Supervised machine learning with tensor-based kernel machines

Kim Batselier, Afra Kilic, Eva Memmel,
Clara Menzen, Albert Saiapin, Frederiek Wesel

TUDelft

Setting the stage...

Parametric supervised learning problem:

Given a set of measurements $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ and a parameterization $\boldsymbol{w}$ of $\hat{y}_n(\cdot)$ in

$$y_n = \hat{y}_n(\boldsymbol{x}_n | \boldsymbol{w}) + e_n,$$

determine $\boldsymbol{w}$.

Kernel machines

$$\hat{y}_n(\boldsymbol{x}_n|\boldsymbol{w}) = \varphi(\boldsymbol{x}_n)^T\ \boldsymbol{w}$$

$\varphi(\cdot)$: collection of basis functions

LTI systems

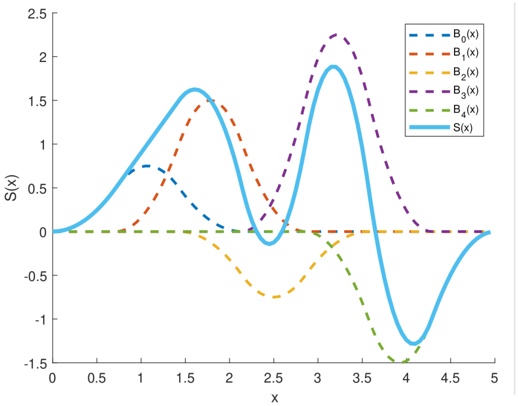
$$\text{OE:} \quad \hat{y}_n = \begin{pmatrix} u_n & \cdots & \hat{y}_{n-1} & \cdots \end{pmatrix} \boldsymbol{w}$$

$$\text{ARMAX/ Box Jenkins:} \quad \hat{y}_n = \begin{pmatrix} u_n & \cdots & y_{n-1} & \cdots & \hat{y}_{n-1} & \cdots \end{pmatrix} \boldsymbol{w}$$

$\Rightarrow$ Learn $\boldsymbol{w}$ from data = solving multivariate polynomial system

Pick your favorite basis function

$$\hat{y}(x) = w_0\ B_0(x) + w_1\ B_1(x) + w_2\ B_2(x) + w_3\ B_3(x) + \cdots$$



TU Delft

A simple univariate example

$$\hat{y}(x) = w_0 + w_1\, x + w_2\, x^2$$

$$= \underbrace{\begin{pmatrix} 1 & x & x^2 \end{pmatrix}}_{\varphi(x)^T} \underbrace{\begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}}_{\boldsymbol{w}}$$

Another simple univariate example

Ground truth:

$$y = f(x) + e = 3.2\,x^2 - 9.6\,\cos\left(5.1\,x^3 - e^{-3\,x}\right) + e$$

Assume $x \in [-1, 1]$. How many basis functions I do we need?

$$f(x) = 3.2\,x^2 - 9.6\,\cos\left(5.1\,x^3 - e^{-3\,x}\right) \approx \sum_{i=1}^{I} \varphi_i(x)\,w_i$$

The total number of basis functions I we need depends on $\varphi_i(x)$ and $f(x)$.

Multivariate basis functions as products of univariate basis functions

$$\underbrace{\varphi(\boldsymbol{x})}_{I^D} = \underbrace{\varphi(x_1)}_{I} \otimes \underbrace{\varphi(x_2)}_{I} \otimes \cdots \otimes \underbrace{\varphi(x_D)}_{I}$$

Bivariate monomial basis functions from a Kronecker product

$$\varphi(\boldsymbol{x}) = \varphi(x_1) \otimes \varphi(x_2)$$

$$\begin{pmatrix} 1 \\ x_1 \\ x_1^2 \\ x_2 \\ x_1 x_2 \\ x_1^2 x_2 \\ x_2^2 \\ x_1 x_2^2 \\ x_1^2 x_2^2 \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \\ x_1^2 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ x_2 \\ x_2^2 \end{pmatrix}$$

