Neural networks, flexible activation functions, and tensor approximations

joint work with Y. Qi, P. Comon, P. Dreesen, M. Ishteva, Y. Zniyed, S. Miron, D. Brie

Konstantin Usevich, CNRS (CRAN, Nancy)

23.11.2022, Workshop on Tensor Theory and Methods, Paris

Overview

Flexible activations

Link to CPD

Coupled matrix/tensor factorization

Low-rank tensor recovery

Overview

Flexible activations

Link to CPD

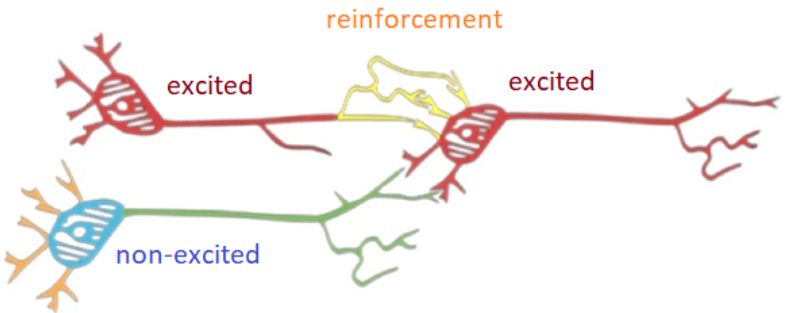
Coupled matrix/tensor factorization

Low-rank tensor recovery

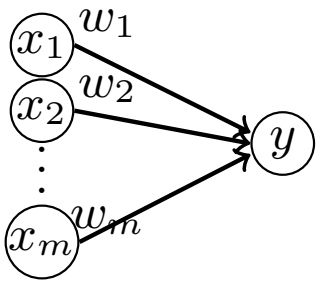
Feed-forward neural networks

- Huge success in computer vision [Krizhevsky et al., 2014]
- Very popular in data science, signal processing, engineering
- Basic problem supervised classification:
  Given a *training set* of objects $\mathbf{x}^{(k)} \in \mathbb{R}^m$ and (discrete) labels y_k,
  find ("learn") the map f such that $f(\mathbf{x}^{(k)}) \approx y_k$,
  which generalizes well: ($f(\mathbf{x}_{test}) \approx y_{test}$ on unseen *test* data)

Neural networks: early ideas



real neuron

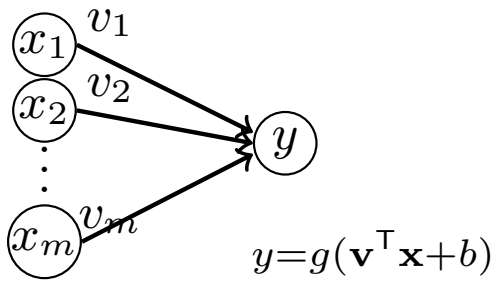


artificial neurons

$$y = \begin{cases} 1, & w_1 x_1 + \cdots + w_m x_m \geq \theta \quad \text{(a threshold)}, \\ 0, & \text{otherwise} \end{cases}$$

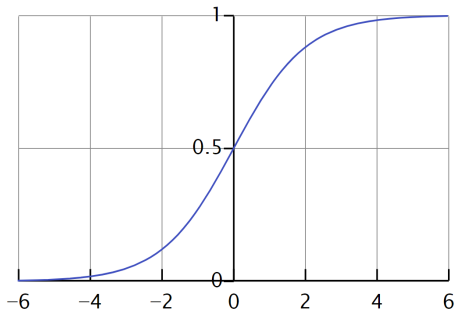
(McCulogh, Pitts, 1943), (Rosenblatt, 1958)

Perceptron and activation functions



- $\mathbf{v} \in \mathbb{R}^m$ — weights, b — "bias"
- $g(t)$ — activation ("thresholding") function

e.g., sigmoid



$$g(t) = \frac{1}{1+e^{-t}}$$

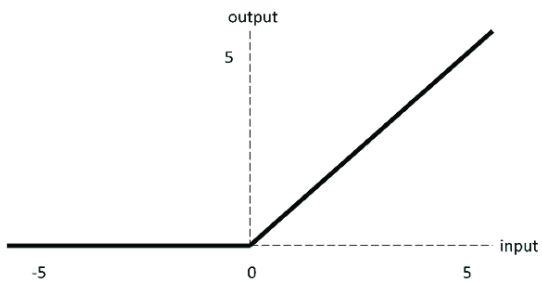
Main issues:

- level sets are hyperplanes
- classifier only for linearly separable classes

Modern feed-forward neural networks

Some key features:

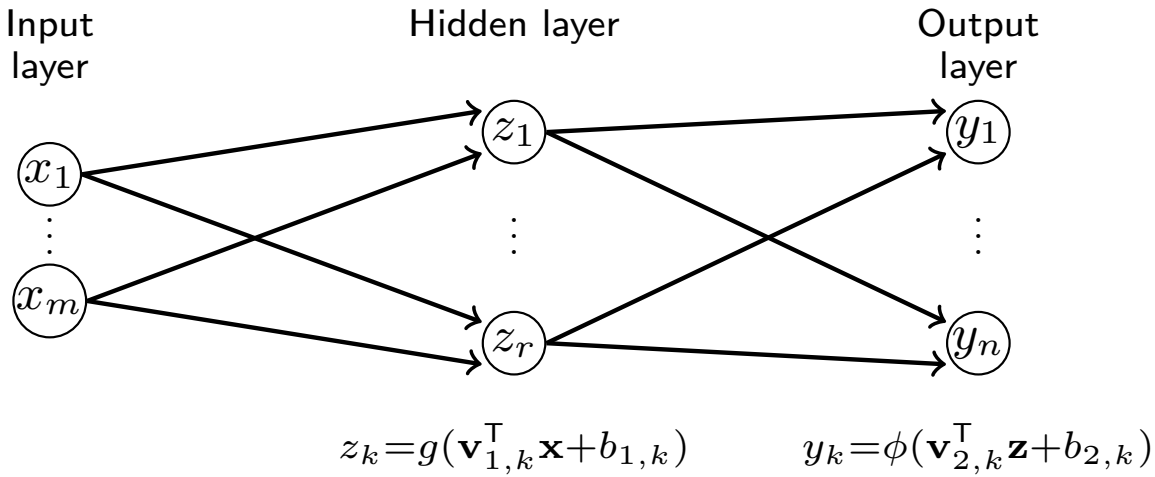
- Stacking several layers:
  lots of layers (deep learning) $\rightarrow$ better performance
- Shift to other activation functions, e.g. ReLu:



$$g(t) = \max(0, t)$$

- Structured weights (e.g., convolutional)
- Clever optimization algorithms, parallel computations on GPUs

Multilayer perceptron



- $\mathbf{v}_{1,k}$, $\mathbf{v}_{2,k}$ — vectors of weights, $b_{1,k}, b_{2,k}$ — biases
- g, ϕ — univariate activation functions

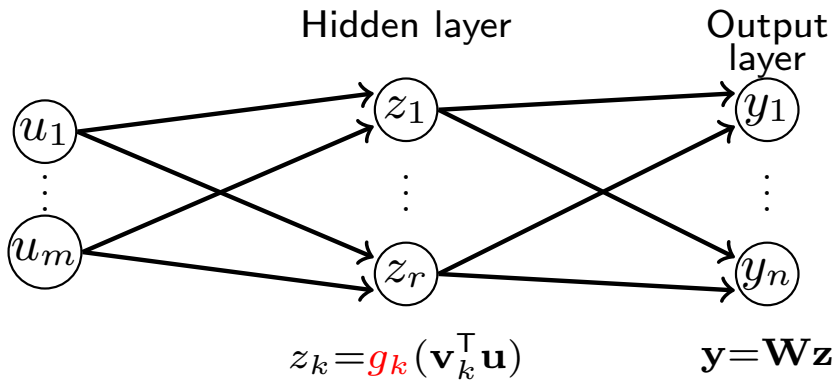
Theorem (Universal Approximation Theorem, e.g., (Pinkus, 1997))

Any continuous $\mathbf{f} : \Omega \to \mathbb{R}^n$ on compact Ω can be approximated to arbitrary accuracy by a multilayer perceptron $\mathbf{f}_\theta$ (with $\phi(t) = t$ and non-polynomial continuous g).

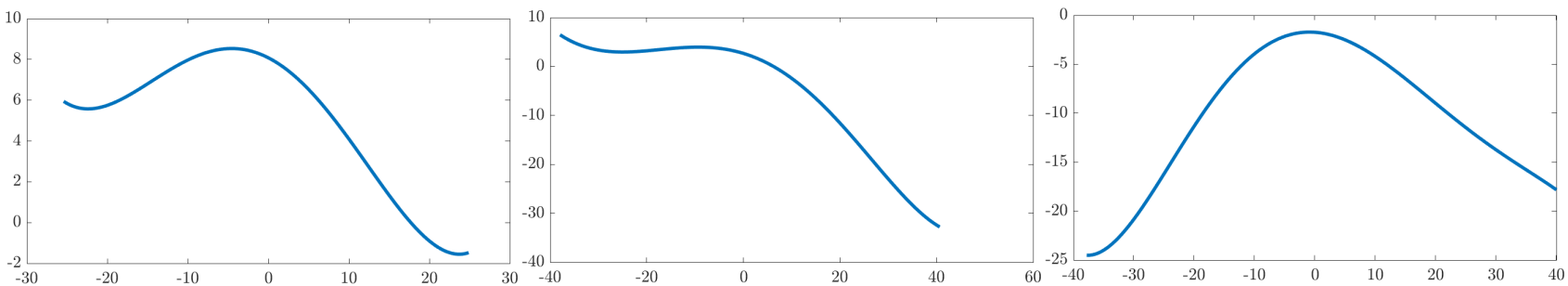
Can we add flexibility to g to achieve more compact representation?

