Machine Learning with Tensors for Structured Data

Guillaume Rabusseau

IVADO - McGill University - Reasoning and Learning Lab

June 11, 2018
RIKEN AIP

Learning with Structured Data

Supervised Learning:

Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

Learning with Structured Data

Supervised Learning:

Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- Classical learning algorithms assume $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^p$.
- How to handle input/output structured data?

Learning with Structured Data

Supervised Learning:

Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- Classical learning algorithms assume $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^p$.
- How to handle input/output structured data?
  - Tensor structured data: Images, videos, spatio-temporal data, ...

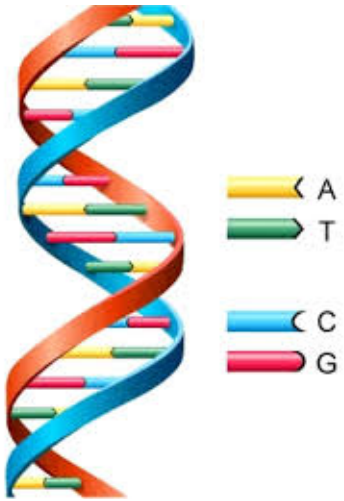
$\in \mathbb{R}^{32 \times 32 \times 3} \simeq \mathbb{R}^{3072}$

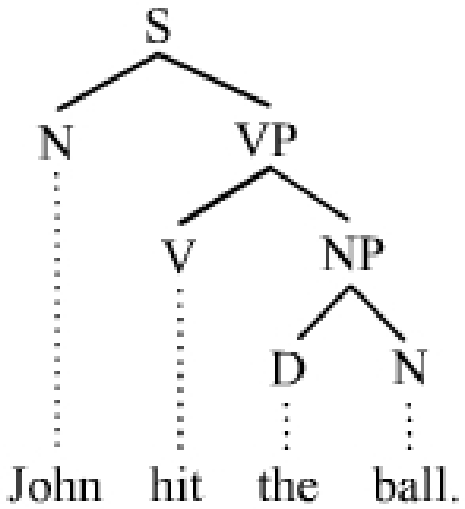
Learning with Structured Data

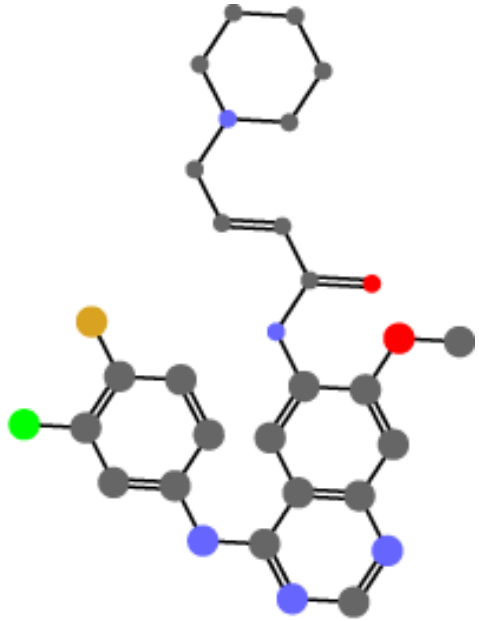
Supervised Learning:

Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- Classical learning algorithms assume $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^p$.
- How to handle input/output structured data?
  - Tensor structured data: Images, videos, spatio-temporal data, ...
  - Discrete structured data: strings, trees, graphs, ...







Learning with Structured Data

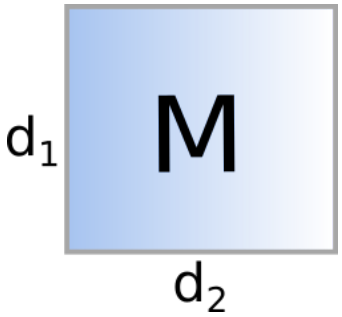
Supervised Learning:

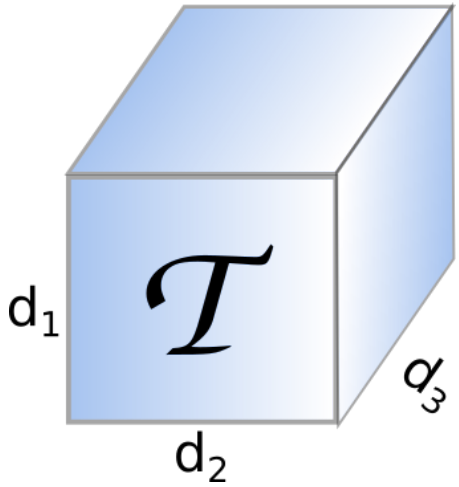
Learn $f : \mathcal{X} \to \mathcal{Y}$ from a sample $\{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- Classical learning algorithms assume $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^p$.
- How to handle input/output structured data?
  - Tensor structured data: Images, videos, spatio-temporal data, ...
  - Discrete structured data: strings, trees, graphs, ...

- In both cases, one can leverage linear and tensor algebra to design learning algorithms.

Tensors





$\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$

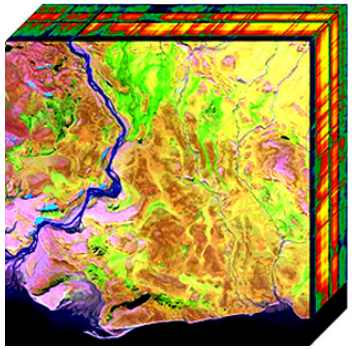
$\mathbf{M}_{ij} \in \mathbb{R}$ for $i \in [d_1], j \in [d_2]$

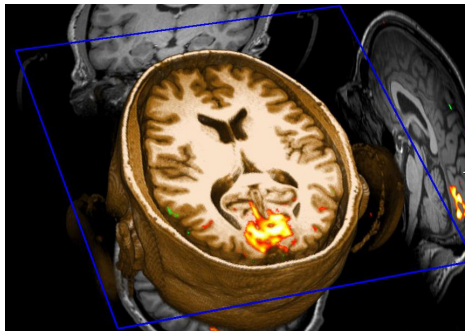
$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$

$(\mathcal{T}_{ijk}) \in \mathbb{R}$ for $i \in [d_1], j \in [d_2], k \in [d_3]$

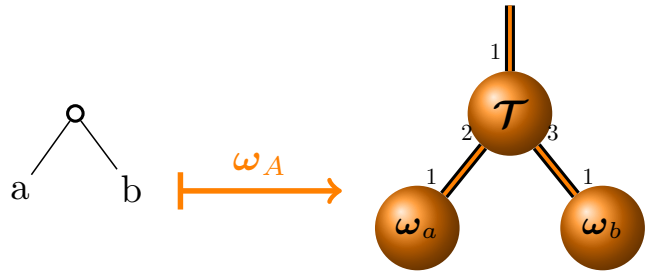
Tensors and Machine Learning

(i) Data has a tensor structure: color image, video, multivariate time series...





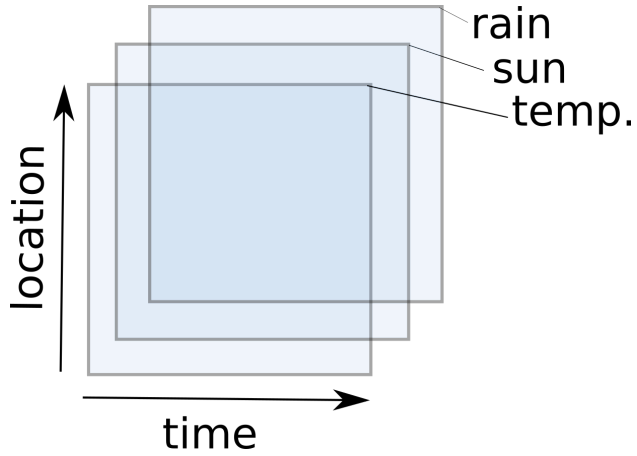
(ii) Tensors as parameters of a model: polynomial regression, higher-order RNNs, weighted automata on trees and graphs...



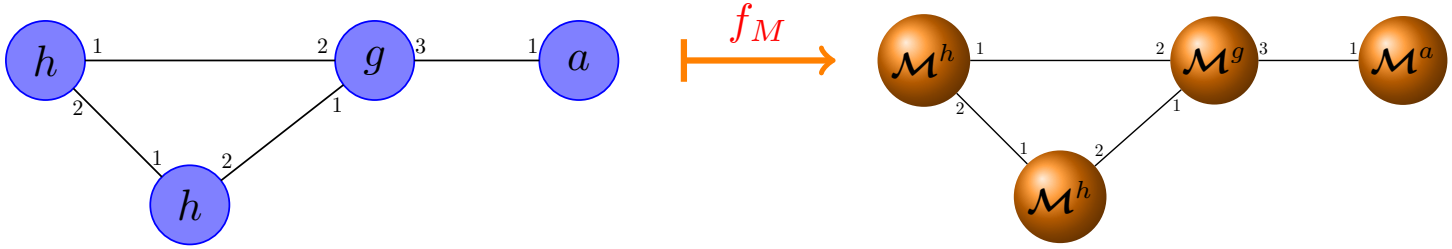
(iii) Tensors as tools: tensor method of moments [Anandkumar et al., 2014], layer compression in neural networks [Novikov et al., 2015], deep learning theoretical analysis [Cohen et al., 2015]...

Contributions

- Low rank regression for tensor data [NIPS'16, arXiv'17]



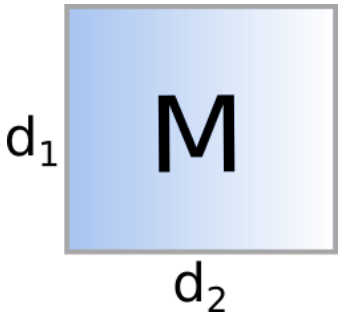
- Weighted automata for learning with discrete structured data [NIPS'17-a, AISTATS'18, JCSS'18, FoSSaCS'18]

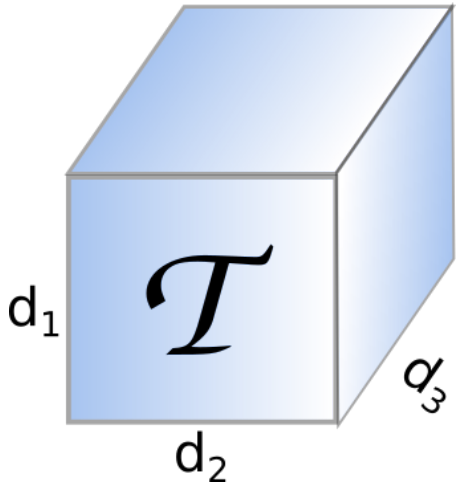


- Tensor Method of Moments [NIPS'17-b, CAP'14]

Outline

Tensors





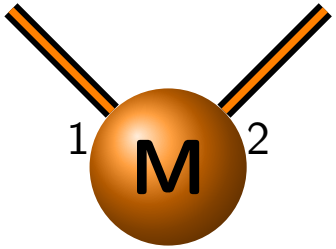
$\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$

$\mathbf{M}_{ij} \in \mathbb{R}$ for $i \in [d_1], j \in [d_2]$

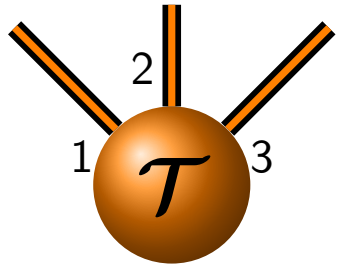
$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$

$(\mathcal{T}_{ijk}) \in \mathbb{R}$ for $i \in [d_1], j \in [d_2], k \in [d_3]$

Tensor Networks

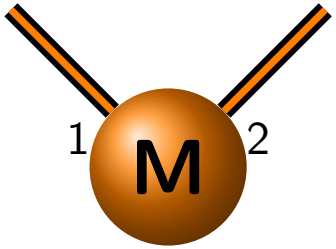


Matrix: $\mathbf{M}_{i_1 i_2}$

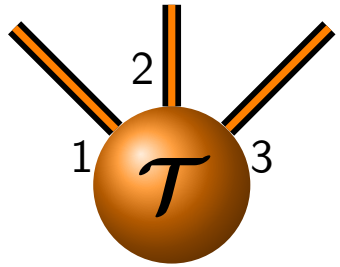


3rd order tensor: $\mathcal{T}_{i_1 i_2 i_3}$

Tensor Networks

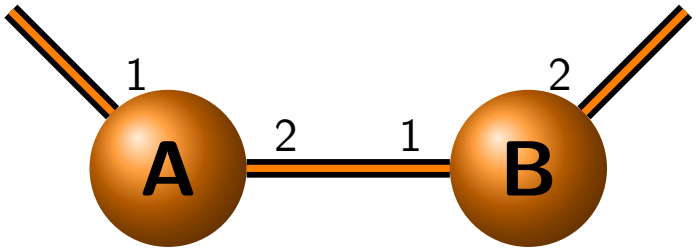


Matrix: $\mathbf{M}_{i_1 i_2}$



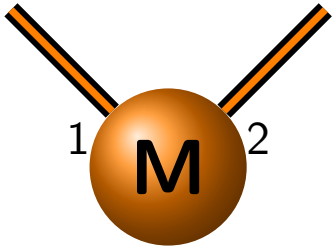
3rd order tensor: $\mathcal{T}_{i_1 i_2 i_3}$

Matrix product:

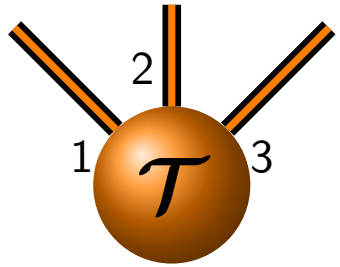


$$(\mathbf{AB})_{i_1,i_2} = \sum_k \mathbf{A}_{i_1 k} \mathbf{B}_{k i_2}$$

Tensor Networks

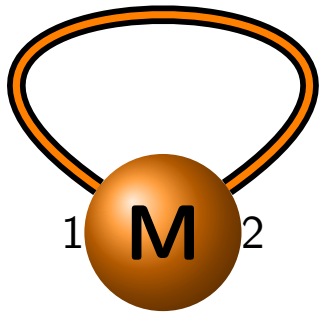


Matrix: $\mathbf{M}_{i_1 i_2}$



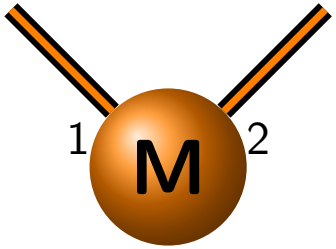
3rd order tensor: $\mathcal{T}_{i_1 i_2 i_3}$

Trace:

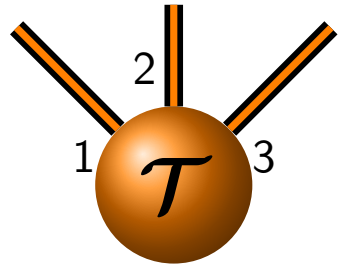


$$\mathrm{Tr}(\mathbf{M}) = \sum_i \mathbf{M}_{ii}$$

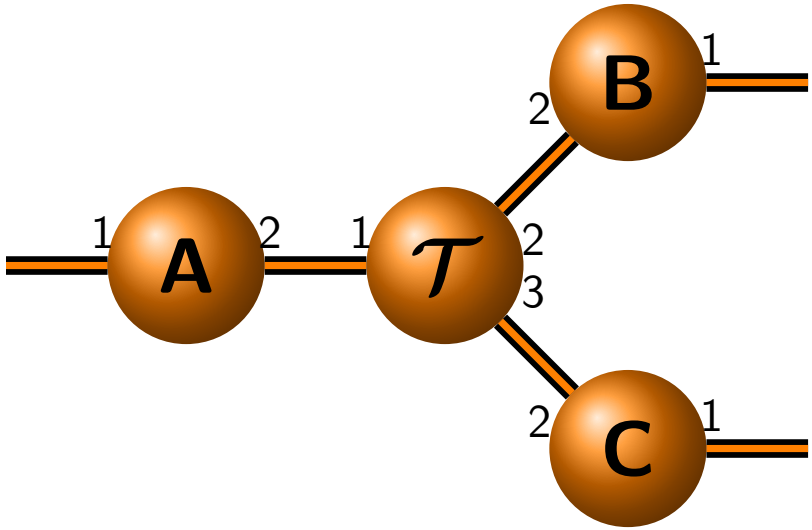
Tensor Networks



Matrix: $\mathbf{M}_{i_1 i_2}$



3rd order tensor: $\mathcal{T}_{i_1 i_2 i_3}$

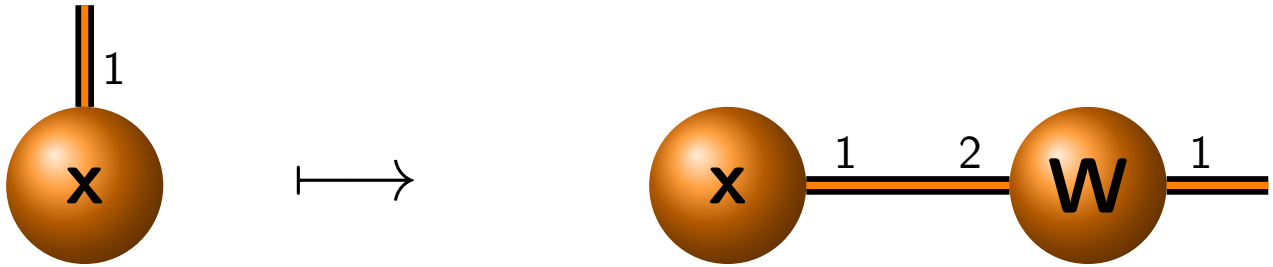


Tensor times matrices:

$$(\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C})_{i_1,i_2,i_3} = \sum_{k_1 k_2 k_3} \mathcal{T}_{k_1 k_2 k_3} \mathbf{A}_{i_1 k_1} \mathbf{B}_{i_2 k_2} \mathbf{C}_{i_3 k_3}$$

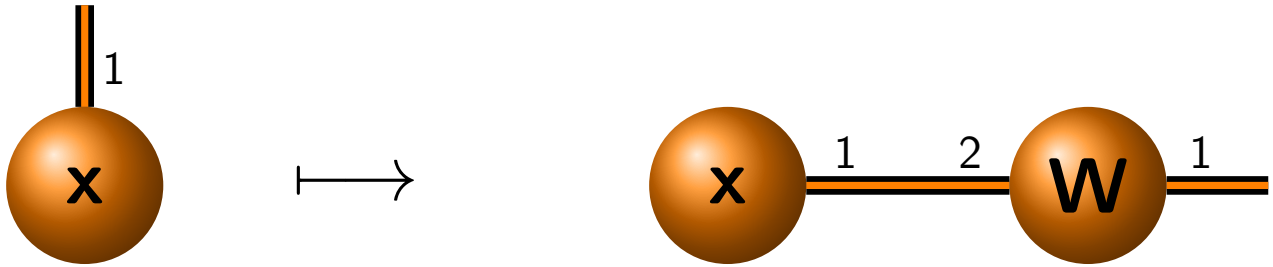
Multilinear Maps

- Liner map $f : \mathbb{R}^d \to \mathbb{R}^p$ maps $\mathbf{x}$ to $\mathbf{W}\mathbf{x} = \mathbf{W} \times_2 \mathbf{x}$ for some $\mathbf{W} \in \mathbb{R}^{p \times d}$:

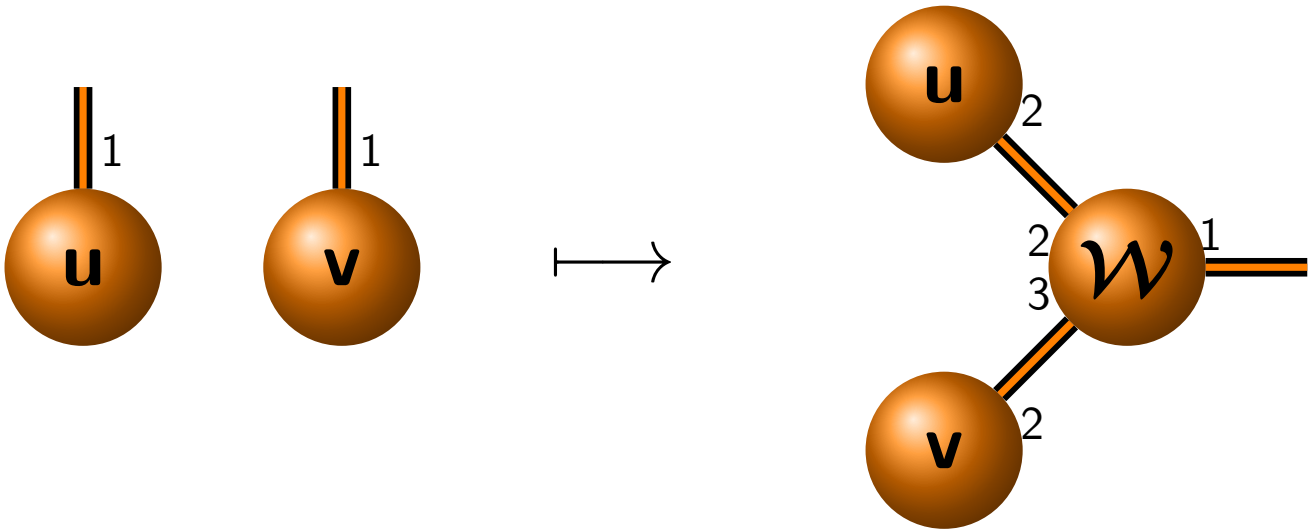


Multilinear Maps

- Liner map $f : \mathbb{R}^d \to \mathbb{R}^p$ maps $\mathbf{x}$ to $\mathbf{W}\mathbf{x} = \mathbf{W} \times_2 \mathbf{x}$ for some $\mathbf{W} \in \mathbb{R}^{p \times d}$:



- Multilinear map $g : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^p$ maps $(\mathbf{u}, \mathbf{v})$ to $\mathcal{W} \times_2 \mathbf{u} \times_3 \mathbf{v}$ for some $\mathcal{W} \in \mathbb{R}^{p \times d_1 \times d_2}$:



Example: Multilinear Maps in Higher-Order RNNs

- Recurrent Neural Network (RNN):

$$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots) \mapsto (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \cdots)$$

- Simple RNN:

$$\mathbf{h}_t = g(\mathbf{U}\mathbf{x}_t + \mathbf{V}\mathbf{h}_{t-1}), \qquad \mathbf{y}_t = g(\mathbf{M}\mathbf{h}_t)$$

Example: Multilinear Maps in Higher-Order RNNs

- Recurrent Neural Network (RNN):

$$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots) \mapsto (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \cdots)$$

- Simple RNN:

$$\mathbf{h}_t = g(\mathbf{U}\mathbf{x}_t + \mathbf{V}\mathbf{h}_{t-1}), \qquad \mathbf{y}_t = g(\mathbf{M}\mathbf{h}_t)$$

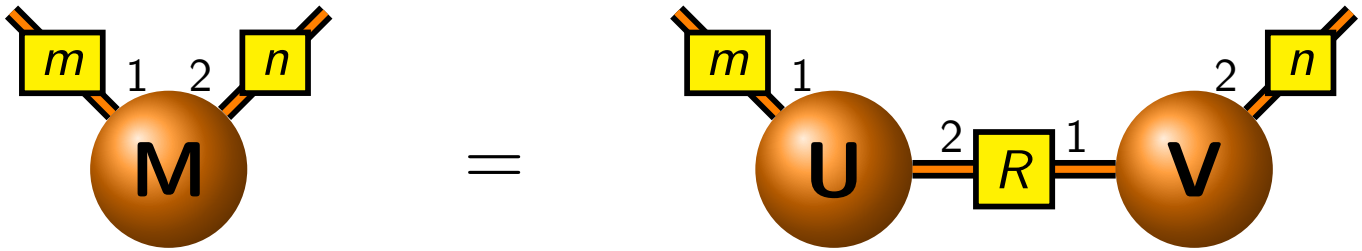
- Second-order RNN [Giles et al., NIPS'90]:

$$\mathbf{h}_t = g(\mathcal{W} \times_2 \mathbf{x}_t \times_3 \mathbf{h}_{t-1})$$

$\rightarrow$ order 2 multiplicative interactions: $[\mathbf{h}_t]_i = g\left(\sum_{j,k} \mathcal{W}_{ijk}[\mathbf{x}_t]_j[\mathbf{h}_{t-1}]_k\right)$.

Tensor Decomposition Techniques

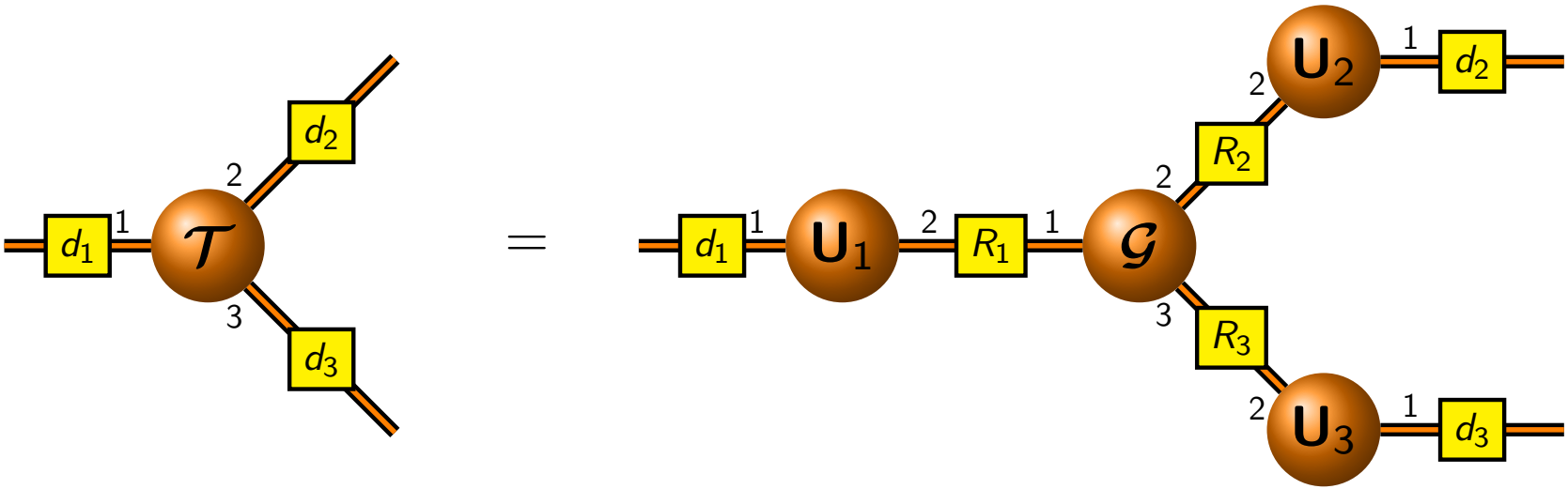
- Matrix Decomposition:



$\Rightarrow$ Rank of $\mathbf{M}$: smallest R such that $\mathbf{M} = \mathbf{UV}$ (with $\mathbf{U} \in \mathbb{R}^{m \times R}$, $\mathbf{V} \in \mathbb{R}^{R \times n}$).

