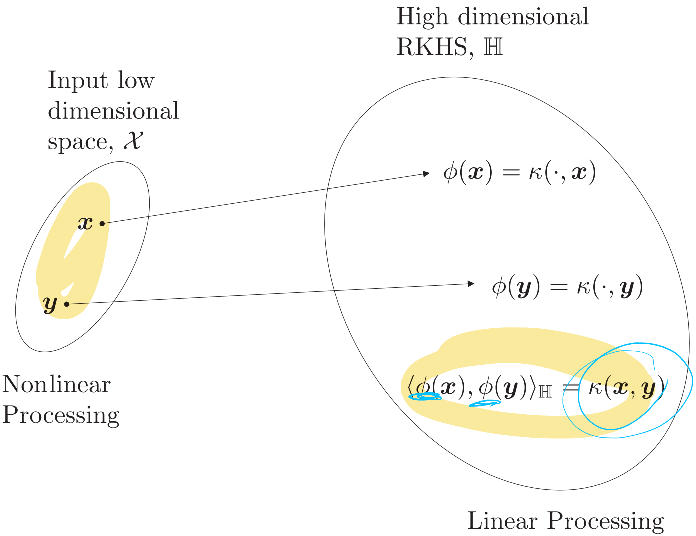
- Parameter estimation [Regularization, biased estimation, mean squared error minimization].
- Signal representations [Time frequency analysis, sparsity aware learning, factor models].
- Filtering signals [Linear regression, adaptive filtering using stochastic gradient decent (LMS, NLMS, RLS), adaptive filtering using regularization].
- Sequential models [hidden markov models, linear dynamical systems, kalman filter].

Next weeks

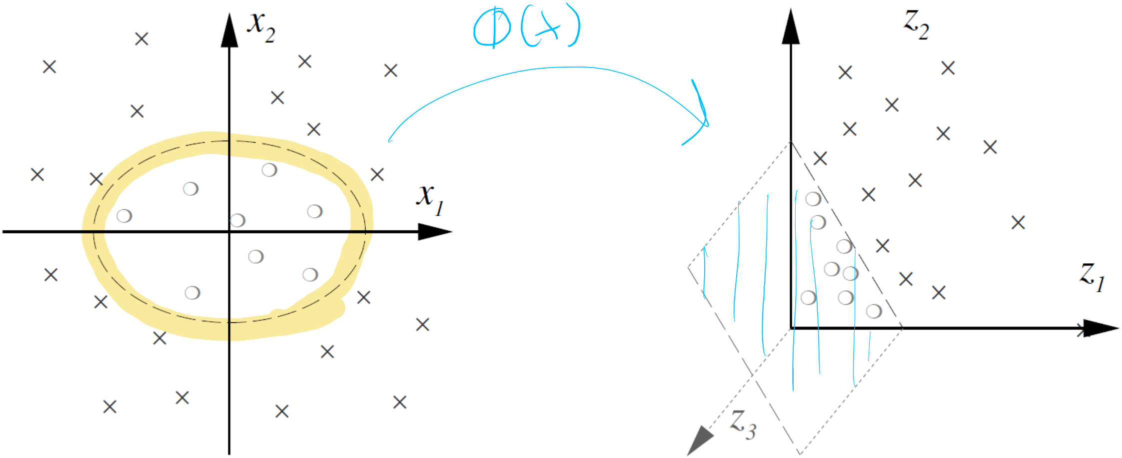
- Kernel methods [kernel ridge regression, support vector regression].

Last week review

Consequences – linear processing



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.