From fixed to flexible activations

Simplest 2-layer model: no activation/bias at output



- fixed activations (classic setup): $g_k(t) = g(t + b_k)$
- flexible activations: different (ideally learned) g_k



Related work

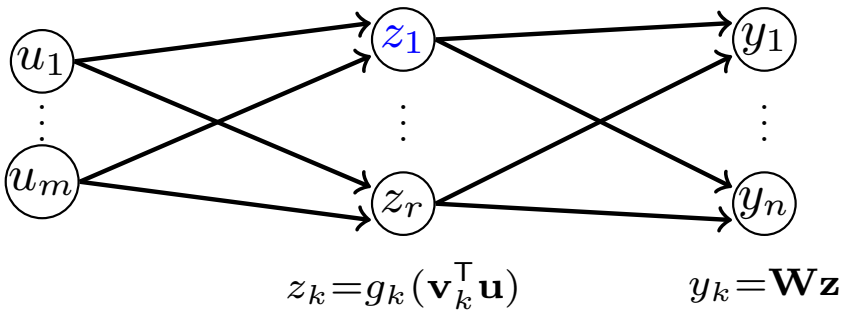
Neural network/approximation literature:

- deep spline networks [Aziznejad, Unser, 2019]
- ridge approximations [Lin, Pinkus, 1993]

Other work:

- independent component analysis [Comon, Jutten, 2010]
- block-structured nonlinear system identification [Dreesen et al., 2015]

Decomposition/approximation problem



Given a multivariate map $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n$, decompose (approximate) it as

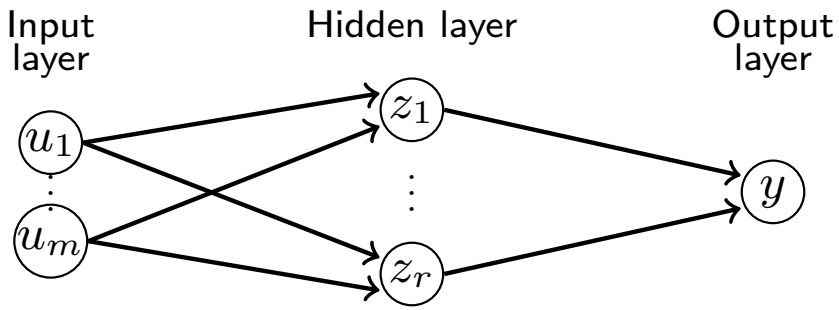
$$\mathbf{f}(\mathbf{u}) = \underbrace{\mathbf{w}_1 g_1(\mathbf{v}_1^\mathsf{T}\mathbf{u})}_{\text{single node (branch)}} + \cdots + \mathbf{w}_r g_r(\mathbf{v}_r^\mathsf{T}\mathbf{u}),$$

where

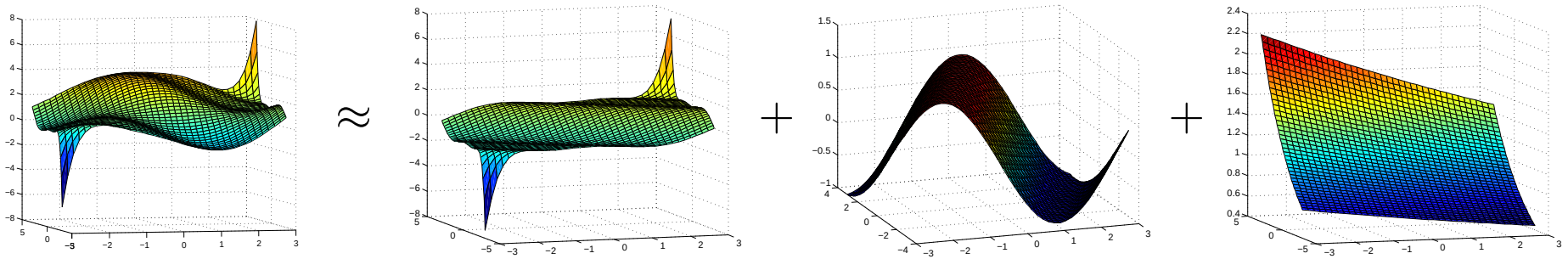
- $\mathbf{v}_k \in \mathbb{R}^m$, g_k — different univariate functions
- $\mathbf{W} = \begin{bmatrix}\mathbf{w}_1 & \cdots & \mathbf{w}_r\end{bmatrix} \in \mathbb{R}^{n\times r}$ — output weights

Geometric interpretation: ridge functions

Single output:



$$f(\mathbf{u}) = g_1(\mathbf{v}_1^\mathsf{T}\mathbf{u}) + \cdots + g_r(\mathbf{v}_r^\mathsf{T}\mathbf{u})$$

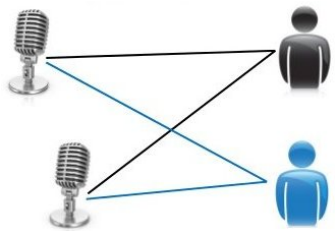


sum of "plane waves" or ridge functions *[Lin, Pinkus, 1993]*

Independent component analysis (source separation)

Linear mixing model:

$$\mathbf{x} = V\mathbf{s}$$



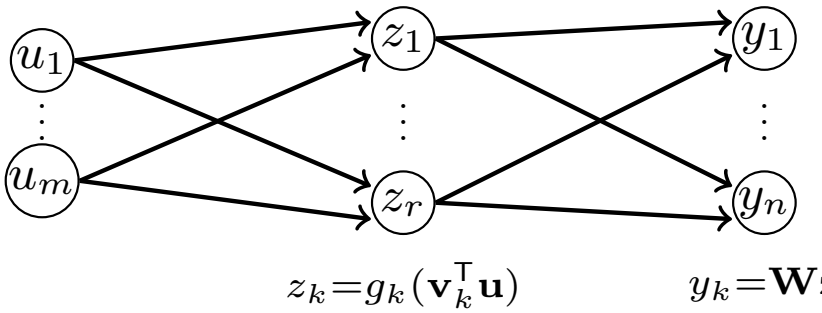
$$\underbrace{\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}}_{\text{sensors}} \in \mathbb{R}^n, \quad \underbrace{V = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_r \end{bmatrix}}_{\text{mixing matrix}}, \quad \underbrace{\mathbf{s} = \begin{bmatrix} s_1 \\ \vdots \\ s_r \end{bmatrix}}_{\text{sources}} \in \mathbb{R}^r$$

if s_k are **independent**, the 2nd characteristic function has expansion

$$\Psi_{\mathbf{x}}(\mathbf{u}) = \psi_{s_1}(\mathbf{v}_1^{\mathsf{T}}\mathbf{u}) + \cdots + \psi_{s_r}(\mathbf{v}_r^{\mathsf{T}}\mathbf{u})$$

See [Comon, Jutten, 2010], [Rajih, Comon, 2006]

Compact matrix notation

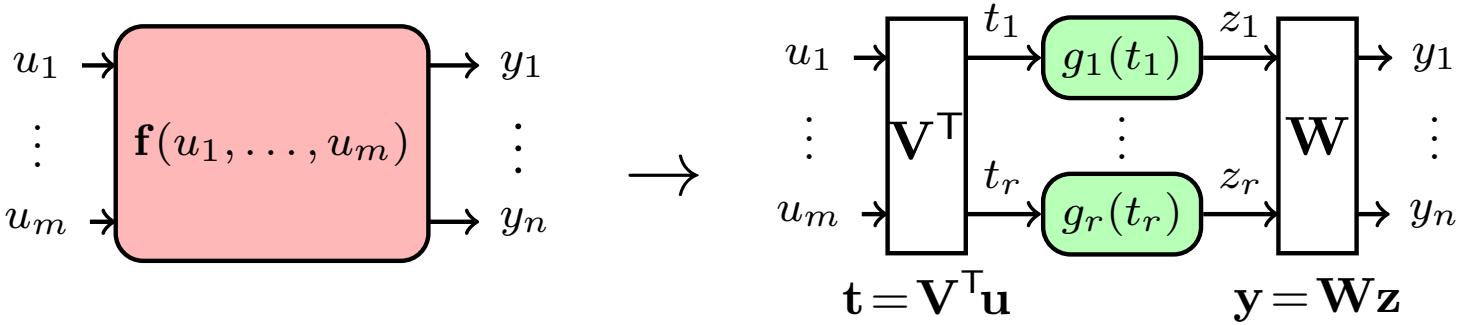


rewrite as $\boxed{\mathbf{f}(\mathbf{u}) = \mathbf{W}\mathbf{g}(\mathbf{V}^\mathsf{T}\mathbf{u})}$

with

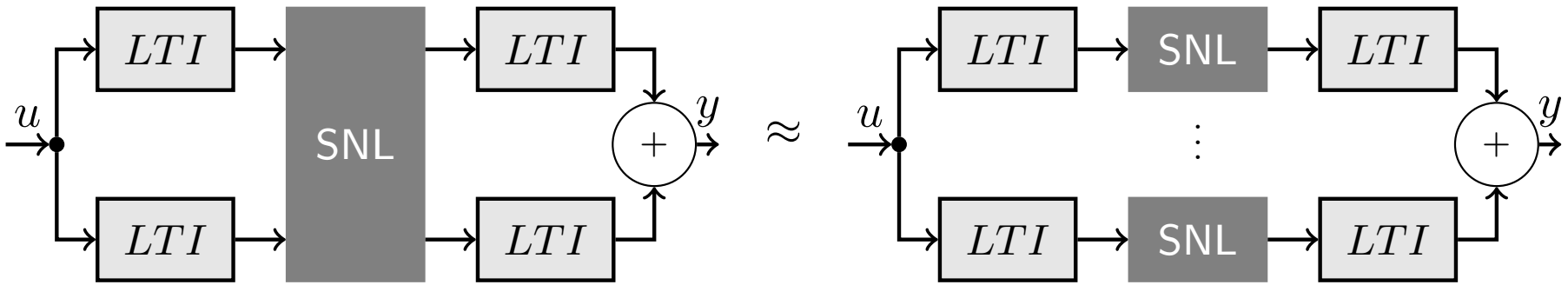
- $\mathbf{V} = \begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_r\end{bmatrix} \in \mathbb{R}^{m \times r}$
- $\mathbf{g}(t_1, \cdots, t_r) = [g_1(t_1) \cdots g_r(t_r)]^\mathsf{T}$

