6 Machine Learning in Computational Mechanics

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book

Contents

- 5 Advanced Physics-Informed Neural Networks
- 6.1.1 Singular Value Decomposition
- 6.2 Reduced Order Models
- 6.3 Sparse Identification of Non-Linear Dynamical Systems – SINDy
- 6.4 Clustering
- 6.5 Support Vector Machines
- 7 Material Modeling with Neural Networks

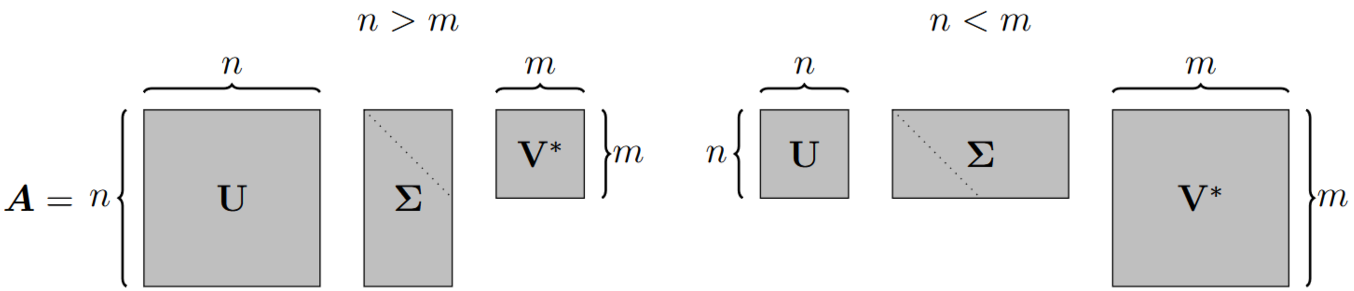
6.1.1 Singular Value Decomposition

Singular value decomposition is a unique matrix factoriziation ($\boldsymbol{A} \in \mathbb{C}^{n \times m}$)

$$\boldsymbol{A} = \boldsymbol{U\Sigma V}^*$$

V^* is the conjugate transpose of V

With $\boldsymbol{U} \in \mathbb{C}^{n \times n}, \boldsymbol{\Sigma} \in \mathbb{R}_0^{+\,n \times m}, \boldsymbol{V}^* \in \mathbb{C}^{m \times m}$



Matrices $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$ are obtained through eigendecompositions of $\boldsymbol{AA}^*$ and $\boldsymbol{A}^*\boldsymbol{A}$

- Columns of $\boldsymbol{U}$ consist of the eigenvectors of $\boldsymbol{AA}^*$
- Columns of $\boldsymbol{V}$ consist of the eigenvectors of $\boldsymbol{A}^*\boldsymbol{A}$
- Diagonal entries of $\boldsymbol{\Sigma}$ are the square roots of the corresponding eigenvalues

$\boldsymbol{U}, \boldsymbol{V}$ are unitary matrices, i.e., $\boldsymbol{U}^*\boldsymbol{U} = \boldsymbol{I}, \boldsymbol{V}^*\boldsymbol{V} = \boldsymbol{I}$
(for real matrices: orthogonal matrices)

Non-zero eigenvalues of $\boldsymbol{AA}^*$ and $\boldsymbol{A}^*\boldsymbol{A}$ are the same

Signs of the eigenvectors of $\boldsymbol{U}$ and $\boldsymbol{V}$ must be consistent and can be checked with

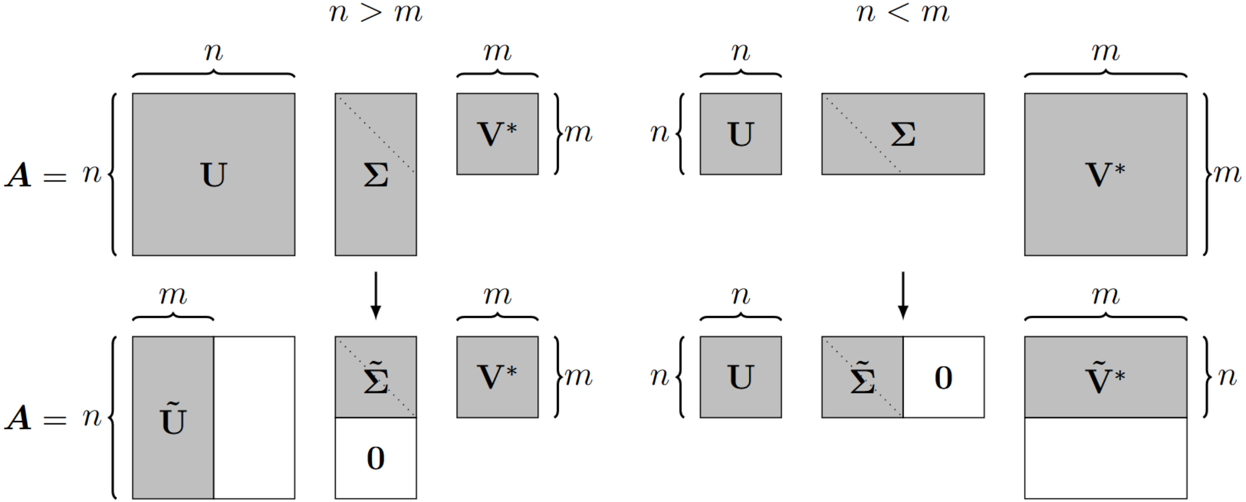
$$\boldsymbol{\Sigma} = \boldsymbol{U}^*\boldsymbol{AV}$$

If an entry of $\boldsymbol{\Sigma}$ is negative, the signs of the corresponding eigenvector in $\boldsymbol{U}$ or $\boldsymbol{V}$ must be flipped.

6.1.1 Singular Value Decomposition

Singular value decomposition as dimensionality reduction technique

- **Economy singular value decomposition**



- Exact reconstruction of $\boldsymbol{A}$ with truncated $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$ (for $n \neq m$)

for $n > m \quad \boldsymbol{A} = \widetilde{\boldsymbol{U}}\widetilde{\boldsymbol{\Sigma}}\boldsymbol{V}^*$

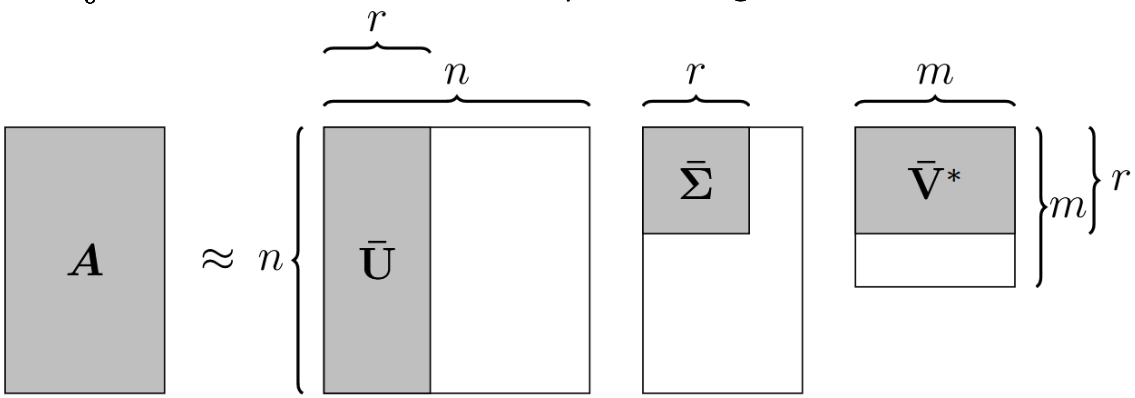
for $n < m \quad \boldsymbol{A} = \boldsymbol{U}\widetilde{\boldsymbol{\Sigma}}\widetilde{\boldsymbol{V}}^*$

- **Truncated singular value decomposition**: truncation of $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$ beyond the mismatch in dimensionality

6.1.1 Singular Value Decomposition

Truncated singular value decomposition: truncation of $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$ beyond the mismatch in dimensionality