Algorithms can generally be kernelized, if they can be expressed as inner products

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

Simply example:

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

The corresponding kernel would be

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

Some kernel properties

Representer theorem in a nutshell

Linearity comes from representer theorem, stating a solution can (under certain circumstances) be written as

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

fix today

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

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

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

$$\mathcal{K} = \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots \\ \kappa(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

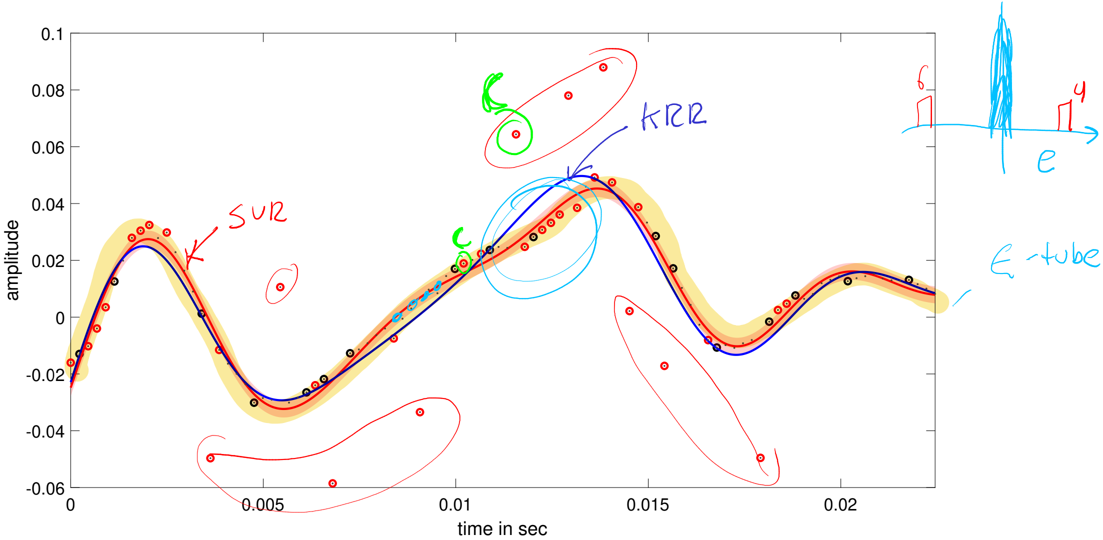
$$\mathcal{K} = \mathcal{K}^T$$

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

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 x, only on the kernels.
- Requires tuning of parameters, but is not "that" sensitive.

Results - support vector regression

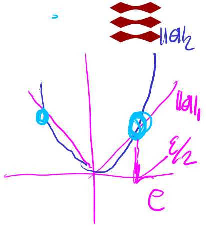


One solution – the Huber loss

Assume the regression task (η_n is i.i.d. noise)

$\mathcal{N}(\cdot,\cdot) + P_{df,\,outlier}$ Sym.

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



Huber showed that, assuming the noise follows a symmetric pdf, the optimal loss is

Huber loss

DESIGN

$$\mathcal{L}(y, f(\boldsymbol{x})) = \begin{cases} \epsilon|y - f(\boldsymbol{x})| - \frac{\epsilon^2}{2}, & \text{if} \quad |y - f(\boldsymbol{x})| > \epsilon \\ \frac{1}{2}|y - f(\boldsymbol{x})|^2, & \text{if} \quad |y - f(\boldsymbol{x})| \le \epsilon \end{cases}$$

$\|\theta\|_1$ $\|\theta\|_2^2$ $\mathcal{N}(\cdot\cdot)$

$\epsilon|\epsilon| - \epsilon^2/2 = \epsilon^2/2$

$\frac{1}{2}|\epsilon|^2 = \epsilon^2/2$

An alternative, and more computationally efficient loss is the ϵ–insensitive loss