Tensor Decomposition Techniques

- Tucker decomposition [Tucker, 1966 / Hitchcock, 1927]:

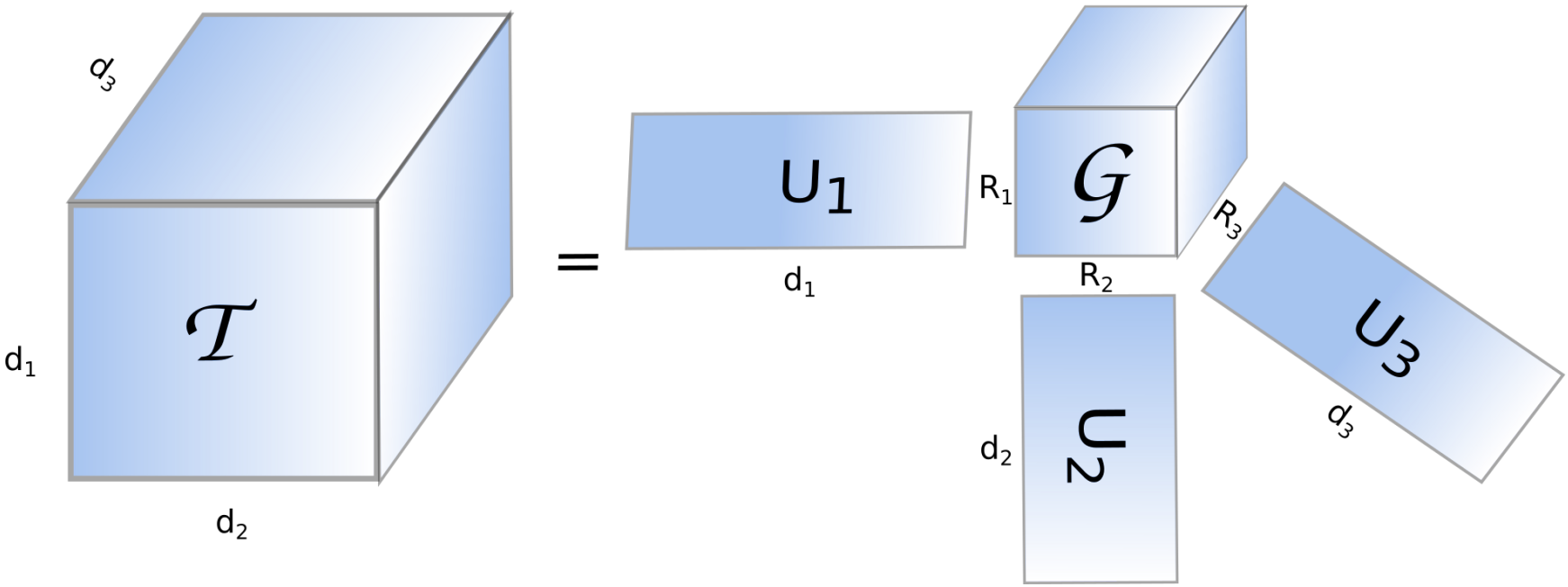


$\Rightarrow$ Multilinear rank of $\mathcal{T}$: smallest (R_1, R_2, R_3) such that

$$\mathcal{T} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$$

Tensor Decomposition Techniques

- Tucker decomposition [Tucker, 1966 / Hitchcock, 1927]:



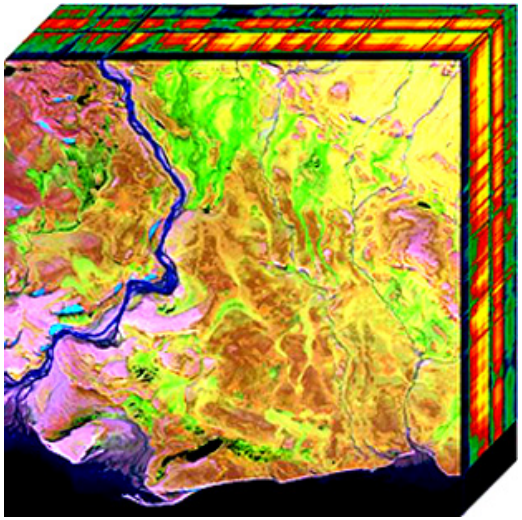
$\Rightarrow$ Multilinear rank of $\mathcal{T}$: smallest (R_1, R_2, R_3) such that

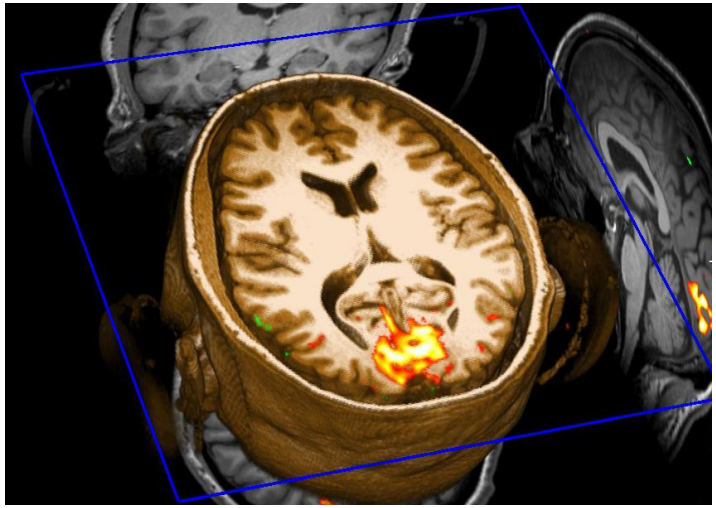
$$\mathcal{T} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$$

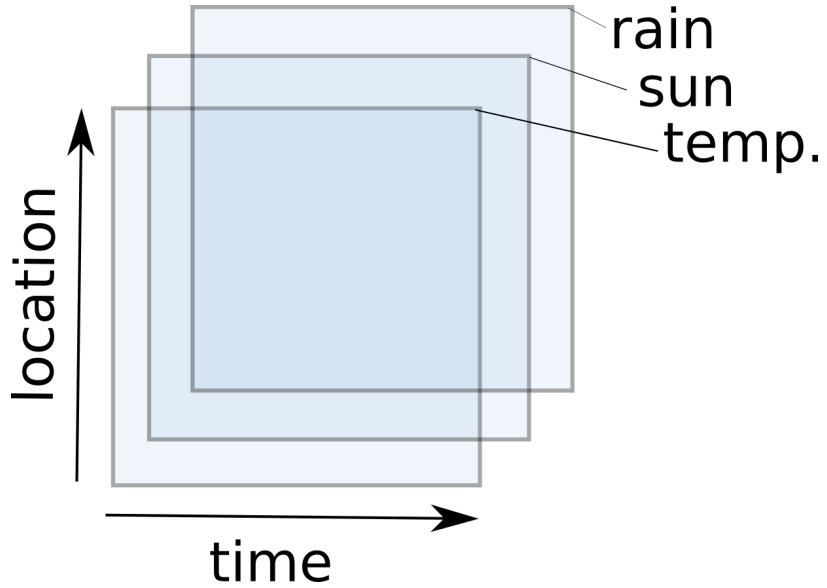
Outline

Tensor Structured Data

- Data with tensor structure: EEG, hyperspectral images, videos, ...







Problem

Learn $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1 \times \cdots \times d_p}$ from $\{(\mathbf{x}^{(n)}, \boldsymbol{\mathcal{Y}}^{(n)})\}_{n=1}^{N}$ where $\boldsymbol{\mathcal{Y}}^{(n)} \simeq f(\mathbf{x}^{(n)})$.

Problem

Learn $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1 \times \cdots \times d_p}$ from $\{(\mathbf{x}^{(n)}, \mathcal{Y}^{(n)})\}_{n=1}^{N}$ where $\mathcal{Y}^{(n)} \simeq f(\mathbf{x}^{(n)})$.

- Multilinear Multitask Learning [Romera-Paredes et al., 2013]

$$f(\mathbf{x}) \in \mathbb{R}^{(\text{Restaurant Critics}) \times (\text{Evaluation Criteria})}$$

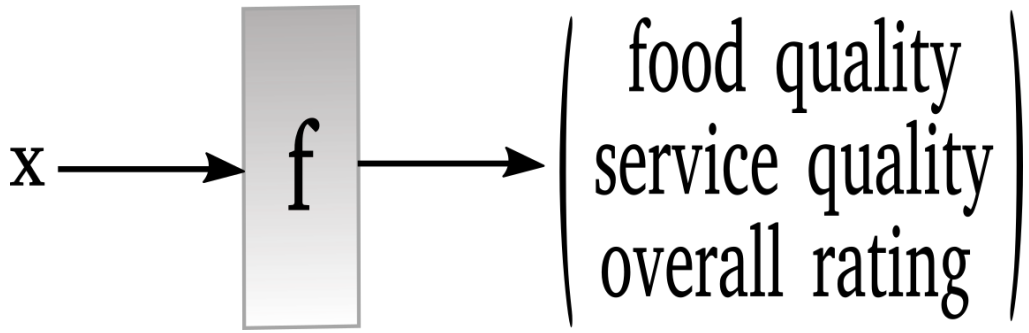
Rest. 1	Critic 1	Critic 2	Critic 3
food quality	5	3	6
service quality	7	8	6.5
overall rating	5	6.5	4

Rest. 2	Critic 1	Critic 2	Critic 3
food quality	7	8	6
service quality	8.5	9	9
overall rating	8	9.5	7

...

Multivariate Regression

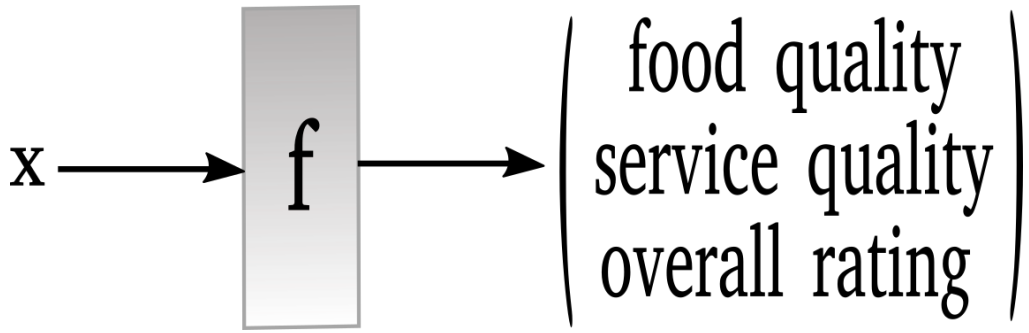
Learn $f : \mathbb{R}^d \to \mathbb{R}^p$ from samples $\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ where $\mathbf{y}^{(n)} \simeq f(\mathbf{x}^{(n)})$.



- Linear model: $f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ $\qquad (\mathbf{W} \in \mathbb{R}^{d \times p})$

Multivariate Regression

Learn $f : \mathbb{R}^d \to \mathbb{R}^p$ from samples $\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ where $\mathbf{y}^{(n)} \simeq f(\mathbf{x}^{(n)})$.

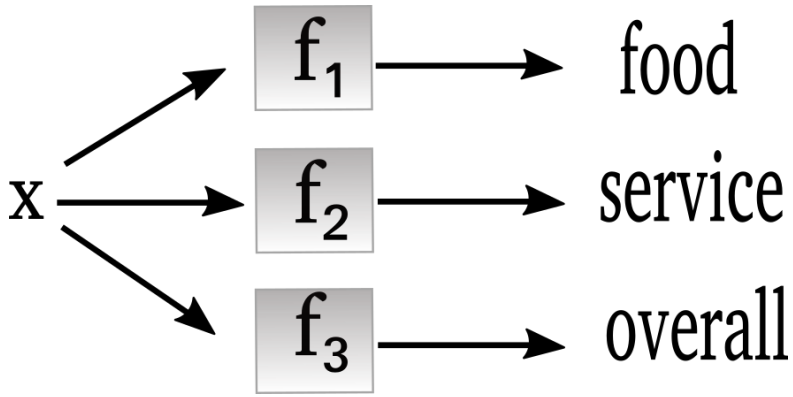


- Linear model: $f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ $\qquad (\mathbf{W} \in \mathbb{R}^{d \times p})$
- Ordinary Least Squares

$$\hat{\mathbf{W}} = \underset{\mathbf{W} \in \mathbb{R}^{d \times p}}{\arg\min} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 \qquad (\mathbf{X} \in \mathbb{R}^{N \times d}, \mathbf{Y} \in \mathbb{R}^{N \times p})$$

Multivariate Regression

Learn $f : \mathbb{R}^d \to \mathbb{R}^p$ from samples $\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ where $\mathbf{y}^{(n)} \simeq f(\mathbf{x}^{(n)})$.



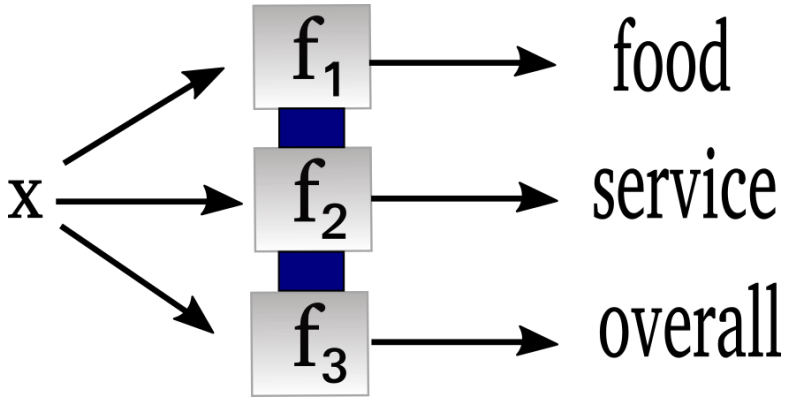
- Linear model: $f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ $\qquad (\mathbf{W} \in \mathbb{R}^{d \times p})$
- Ordinary Least Squares

$$\hat{\mathbf{W}} = \underset{\mathbf{W} \in \mathbb{R}^{d \times p}}{\arg\min} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 \qquad (\mathbf{X} \in \mathbb{R}^{N \times d}, \mathbf{Y} \in \mathbb{R}^{N \times p})$$

$\Rightarrow$ Equivalent to perform p independent linear regressions!
How can we capture linear dependencies in the output?

Multivariate Regression

Learn $f : \mathbb{R}^d \to \mathbb{R}^p$ from samples $\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ where $\mathbf{y}^{(n)} \simeq f(\mathbf{x}^{(n)})$.



- Linear model: $f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ $\qquad (\mathbf{W} \in \mathbb{R}^{d \times p})$
- Ordinary Least Squares

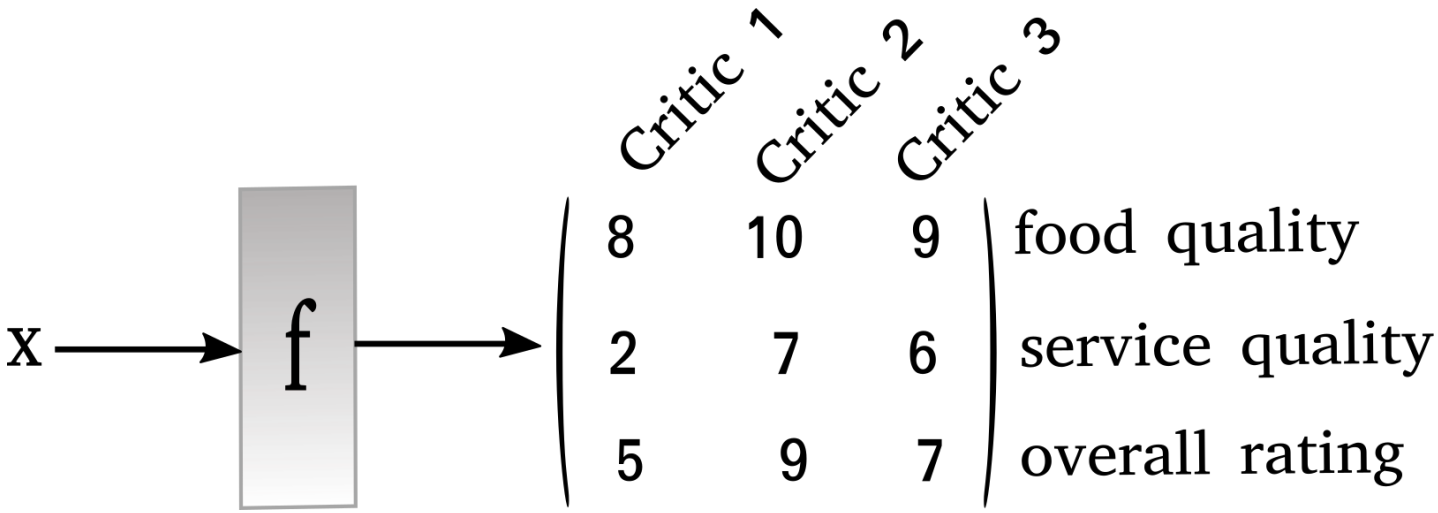
$$\hat{\mathbf{W}} = \underset{\mathbf{W} \in \mathbb{R}^{d \times p}}{\arg\min} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 \qquad (\mathbf{X} \in \mathbb{R}^{N \times d}, \mathbf{Y} \in \mathbb{R}^{N \times p})$$