- $\overline{\boldsymbol{U}} \in \mathbb{C}^{n \times r}, \overline{\boldsymbol{\Sigma}} \in \mathbb{R}_0^{+\,r \times r}, \overline{\boldsymbol{V}} \in \mathbb{C}^{m \times r}$ are truncated up to r^{th} singular value



- Reconstruction is now approximative!

$$\boldsymbol{A} \approx \overline{\boldsymbol{U}}\overline{\boldsymbol{\Sigma}}\overline{\boldsymbol{V}}^*$$

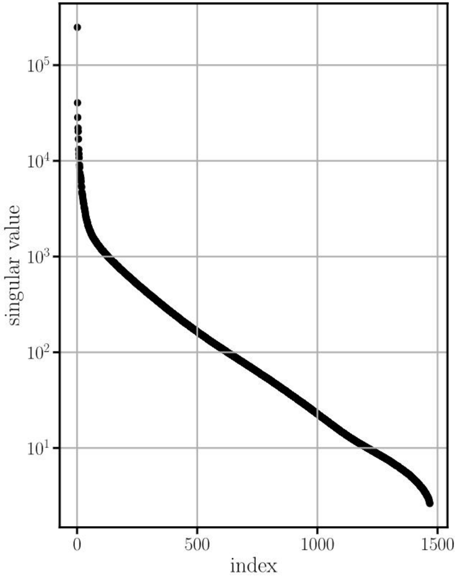
- The truncated singular value decomposition enables dimensionality reduction
  - Principal components analysis is a related dimensionality reduction technique (see 6.1.2)

6.1.1.1 Image Compression

Image compression as example of dimensionality reduction with singular value decomposition

- Gray-scale image as matrix with pixel entries in x_1 and x_2 direction

Original image ($1'868 \times 1468$)



Singular values (diagonal entries of Σ)

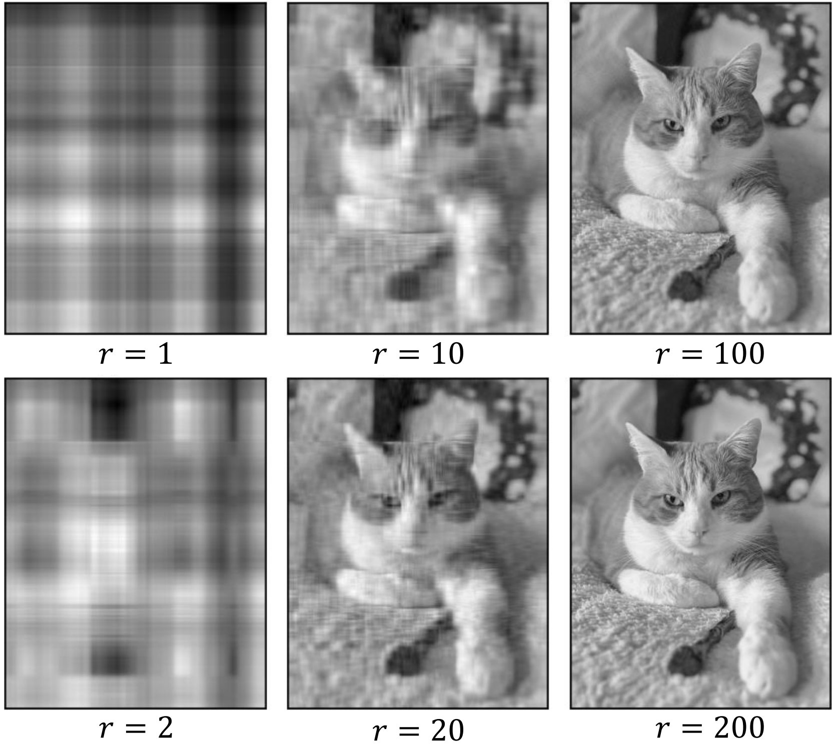
Truncated singular value decomposition leads to $nr + rr + mr$ values instead of nm values

6.1.1.1 Image Compression

Low-dimensional procedure for the characterization of human faces, Sirovich et al. 1987

Compression ratios for $r = 1, 2, 10, 20, 100, 200$:

$$0.116\%, 0.243\%, 1.22\%, 2.43\%, 12.5\%, 25.8\%$$



Further compression possible if multiple images are considered simultaneously, i.e., a common compression, e.g., eigenfaces from a library of faces can reconstruct yet unseen faces

Exercises

- E.22 Singular Value Decomposition (P & C)
  - First, perform a singular value decomposition on a given matrix. Next implement a general singular value decomposition and apply it to an image.

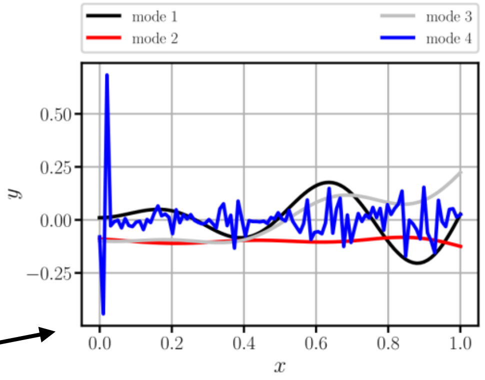
6.1.1.2 Identification of a Reduced Basis

Consider the spatio-temporal function

$$y = f(x,t) = \sin(2\pi x) + 2\sin(4\pi x)xt^2 + x^2 t + 5$$

Measurements obtained at m temporal snapshots on a spatial grid with n points (stored in a snapshot matrix $\boldsymbol{X}$)

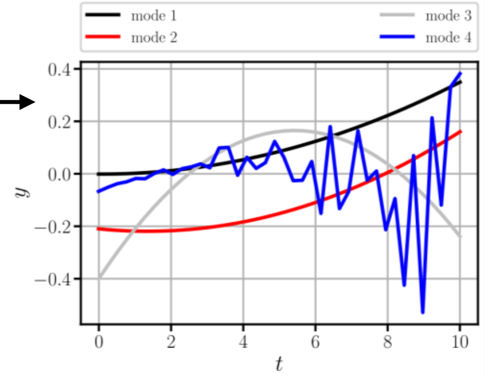
$$\boldsymbol{X} = \begin{pmatrix} f(x_1,t_1) & f(x_2,t_1) & \dots & f(x_n,t_1) \\ f(x_1,t_2) & f(x_2,t_2) & \dots & f(x_n,t_2) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_1,t_m) & f(x_2,t_m) & \dots & f(x_n,t_m) \end{pmatrix}$$



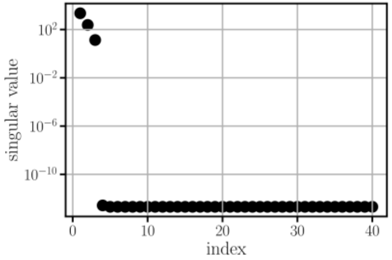
Singular value decomposition of snapshot matrix

$$\boldsymbol{X} = \boldsymbol{U\Sigma V}^*$$

- Spatial modes can be extracted from columns of $\boldsymbol{V}$
- Temporal modes can be extracted from columns of $\boldsymbol{U}$



Only the first 3 modeshapes are associated with non-zero singular values:
→ $\boldsymbol{X}$ can be reconstructed exactly with $r = 3$



6.1.1.2 Identification of a Reduced Basis

Modeshapes identified from $\boldsymbol{X}$ using singular value decomposition

$$\boldsymbol{X} = \overline{\boldsymbol{U}}\boldsymbol{\Sigma}\overline{\boldsymbol{V}}^*$$

provide a reduced basis (using time-dependent coefficients $\boldsymbol{c}(t) \in \mathbb{R}^{m\times r}$)

$$\boldsymbol{c}(t) = \boldsymbol{X}\overline{\boldsymbol{V}}$$

$$\boldsymbol{X} \approx \boldsymbol{c}(t)\overline{\boldsymbol{V}}^T$$

Exploiting the orthonormal property of $\boldsymbol{V}$ ($\boldsymbol{V}^*\boldsymbol{V} = \boldsymbol{I}$)