"Block-structured" form:

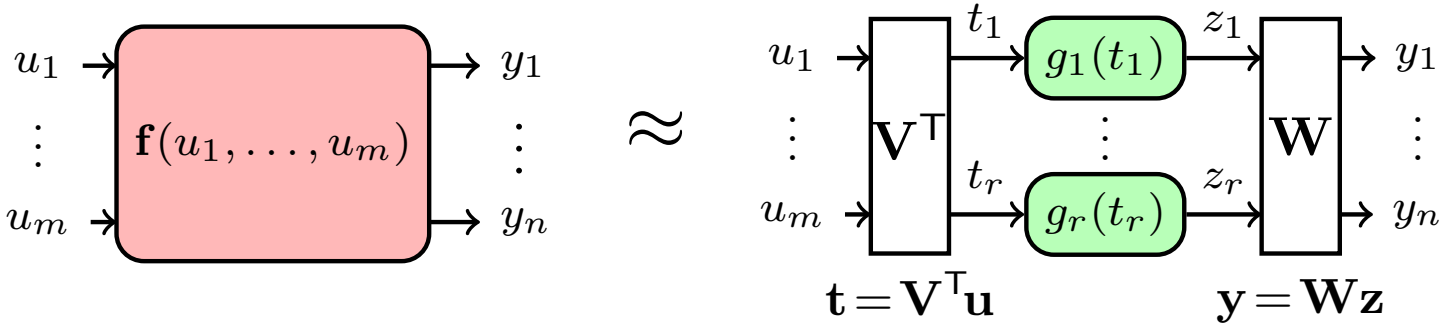


Nonlinear dynamical system identification

Identification of Parallel Wiener-Hammerstein systems



$\rightsquigarrow$ reduced to approximation of a multivariate map:



- called *decoupling* in system identification literature
- applicable to other block structures

Overview

Flexible activations

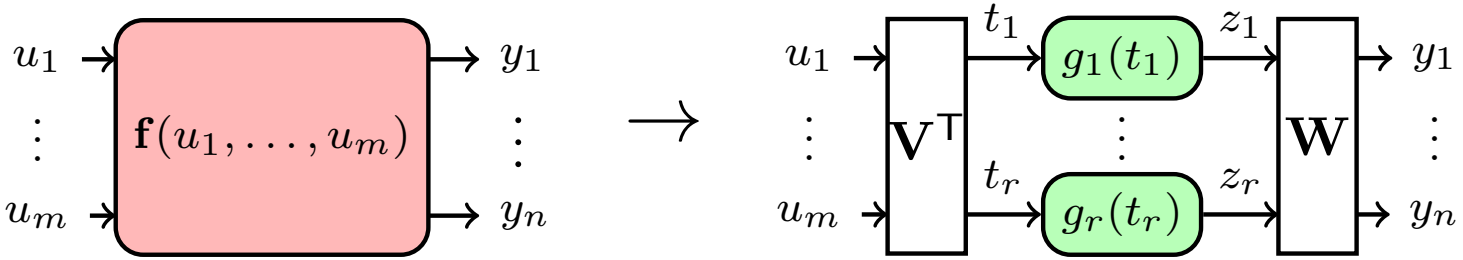
Link to CPD

Coupled matrix/tensor factorization

Low-rank tensor recovery

Exact factorization (decoupling)

Goal: Given $\mathbf{f}$, decompose it as $\mathbf{f}(\mathbf{u}) = \mathbf{W}\mathbf{g}(\mathbf{V}^\mathsf{T}\mathbf{u})$



with $\mathbf{V} \in \mathbb{R}^{m \times r}$, $\mathbf{W} \in \mathbb{R}^{n \times r}$, $\mathbf{g}(t_1, \cdots, t_r) = [g_1(t_1) \cdots g_r(t_r)]^\mathsf{T}$

Key idea of [Dreesen et al, 2015]: Jacobian of $\mathbf{f}$ has form

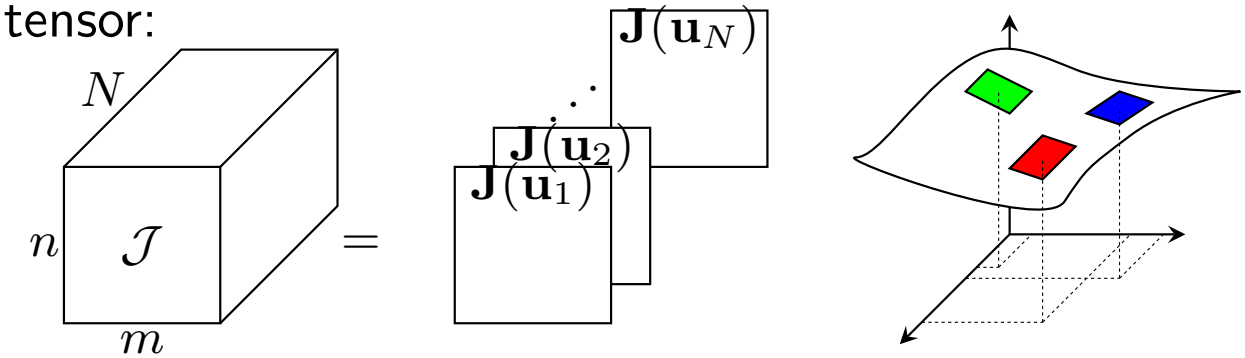
$$\boxed{\mathbf{J}_\mathbf{f}(\mathbf{u}) = \mathbf{W} \begin{bmatrix} g_1'(\mathbf{v}_1^\mathsf{T}\mathbf{u}) & & \\ & \ddots & \\ & & g_r'(\mathbf{v}_r^\mathsf{T}\mathbf{u}) \end{bmatrix} \mathbf{V}^\mathsf{T}}$$

only diagonal term depends on $\mathbf{u}$

Exact factorization: CPD of Jacobian tensor

Algorithm.

1. Evaluate $\mathbf{J_f}(\mathbf{u})$ at N "operating points" $\mathbf{u}_1, \ldots, \mathbf{u}_N \in \mathbb{R}^m$
2. Stack them into a tensor:



3. Joint matrix diagonalization $\leftrightarrow$ CPD $\boldsymbol{\mathcal{J}} = [\![\mathbf{W}, \mathbf{V}, \mathbf{H}]\!]$

$$\begin{matrix}\mathbf{J}(\mathbf{u}^{(1)}) = \mathbf{W}\mathbf{D}^{(1)}\mathbf{V}^{\mathsf{T}}, \\ \vdots \\ \mathbf{J}(\mathbf{u}^{(N)}) = \mathbf{W}\mathbf{D}^{(N)}\mathbf{V}^{\mathsf{T}}\end{matrix} \quad \longleftrightarrow \quad \mathcal{J} = \mathbf{w}_1 \circ \mathbf{v}_1 \circ \mathbf{h}_1 + \cdots + \mathbf{w}_r \circ \mathbf{v}_r \circ \mathbf{h}_r$$

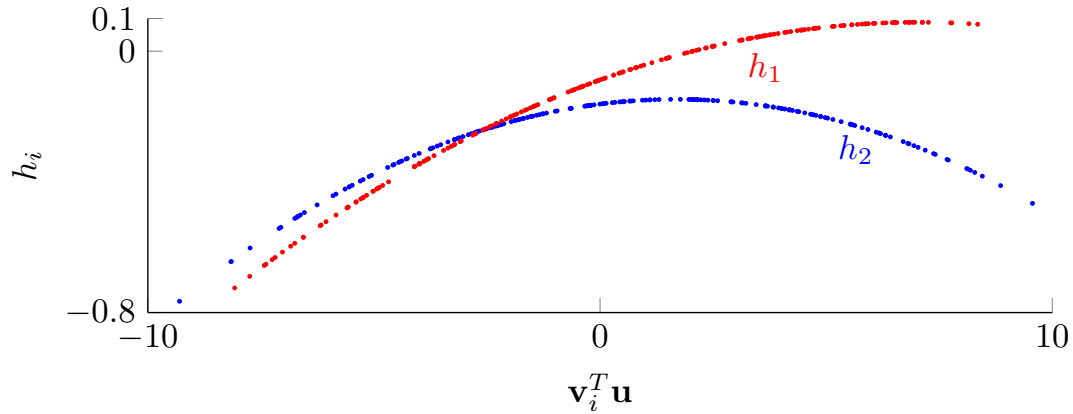
4. Retrieve $\mathbf{v}_k$, $\mathbf{w}_k$ from factors $\mathbf{W}, \mathbf{V}$ of the CPD
5. $\mathbf{h}_k$ contains evaluations of g_k'