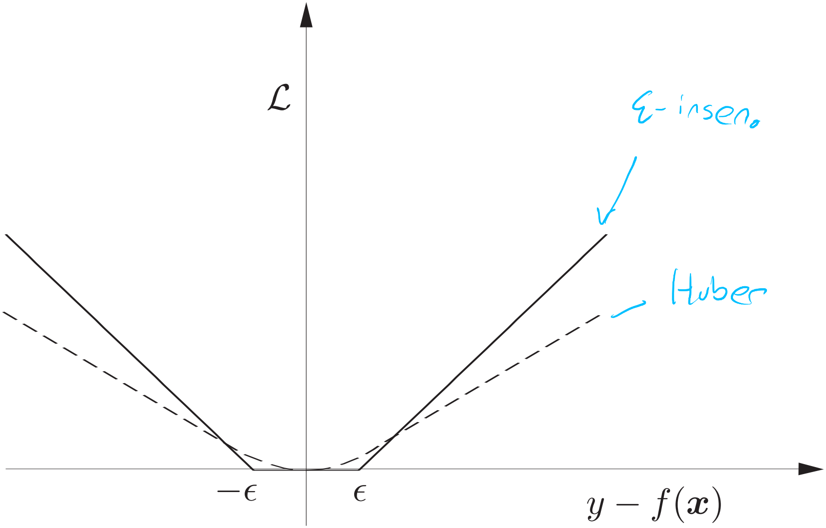
ϵ–insensitive loss

$$\mathcal{L}(y, f(\boldsymbol{x})) = \begin{cases} |y - f(\boldsymbol{x})| - \epsilon, & \text{if} \quad |y - f(\boldsymbol{x})| > \epsilon \\ 0, & \text{if} \quad |y - f(\boldsymbol{x})| \le \epsilon \end{cases}$$

$\|\theta\|_1$

$\Rightarrow$ SVR

The loss functions



Support vector regression

Introduce slack variables

$L = 0$ if $|\text{error}| \le \varepsilon$

This loss leads to support vector regression through a couple of steps.
First, rewrite the loss by introducing slack variables ξ_n and $\tilde{\xi}_n$:

$-0{,}2 \le 0{,}1 + 0$

error

$$y_n - f(\boldsymbol{x}_n) \le \epsilon + \xi_n$$

$$-(y_n - f(\boldsymbol{x}_n)) \le \epsilon + \tilde{\xi}_n$$

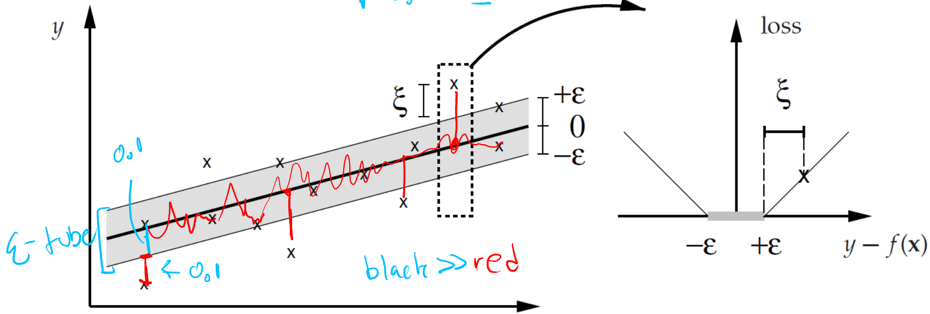
$-$ error $\le$

$+0{,}2 \le 0.1 + 0.1$

Min ξ_n

$\xi \ge 0$

$\xi_n \cdot \tilde{\xi}_n = 0$



Regularized minimization of slack variables

13.1.1

DESIGN

func. smoother

model misfit

$$\underset{\boldsymbol{\theta},\boldsymbol{\xi},\tilde{\boldsymbol{\xi}},}{\arg\min}\ J(\boldsymbol{\theta},\boldsymbol{\xi},\tilde{\boldsymbol{\xi}}) := \lambda\frac{1}{2}\|\boldsymbol{\theta}\|^2 + \cancel{C}\left(\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N}\tilde{\xi}_n\right)$$

s.t. $y_n - f(\boldsymbol{x}_n) \leq \epsilon + \xi_n$

$-(y_n - f(\boldsymbol{x}_n)) \leq \epsilon + \tilde{\xi}_n$

$\xi_n \geq 0$

$\tilde{\xi}_n \geq 0$

Next step ??