Learning problem

$$\boldsymbol{y} = \boldsymbol{\Phi}\, \boldsymbol{w} + \boldsymbol{e}$$

$$\min_{\boldsymbol{w}} L(\boldsymbol{y}, \boldsymbol{\Phi}, \boldsymbol{w})$$

Possible loss functions $L(\boldsymbol{y}, \boldsymbol{\Phi}, \boldsymbol{w})$:

- logistic (logistic regression)
- hinge (support vector machines / classification)
- Vapnik ϵ-insensitive (support vector machines / regression)
- squared (least-squares support vector machines, Gaussian Processes)

Squared loss

$$\min_{\boldsymbol{w}} ||\boldsymbol{y} - \boldsymbol{\Phi}\ \boldsymbol{w}||_2^2$$

$$\boldsymbol{w} = (\underbrace{\boldsymbol{\Phi}^T \boldsymbol{\Phi}}_{I^D \times I^D})^{-1}\ \boldsymbol{\Phi}^T\ \boldsymbol{y}$$

Assumptions:

- $N \geq I^D$
- Persistency of excitation: $\operatorname{rank}(\boldsymbol{\Phi}) = I^D$

TU Delft

What if $\mathrm{rank}(\mathbf{\Phi}) < I^D$?

Kernel ridge regression

$$\min_{\boldsymbol{w}} ||\boldsymbol{y} - \mathbf{\Phi}\ \boldsymbol{w}||_2^2 + \lambda\ ||\boldsymbol{w}||_2^2$$

$$\boldsymbol{w} = (\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\ \boldsymbol{I})^{-1}\ \mathbf{\Phi}^T\ \boldsymbol{y}$$

A simple example

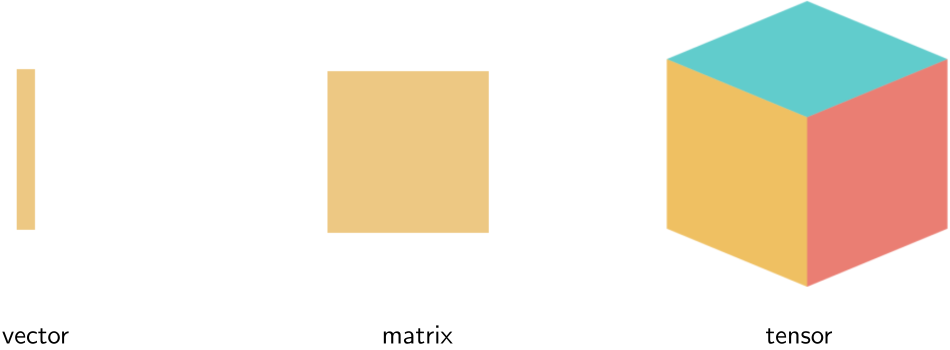
- $D = 18$ variables $x_1, \ldots, x_{18}$
- $I = 20$ basis functions
- $\boldsymbol{w}$ has $20^{18} = 2.6\ 10^{23}$ entries $\approx 10^{10}$ ChatGPTs

Tensor networks allow you to efficiently solve

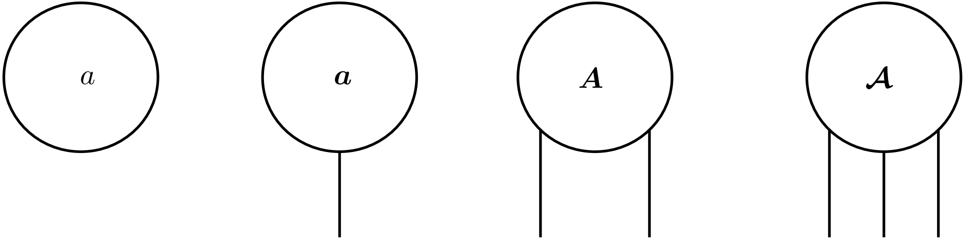
$$\left(\mathbf{\Phi}^T \mathbf{\Phi} + \boldsymbol{P}\right) \boldsymbol{w} = \mathbf{\Phi}^T \boldsymbol{y}$$

without explicitly making / inverting $(\mathbf{\Phi}^T \mathbf{\Phi} + \boldsymbol{P})$.