Factorization and exact CPD

$$\mathbf{f}(\mathbf{u}) = \mathbf{W}\mathbf{g}(\mathbf{V}^\mathsf{T}\mathbf{u}) \Rightarrow \mathcal{J} = \mathbf{w}_1 \circ \mathbf{h}_1 \circ \mathbf{v}_1 + \cdots + \mathbf{w}_r \circ \mathbf{h}_r \circ \mathbf{v}_r$$

where $(\mathbf{h}_k)_i = g_k'(\mathbf{v}_k^\mathsf{T}\mathbf{u}_i)$

An example of recovery:
(exact decomposition,
$m = n = 2$, $r = 2$)



thanks to uniqueness of CPD of $\mathcal{J}$

Issues:

- CPD is a relaxation: structure of $\mathbf{H}$ is lost
- need extra step to estimate g_k

Polynomial case

Decompose polynomial map $\mathbf{f}\colon \mathbb{R}^m \to \mathbb{R}^n$ (degree d) as

$$\mathbf{f}(\mathbf{u}) = \mathbf{w}_1 g_1(\mathbf{v}_1^\mathsf{T}\mathbf{u}) + \cdots + \mathbf{w}_r g_r(\mathbf{v}_r^\mathsf{T}\mathbf{u}), \tag{1}$$

with $\mathbf{v}_k \in \mathbb{R}^m, \mathbf{w}_k \in \mathbb{R}^n$, $g_k(t) = c_{1,k}t + c_{2,k}t^2 + \cdots + c_{d,k}t^d$

[Comon, Qi, U., 2015], [U., Dreesen, Ishteva, 2020]: (1) is equivalent to a coupled tensor decomposition

$$\begin{aligned} \mathcal{T}^1 &= [\![\mathbf{W}, \mathbf{V}, \mathbf{c}_1^\mathsf{T}]\!], \\ \mathcal{T}^2 &= [\![\mathbf{W}, \mathbf{V}, \mathbf{V}, \mathbf{c}_2^\mathsf{T}]\!], \\ &\vdots \\ \mathcal{T}^d &= [\![\mathbf{W}, \underbrace{\mathbf{V}, \ldots, \mathbf{V}}_{d \text{ times}}, \mathbf{c}_d^\mathsf{T}]\!], \end{aligned}$$

Useful for studying uniqueness [Comon, Qi, U., 2017] (X-rank theory)

Generic uniqueness of the factorization

$$\mathbf{f}(\mathbf{u}) = \mathbf{w}_1 g_1(\mathbf{v}_1^{\mathsf{T}}\mathbf{u}) + \cdots + \mathbf{w}_r g_r(\mathbf{v}_r^{\mathsf{T}}\mathbf{u}),\ g_k(t) = c_{1,k}t + c_{2,k}t^2 + \cdots + c_{d,k}t^d$$

Theorem ([Comon, Qi, U., 2017], simplified)

Let s, $1 \leq s \leq d$, and r be such that

$$r \leq \min\left(\binom{m+s-1}{s}, C(m,n,d)\right) n.$$

Then for a general r-term $\mathbf{f}(\mathbf{u})$, we can recover uniquely (up to scaling ambiguities) $\mathbf{w}_k$, $\mathbf{v}_k$ and $c_{j,k}$, $j \geq s$

Examples:

- overall identifiability: $r \leq nm$
- nonlinear terms can be identifiable for $r \leq n\frac{m(m+1)}{2}$

Overview

Flexible activations

Link to CPD

Coupled matrix/tensor factorization

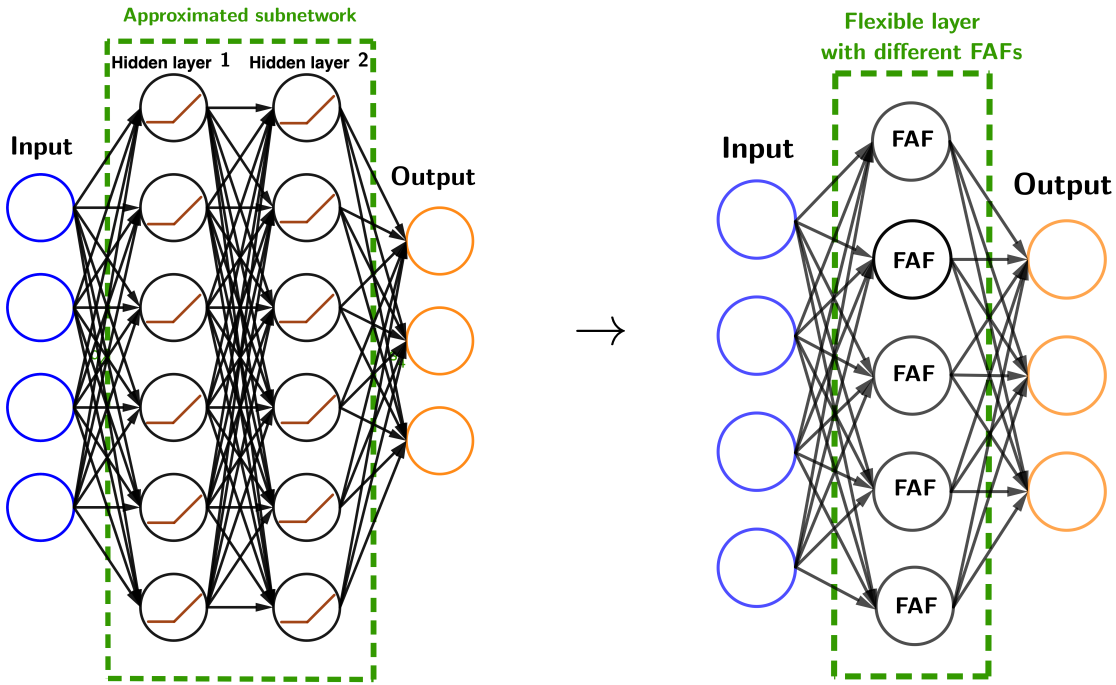
Low-rank tensor recovery

Tensor approximations for NN compression/learning

1. *(Lebedev et al., 2015)*
   Compress convolutional layer tensors ($\mathbf{W}$ is sparse and structured).
2. *(Novikov et al., 2016)*, Compression of fully-connected layers (e.g. tensorization of large matrix $\mathbf{W}$ +TT approximation)
3. *(Cohen et al., 2016)*: replace the nonlinear transfromation $g(\mathbf{W}^{\mathsf{T}}\mathbf{x} + \mathbf{b})$ with product pooling unit
   $\rightarrow$ (Khrulkov et al, 2018) link with tensor formats (TT, HT, ...)
4. *(Janzamin, et al. 2016)*: learning the linear layer $\mathbf{W}$ from tensor approximation of the score function

Framework: neural network compression

Goal: compress a part of a neural network:



with the flexible model: $\mathbf{f}(\mathbf{u}) \approx \mathbf{W}\mathbf{g}(\mathbf{V}^{\mathsf{T}}\mathbf{u})$

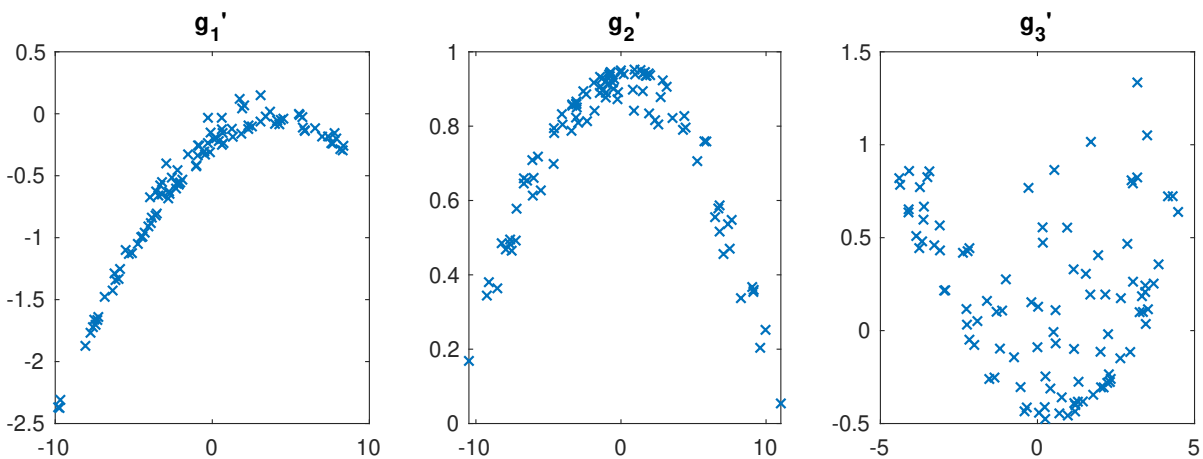
- pretrained network $\Rightarrow$ can evaluate derivatives
- use Jacobian tensor CP approximation