- Reduced Rank Regression (Izenman, 1975)

$$\hat{\mathbf{W}} = \underset{\mathbf{W} \in \mathbb{R}^{d \times p}}{\arg\min} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 \quad \text{s.t. } \mathrm{rank}(\mathbf{W}) \leq R$$

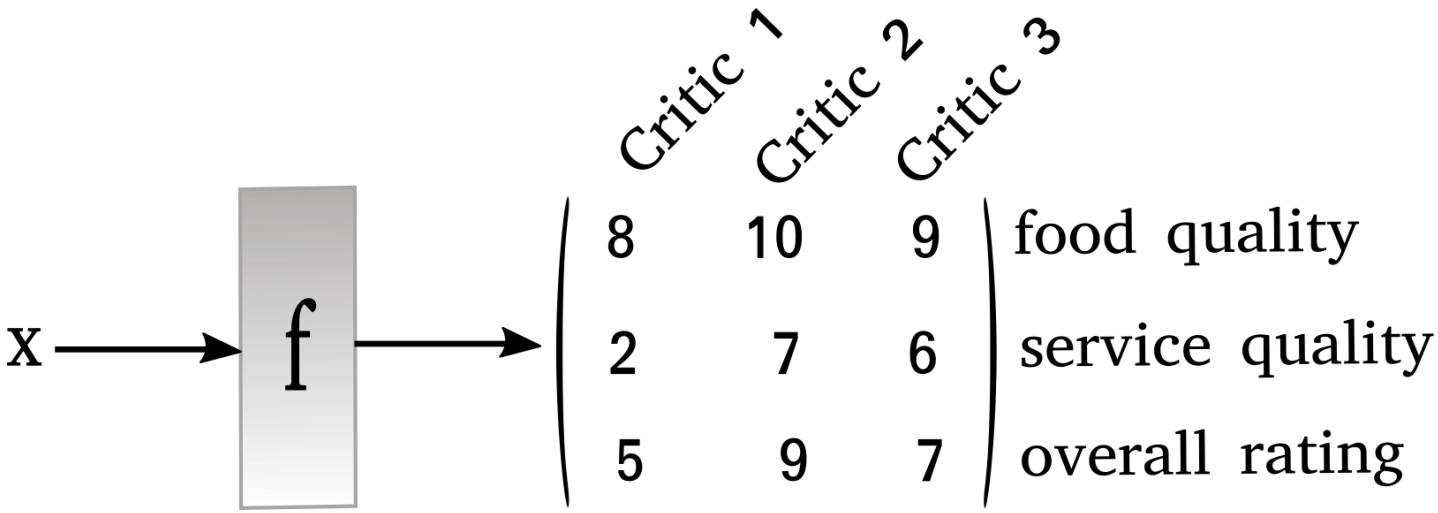
Tensor-valued Regression

Learn $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1 \times d_2}$ from $\{(\mathbf{x}^{(n)}, \mathcal{Y}^{(n)})\}_{n=1}^{N}$ where $\mathcal{Y}^{(n)} \simeq f(\mathbf{x}^{(n)})$.



Tensor-valued Regression

Learn $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1 \times d_2}$ from $\{(\mathbf{x}^{(n)}, \mathcal{Y}^{(n)})\}_{n=1}^{N}$ where $\mathcal{Y}^{(n)} \simeq f(\mathbf{x}^{(n)})$.

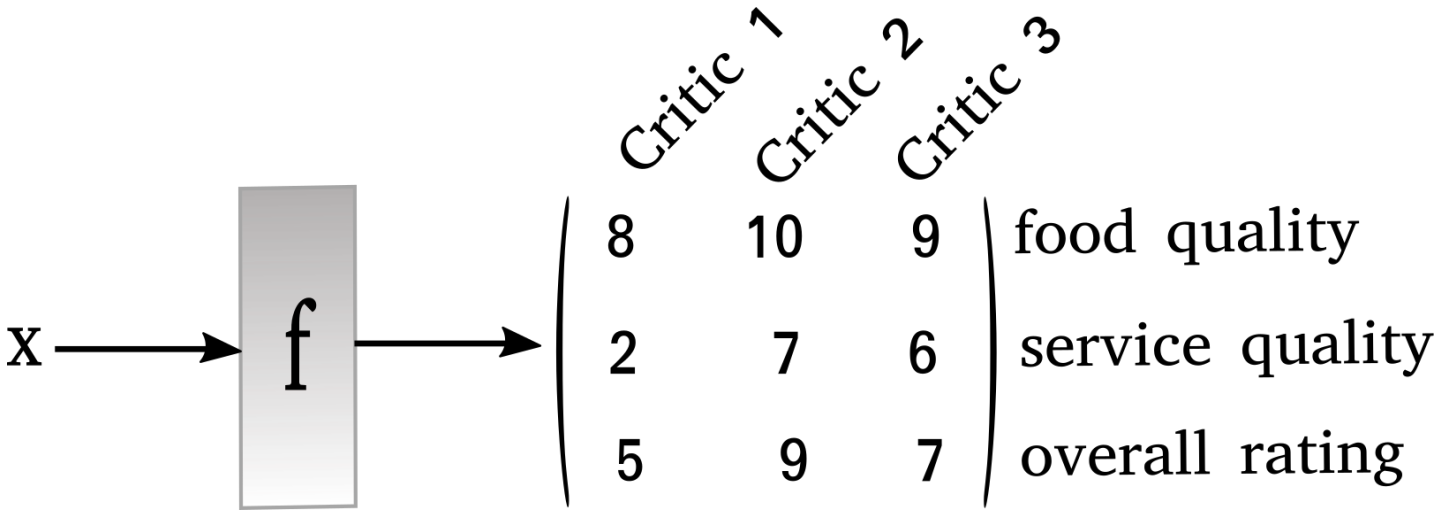


- Vectorize outputs and use reduced rank regression?

→ Need to capture higher order dependencies: multilinear rank constraint.

Tensor-valued Regression [**GR**, H. Kadri, NIPS'16]

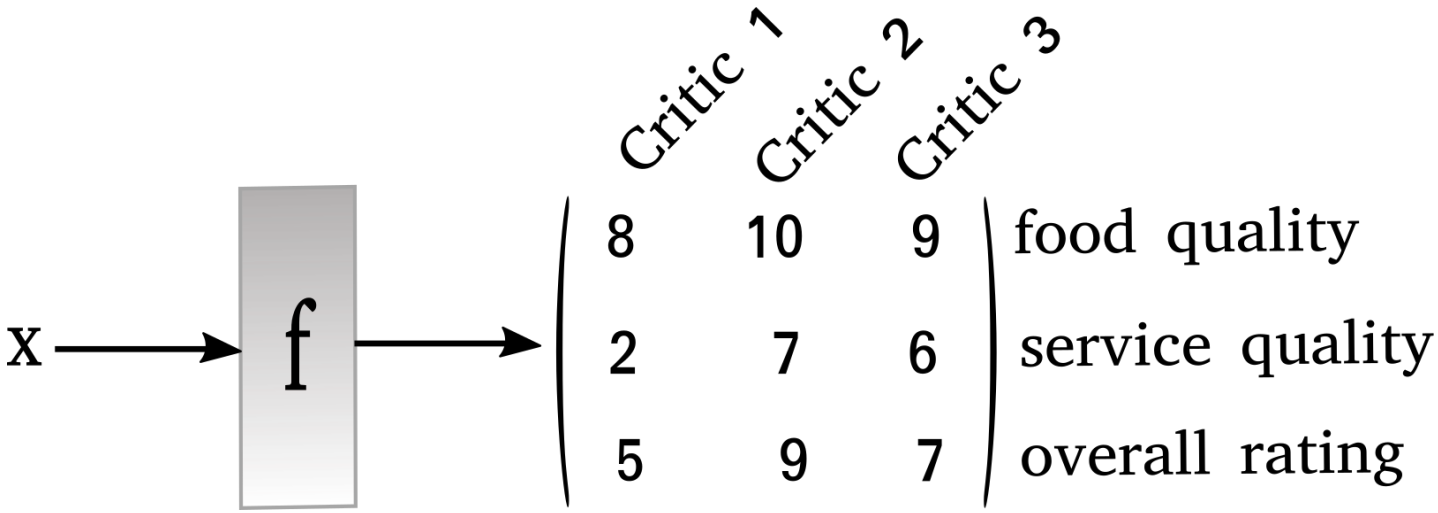
Learn $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1 \times d_2}$ from $\{(\mathbf{x}^{(n)}, \mathcal{Y}^{(n)})\}_{n=1}^{N}$ where $\mathcal{Y}^{(n)} \simeq f(\mathbf{x}^{(n)})$.



- Linear model: $f(\mathbf{x}) = \mathcal{W} \times_1 \mathbf{x}$ $\qquad (\mathcal{W} \in \mathbb{R}^{d_0 \times d_1 \times d_2})$

Tensor-valued Regression [**GR**, H. Kadri, NIPS'16]

Learn $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1 \times d_2}$ from $\{(\mathbf{x}^{(n)}, \mathcal{Y}^{(n)})\}_{n=1}^{N}$ where $\mathcal{Y}^{(n)} \simeq f(\mathbf{x}^{(n)})$.



- Linear model: $f(\mathbf{x}) = \mathcal{W} \times_1 \mathbf{x}$ $\qquad (\mathcal{W} \in \mathbb{R}^{d_0 \times d_1 \times d_2})$
- **Low-Rank Regression for Tensor Structured Response**

$$\underset{\mathcal{W} \in \mathbb{R}^{d_0 \times d_1 \times d_2}}{\arg\min} \; \|\mathcal{W} \times_1 \mathbf{X} - \mathcal{Y}\|_F^2 \quad \text{s.t. } \operatorname{rank}_{ml}(\mathcal{W}) \leq (R_0, R_1, R_2)$$

Solving the Minimization Problem [**GR**, H. Kadri, NIPS'16]

Problem

$$\underset{\mathcal{W} \in \mathbb{R}^{d_0 \times d_1 \times d_2}}{\arg\min} \|\mathcal{W} \times_1 \mathbf{X} - \mathcal{Y}\|_F^2 \qquad s.t. \ \operatorname{rank}_{ml}(\mathcal{W}) \leq (R_0, R_1, R_2)$$

Solving the Minimization Problem [**GR**, H. Kadri, NIPS'16]

Problem

$$\underset{\mathcal{W} \in \mathbb{R}^{d_0 \times d_1 \times d_2}}{\arg\min} \|\mathcal{W} \times_1 \mathbf{X} - \mathcal{Y}\|_F^2 \qquad s.t.\ \operatorname{rank}_{ml}(\mathcal{W}) \leq (R_0, R_1, R_2)$$

is equivalent to:

Problem

$$\underset{\mathbf{U}_0, \mathbf{U}_1, \mathbf{U}_2}{\arg\min} \|\mathcal{Y} \times_1 \mathbf{\Pi}_0 \times_2 \mathbf{\Pi}_1 \times_3 \mathbf{\Pi}_2 - \mathcal{Y}\|_F^2 \qquad w.r.t.\ \mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}$$

$$s.t.\ \mathbf{U}_i^\top \mathbf{U}_i = \mathbf{I}\ for\ 0 \leq i \leq 2, \mathbf{\Pi}_0 = \mathbf{X}\mathbf{U}_0 \left(\mathbf{U}_0^\top \mathbf{X}^\top \mathbf{X} \mathbf{U}_0\right)^{-1} \mathbf{U}_0^\top \mathbf{X}^T,\ \mathbf{\Pi}_i = \mathbf{U}_i \mathbf{U}_i^\top\ for\ i = 1, 2$$

- Find 3 low-dimensional subspaces U_0, U_1, U_2 such that projecting $\mathcal{Y}$ along the corresponding modes is close to $\mathcal{Y}$.
- NP-hard... Solve $\arg\min_{U_i} \|\mathcal{Y} \times_{i+1} \mathbf{\Pi}_i - \mathcal{Y}\|_F^2$ instead.

Theoretical Guarantees [**GR**, H. Kadri, NIPS'16]

Problem

$$(*) \quad \underset{\mathcal{W} \in \mathbb{R}^{d_0 \times d_1 \times d_2}}{\arg\min} \|\mathcal{W} \times_1 \mathbf{X} - \mathcal{Y}\|_F^2 \qquad s.t.\ \mathrm{rank}_{ml}(\mathcal{W}) \leq (R_0, R_1, R_2)$$

- HOLRR is an order 3 approximation algorithm:

Theorem

Let $\mathcal{W}^*$ be a solution of $(*)$ and let $\hat{\mathcal{W}}$ be the regression tensor returned by HOLRR. Then,

$$\|\hat{\mathcal{W}} \times_1 \mathbf{X} - \mathcal{Y}\|_F^2 \leq 3\|\mathcal{W}^* \times_1 \mathbf{X} - \mathcal{Y}\|_F^2.$$

Theoretical Guarantees (cont'd) [**GR**, H. Kadri, NIPS'16]

Problem

$$(*) \quad \underset{\mathcal{W} \in \mathbb{R}^{d_0 \times d_1 \times d_2}}{\arg\min} \; \|\mathcal{W} \times_1 \mathbf{X} - \mathcal{Y}\|_F^2 \qquad s.t. \; \mathrm{rank}_{ml}(\mathcal{W}) \leq (R_0, R_1, R_2)$$

- HOLRR is statistically consistent

Theoretical Guarantees (cont'd) [**GR**, H. Kadri, NIPS'16]

Problem

$$(*) \quad \underset{\mathcal{W} \in \mathbb{R}^{d_0 \times d_1 \times d_2}}{\arg\min} \; \|\mathcal{W} \times_1 \mathbf{X} - \mathcal{Y}\|_F^2 \qquad s.t. \; \mathrm{rank}_{ml}(\mathcal{W}) \leq (R_0, R_1, R_2)$$