Consider a system of equations (task is to solve for $\boldsymbol{y}$)

$$\boldsymbol{K}\boldsymbol{y} = \boldsymbol{f}$$

Prior knowledge of the solution $\boldsymbol{y}$ is stored in the snapshot matrix $\boldsymbol{X}$

Singular value decomposition yields $\boldsymbol{V}$, i.e., $\overline{\boldsymbol{V}}$

Projection into a lower dimensional system

$$\boldsymbol{K}\boldsymbol{y}\overline{\boldsymbol{V}} = \boldsymbol{f}\overline{\boldsymbol{V}}$$

$$\boldsymbol{K}\boldsymbol{c} = \boldsymbol{f}\overline{\boldsymbol{V}}$$

Solve for $\boldsymbol{c}$ instead of $\boldsymbol{y}$ and recover $\boldsymbol{y}$ with

$$\boldsymbol{y} \approx \boldsymbol{c}\overline{\boldsymbol{V}}$$

Exercises

- E.23 Introduction to Reduced Order Models (C)
  - Sample a basic spatio-temporal function and extract the essential spatial and temporal modes using a singular value decomposition

6.2 Reduced Order Models

Acceleration of a dynamic finite element simulation using a reduced basis

Consider the semi-discrete finite element equations

$$\boldsymbol{M}\ddot{\boldsymbol{u}}(t) + \boldsymbol{K}\boldsymbol{u}(t) = \boldsymbol{F}(t)$$

$\boldsymbol{M}$ is the mass matrix, $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{F}$ is the source vector

Solvable by integration through time, e.g., central difference approximation of $\ddot{\boldsymbol{u}}$

$$\dot{\boldsymbol{u}}(t) \approx \frac{\boldsymbol{u}(t+\Delta t) - \boldsymbol{u}(t+\Delta t)}{2\Delta t}$$

$$\ddot{\boldsymbol{u}}(t) \approx \frac{\boldsymbol{u}(t+\Delta t) - 2\boldsymbol{u}(t) + \boldsymbol{u}(t-\Delta t)}{\Delta t^2}$$

Inserted in the finite element equations

$$\boldsymbol{M}\left(\frac{\boldsymbol{u}(t+\Delta t) - 2\boldsymbol{u}(t) + \boldsymbol{u}(t-\Delta t)}{\Delta t^2}\right) + \boldsymbol{K}\boldsymbol{u}(t) \approx \boldsymbol{F}(t)$$

Rewritten in an explicit form

$$\frac{1}{\Delta t^2}\boldsymbol{u}(t+\Delta t)\boldsymbol{M} \approx \boldsymbol{F}(t) - \left(\boldsymbol{K} - \frac{2}{\Delta t^2}\boldsymbol{M}\right)\boldsymbol{u}(t) - \frac{1}{\Delta t^2}\boldsymbol{M}\boldsymbol{u}(t-\Delta t)$$

Compute next time step $\boldsymbol{u}(t+\Delta t)$ by inverting $\boldsymbol{M}$

6.2.2 Reduced Order Modeling with Finite Elements

From a snapshot matrix

$$\boldsymbol{X} = \left(\boldsymbol{u}(t_1), \boldsymbol{u}(t_2), \dots, \boldsymbol{u}(t_m)\right)^T \in \mathbb{R}^{n\times m}$$

A reduced basis is identified (with the truncated singular value decomposition)

$$\boldsymbol{\psi} = \bar{\boldsymbol{V}} \in \mathbb{R}^{n\times r}$$

Projection of degrees of freedom $\boldsymbol{u}(t) \in \mathbb{R}^n$ onto reduced basis $\boldsymbol{c}(t) \in \mathbb{R}^r$

$$\boldsymbol{u}(t) \approx \boldsymbol{\psi}\boldsymbol{c}(t)$$

Insertion into the semi-discrete finite element equations

$$\boldsymbol{M}\boldsymbol{\psi}\ddot{\boldsymbol{c}}(t) + \boldsymbol{K}\boldsymbol{\psi}\boldsymbol{c}(t) = \boldsymbol{F}(t)$$

r degrees of freedom with n equations

Pre-multiplication with $\boldsymbol{\psi}^T$ to project the system onto the reduced space (to reduce number of equations to r)

$$\boldsymbol{\psi}^T\boldsymbol{M}\boldsymbol{\psi}\ddot{\boldsymbol{c}}(t) + \boldsymbol{\psi}^T\boldsymbol{K}\boldsymbol{\psi}\boldsymbol{c}(t) = \boldsymbol{\psi}^T\boldsymbol{F}(t)$$

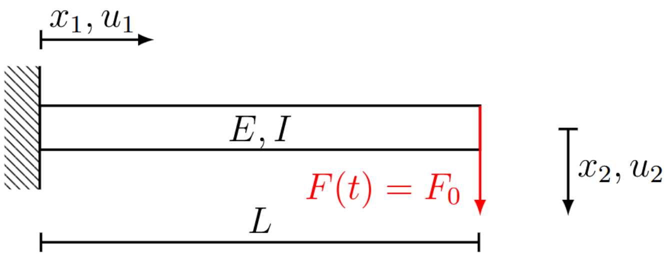
Solve for $\boldsymbol{c}(t)$ and recover full solution vector with

$$\boldsymbol{u}(t) \approx \boldsymbol{c}(t)\boldsymbol{\psi}^T$$

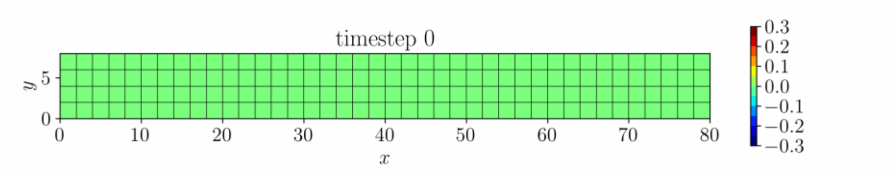
Proper orthogonal decomposition: projection of solution (to differential equation) to lower-dimensional subspace using orthogonal basis functions

6.2.3 Cantilever Beam

Simulation of a cantilever beam with two-dimensional finite elements

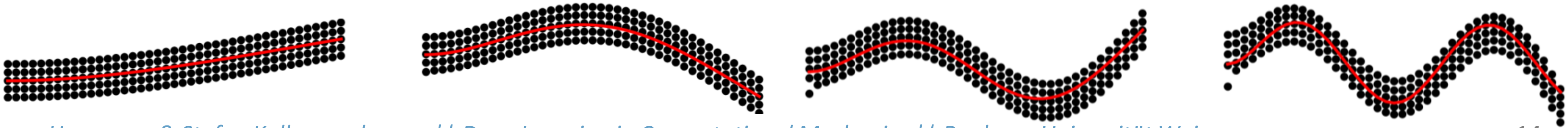


- Full finite element simulation for $m = 5000$ timesteps with $n = 410$ degrees of freedom → X

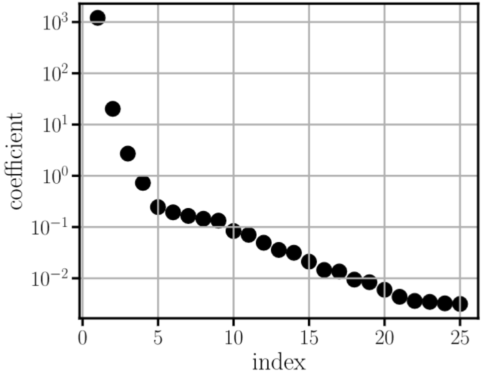


- Singular value decomposition on $\boldsymbol{X}$ to obtain reduced basis $\boldsymbol{\psi} = \overline{\boldsymbol{V}}$
  - Identification of the first four mode shapes of an Euler-Bernoulli beam