$\Rightarrow \xi \cdot \tilde{\xi} = 0$

Lagrarian multipliers

13.1.2 + .3

"skim"

From appendix C.2. If we have the problem:

prob. statement

$$\min_{\boldsymbol{\theta}} \quad J(\boldsymbol{\theta})$$

$$\text{s.t.} \quad f_i(\boldsymbol{\theta}) \geq 0, \quad i = 1, 2, \cdots, m$$

ξ, $\hat{\xi}$, errors..

then the Lagrangian is

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) - \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{\theta})$$

And is solved by:

$$\left. \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) \right|_{\theta = \theta_*} = \mathbf{0}$$

$$\lambda_i \geq 0 \qquad i = 1, 2, \cdots, m$$

$$\lambda_i f_i(\boldsymbol{\theta}_*) = 0 \quad i = 1, 2, \cdots, m$$

The solution

\xi and \xi tilde

L.M. for ξ and

Assume $f(\boldsymbol{x}) = \boldsymbol{\theta}^T\boldsymbol{x} + \theta_0$, this leads to the solution

$$\hat{\boldsymbol{\theta}}^T = \sum_{n=1}^{N} (\tilde{\lambda}_n - \lambda_n)\boldsymbol{x}_n^T$$

which leads to the solution

SVR

$$\hat{y} = f(\boldsymbol{x}) = \sum_{n=1}^{N} (\tilde{\lambda}_n - \lambda_n)\boldsymbol{x}_n^T\boldsymbol{x} + \hat{\theta}_0$$

$k(x_n, x)$

This is an unusual solution! Can we leverage on this?

Do you see any weakness in this solution?

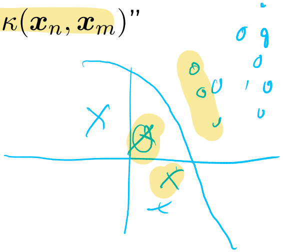
$\sum$

$n \in$ outside or on the ε-tube

Remember our kernelization recipe

Step 1–4 basically:

> "Replace each inner product by a kernel evaluation, that is, $\langle \boldsymbol{x}_n, \boldsymbol{x}_m \rangle = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$"



The prediction equation was

$$\hat{y} = f(\boldsymbol{x}) = \sum_{n=1}^{N} (\tilde{\lambda}_n - \lambda_n) \boldsymbol{x}_n^T \boldsymbol{x} + \hat{\theta}_0$$

The kernel trick results in

Support vector regression prediction

$N' << N$

$$\hat{y} = \sum_{n=1}^{N} (\tilde{\lambda}_n - \lambda_n) \kappa(\boldsymbol{x}_n, \boldsymbol{x}) + \hat{\theta}_0$$

The points for which either λ_n or $\tilde{\lambda}_n$ are non-zero are called the support vectors. ex 13,64

Additionally, it can be shown that; $\lambda_n \tilde{\lambda}_n = 0$, and $0 \leq \tilde{\lambda}_n \leq C$, $0 \leq \lambda_n \leq C$.

ϵ–support vector regression – the complete problem

Support vector regression prediction

Sol:

$$\hat{y} = \sum_{n=1}^{N} (\tilde{\lambda}_n - \lambda_n)\kappa(\boldsymbol{x}_n, \boldsymbol{x}) + \hat{\theta}_0$$

Support vector regression minimization

LM =>

$$\underset{\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}},}{\arg\max} \quad \sum_{n=1}^{N} (\tilde{\lambda}_n - \lambda_n) y_n - \epsilon(\tilde{\lambda}_n - \lambda_n)$$

$$- \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} (\tilde{\lambda}_n - \lambda_n)(\tilde{\lambda}_m - \lambda_m)\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

$$\text{s.t.} \quad 0 \leq \tilde{\lambda}_n \leq C, \quad 0 \leq \lambda_n \leq C, \quad n = 1, 2, \cdots, N$$

$$\sum_{n=1}^{N} \tilde{\lambda}_n = \sum_{n=1}^{N} \lambda_n$$

Summary

- Huber loss is the optimal robust loss if the noise follows a symmetric pdf.
- A tractable loss that leads to support vector regression is the ϵ-insensitive loss.
- Support vector regression:
  - seems to not have "training", but store points instead.
  - is solved by Lagrangian multipliers.
  - is very useful for de-noising and for anomaly detection.

Next week

Winter is ~~coming~~ here... exam...

Once more, thanks for all who evaluated. We use the feedback!

The course usually lacks TAs, please consider being TA next year.

Thank you for attending.