Drawbacks of the Jacobian approach

$$\min_{\mathbf{w}_k,\mathbf{v}_k,\mathbf{h}_k} \left\| \mathcal{J} - [\![\mathbf{W},\mathbf{V},\mathbf{H}]\!] \right\|_F, \quad \text{where} \quad \mathcal{J}_{:,:,i} = \mathbf{J}_{\mathbf{f}}(\mathbf{u}_i)$$

Issues:

- Need to estimate activation functions + loss of uniqueness

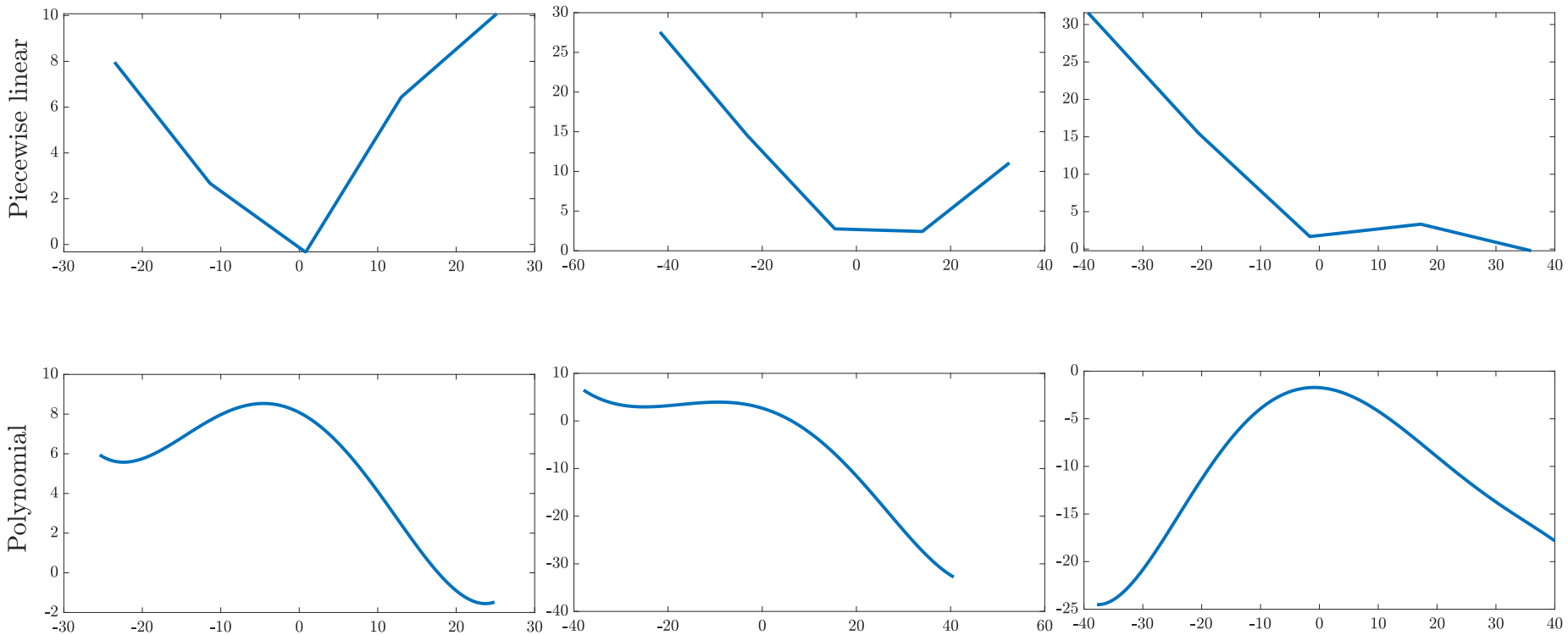


- derivatives are approximated instead of function values

Idea: add constraints

Activation function from a parameterized basis:

$$g_l(t) = c_{0,l} + c_{1,l}\phi_1(t) + \cdots + c_{d,l}\phi_d(t).$$



Proposed approach: coupled tensor-matrix factorization

For a base $g_l(t) = c_{0,l} + c_{1,l}\phi_1(t) + \cdots + c_{d,l}\phi_d(t)$, we solve

$$\min_{\mathbf{w}_k,\mathbf{v}_k,\mathbf{h}_k,\mathbf{z}_k} \underbrace{\left\|\mathcal{J} - [|\mathbf{W},\mathbf{V},\mathbf{H}|]\right\|^2}_{\text{Jacobian}} + \lambda \cdot \underbrace{\left\|\mathbf{F} - \mathbf{W}\mathbf{Z}^{\mathsf{T}}\right\|^2}_{\text{function evaluation (data fidelity)}}$$

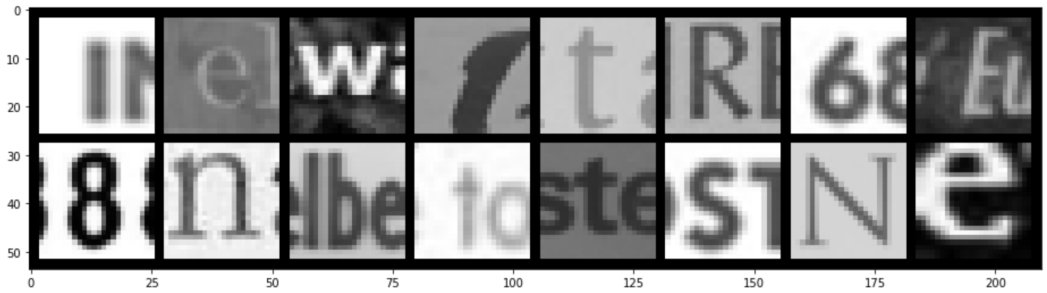
$$\text{subject to} \quad \mathbf{h}_l = \mathbf{X}_l \cdot \mathbf{c}_l, \quad \mathbf{z}_l = \mathbf{Y}_l \cdot \mathbf{c}_l, \quad \text{with}$$

$$\mathbf{X}_l = \begin{bmatrix} 0 & \phi_1'(\mathbf{v}_l^{\mathsf{T}}\mathbf{u}^{(1)}) & \cdots & \phi_d'(\mathbf{v}_l^{\mathsf{T}}\mathbf{u}^{(1)}) \\ \vdots & \vdots & & \vdots \\ 0 & \phi_1'(\mathbf{v}_l^{\mathsf{T}}\mathbf{u}^{(N)}) & \cdots & \phi_d'(\mathbf{v}_l^{\mathsf{T}}\mathbf{u}^{(N)}) \end{bmatrix}, \quad \mathbf{Y}_l = \begin{bmatrix} 1 & \phi_1(\mathbf{v}_l^{\mathsf{T}}\mathbf{u}^{(1)}) & \cdots & \phi_d(\mathbf{v}_l^{\mathsf{T}}\mathbf{u}^{(1)}) \\ \vdots & \vdots & & \vdots \\ 1 & \phi_1(\mathbf{v}_l^{\mathsf{T}}\mathbf{u}^{(N)}) & \cdots & \phi_d(\mathbf{v}_l^{\mathsf{T}}\mathbf{u}^{(N)}) \end{bmatrix}.$$

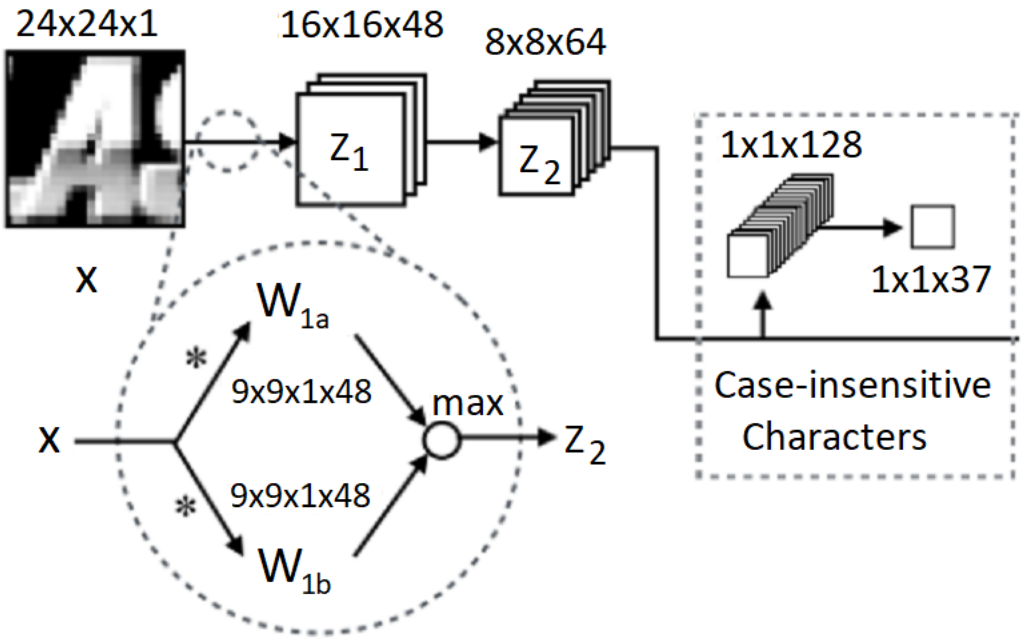
- constraints to preserve the structure
- data fidelity term to approximate well the predictions
  $\Rightarrow$ avoid fine-tuning

Test of compression

ICDAR 2003 dataset:
$\approx$163000 train, $\approx$ 5300 test,
36 classes



CharNet
[Jaderberg et al., 2014]