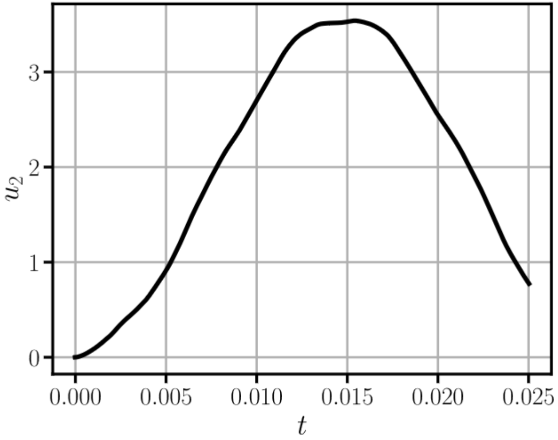
$$m_n(x_1) = \cosh(\beta_n x_1) - \cos(\beta_n x_1) - \sigma_n\big(\sinh(\beta_n x_1) - \sin(\beta_n x_1)\big)$$



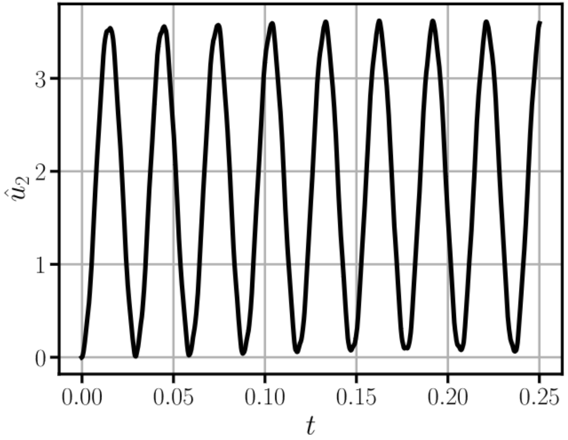
6.2.3 Cantilever Beam



From the singular values, a truncation level of $r = 5$ is selected (reducing the system from 410×410 to 5×5)



Measurement for snapshot matrix $\boldsymbol{X}$



Simulation with reduced order model

Full system: $2.87 \cdot 10^{-3}$ s per time step
Reduced system: $1.65 \cdot 10^{-4}$ s per time step

Exercises

- E.24 Reduced Order Models with Finite Elements (C)
  - Modify a finite element code to incorporate a reduced order model. After collecting a snapshot matrix and performing a singular value decomposition, the reduced basis is to be applied within the system of equations.

6.3 SINDy

Discovering governing equations from data: Sparse identification of nonlinear dynamical systems, Brunton et al. 2015

SINDy: Sparse Identification of Non-Linear Dynamical Systems

The goal of SINDy is the discovery of sparse dynamical system models described by systems of differential equations of the form

$$\frac{d}{dt}\boldsymbol{x}(t) = f(\boldsymbol{x}(t))$$

Where $\boldsymbol{x}(t) = \left(x_1(t), x_2(t), \dots, x_m(t)\right)^T$

6.3 SINDy

1. Sample temporal snapshots of $\boldsymbol{x}(t)$ and its first temporal derivative $\dot{\boldsymbol{x}}(t)$ in snapshot matrices $\boldsymbol{X}, \dot{\boldsymbol{X}}$

$$\boldsymbol{X} = \begin{pmatrix} - & \boldsymbol{x}(t_1) & - \\ - & \boldsymbol{x}(t_2) & - \\ & \vdots & \\ - & \boldsymbol{x}(t_n) & - \end{pmatrix}, \dot{\boldsymbol{X}} = \begin{pmatrix} - & \dot{\boldsymbol{x}}(t_1) & - \\ - & \dot{\boldsymbol{x}}(t_2) & - \\ & \vdots & \\ - & \dot{\boldsymbol{x}}(t_n) & - \end{pmatrix}$$

If the temporal derivatives $\dot{\boldsymbol{x}}(t)$ are unavailable, numerical schemes can be used

2. Select a library of candidate functions Θ for the sparse regression

$$\boldsymbol{\Theta} = \begin{pmatrix} 1 & \boldsymbol{x}_1(t_1) & \boldsymbol{x}_2(t_1) & \cdots & \boldsymbol{x}_1(t_1)\boldsymbol{x}_2(t_1) & \cdots & \boldsymbol{x}_1(t_1)^2 & \boldsymbol{x}_2(t_1)^2 & \cdots \\ 1 & \boldsymbol{x}_1(t_2) & \boldsymbol{x}_2(t_2) & \cdots & \boldsymbol{x}_1(t_2)\boldsymbol{x}_2(t_2) & \cdots & \boldsymbol{x}_1(t_2)^2 & \boldsymbol{x}_2(t_2)^2 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & \boldsymbol{x}_1(t_n) & \boldsymbol{x}_2(t_n) & \cdots & \boldsymbol{x}_1(t_n)\boldsymbol{x}_2(t_n) & \cdots & \boldsymbol{x}_1(t_n)^2 & \boldsymbol{x}_2(t_n)^2 & \cdots \end{pmatrix}$$

3. The sparse regression problem with sparse regression coefficients $\boldsymbol{\Xi}$

$$\dot{\boldsymbol{X}} = \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\Xi}$$

4. Compute the sparse regression coefficients Ξ with

- Sequential thresholded least squares
- (or Lasso) Least squares with a L^1 penalty to promote sparsity

6.3 SINDy – Sequential Thresholded Least Squares

The regression is solved iteratively with least squares. In each iteration the coefficients smaller than a threshold t are set to zero. This results in a sparse coefficient matrix

Algorithm 12 Sequential thresholded least squares algorithm

Require: snapshot matrix of first time derivatives $\dot{X}$, library evaluated with snapshots $\Theta(X)$, number of iteration k, sparse tolerance t

Initial least squares regression
$\Xi = (\Theta^\intercal\Theta)^{-1}\Theta^\intercal\dot{X}$

for all k **do**
Identify indices i of sparse terms
$i_{\text{sparse}} = |\Xi| < t$
$\Xi i = 0$

Perform the least squares regression sequentially for the number of degrees of freedom m on the non-sparse terms.
$j = 1$
for all m **do**
$i_{\text{nonsparse}} = \neg i_{\text{sparse}}[:, j]$
$\Xi i_{\text{nonsparse}}, j = (\Theta i_{\text{nonsparse}}, :^\intercal \Theta i_{\text{nonsparse}}, :))^{-1}\Theta i_{\text{nonsparse}}, :^\intercal \dot{X}[:, j]$
$j = j + 1$
end for

end for

6.3.1 Learning a Differential Equation

A system of differential equations is given as

$$\dot{x}_1 = -x_1 + 2x_2$$
$$\dot{x}_2 = -x_1 + x_2$$

With the initial conditions

$$x_1(0) = 1$$
$$x_2(0) = 1$$

And with the solution

$$x_1(t) = \sin(t) + \cos(t)$$
$$x_2(t) = \cos(t)$$

The goal of SINDy is to find the underlying differential equation with snapshots of the solution $x_1(t), x_2(t)$

6.3.1 Learning a Differential Equation

$$\boldsymbol{x}(t) = (\sin(t) + \cos(t), \cos(t))^T$$

Snapshots collected at $\boldsymbol{t} = [0, \frac{\pi}{2}, \pi, \frac{3}{4\pi}, 2\pi]$

$$\boldsymbol{X} = \begin{pmatrix} x_1(t_1) & x_2(t_1) \\ x_1(t_2) & x_2(t_2) \\ x_1(t_3) & x_2(t_3) \\ x_1(t_4) & x_2(t_4) \\ x_1(t_5) & x_2(t_5) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ -1 & -1 \\ -1 & 0 \\ 1 & 1 \end{pmatrix}, \dot{\boldsymbol{X}} = \begin{pmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) \\ \dot{x}_1(t_3) & \dot{x}_2(t_3) \\ \dot{x}_1(t_4) & \dot{x}_2(t_4) \\ \dot{x}_1(t_5) & \dot{x}_2(t_5) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & -1 \\ -1 & 0 \\ 1 & 1 \\ 1 & 0 \end{pmatrix}$$