A tensor is a high-dimensional generalization of vectors and matrices

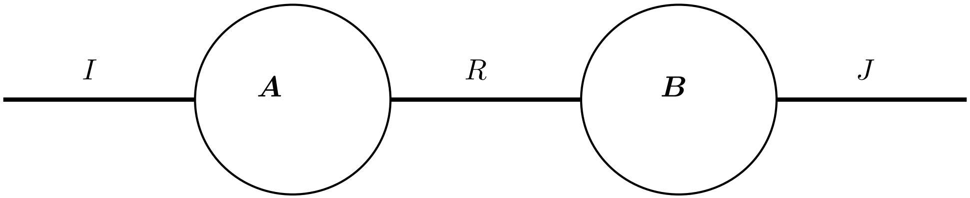


Tensor diagrams



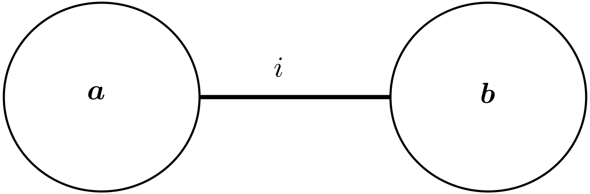
Index summation is visualized as a common edge

$$\boldsymbol{C}(i,j) = \sum_{r=1}^{R} \boldsymbol{A}(i,r)\ \boldsymbol{B}(r,j)$$



TUDelft

Brain teaser



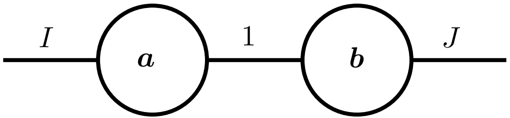
inner product: $\sum_{i=1}^{I} \boldsymbol{a}(i)\, \boldsymbol{b}(i) = \boldsymbol{a}^T \boldsymbol{b}$

An important operation: Outer product of vector $\boldsymbol{a}$ with vector $\boldsymbol{b}$

$$\boldsymbol{C} = \boldsymbol{a}\,\boldsymbol{b}^T = \boldsymbol{a} \circ \boldsymbol{b}$$

$$\boldsymbol{C}(i,j) = \boldsymbol{a}(i)\ \ \boldsymbol{b}(j)$$

$$\boldsymbol{C}(i,j) = \sum_{r=1}^{1} \boldsymbol{a}(i,r)\ \boldsymbol{b}(r,j)$$



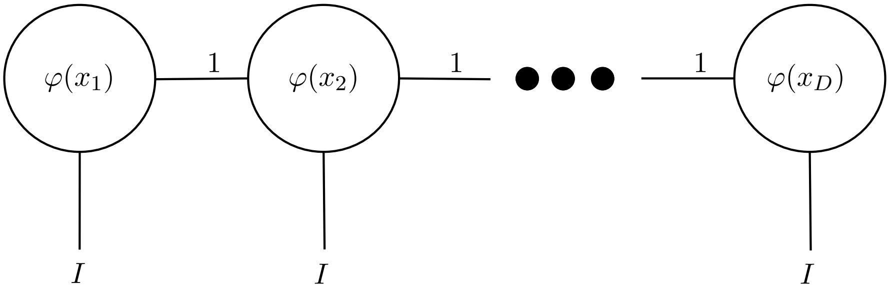
Key idea

Replacing $\mathbf{\Phi}, \boldsymbol{P}, \boldsymbol{w}$ by networks of much smaller tensors avoids explicit construction/inversion of $\mathbf{\Phi}, \boldsymbol{P}, \boldsymbol{w}$ and enables efficient learning.