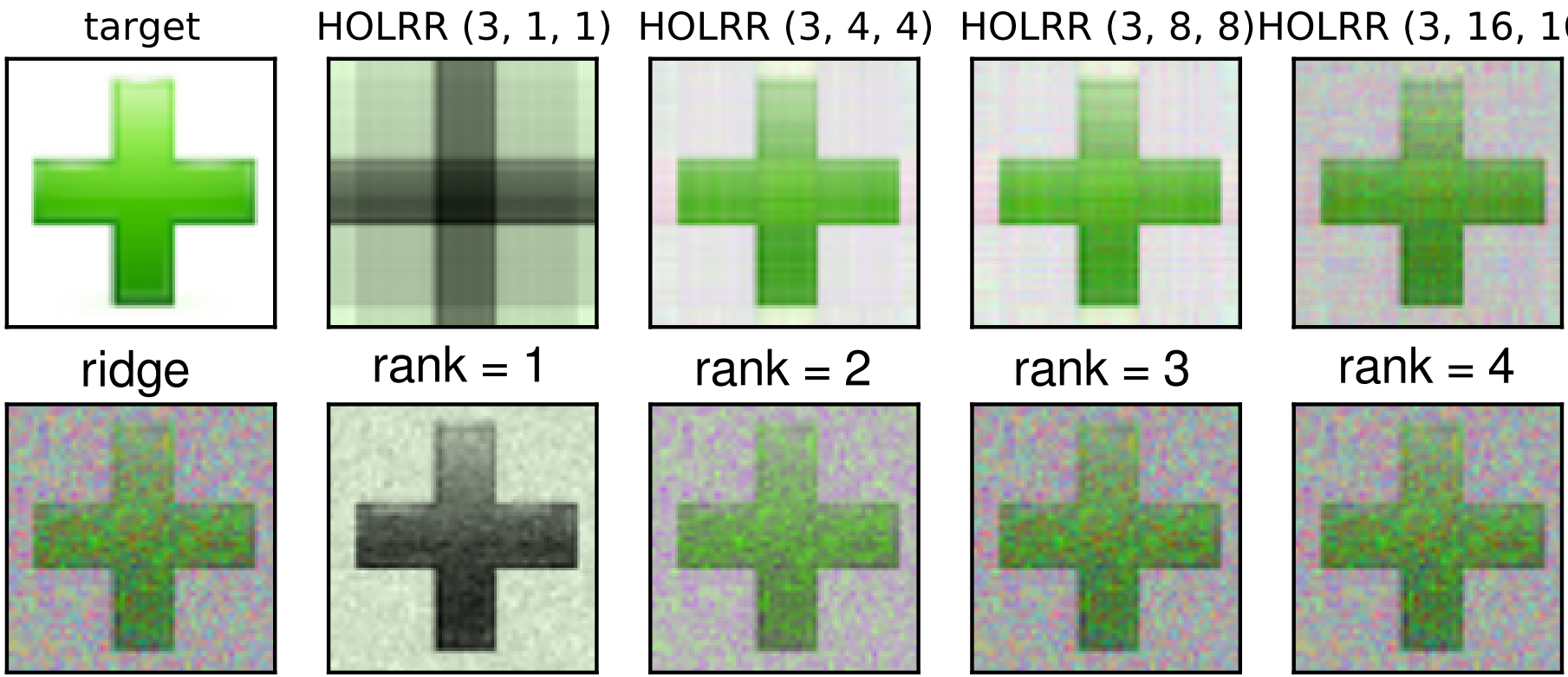
- HOLRR is statistically consistent
- Generalization bound for the class of functions

$$\mathcal{F}_{ml} = \{\mathbf{x} \mapsto \mathcal{W} \times_1 \mathbf{x} \; : \; \mathrm{rank}_{ml}(\mathcal{W}) = (R_0, R_1, R_2)\}\,.$$

$\rightarrow$ VC-dimension of $\mathcal{F}_{ml}$ is in $\mathcal{O}\left(\sqrt{R_0 R_1 R_2 \log(d_1 d_2 d_3)}\right)$ instead of $\mathcal{O}\left(\sqrt{d_1 d_2 d_3}\right)$.

Image Reconstruction from Noisy Measurements

- $\mathcal{W} \in \mathbb{R}^{3\times 50\times 50}$ is an RGB image.
- Data is generated by $\mathbf{Y} = \mathcal{W} \times_1 \mathbf{x} + \boldsymbol{\xi}$ where $\mathbf{x} \sim \mathcal{N}(0, \mathbf{I})$ and $\boldsymbol{\xi}_{ij} \sim \mathcal{N}(0, 1)$.
- Training set of size 200.



Experiments on Real Data

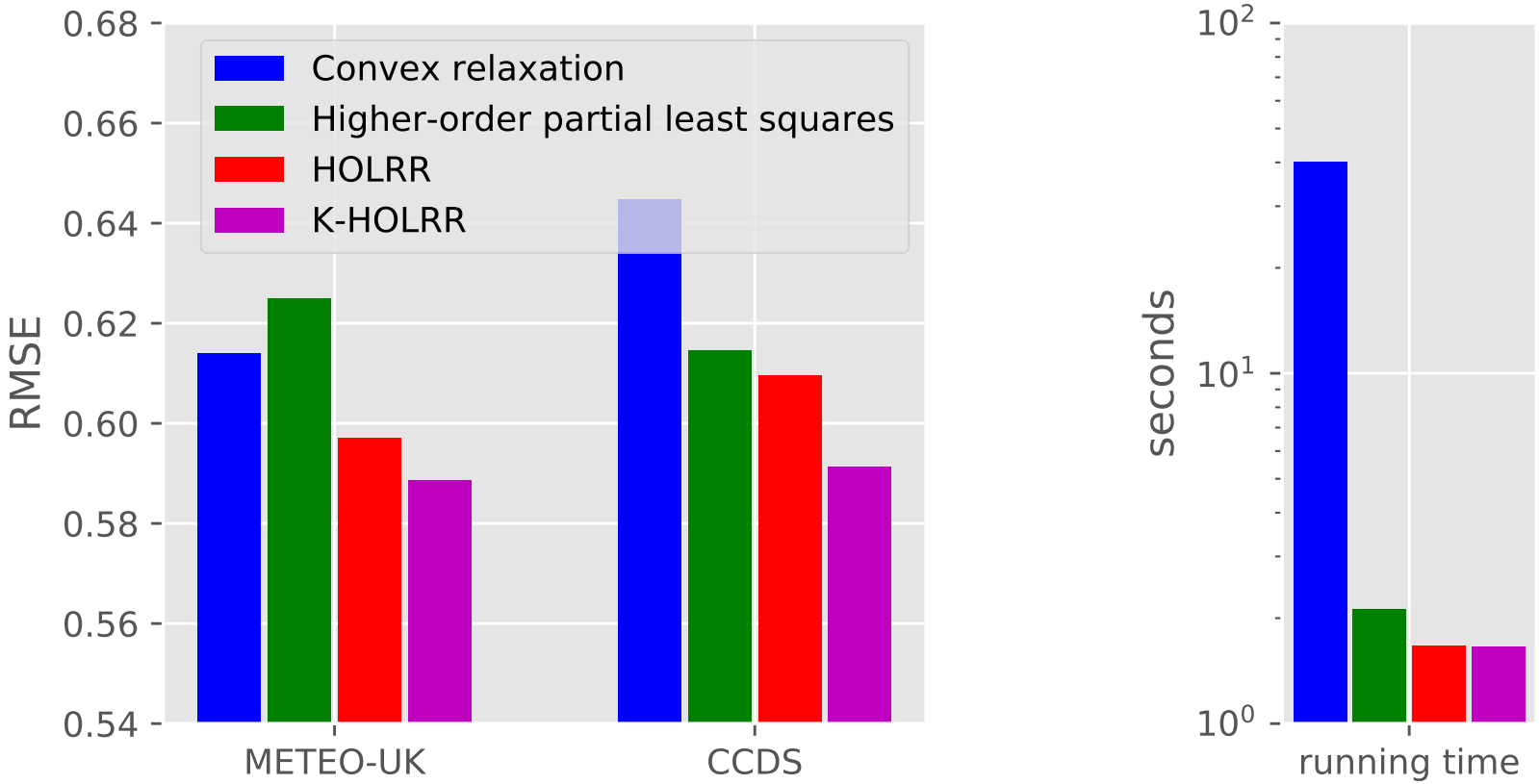


Figure: Task: predict meteorological variables in different locations from their values in the preceding 3 time steps (average over 10 runs). Output is of size 17×125 for CCDS and $5 \times 16 \times 5$ for METEO-UK.

Discussion

- Multilinear extension of low/reduced-rank regression.
- Approximation algorithm rather than convex relaxation.
- Kernel extension $\rightarrow$ nonlinear setting.
- Fast, efficient, theoretical guarantees.

Discussion

- Multilinear extension of low/reduced-rank regression.
- Approximation algorithm rather than convex relaxation.
- Kernel extension $\rightarrow$ nonlinear setting.
- Fast, efficient, theoretical guarantees.

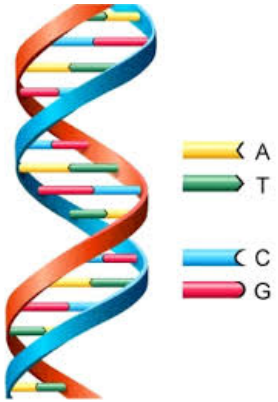
Leverage the tensor structure $\Rightarrow \begin{cases} \text{faster algorithms} \\ \text{better sample efficiency} \end{cases}$

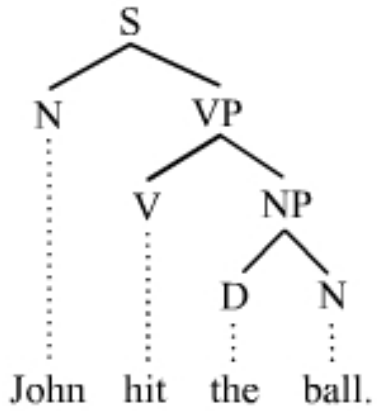
Outline

1. Preliminaries: Tensors and Multilinear Algebra
2. Low-Rank Regression with Tensor Responses
3. Weighted Automata for Learning with Structured Data
   - Weighted Automata (WA) and Spectral Learning
   - Connections betweens WAs and RNNs
   - Beyond Strings and Trees: Graph Weighted Models
4. Conclusion and Future Lines of Research

Problem Statement

- How can one learn with structured objects such as strings and trees?

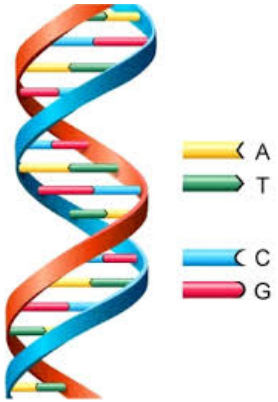


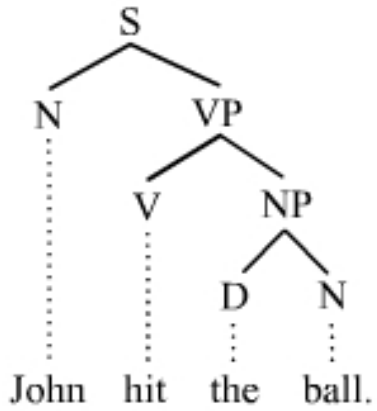


- Intersection of Theoretical Computer Science and Machine Learning...

Problem Statement

- How can one learn with structured objects such as strings and trees?



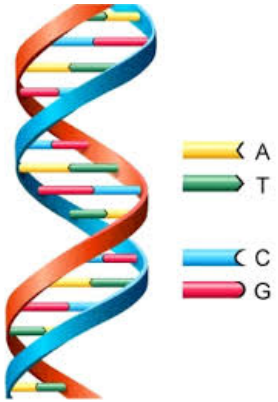


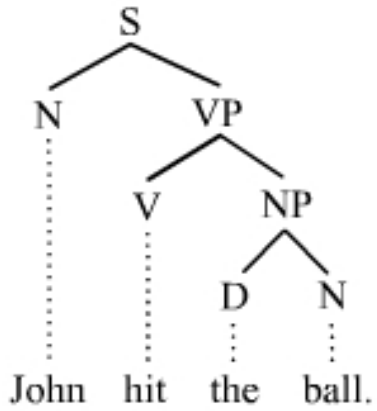
- Intersection of Theoretical Computer Science and Machine Learning...

$\rightarrow$ Weighted Automata: robust model to represent functions defined over structured objects (for example probability distributions).

Problem Statement

- How can one learn with structured objects such as strings and trees?





- Intersection of Theoretical Computer Science and Machine Learning...

→ Weighted Automata: robust model to represent functions defined over structured objects (for example probability distributions).

- String Weighted Automata (WA): generalize *Hidden Markov Models*, *Predictive State Representations* and closely related to *RNNs*.

String Weighted Automata (WA)

- Σ a finite alphabet (e.g. $\{a, b\}$), Σ^* strings on Σ (e.g. $abba$)
- A WA computes a function $f : \Sigma^* \to \mathbb{R}$

String Weighted Automata (WA)

- Σ a finite alphabet (e.g. $\{a, b\}$), Σ^* strings on Σ (e.g. $abba$)
- A WA computes a function $f : \Sigma^* \to \mathbb{R}$
- Weighted Automaton: $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma \in \Sigma}, \boldsymbol{\omega})$ where

$\boldsymbol{\alpha} \in \mathbb{R}^n$ initial weights vector
$\boldsymbol{\omega} \in \mathbb{R}^n$ final weights vector
$\mathbf{A}^\sigma \in \mathbb{R}^{n \times n}$ transition weights matrix for each $\sigma \in \Sigma$

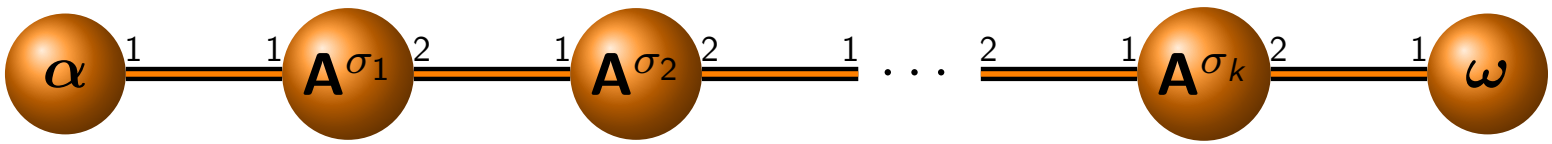
String Weighted Automata (WA)

- Σ a finite alphabet (e.g. $\{a, b\}$), Σ^* strings on Σ (e.g. $abba$)
- A WA computes a function $f : \Sigma^* \to \mathbb{R}$
- Weighted Automaton: $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$ where