Selected candidate functions and the corresponding matrix

$$1, x_1, x_2, x_1^2, x_2^2, x_1 x_2$$

$$\boldsymbol{\Theta} = \begin{pmatrix} 1 & x_1(t_1) & x_2(t_1) & x_1(t_1)^2 & x_2(t_1)^2 & x_1(t_1)x_2(t_1) \\ 1 & x_1(t_2) & x_2(t_2) & x_1(t_2)^2 & x_2(t_2)^2 & x_1(t_2)x_2(t_2) \\ 1 & x_1(t_3) & x_2(t_3) & x_1(t_3)^2 & x_2(t_3)^2 & x_1(t_3)x_2(t_3) \\ 1 & x_1(t_4) & x_2(t_4) & x_1(t_4)^2 & x_2(t_4)^2 & x_1(t_4)x_2(t_4) \\ 1 & x_1(t_5) & x_2(t_5) & x_1(t_5)^2 & x_2(t_5)^2 & x_1(t_5)x_2(t_5) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

6.3.1 Learning a Differential Equation

$$\mathbf{\Theta} = \begin{pmatrix} 1 & x_1(t_1) & x_2(t_1) & x_1(t_1)^2 & x_2(t_1)^2 & x_1(t_1)x_2(t_1) \\ 1 & x_1(t_2) & x_2(t_2) & x_1(t_2)^2 & x_2(t_2)^2 & x_1(t_2)x_2(t_2) \\ 1 & x_1(t_3) & x_2(t_3) & x_1(t_3)^2 & x_2(t_3)^2 & x_1(t_3)x_2(t_3) \\ 1 & x_1(t_4) & x_2(t_4) & x_1(t_4)^2 & x_2(t_4)^2 & x_1(t_4)x_2(t_4) \\ 1 & x_1(t_5) & x_2(t_5) & x_1(t_5)^2 & x_2(t_5)^2 & x_1(t_5)x_2(t_5) \end{pmatrix}$$

Sparse regression problem $\dot{\boldsymbol{X}} = \mathbf{\Theta}(\mathbf{X})\mathbf{\Xi}$ solved with sequential thresholded least squares with $t = 0.01$

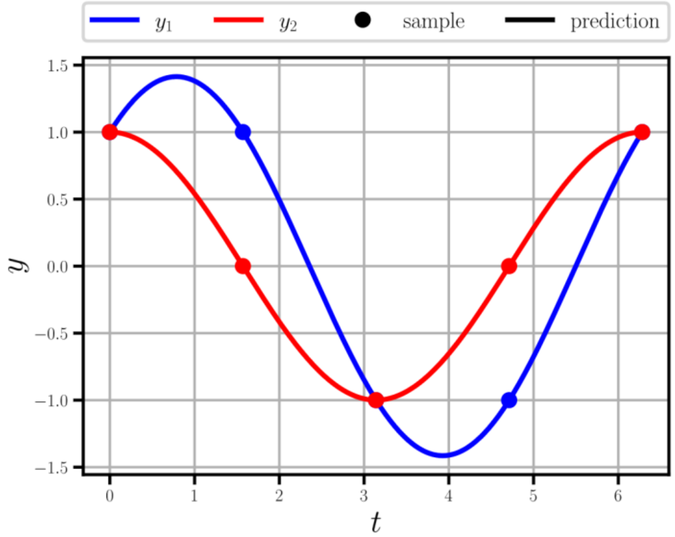
Initial least squares regression

$$\mathbf{\Xi} = (\mathbf{\Theta}^T\mathbf{\Theta})^{-1}\mathbf{\Theta}^T\dot{\boldsymbol{X}} = \begin{pmatrix} -5.98e-15 & -3.93e-15 \\ -1 & -1 \\ 2 & 1 \\ 6.56e-15 & 3.84e-15 \\ -2.35e-15 & -1.06e-15 \\ 2.19e-15 & 1.81e-15 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 \\ -1 & -1 \\ 2 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

6.3.1 Learning a Differential Equation

Recovered system

	$\dot{x}_1 =$	$\dot{x}_2 =$
1	0	0
x_1	-1	-1
x_2	2	1
x_1^2	0	0
x_2^2	0	0
$x_1 x_2$	0	0

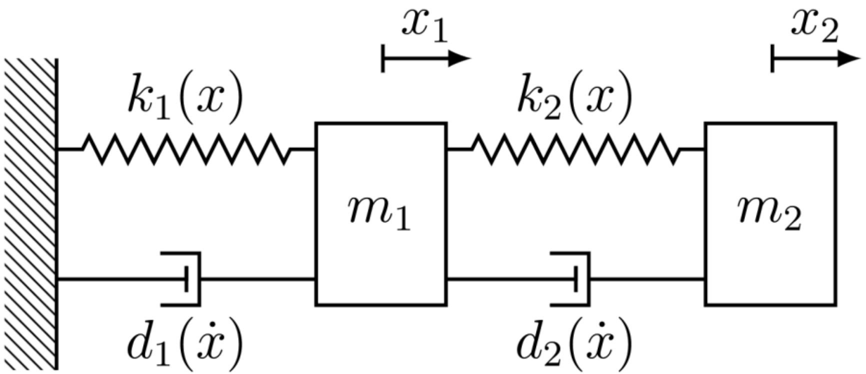


Original system

$$\dot{x}_1 = -x_1 + 2x_2$$
$$\dot{x}_2 = -x_1 + x_2$$

6.3.2 Structural Dynamics Model

A two degree of freedom mass-spring-damper model



Governing equations obtained from equilibrium

$$m_1\ddot{x}_1 + k_1(x_1) + d_1(\dot{x}_1) + k_2(x_1 - x_2) + d_2(\dot{x}_1 - \dot{x}_2) = 0$$
$$m_2\ddot{x}_2 + k_2(x_2 - x_1) + d_2(\dot{x}_2 - \dot{x}_1) = 0$$

And initial conditions

$$x_1(0) = x_{1_0}, \dot{x}_1(0) = \dot{x}_{1_0}, x_2(0) = x_{2_0}, \dot{x}_2(0) = \dot{x}_{2_0}$$

6.3.2 Structural Dynamics Model

Governing equations obtained from equilibrium

$$m_1\ddot{x}_1 + k_1(x_1) + d_1(\dot{x}_1) + k_2(x_1 - x_2) + d_2(\dot{x}_1 - \dot{x}_2) = 0$$
$$m_2\ddot{x}_2 + k_2(x_2 - x_1) + d_2(\dot{x}_2 - \dot{x}_1) = 0$$

Substitutiono to transform the system of second order differential equations into a system of first order differential equations

$$y_1 = x_1, y_2 = \dot{x}_1, y_3 = x_2, y_4 = \dot{x}_2$$

Any higher order system of differential equations can be rewritten in terms of a system of first order differential equations

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = \frac{-k_1(y_1) - d_1(y_2) - k_2(y_1 - y_3) - d_2(y_2 - y_4)}{m_1}$$
$$\dot{y}_3 = y_4$$
$$\dot{y}_4 = \frac{-k_2(y_3 - y_1) - d_2(y_4 - y_2)}{m_2}$$

6.3.2 Structural Dynamics Model – Linear

Assuming linear springs and dampers

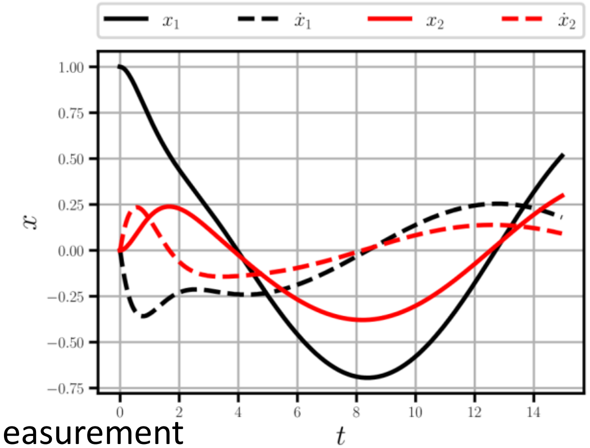
$$k_i(x) = K_i x$$
$$d_i(\dot{x}) = D_i \dot{x}$$