TU Delft

Back to our kernel machines

$$\boldsymbol{\phi} = \varphi(x_1) \circ \varphi(x_2) \circ \cdots \circ \varphi(x_D)$$



$$I^D \rightarrow D\, I$$

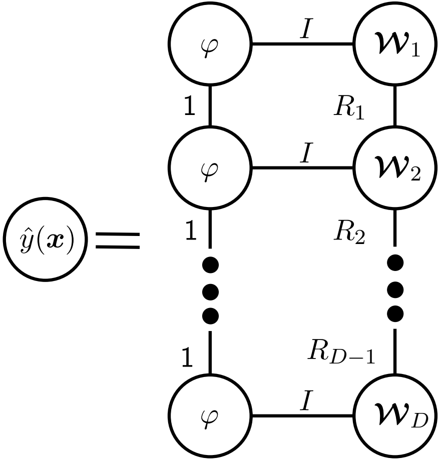
Revisit bivariate monomial basis functions as a tensor product

$$\boldsymbol{\phi} = \varphi(x_1) \circ \varphi(x_2)$$

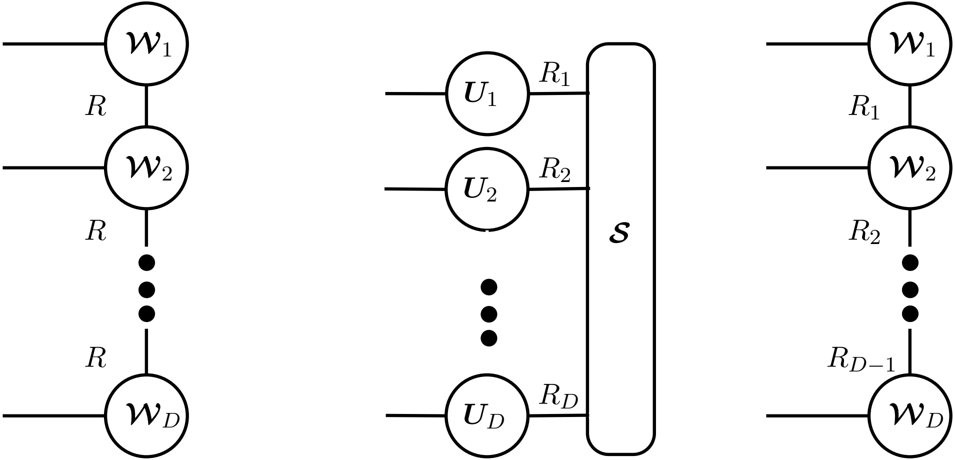
$$\begin{pmatrix} 1 & x_2 & x_2^2 \\ x_1 & x_1 x_2 & x_1 x_2^2 \\ x_1^2 & x_1^2 x_2 & x_1^2 x_2^2 \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \\ x_1^2 \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_2 \\ x_2^2 \end{pmatrix}$$

Tensor-based kernel machine

$$y = \langle \boldsymbol{\phi}, \boldsymbol{\mathcal{W}} \rangle + e$$



Possible tensor network topologies...



Polyadic
DIR

Tucker
$R^D + DIR$

Tensor Train
DIR^2

TU Delft

From linear least-squares to nonlinear least-squares

$$\min_{\mathcal{W}_1,\mathcal{W}_2,\ldots,\mathcal{W}_D} ||\boldsymbol{y}-\langle \boldsymbol{\phi},\ \mathrm{TN}(\mathcal{W}_1,\mathcal{W}_2,\ldots,\mathcal{W}_D)\rangle||_2^2\ +\ R(\mathrm{TN}(\mathcal{W}_1,\mathcal{W}_2,\ldots,\mathcal{W}_D))$$

Solving a polynomial system of equations!