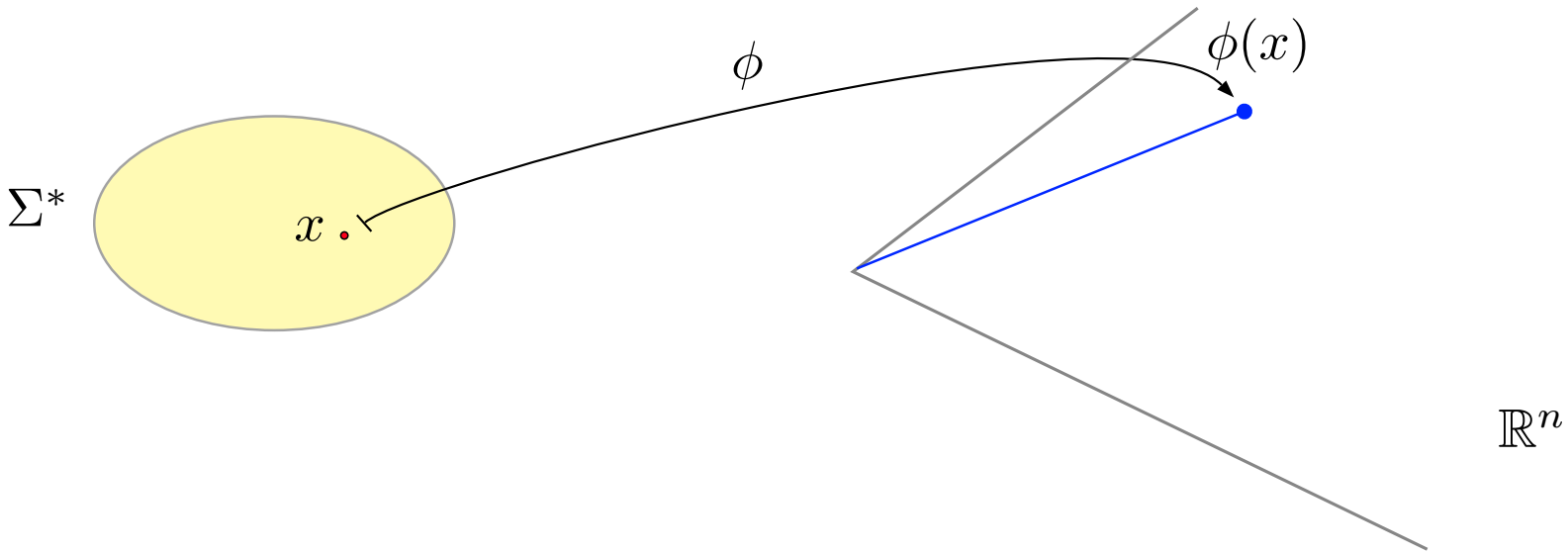
$$\begin{aligned} \boldsymbol{\alpha} &\in \mathbb{R}^n && \text{initial weights vector} \\ \boldsymbol{\omega} &\in \mathbb{R}^n && \text{final weights vector} \\ \mathbf{A}^\sigma &\in \mathbb{R}^{n\times n} && \text{transition weights matrix for each } \sigma \in \Sigma \end{aligned}$$

- A computes a function $f_A : \Sigma^* \to \mathbb{R}$ defined by

$$f_A(\sigma_1\sigma_2\cdots\sigma_k) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}\mathbf{A}^{\sigma_2}\cdots\mathbf{A}^{\sigma_k}\boldsymbol{\omega}$$

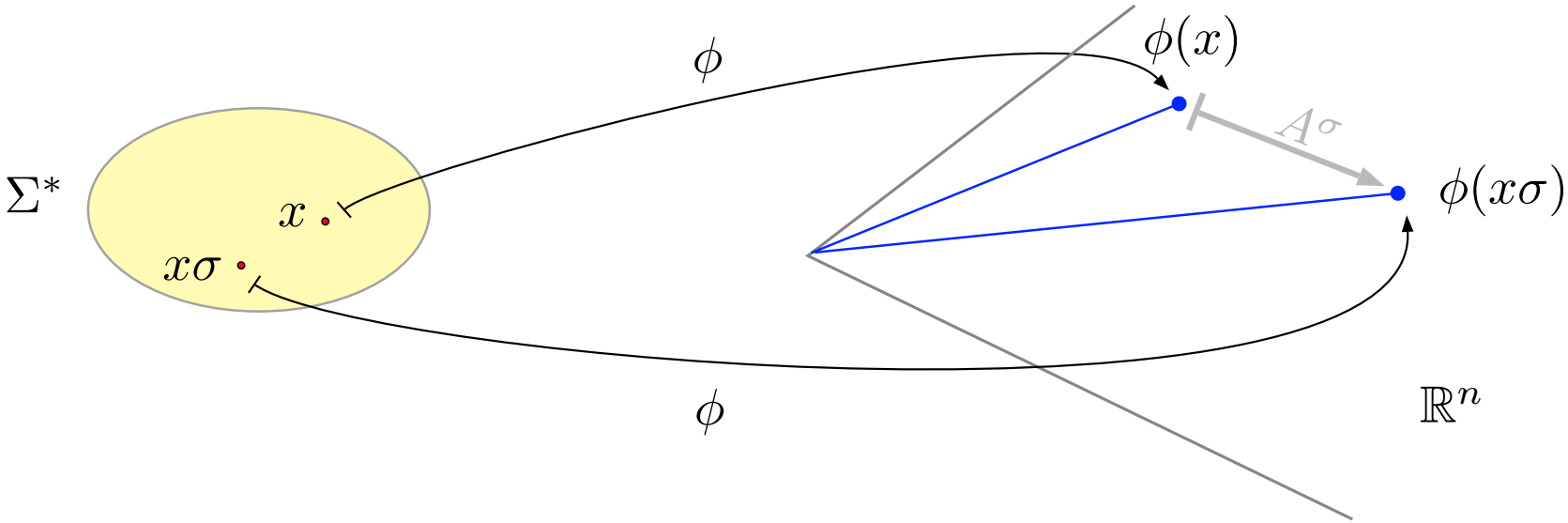


Weighted Automata and Representation Learning



- A WA induces a mapping $\phi : \Sigma^* \to \mathbb{R}^n$ ($\sim$ word embedding)

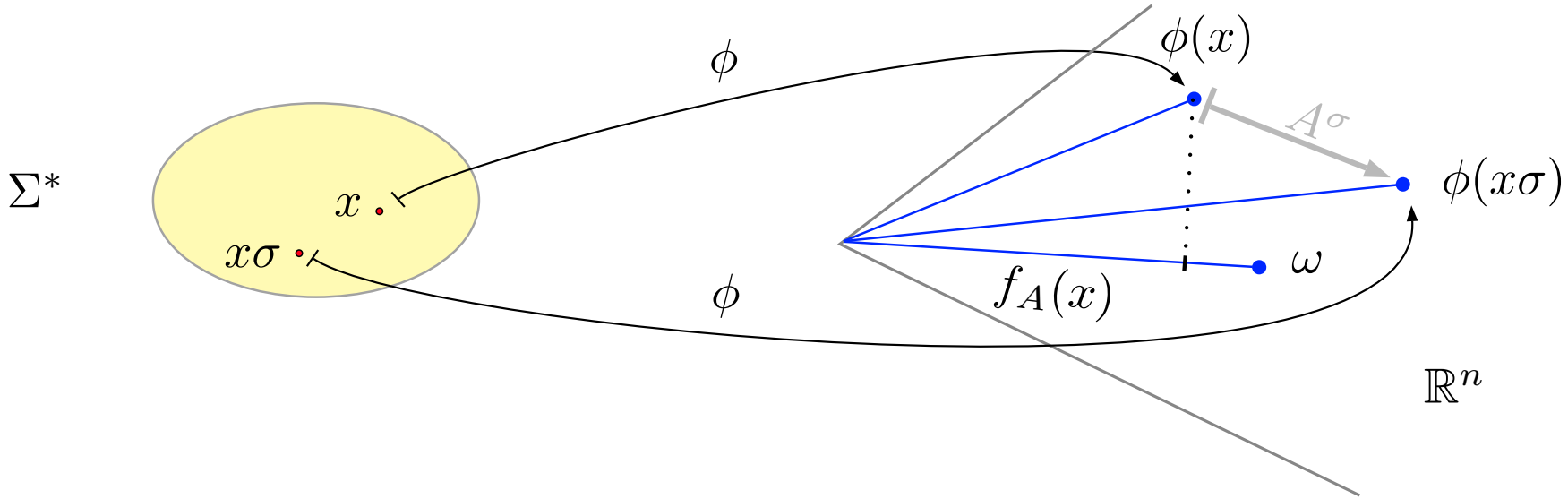
Weighted Automata and Representation Learning



- A WA induces a mapping $\phi : \Sigma^* \to \mathbb{R}^n$ ($\sim$ word embedding)
- The mapping ϕ is compositional:

$$\phi(\lambda) = \boldsymbol{\alpha}^\top, \;\; \phi(\sigma_1) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}, \;\; \phi(\sigma_1\sigma_2) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}\mathbf{A}^{\sigma_2} = \phi(\sigma_1)\mathbf{A}^{\sigma_2}, \;\; \ldots$$

Weighted Automata and Representation Learning



- A WA induces a mapping $\phi : \Sigma^* \to \mathbb{R}^n$ ($\sim$ word embedding)
- The mapping ϕ is compositional:

$$\phi(\lambda) = \boldsymbol{\alpha}^\top, \;\; \phi(\sigma_1) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}, \;\; \phi(\sigma_1\sigma_2) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1}\mathbf{A}^{\sigma_2} = \phi(\sigma_1)\mathbf{A}^{\sigma_2}, \;\; \ldots$$

- The output $f_A(x) = \langle \phi(x), \boldsymbol{\omega} \rangle$ is linear in $\phi(x)$.

Spectral Learning of Weighted Automata

Spectral Learning of Weighted Automata

- $\mathbf{H}_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$: Hankel matrix of $f : \Sigma^* \to \mathbb{R}$

  Definition: prefix p, suffix $s \Rightarrow (\mathbf{H}_f)_{p,s} = f(ps)$

Spectral Learning of Weighted Automata

- $\mathbf{H}_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$: Hankel matrix of $f : \Sigma^* \to \mathbb{R}$

  Definition: prefix p, suffix $s \Rightarrow (\mathbf{H}_f)_{p,s} = f(ps)$

- Fundamental theorem [Carlyle and Paz, 1971; Fliess 1974]:

  $\text{rank}(\mathbf{H}_f) < \infty \iff f$ can be computed by a WA

Spectral Learning of Weighted Automata

- $\mathbf{H}_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$: Hankel matrix of $f : \Sigma^* \to \mathbb{R}$

  Definition: prefix p, suffix $s \Rightarrow (\mathbf{H}_f)_{p,s} = f(ps)$

- Fundamental theorem [Carlyle and Paz, 1971; Fliess 1974]:

  $\text{rank}(\mathbf{H}_f) < \infty \iff f$ can be computed by a WA

- Proof is constructive $\Rightarrow$ Spectral Learning of WA:
  1. Estimate a sub-block of $\mathbf{H}_f$ from training data
  2. Low rank decomposition $\mathbf{H} \simeq \mathbf{PS}$
  3. Build WA $\hat{A}$ using $\mathbf{H}, \mathbf{P}$ and $\mathbf{S}$.

$\rightarrow$ Efficient and consistent learning algorithms for weighted automata [Hsu et al., 2009; Bailly et al. 2009; Balle et al., 2014, ...].

Connections betweens WAs and RNNs

Weighted Automata and Recurrent Neural Networks

- Recall that the hidden state of a second-order RNN (2-RNN) is computed by

$$\mathbf{h}_t = g(\mathcal{W} \times_2 \mathbf{x}_t \times_3 \mathbf{h}_{t-1})$$

Weighted Automata and Recurrent Neural Networks

- Recall that the hidden state of a second-order RNN (2-RNN) is computed by

$$\mathbf{h}_t = g(\mathcal{W} \times_2 \mathbf{x}_t \times_3 \mathbf{h}_{t-1})$$

- Similarly, the feature map of a WA $(\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma \in \Sigma}, \boldsymbol{\omega})$ can be written as

$$\phi(x\sigma) = \mathcal{A} \times_2 \mathbf{e}_\sigma \times_3 \phi(x)$$

where

  - $\mathcal{A} \in \mathbb{R}^{n \times \Sigma \times n}$ is defined by $\mathcal{A}_{:,\sigma,:} = (\mathbf{A}^\sigma)^\top$,
  - $\mathbf{e}_\sigma$ is the one-hot encoding of σ.

Spectral Learning of Linear 2-RNNs [**GR**, T.Li, D. Precup]

- For sequences of discrete symbols, WAs and second-order RNNs with linear activation functions are equivalent!

Spectral Learning of Linear 2-RNNs [**GR**, T.Li, D. Precup]

- For sequences of discrete symbols, WAs and second-order RNNs with linear activation functions are equivalent!

$\Rightarrow$ For the discrete case, the spectral learning algorithm is a consistent learning algorithm for linear second-order RNNs.

Spectral Learning of Linear 2-RNNs [**GR**, T.Li, D. Precup]

- For sequences of discrete symbols, WAs and second-order RNNs with linear activation functions are equivalent!

$\Rightarrow$ For the discrete case, the spectral learning algorithm is a consistent learning algorithm for linear second-order RNNs.

- What about sequences of continuous vectors?
  - $\rightarrow$ Can we extend the spectral learning algorithm to linear 2-RNNs defined over continuous vectors?

Spectral Learning of Linear 2-RNNs [**GR**, T.Li, D. Precup]

- For sequences of discrete symbols, WAs and second-order RNNs with linear activation functions are equivalent!

$\Rightarrow$ For the discrete case, the spectral learning algorithm is a consistent learning algorithm for linear second-order RNNs.

- What about sequences of continuous vectors?
  - $\rightarrow$ Can we extend the spectral learning algorithm to linear 2-RNNs defined over continuous vectors?
- YES! By leveraging multilinear properties of linear RNNs and tensor sensing techniques.

Non-Linear Weighted Automata [T.Li, **GR**, D. Precup, AISTATS'18]

- WA $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma \in \Sigma}, \boldsymbol{\omega})$: linear transition maps and linear termination function...

Non-Linear Weighted Automata [T.Li, **GR**, D. Precup, AISTATS'18]

- WA $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$: linear transition maps and linear termination function...
- Non-linear Weighted Automaton: $(\boldsymbol{\alpha}, \{G_\sigma\}_{\sigma\in\Sigma}, F)$
  - $\boldsymbol{\alpha}$ is the initial latent state
  - $G_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ are non-linear transition maps
  - $F : \mathbb{R}^n \to \mathbb{R}$ is a non-linear termination function

$$f(\sigma_1\sigma_2\cdots\sigma_k) = F(G_{\sigma_k}(\cdots G_{\sigma_2}(G_{\sigma_1}(\boldsymbol{\alpha}))\cdots)$$

($\simeq$ RNNs with one-hot encoding of the inputs)

Non-Linear Weighted Automata [T.Li, **GR**, D. Precup, AISTATS'18]

- WA $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\omega})$: linear transition maps and linear termination function...
- Non-linear Weighted Automaton: $(\boldsymbol{\alpha}, \{G_\sigma\}_{\sigma\in\Sigma}, F)$
  - $\boldsymbol{\alpha}$ is the initial latent state
  - $G_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ are non-linear transition maps
  - $F : \mathbb{R}^n \to \mathbb{R}$ is a non-linear termination function

$$f(\sigma_1\sigma_2\cdots\sigma_k) = F(G_{\sigma_k}(\cdots G_{\sigma_2}(G_{\sigma_1}(\boldsymbol{\alpha}))\cdots)$$

($\simeq$ RNNs with one-hot encoding of the inputs)

- Two-stage learning algorithm:
  - Learning ϕ: $\Sigma^* \to \mathbb{R}^n$ using an encoder-decoder network to non-linearly decompose the Hankel matrix $\mathbf{H}_f$.
  - Learning G_σ: feed-forward neural networks.