Example values $K_1 = 2, K_2 = 4, D_1 = 2, D_2 = 3, m_1 = 5, m_2 = 2$

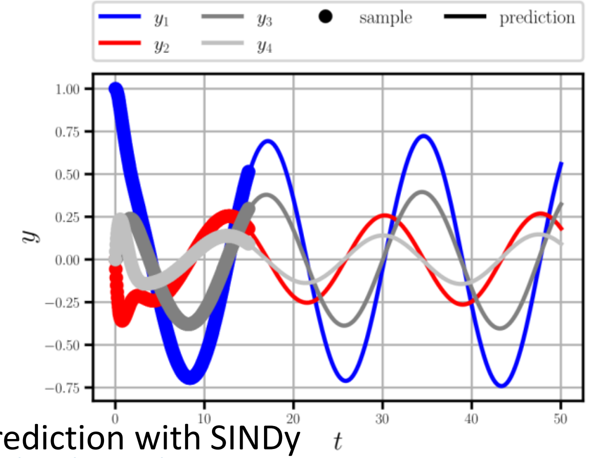
SINDy estimation with 300 snapshots and the original system

	$\dot{y}_1 =$	$\dot{y}_2 =$	$\dot{y}_3 =$	$\dot{y}_4 =$
1	0	0	0	0
y_1	0	-1.1340	0	0.9341
y_2	0.9999	-0.9677	0	0.9628
y_3	0	1.8778	0	-1.8779
y_4	0	1.4562	0.9997	-1.4564
y_1^2	0	0	0	0
y_2^2	0	0	0	0
y_3^2	0	0	0	0
y_4^2	0	0	0	0
y_1y_2	0	0	0	0
y_1y_3	0	0	0	0
y_1y_4	0	0	0	0
y_2y_3	0	0	0	0
y_2y_4	0	0	0	0
y_3y_4	0	0	0	0

	$\dot{y}_1 =$	$\dot{y}_2 =$	$\dot{y}_3 =$	$\dot{y}_4 =$
1	0	0	0	0
y_1	0	-1.2	0	1
y_2	1	-1	0	1
y_3	0	2	0	-2
y_4	0	1.5	1	-1.5
y_1^2	0	0	0	0
y_2^2	0	0	0	0
y_3^2	0	0	0	0
y_4^2	0	0	0	0
y_1y_2	0	0	0	0
y_1y_3	0	0	0	0
y_1y_4	0	0	0	0
y_2y_3	0	0	0	0
y_2y_4	0	0	0	0
y_3y_4	0	0	0	0



Measurement



Prediction with SINDy

6.3.2 Structural Dynamics Model – Linear

SINDy is made for nonlinear dynamics

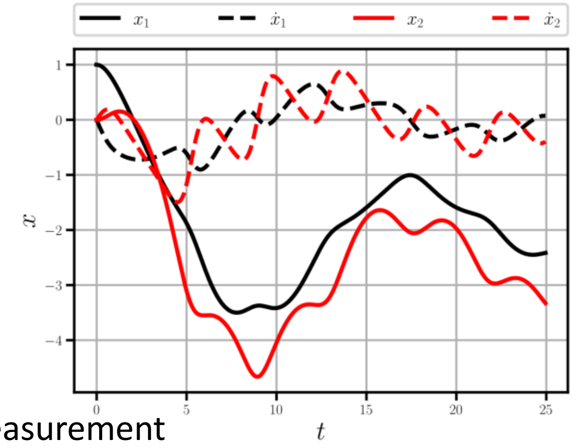
Nonlinear springs and dampers

$$k_1(x) = x, k_2(x) = 2x^3$$
$$d_1(\dot{x}) = \dot{x}, d_2(\dot{x}) = 1$$
$$m_1 = 5, m_2 = 2$$

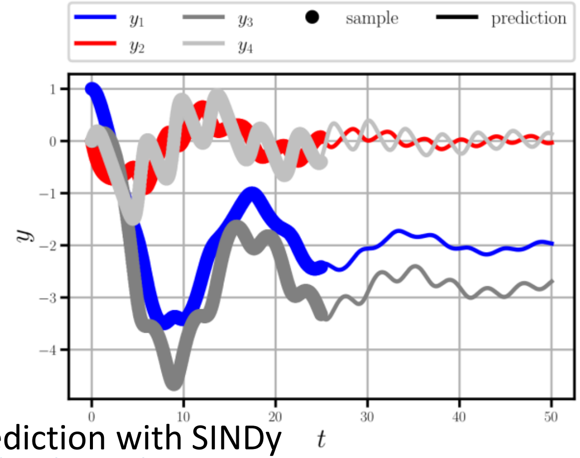
SINDy estimation with 500 snapshots and the original system

	$\dot{y}_1 =$	$\dot{y}_2 =$	$\dot{y}_3 =$	$\dot{y}_4 =$
1	0	-0.2029	0	-0.4986
y_1	0	-0.2006	0	0
y_2	0.9998	-0.2119	0	0
y_3	0	0	0	0
y_4	0	0	0.9998	0
y_1^2	0	0	0	0
y_3^2	0	0	0	0
y_1y_3	0	0	0	0
y_1^3	0	-0.3844	0	0.9756
y_3^3	0	0.3935	0	-0.9915
$y_1^2y_3$	0	1.1652	0	-2.9479
$y_1y_3^2$	0	-1.1741	0	2.9634

	$\dot{y}_1 =$	$\dot{y}_2 =$	$\dot{y}_3 =$	$\dot{y}_4 =$
1	0	-0.2	0	-0.5
y_1	0	-0.2	0	0
y_2	1	-0.2	0	0
y_3	0	0	0	0
y_4	0	0	1	0
y_1^2	0	0	0	0
y_3^2	0	0	0	0
y_1y_3	0	0	0	0
y_1^3	0	-0.4	0	1
y_3^3	0	0.4	0	-1
$y_1^2y_3$	0	1.2	0	-3
$y_1y_3^2$	0	-1.2	0	3



Measurement

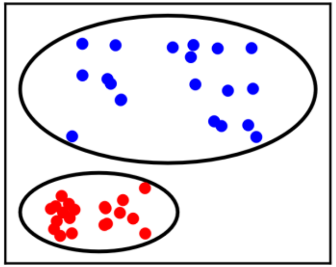


Prediction with SINDy

Exercises

- E.25 Introduction to SINDy (P & C)
  - Apply SINDy to a simple system of first order differential equations using pen-and-paper. Next implement SINDy and the sequential thresholded least sqaures algorithm and recompute the pen-and-paper computations.
- E.26 SINDy for Structural Dynamics (P & C)
  - Apply the implementation of SINDy to a two-degree-of-freedom example from structural dynamics.