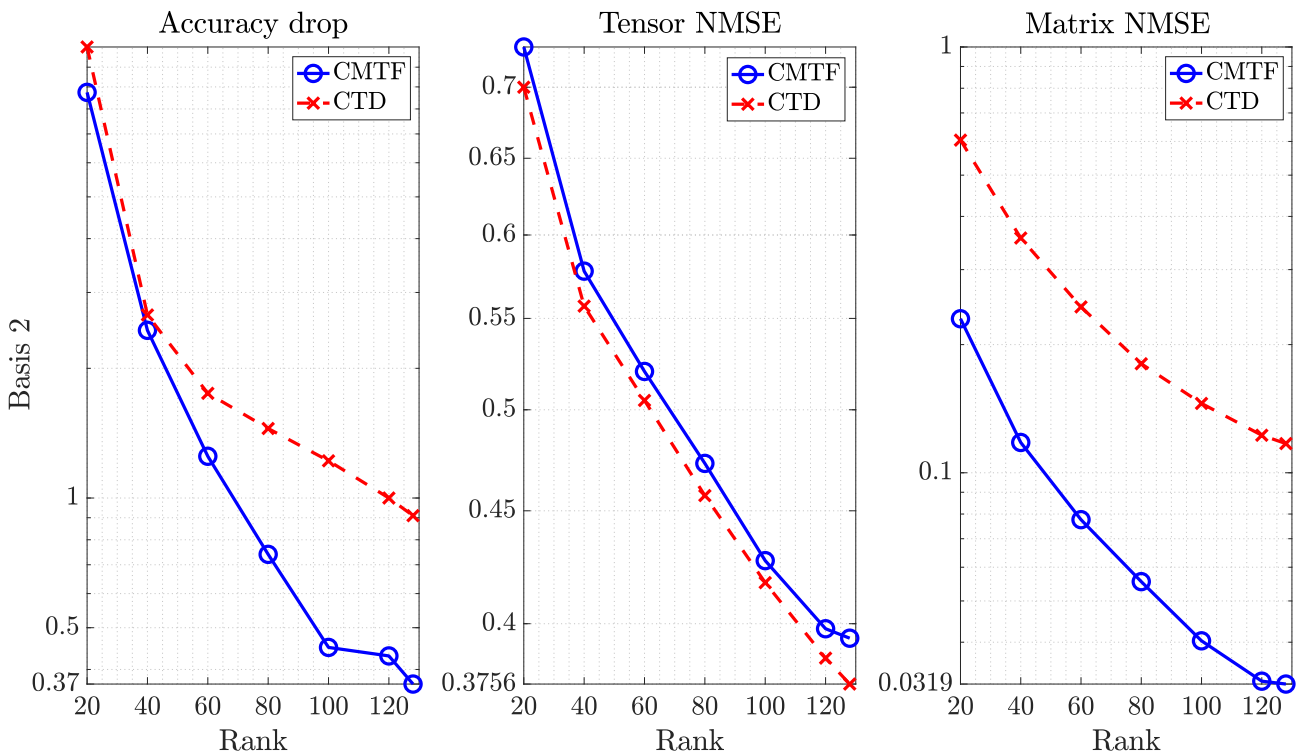
compress the conv3 layer (viewed as fully connected layer $\mathbb{R}^{4096} \to \mathbb{R}^{128}$)

CMTF compression performance

- use only 360 operating points $\mathbf{u}_i$ (10 per class from training set)
- no fine-tuning and 4x compression

$$\min_{\mathbf{w}_k,\mathbf{v}_k,\mathbf{h}_k,\mathbf{z}_k} \underbrace{\left\| \mathcal{J} - [|\mathbf{W},\mathbf{V},\mathbf{H}|] \right\|^2}_{\text{Jacobian}} + \lambda \cdot \underbrace{\left\| \mathbf{F} - \mathbf{W}\mathbf{Z}^{\mathsf{T}} \right\|^2}_{\text{function evaluation}}$$

subject to constraints on $\mathbf{H}$ and $\mathbf{Z}$



Overview

Flexible activations

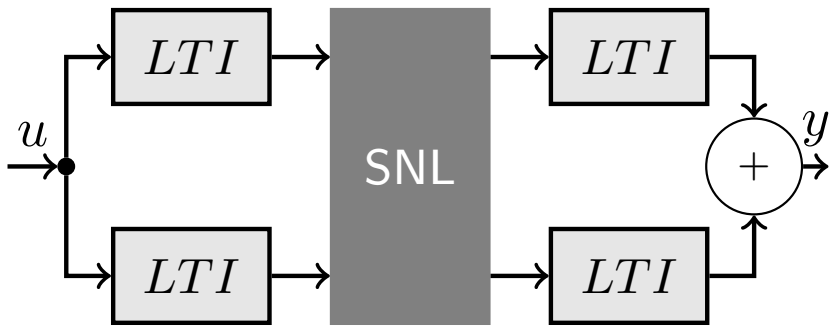
Link to CPD

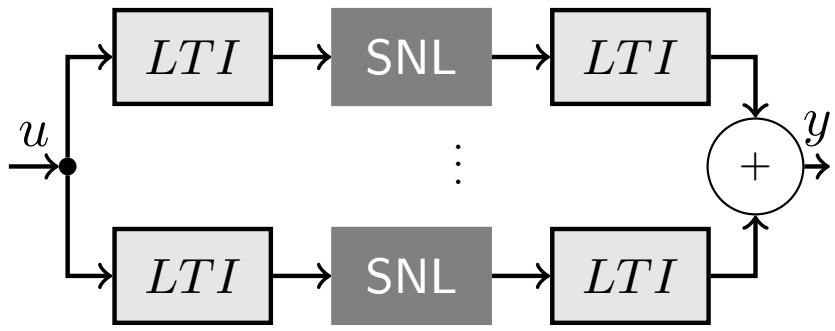
Coupled matrix/tensor factorization

Low-rank tensor recovery

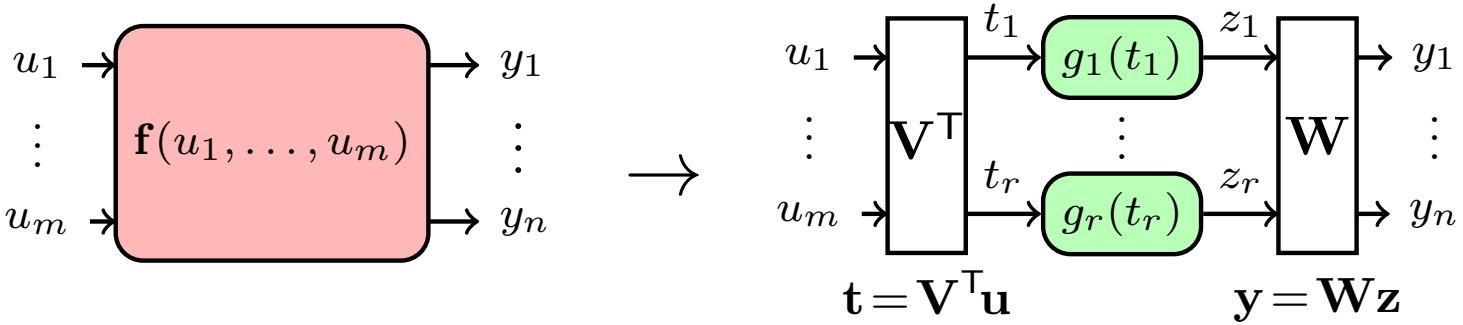
Nonlinear dynamical system identification

Identification of Parallel Wiener-Hammerstein systems





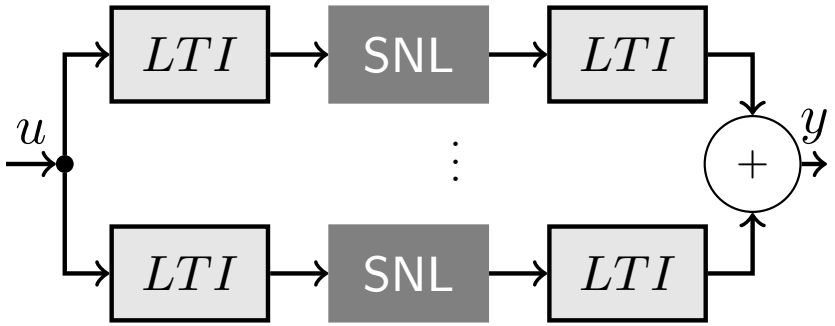
$\rightsquigarrow$ reduced to approximation of a multivariate map: $\mathbf{f}(\mathbf{u}) \approx \mathbf{W}\mathbf{g}(\mathbf{V}^{\mathsf{T}}\mathbf{u})$



- called *decoupling* in system identification literature
- applicable to other block structures

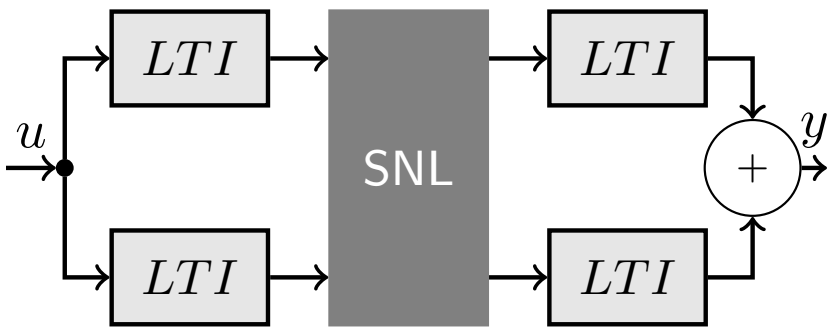
Parallel Wiener-Hammerstein system identification

Goal: estimate



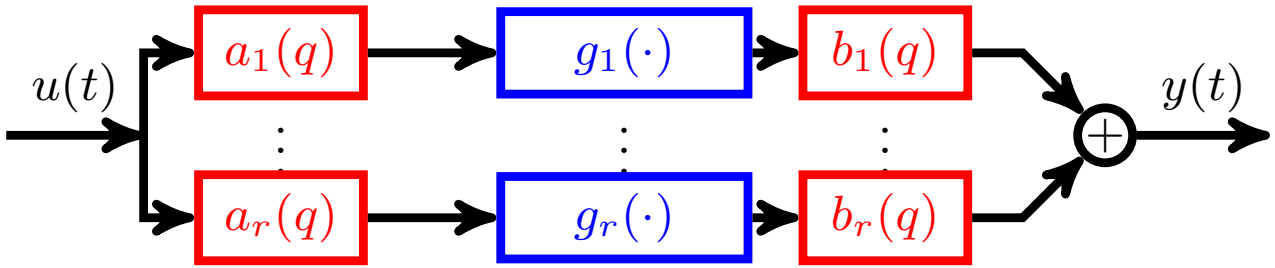
from input and output data $u(t)$, $v(t)$.