Exploiting tensor network structure enables efficient learning

$\mathrm{TN}(\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_D)$ is a linear map in $\mathcal{W}_1$

$$\mathrm{TN}(\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_D) \rightarrow \mathrm{TN}_1(\mathcal{W}_1) = \boldsymbol{T}_1 \, \boldsymbol{w}_1$$

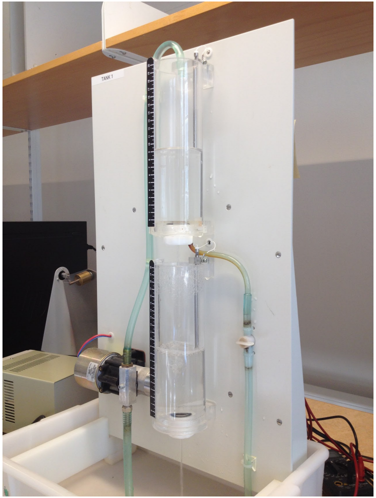
$$\min_{\boldsymbol{w}_1} ||\boldsymbol{y} - \boldsymbol{\Phi} \, \boldsymbol{T}_1 \, \boldsymbol{w}_1||_2^2 \, + \, \mathcal{R}(\boldsymbol{T}_1 \, \boldsymbol{w}_1)$$

Computational cost polyadic network: $O\left(N(IR)^2 + (IR)^3\right)$

Example: Cascaded tanks

- u_n : pump voltage
- y_n : water level lower tank
- Task: learn

$$\hat{y}_n\left(u_{n-1}, u_{n-2}, \ldots, y_{n-1}, y_{n-2}, \ldots\right)$$



Example: SUSY binary classification problem

$I = 20, D = 18, N = 5,000,000$

	AUC		1-Accuracy (%)	Time (s)
Technique	Low-level	Complete	Complete	Complete
BDT	0.850 ± 0.003	0.863 ± 0.003	NA	NA
NN	0.867 ± 0.002	0.875 ± 0.001	NA	NA
Dropout NN	0.856 ± 0.001	0.873 ± 0.001	NA	NA
DNN	0.872 ± 0.001	0.876 ± 0.001	NA	NA
Dropout DNN	$\mathbf{0.876 \pm 0.001}$	$\mathbf{0.879 \pm 0.001}$	NA	NA
VISH	0.859 ± 0.001	NA	NA	NA
CPD-SIP $(R = 5)$	0.860 ± 0.001	0.871 ± 0.001	20.14 ± 0.04	1416 ± 12
CPD-SIP $(R = 10)$	0.869 ± 0.001	0.872 ± 0.002	19.97 ± 0.10	3567 ± 1
CPD-SIP $(R = 15)$	0.871 ± 0.001	0.873 ± 0.001	19.86 ± 0.10	5628 ± 153
CPD-SIP $(R = 20)$	0.872 ± 0.001	0.875 ± 0.001	19.74 ± 0.04	9437 ± 721

References

- Batselier K., Low-rank tensor decompositions for nonlinear system identification, in IEEE Control Systems Magazine, vol. 42, no. 1, pp. 54-74, Feb. 2022
- Wesel F., Batselier K., Large-Scale Learning with Fourier Features and Tensor Decompositions, in Proc. of the 2021 Conference on Neural Information Processing Systems (NeuRIPS), December 2021
- Karagoz R., Batselier K., Nonlinear system identification with regularized Tensor Network B-splines, Automatica, vol. 122, 2020
- Wesel F., Batselier K., Tensor-based Kernel Machines with Structured Inducing Points for Large and High-Dimensional Data, in Proc. of The 26th International Conference on Artificial Intelligence and Statistics (AISTATS), April 2023
- Wesel F., Batselier K., Quantized Fourier and Polynomial Features for more Expressive Tensor Network Models, in Proc. of The 27th International Conference on Artificial Intelligence and Statistics (AISTATS), April 2024.
- Schuurmans J., Batselier K., Kooij J., How informative is the Approximation Error from Tensor Decomposition for Neural Network Compression?, in Proc. of The Eleventh International Conference on Learning Representations (ICLR), May 2023