Non-Linear Weighted Automata [T.Li, **GR**, D. Precup, AISTATS'18]

- Experiments on Penn Tree Bank data: $5,987$ sentences over an alphabet of 33 symbols.
- Two evaluation metrics:

Table: Pautomac Score ($\sim$ perplexity) on test data.

Sample Size	SP	EM	RNN	NL-WA
1000	9.098	4.252	4.765	**2.937**
2000	4.995	3.723	4.6053	**2.923**
3000	4.532	3.570	4.398	**2.894**
4000	4.235	3.542	4.244	**2.880**
ALL	4.234	3.496	4.191	**2.748**

Table: Word error rate (one-step ahead prediction) on test data.

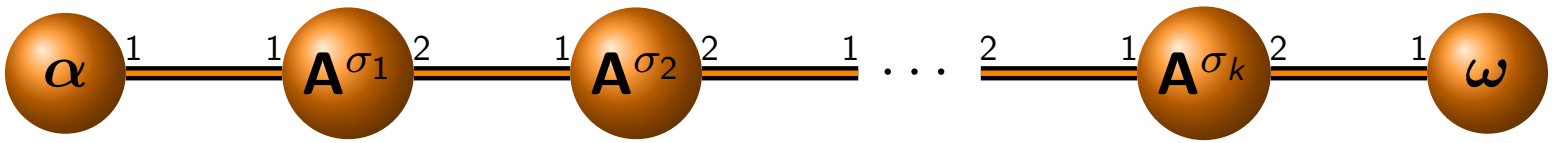
Sample Size	SP	EM	RNN	NL-WA
1000	0.8432	0.808	0.806	**0.7630**
2000	0.8342	0.793	0.788	**0.7332**
3000	0.8195	0.781	0.736	**0.7134**
4000	0.8141	0.776	**0.692**	0.6935
ALL	0.8033	0.753	**0.669**	0.6831

Beyond Strings and Trees

A Look Back on String Weighted Automata

A Weighted Automaton $A = (\boldsymbol{\alpha}, \{\mathbf{A}^{\sigma}\}_{\sigma \in \Sigma}, \boldsymbol{\omega})$ computes a function $f_A : \Sigma^* \to \mathbb{R}$ defined by

$$f_A(\sigma_1 \sigma_2 \cdots \sigma_k) = \boldsymbol{\alpha}^\top \mathbf{A}^{\sigma_1} \mathbf{A}^{\sigma_2} \cdots \mathbf{A}^{\sigma_k} \boldsymbol{\omega}$$

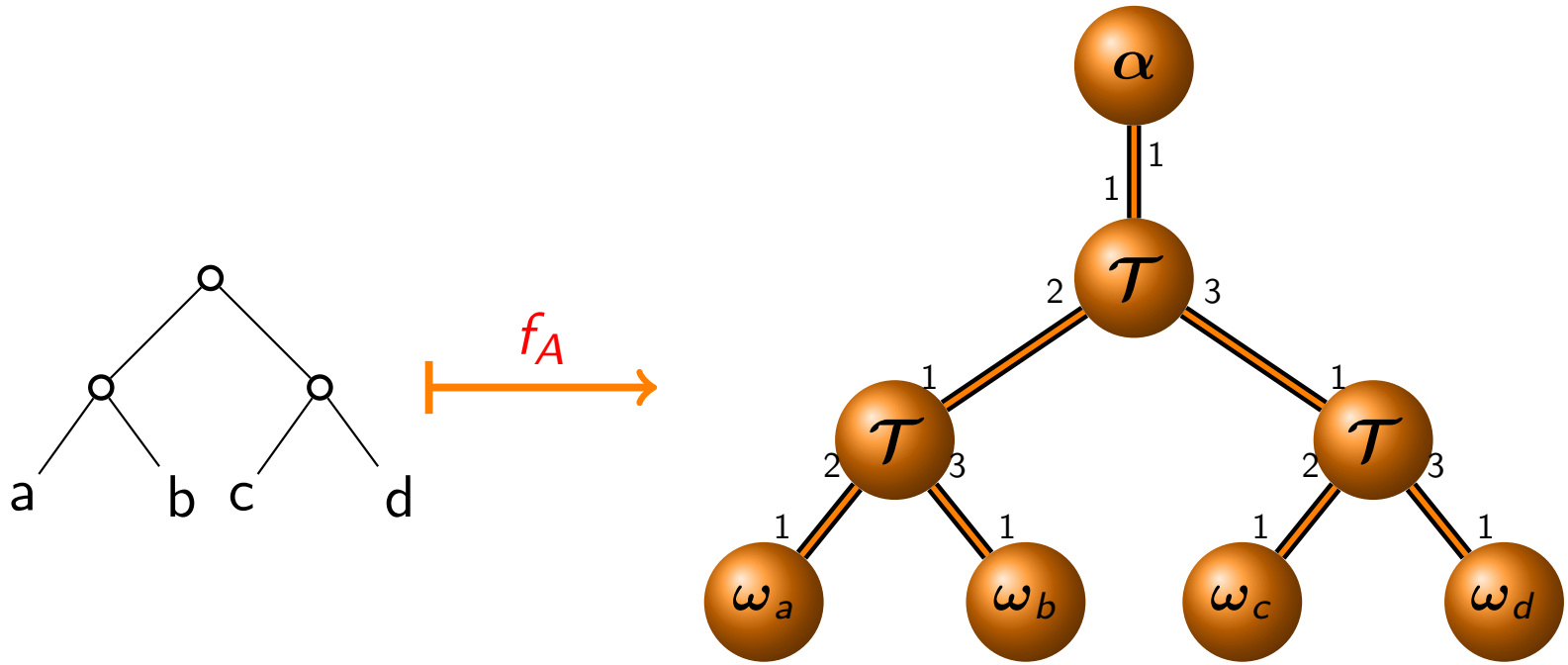


Beyond Strings: Weighted Tree Automata

- A weighted tree automaton (WTA) is a tuple $A = \langle \boldsymbol{\alpha}, \boldsymbol{\mathcal{T}}, \{\boldsymbol{\omega}_\sigma\}_{\sigma \in \Sigma} \rangle$

$$\boldsymbol{\alpha} \in \mathbb{R}^n : \text{ vector of initial weights}$$
$$\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n \times n \times n} : \text{ tensor of transition weights}$$
$$\boldsymbol{\omega}_\sigma \in \mathbb{R}^n : \text{ vector of final weights associated with } \sigma \in \Sigma$$

- A WTA computes a function $f_A : \mathfrak{T}_\Sigma \to \mathbb{R}$.



Graph Weighted Models [R. Bailly*, **GR***, F. Denis, LATA'15/JCSS'18]

- $\mathcal{F} = \{a(\cdot), h(\cdot,\cdot), g(\cdot,\cdot,\cdot)\}$

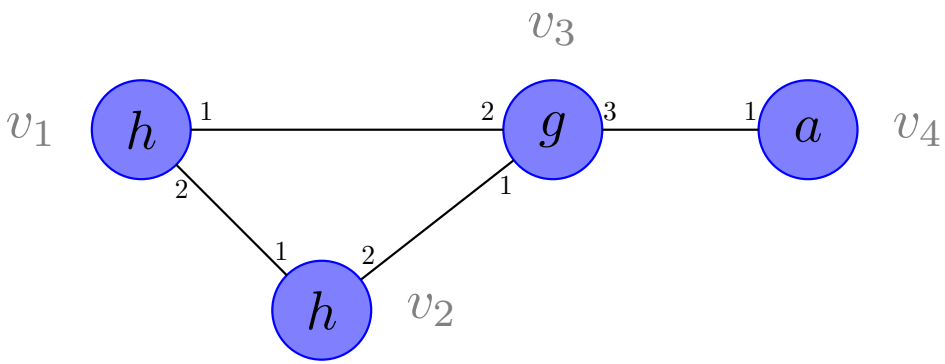


Figure: A graph on the ranked alphabet $\mathcal{F} = \{a(\cdot), h(\cdot,\cdot), g(\cdot,\cdot,\cdot)\}$.

Graph Weighted Models [R. Bailly*, **GR***, F. Denis, LATA'15/JCSS'18]

- $\mathcal{F} = \{a(\cdot), h(\cdot,\cdot), g(\cdot,\cdot,\cdot)\}$

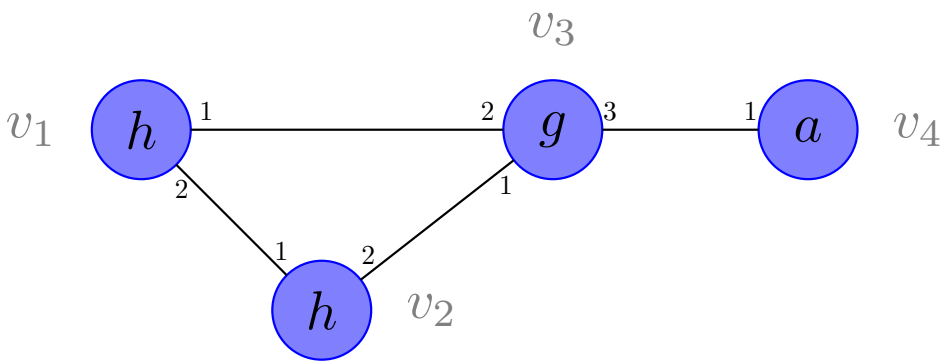
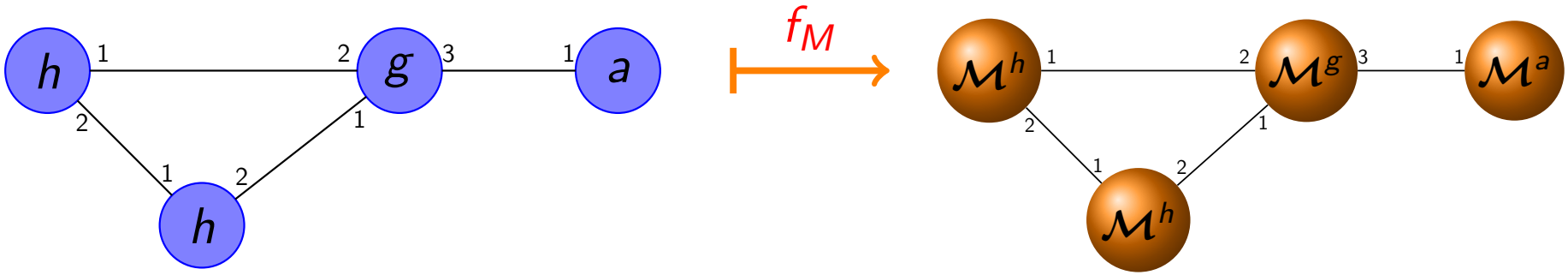


Figure: A graph on the ranked alphabet $\mathcal{F} = \{a(\cdot), h(\cdot,\cdot), g(\cdot,\cdot,\cdot)\}$.

- GWM: vector $\boldsymbol{\mathcal{M}}^a \in \mathbb{R}^n$, matrix $\boldsymbol{\mathcal{M}}^h \in \mathbb{R}^{n\times n}$, tensor $\boldsymbol{\mathcal{M}}^g \in \mathbb{R}^{n\times n\times n}$

Graph Weighted Models [R. Bailly*, **GR***, F. Denis, LATA'15/JCSS'18]

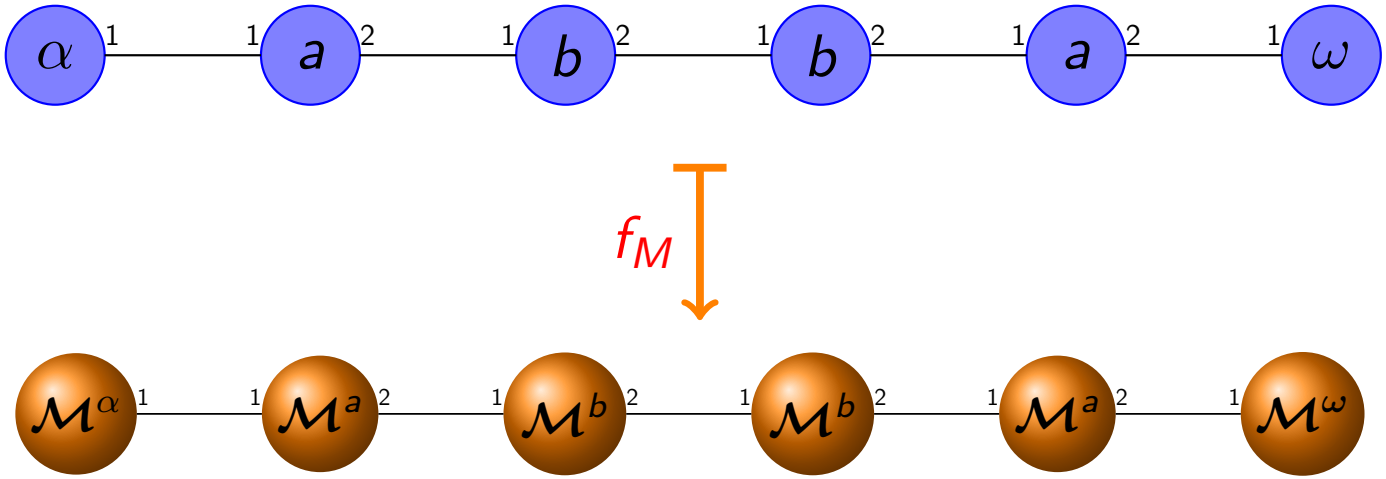
- $\mathcal{F} = \{a(\cdot), h(\cdot,\cdot), g(\cdot,\cdot,\cdot)\}$
- GWM: vector $\boldsymbol{\mathcal{M}}^a \in \mathbb{R}^n$, matrix $\boldsymbol{\mathcal{M}}^h \in \mathbb{R}^{n\times n}$, tensor $\boldsymbol{\mathcal{M}}^g \in \mathbb{R}^{n\times n\times n}$



$$f_M(G) = \sum_{i_1,i_2,i_3,i_4\in[n]} \boldsymbol{\mathcal{M}}^h_{i_1,i_2}\boldsymbol{\mathcal{M}}^h_{i_2,i_3}\boldsymbol{\mathcal{M}}^g_{i_3,i_1,i_4}\boldsymbol{\mathcal{M}}^a_{i_4}$$