6.4 Clustering



Clustering: Discovers similarities between data and creates discrete clusters (unsupervised)

One approach is k-means clustering

- Subdivision of dataset into k clusters
- Each datapoint $\boldsymbol{x}$ belongs to cluster with the closest mean $\boldsymbol{\mu}$
- Framed as minimization problem

$$\arg\min_{\boldsymbol{\mu}_j} \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i | y(\boldsymbol{x}_i = y_j)} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_j \right\|_2^2$$

- Where $\boldsymbol{\mu}_j$ is the mean of the j^{th} cluster consisting of datapoints associated with the cluster, i.e., $y_j = y(\boldsymbol{x}_i)$
- NP-hard problem, solved by heuristic algorithms, e.g., Lloyds algorithm

1. Computation of distances d_{ij}, used to identify closes cluster y_i by taking the minimum distance $\min_j d_{ij}$
2. Recomputation of cluster centers $\boldsymbol{\mu}_j$ with the mean of the associated data points $\boldsymbol{x}_i \in D_j$

Repeat until convergence

6.4 Clustering – Lloyds algorithm

Algorithm 13 Lloyds algorithm for k-means clustering [245]

Require: data $\boldsymbol{x}_i$, number of iterations, number of clusters

Initialize cluster centers $\boldsymbol{\mu}_j$
for all iterations **do**
  for all samples $\boldsymbol{x}_i$ **do**
    for all cluster centers $\boldsymbol{\mu}_j$ **do**
      Compute Euclidean distances $d_{ij} = ||\boldsymbol{\mu}_j - \boldsymbol{x}_i||_2^2$
    end for
    Identify associated cluster $y_i = \min_j d_{ij}$
  end for
  for all cluster centers $\boldsymbol{\mu}_j$ **do**
    Update cluster centers $\boldsymbol{\mu}_j = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}_i$
  end for
end for

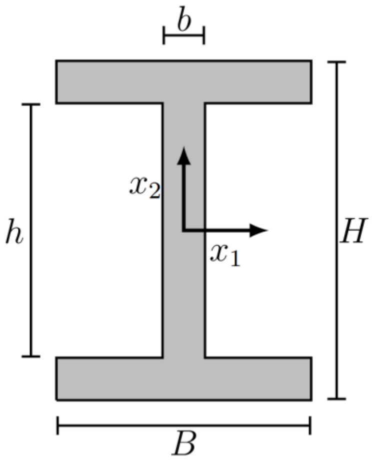
6.4.1 Cross-Section Clustering

Consider the cross-section of an I-beam with the properties

$$I_1 = \frac{bh^3}{12} + \frac{B}{12}(H^3 - h^3)$$

$$I_2 = \frac{b^3 h}{12} + \frac{B^3}{12}(H - h)$$

$$A = HB - h(B - b)$$



Assuming three different cross-sections (with different distributions)

	H	h	B	b
cross-section 1	$\mathcal{U}(1,2)$	$\mathcal{U}(0.5,H)$	$\mathcal{U}(2,4)$	$\mathcal{U}(1,B)$
cross-section 2	$\mathcal{U}(4,5)$	$\mathcal{U}(0.5,H)$	$\mathcal{U}(2,3)$	$\mathcal{U}(0.5,b)$
cross-section 3	$\mathcal{U}(3.5,3.5)$	$\mathcal{U}(3,H)$	$\mathcal{U}(3.5,4.5)$	$\mathcal{U}(3,B)$

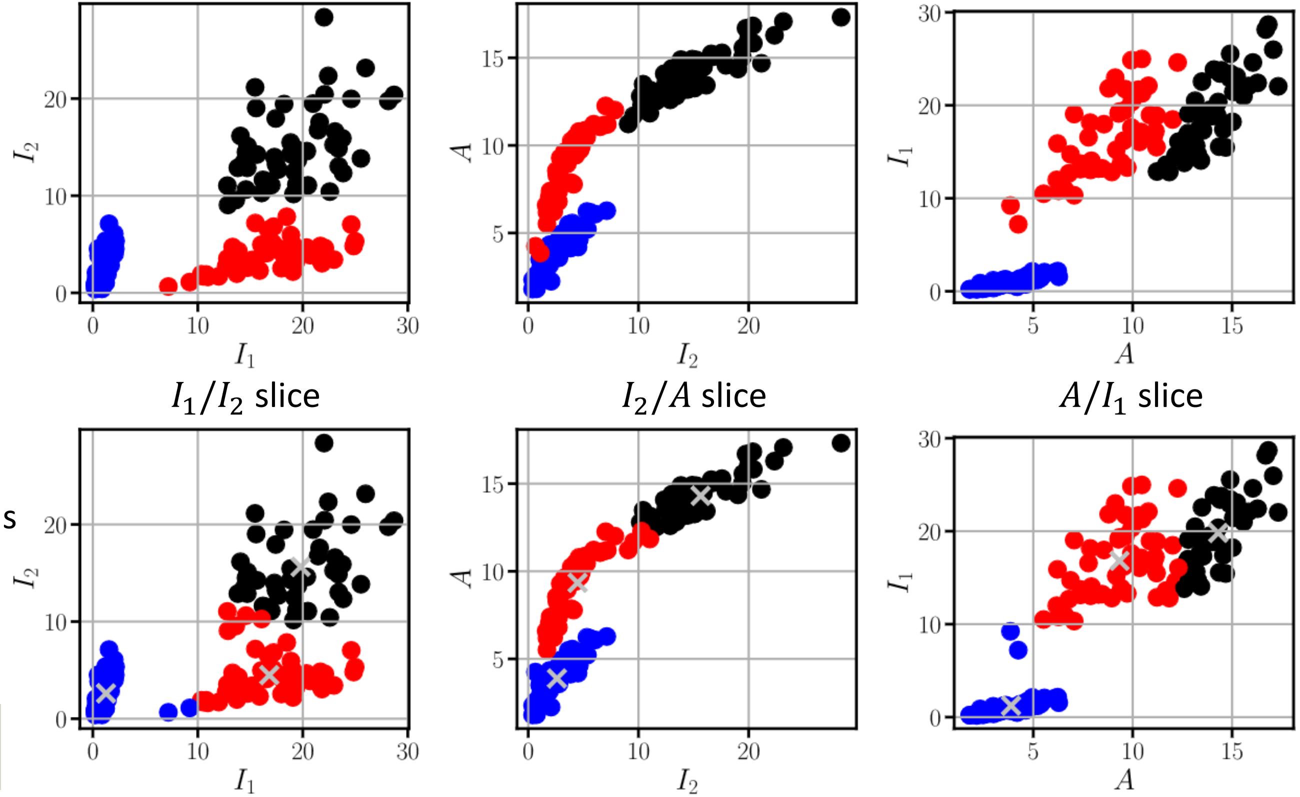
Can we identify the three different cross-sections from the three properties, I_1, I_2, A?

6.4.1 Cross-Section Clustering

k-means clustering with Lloyds algorithm

Original cross-sections

Identified cross-sections with clustering

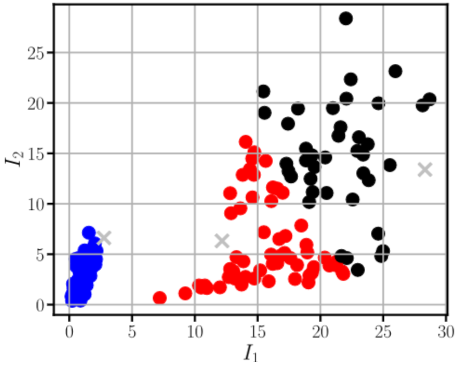


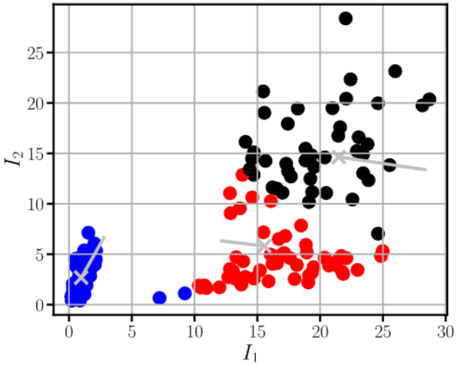
Cluster centers are indicated by silver centers

6.4.1 Cross-Section Clustering

Evolution of cluster identification in I_1/I_2-slice (initial guess + three iterations