Current approaches are two-step: estimate first



Can we identify PWH system directly?

Parallel Wiener Hammerstein system: FIR case



Assumption:

- $a_k(q)$, $b_k(q)$ — finite impulse response (FIR) filters with lags L_1, L_2
- $g_k(\cdot)$ are univariate (static) functions

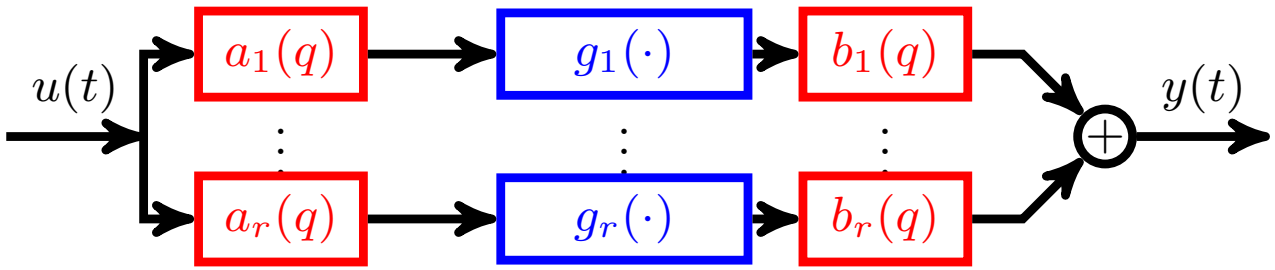
$$\mathbf{y} = \sum_{k=1}^{r} \mathbf{b}_k * (g_k(\mathbf{a}_k * \mathbf{u}))$$

where

- $\mathbf{a}_k \in \mathbb{R}^{L_1}$, $\mathbf{b}_k \in \mathbb{R}^{L_2}$ — filter coefficients
- $g_k(\cdot)$ applied elementwise to $\mathbf{a}_k * \mathbf{u}$ (filtered input)

convolutional NN with flexible activation functions

Our setup: FIR and polynomial activations



1. a_k, b_k are FIR with lags L_1 and L_2
   $\Rightarrow$ y depends on $L = L_1 + L_2 - 1$ past inputs:

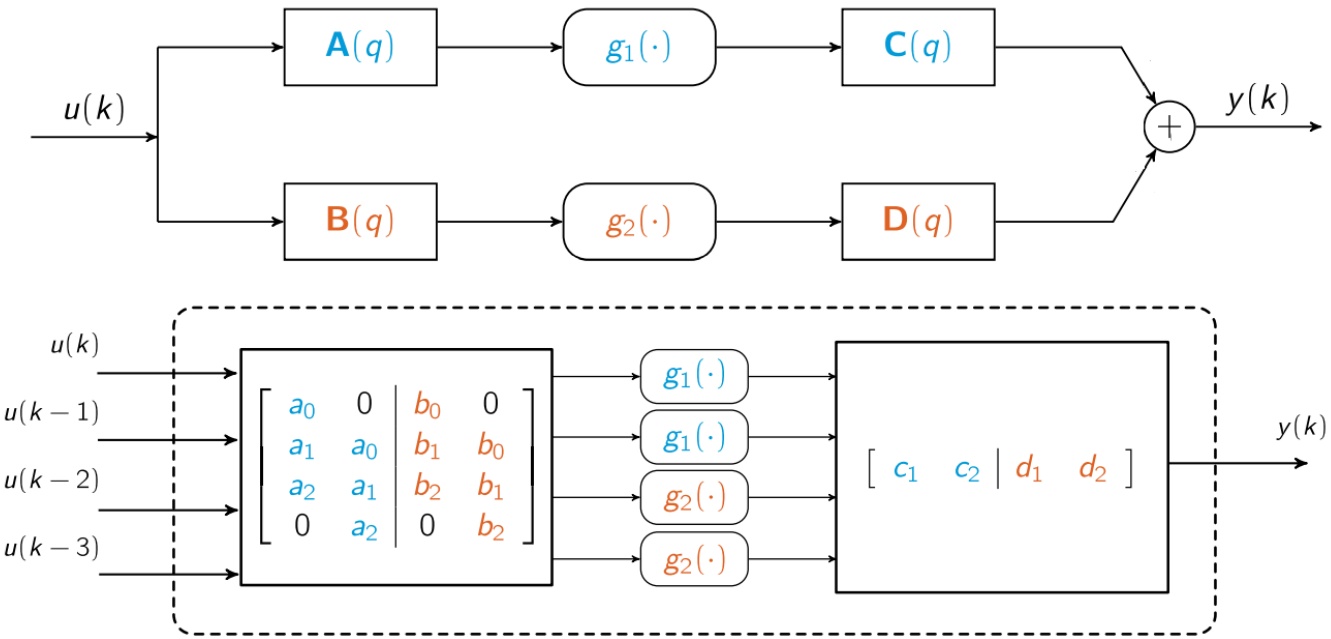
$$y(t) = f(u(t), u(t-1), \ldots, u(t-L+1)),$$

2. g_k — polynomial of degree d
   $\Rightarrow$ f is a multivariate polynomial in $(u(t), u(t-1), \ldots, u(t-L+1))$

   identification from truncated Volterra kernels up to order d

Apply plain decoupling to parallel Wiener-Hammerstein

[Dreesen, Ishteva, 2021]:



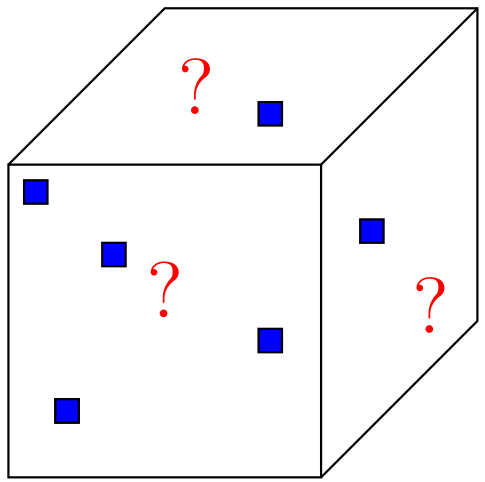
Drawbacks:

- Inflates tensor rank (rL_1 branches)
- Structured factors (similar to [Kibangou, Favier, 2010])

Our approach: use tensor recovery

Low-rank tensor recovery

- **Low-rank** tensor completion from few elements



$$= \quad [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

- Tensor recovery: reconstruct a **low-rank** $\mathcal{T} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ from M samples (projections) $\mathscr{P}(\mathcal{T})$:

$$\mathscr{P}(\mathcal{T}) = \begin{bmatrix} \langle \mathcal{T}, \mathcal{S}_1 \rangle_F & \cdots & \langle \mathcal{T}, \mathcal{S}_M \rangle_F \end{bmatrix}^{\mathsf{T}}$$

— linear sampling (projection) operator

Key ideas of our approach

We mimick the CPD Jacobian method of [Dreesen et al. 2015], but

1. take operating points $\mu_k, k = 1, \ldots, N$ with Vandermonde structure:

$$\mathbf{u}_\mu = \begin{bmatrix} 1 & \mu & \mu^2 & \ldots & \mu^{L-1} \end{bmatrix}^\mathsf{T}, \quad \mu \in \mathbb{C}$$

2. split the nonlinearity into homogeneous parts:

$$f(\mathbf{u}) = f^{(0)} + f^{(1)}(\mathbf{u}) + \cdots + f^{(d)}(\mathbf{u})$$

3. where gradient of $f^{(s)}$ — partial contraction of s-th Volterra kernel:

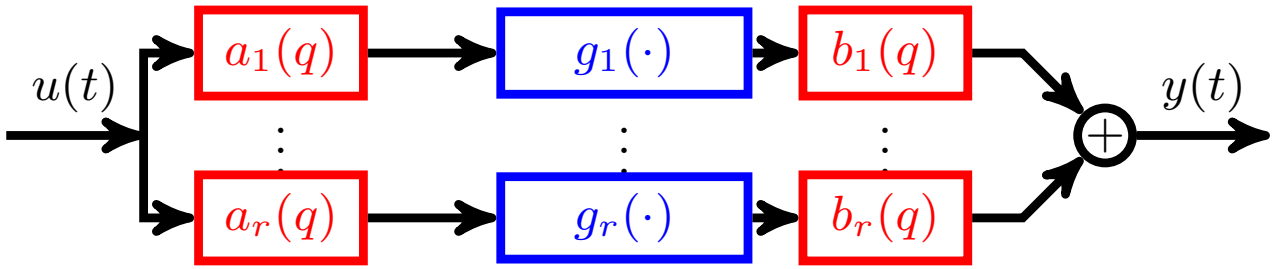
$$\nabla f^{(s)}(\mathbf{u}) = s \cdot \boldsymbol{\mathcal{H}}^{(s)} \bullet_2 \mathbf{u} \cdots \bullet_s \mathbf{u}.$$

4. stack the gradients into a "measurement" vector

$$\mathbf{z}_k = \begin{bmatrix} (\nabla f^{(1)}(\mathbf{u}_{\mu_k}))^\mathsf{T} & (\nabla f^{(2)}(\mathbf{u}_{\mu_k}))^\mathsf{T} & \cdots & (\nabla f^{(2)}(\mathbf{u}_{\mu_k}))^\mathsf{T} \end{bmatrix}$$

5. $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)$ must be a sampling of a low-rank tensor

Main result



Proposition [U., Dreesen, Ishteva, 2021]
$\mathbf{z}$ is a projection of a rank-r polyadic decomposition

$$\mathbf{z} = \mathscr{P}(\boldsymbol{\mathcal{T}}), \quad \boldsymbol{\mathcal{T}} = \underbrace{\sum_{\ell=1}^{r} \mathbf{a}_\ell \otimes \mathbf{b}_\ell \otimes \mathbf{h}_\ell}_{[\![\mathbf{A},\mathbf{B},\mathbf{H}]\!]},$$

$\mathbf{a}_\ell \in \mathbb{R}^{L_1}$, $\mathbf{b}_\ell \in \mathbb{R}^{L_2}$, coefficients of g_ℓ can be recovered from $\mathbf{h}_\ell$

parallel Wiener-Hammerstein system — rank-r tensor recovery

Example

$r = 2$ branches, filter lengths $L_1 = 3$, $L_2 = 3$ with coefficients:

$$\mathbf{A} = \begin{bmatrix} 0.3 & 0.6 \\ -0.4 & 0.2 \\ 0.1 & 0.3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \\ 0.1 & 0.01 \end{bmatrix},$$

and nonlinearities $g_1(x_1) = 3x_1^3 - x_1^2 + 5, \quad g_2(x_2) = -5x_2^3 + 3x_2 - 7.$

Correct recovery with 30 random μ on the unit circle:

$$\widehat{\mathbf{A}} = \begin{bmatrix} 1 & 1 \\ -1.3333 - i0.8305 \cdot 10^{-8} & 0.3333 + i0.4203 \cdot 10^{-9} \\ 0.3333 - i0.2210 \cdot 10^{-8} & 0.4999 + i0.7007 \cdot 10^{-10} \end{bmatrix},$$

$$h_1(t) = t^2 - 0.2222t, \quad h_2(t) = t^2 - 0.2.$$

Tensor recovery

For $\mathbf{z} \in \mathbb{C}^M$, rank r, and sampling operator $\mathscr{P} : \mathbb{C}^{L_1 \times L_2 \times L_3} \to \mathbb{C}^M$ find

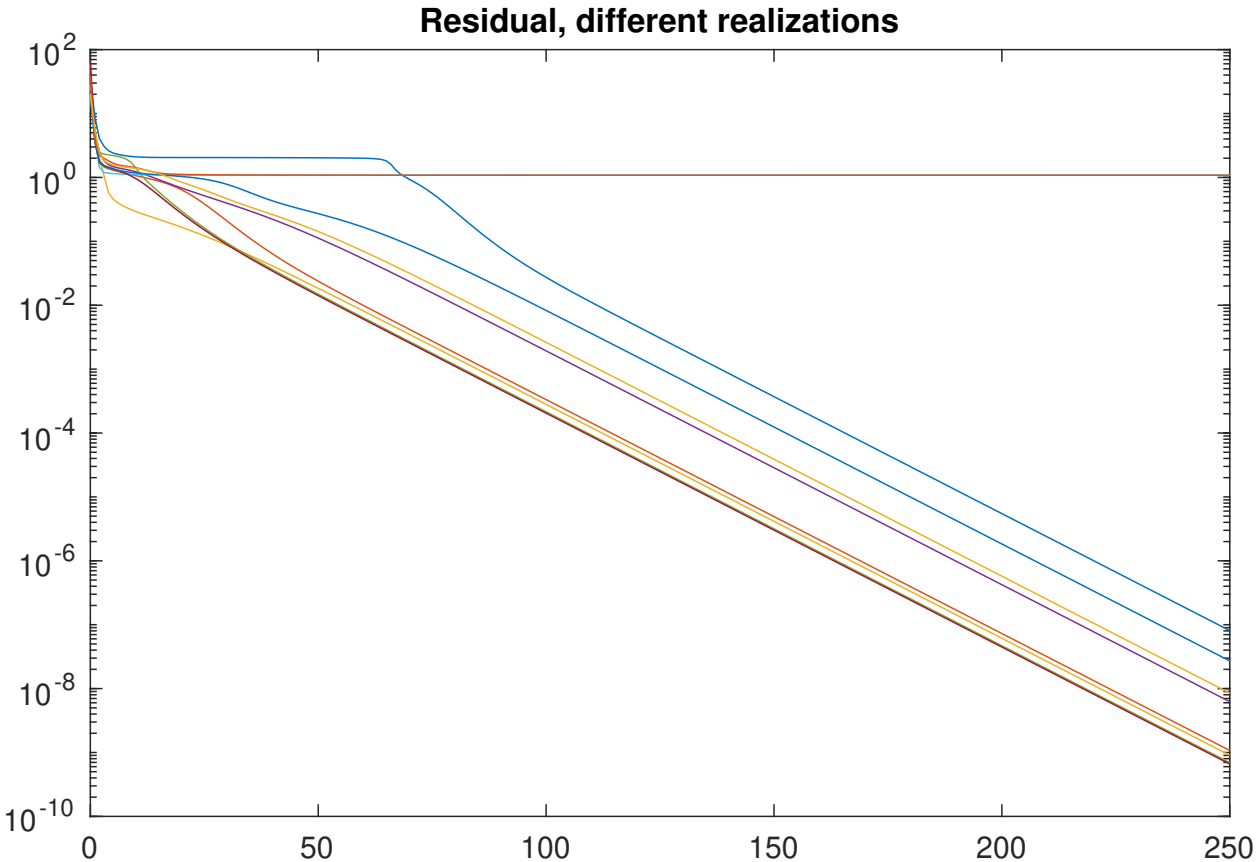
$$\min_{\mathbf{A} \in \mathbb{C}^{L_1 \times r}, \mathbf{B} \in \mathbb{C}^{L_2 \times r}, \mathbf{H} \in \mathbb{C}^{L_3 \times r}} \| \mathscr{P}(\llbracket \mathbf{A}, \mathbf{B}, \mathbf{H} \rrbracket) - \mathbf{z} \|_2^2,$$

Algorithm (alternating least squares).
Input: initializations $\mathbf{A}_0$, $\mathbf{B}_0$, $\mathbf{H}_0$.

1. For k=1,2,.... until a stopping criterion is satisfied
2. $\quad \mathbf{A}_k \leftarrow \arg\min_{\mathbf{A}} \| \mathscr{P}(\llbracket \mathbf{A}, \mathbf{B}_{k-1}, \mathbf{H}_{k-1} \rrbracket) - \mathbf{z} \|_2^2$;
3. $\quad \mathbf{B}_k \leftarrow \arg\min_{\mathbf{B}} \| \mathscr{P}(\llbracket \mathbf{A}_k, \mathbf{B}, \mathbf{H}_{k-1} \rrbracket) - \mathbf{z} \|_2^2$;
4. $\quad \mathbf{H}_k \leftarrow \arg\min_{\mathbf{H}} \| \mathscr{P}(\llbracket \mathbf{A}_k, \mathbf{B}_k, \mathbf{H} \rrbracket) - \mathbf{z} \|_2^2$.
5. End for

Dependence on intialization

Convergence for 10 initializations (prev. example):



Open questions and references

Ongoing work:

- deeper architectures with flexible activation functions: identifiability and algorithms
- optimization (better/scalable algorithms, convergence)
- robustness to perturbations, stability (link with uniqueness)
- link with other tensor approximation methods for NN compression

Thank you!

And all my collaborators:

- Jacobians (CPD-based methods):
  K. U., P. Dreesen and M. Ishteva., *Decoupling multivariate polynomials: interconnections between tensorizations*, JCAM, 2020.
- Generic uniqueness (X-rank):
  P. Comon, Y. Qi and K. U., *X-rank and identifiability for a polynomial decomposition model*, SIAGA, 2017.
- NN compression/ CMTF:
  Y.Zniyed, K. U., S. Miron, D. Brie, *Tensor-based framework for training flexible neural networks*, 2021, arXiv:2106.13542 .
- Tensor recovery for convolutional case:
  K. U., P. Dreesen, M. Ishteva, *Low-rank tensor recovery for Jacobian-based Volterra identificationof parallel Wiener-Hammerstein systems*, proceedings of SYSID 2021, arXiv:2109.09584 .