Graph Weighted Models [R. Bailly*, **GR***, F. Denis, LATA'15/JCSS'18]

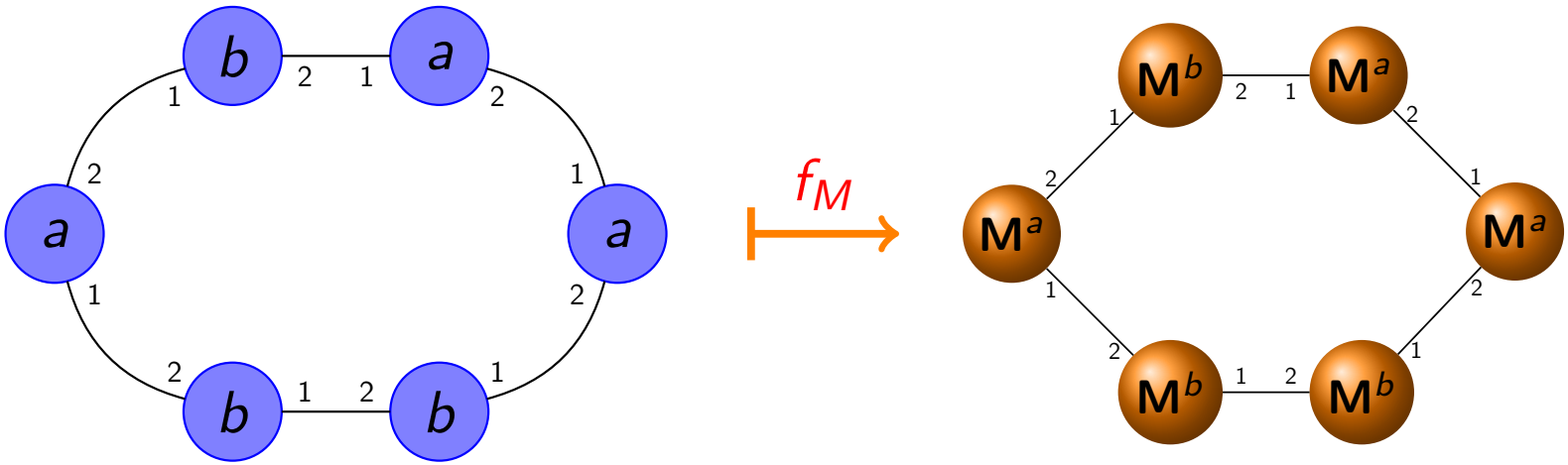
- $\mathcal{F} = \{\alpha(\cdot), a(\cdot,\cdot), b(\cdot,\cdot), \omega(\cdot)\}$
- GWM: $\boldsymbol{\mathcal{M}}^{\alpha}, \boldsymbol{\mathcal{M}}^{\omega} \in \mathbb{R}^{n}$, $\boldsymbol{\mathcal{M}}^{a}, \boldsymbol{\mathcal{M}}^{b} \in \mathbb{R}^{n \times n}$



$$f_M(G) = \sum_{i_1,i_2,i_3,i_4,i_5 \in [n]} \boldsymbol{\mathcal{M}}^{\alpha}_{i_1} \boldsymbol{\mathcal{M}}^{a}_{i_1,i_2} \boldsymbol{\mathcal{M}}^{b}_{i_2,i_3} \boldsymbol{\mathcal{M}}^{b}_{i_3,i_4} \boldsymbol{\mathcal{M}}^{a}_{i_4,i_5} \boldsymbol{\mathcal{M}}^{\omega}_{i_5}$$
$$= \boldsymbol{\alpha}^{\top} \mathbf{M}^{a} \mathbf{M}^{b} \mathbf{M}^{b} \mathbf{M}^{a} \boldsymbol{\omega}$$

Graph Weighted Models [R. Bailly*, **GR***, F. Denis, LATA'15/JCSS'18]

- $\mathcal{F} = \{a(\cdot,\cdot), b(\cdot,\cdot)\}$
- GWM: $\mathbf{M}^a, \mathbf{M}^b \in \mathbb{R}^{n\times n}$



$$f_M(G) = \sum_{i_1,i_2,i_3,i_4,i_5,i_6\in[n]} \mathbf{M}^a_{i_1,i_2}\mathbf{M}^b_{i_2,i_3}\mathbf{M}^a_{i_3,i_4}\mathbf{M}^a_{i_4,i_5}\mathbf{M}^b_{i_5,i_6}\mathbf{M}^b_{i_6,i_1}$$

$$= \mathrm{Tr}(\mathbf{M}^a\mathbf{M}^b\mathbf{M}^a\mathbf{M}^a\mathbf{M}^b\mathbf{M}^b)$$

Learning Graph Weighted Models

- Long term objective: extend the spectral learning algorithm to functions defined over graphs.
  - $\rightarrow$ learning general GWMs is very challenging.
- First step: study the problem of learning GWMs defined over simple families of graphs (circular strings, 2D grids).
- Minimization of GWMs over circular strings [**GR**, FoSSaCS'18]:
  - Minimizing WA $\leftrightarrow$ linear algebra
  - Minimizing GWMs $\leftrightarrow$ theory of finite dimensional algebras

Outline

Conclusion

- Spectral methods for tensor and discrete structured data.

$\Rightarrow$ Leverage fundamental algebraic properties for learning:
  - Take tensor structure into account for better generalization.
  - Learning for structured data with weighted automata.
  - Spectral learning: efficient and consistent learning algorithms.

Multilinear algebra $\leftrightarrow$ powerful models for learning with structured data.

Future Research Directions

- Learning with graphs.
  - Develop efficient learning algorithms for graph structured data.
  - Spectral learning of GWM $\Rightarrow$ consistent learning algorithm.
  - Explore connections with graph neural networks (TCS insight).

Future Research Directions

- Learning with graphs.
  - Develop efficient learning algorithms for graph structured data.
  - Spectral learning of GWM $\Rightarrow$ consistent learning algorithm.
  - Explore connections with graph neural networks (TCS insight).
- Fast and scalable learning algorithms.
  - Tensor networks have been successfully used in numerical analysis and quantum physics to perform very large scale linear algebra.
  - Wide range of potential applications in ML.

Future Research Directions

- Learning with graphs.
  - Develop efficient learning algorithms for graph structured data.
  - Spectral learning of GWM $\Rightarrow$ consistent learning algorithm.
  - Explore connections with graph neural networks (TCS insight).
- Fast and scalable learning algorithms.
  - Tensor networks have been successfully used in numerical analysis and quantum physics to perform very large scale linear algebra.
  - Wide range of potential applications in ML.
- Nonlinear tensor learning.
  - Combine the power of tensor algebra and deep learning.
  - Revisit higher-order RNN through the lens of multilinear algebra.
  - Both directions, e.g. non-linear extensions of tensor decomposition techniques / multilinear regularization in deep networks.

Thank you for your attention.

Image Reconstruction from Noisy Measurements

- $\mathcal{W} \in \mathbb{R}^{3\times 50\times 50}$ is an RGB image.
- Data is generated by $\mathbf{Y} = \mathcal{W} \times_1 \mathbf{x} + \boldsymbol{\xi}$ where $\mathbf{x} \sim \mathcal{N}(0, \mathbf{I})$ and $\boldsymbol{\xi}_{ij} \sim \mathcal{N}(0, 1)$.
- Training set of size 200.

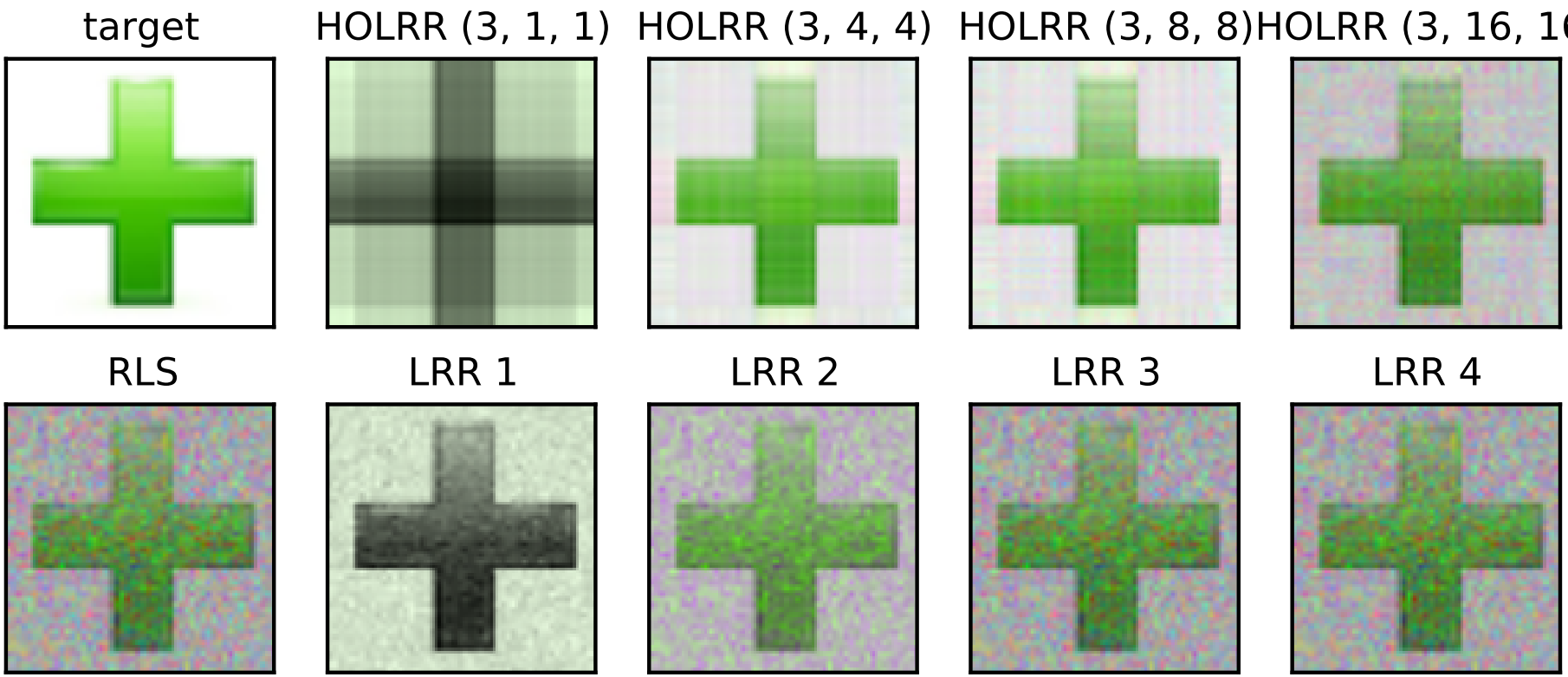
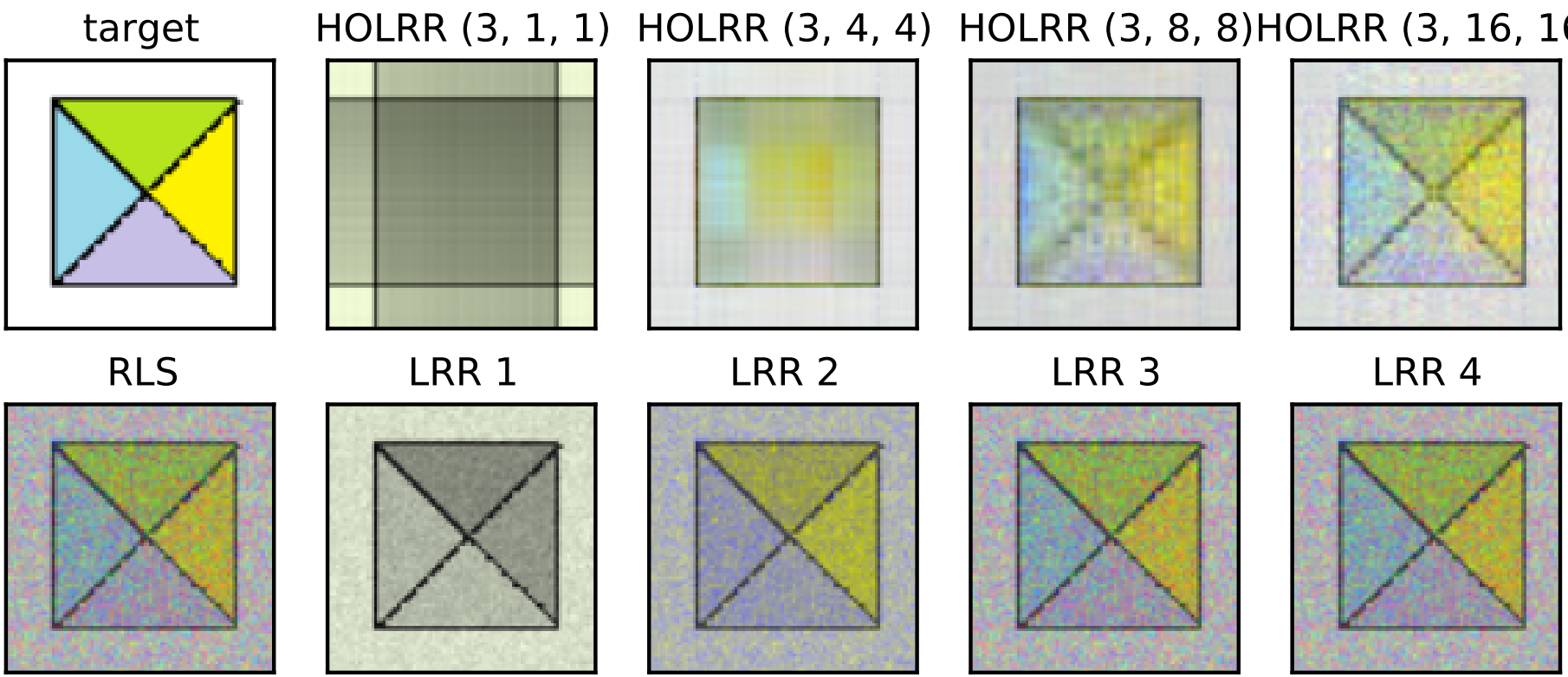


Image Reconstruction from Noisy Measurements

- $\mathcal{W} \in \mathbb{R}^{3\times 70\times 70}$ is an RGB image.



Izenman, A. J. (1975). Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, 5(2):248–264.