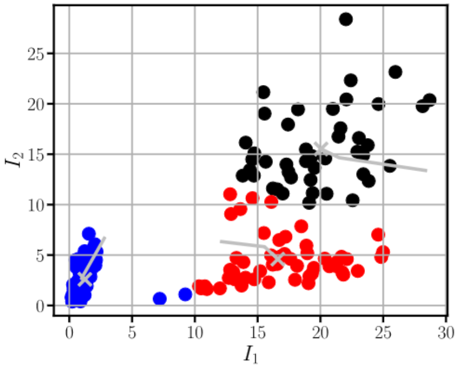
Iteration 0: 86%

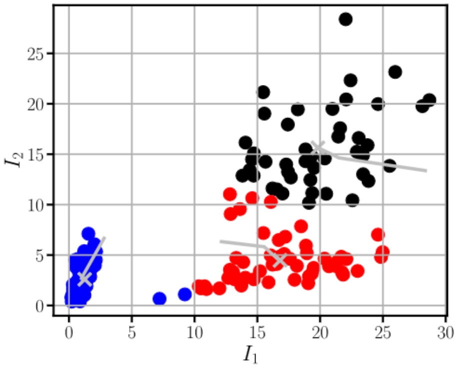




Iteration 1: 94%

Iteration 2: 95%

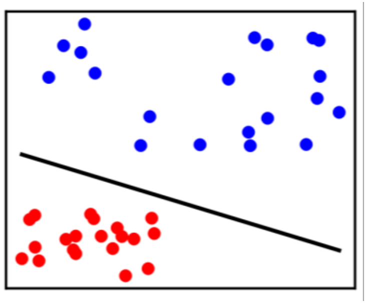




Iteration 3: 95%

6.5 Support Vector Machines

Classification: Prediction of a discrete category via a mapping between input and a (discrete) category



Support vector machines as classification algorithm

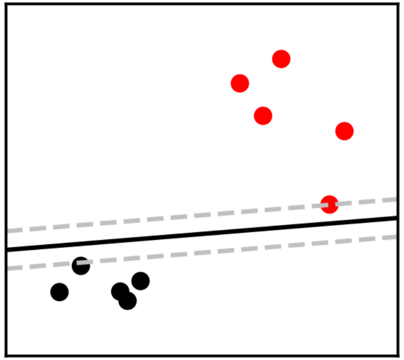
- Binary classifier, i.e., (-1 and 1)
- Predecessor of neural networks

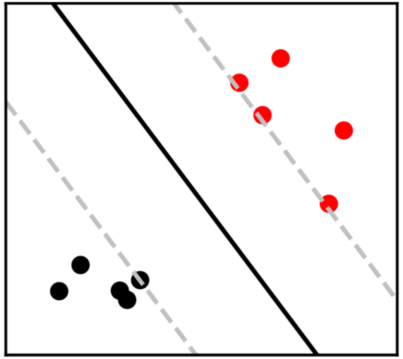
- Linear support vector machines construct a hyperplane between two categories

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$

- Hyperplane should maximize margin between closest points to the hyperplane

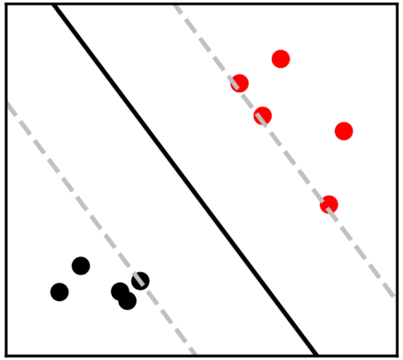
Minimal margin





Maximal margin

6.5 Support Vector Machines



Prediction $\hat{y}$ is made depending on what side of the plane the datapoint $\boldsymbol{x}$ lies

$$\hat{y} = \operatorname{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

Optimization is formulated

- 1. Correct classification of data

Mislabeling a datapoint increases the loss by one

$$\mathcal{L}(\tilde{y}_i, \hat{y}(\tilde{\boldsymbol{x}}_i)) = \begin{cases} 0 \text{ if } \tilde{y}_i = \operatorname{sign}(\boldsymbol{w} \cdot \tilde{\boldsymbol{x}}_i + b) \\ 1 \text{ if } \tilde{y}_i \neq \operatorname{sign}(\boldsymbol{w} \cdot \tilde{\boldsymbol{x}}_i + b) \end{cases}$$

  - With the constraint

$$\min_i |\tilde{\boldsymbol{x}}_i \cdot \boldsymbol{w}| = 1$$

Ensures that closest data points (support vectors) are at a standardized distance

- 2. Maximization of the margin

$$C = \sum_{i=1}^{m} \mathcal{L}(\tilde{y}_i, \hat{y}(\tilde{\boldsymbol{x}}_i)) + \frac{1}{2} \|\boldsymbol{w}\|^2$$

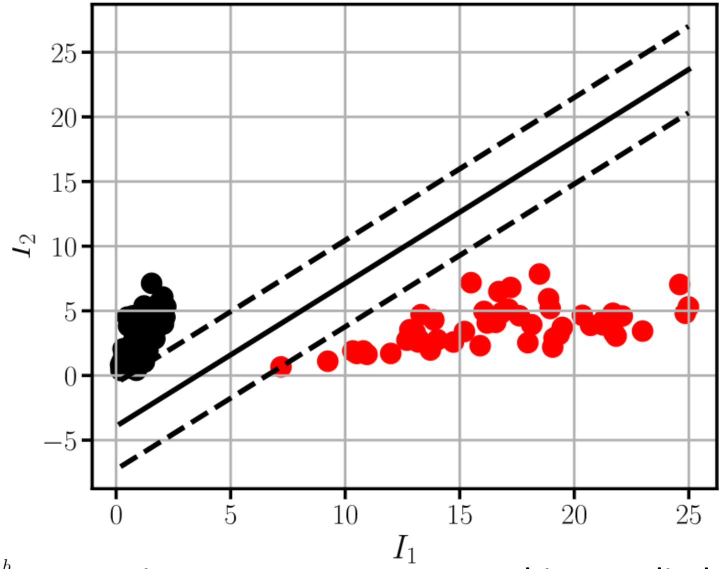
Achieved by the regularization term

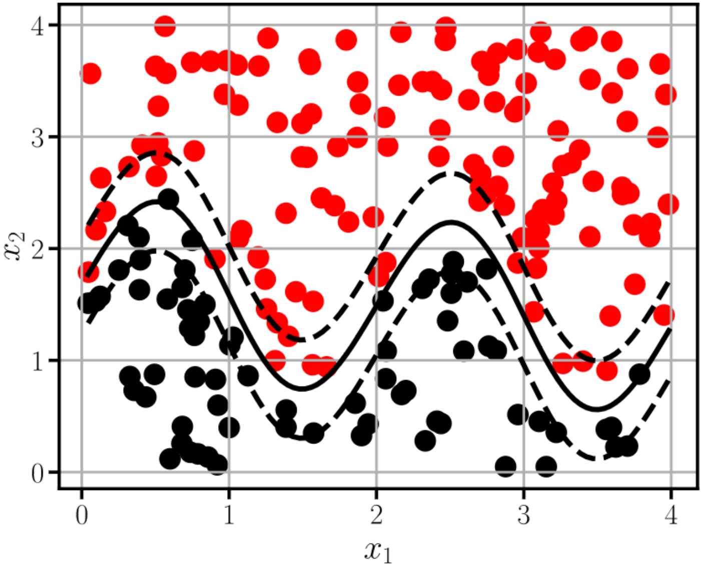
Loss $\mathcal{L}(\tilde{y}_i, \hat{y}(\tilde{\boldsymbol{x}}_i))$ is not differentiable and therefore exchanged with the differentiable Hinge loss function

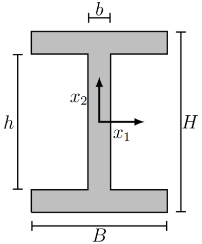
$$\mathcal{L}_H(\tilde{y}_i, \hat{y}(\tilde{x}_i)) = \max(0, 1 - \tilde{y}_i \hat{y}_i)$$

Optimization via gradient-based optimization determines optimal hyperplane

6.5 Support Vector Machines – Examples







Linear support vector machine applied to the cross-section example

Nonlinear support vector machine enabled by projection to higher-dimensional space

$$e(x_1, x_2) = \left(x_1, x_2, \sin(\pi x_1), \sin(2\pi x_1)\right)^T$$

Contents

- 5 Advanced Physics-Informed Neural Networks
- 6.1.1 Singular Value Decomposition
- 6.2 Reduced Order Models
- 6.3 Sparse Identification of Non-Linear Dynamical Systems – SINDy
- 6.4 Clustering
- 6.5 Support Vector Machines
- 7 Material Modeling with Neural Networks

6 Machine Learning in Computational Mechanics

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book