

# Tensor networks for machine learning: from random projections to learning the structure of tensor networks

*Guillaume Rabusseau*

Assistant Professor at DIRO, UdeM  
CIFAR Canada Chair in AI at Mila

October 27, 2020  
RIKEN AIP

# Learning with Structured Data

Supervised Learning:

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

# Learning with Structured Data

Supervised Learning:

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

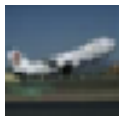
- Classical learning algorithms assume  $\mathcal{X} = \mathbb{R}^d$  and  $\mathcal{Y} = \mathbb{R}^p$ .
- How to handle input/output structured data?

# Learning with Structured Data

## Supervised Learning:

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

- 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, ...



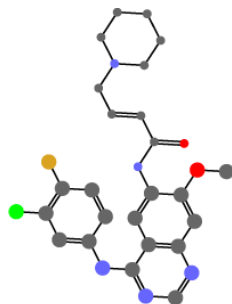
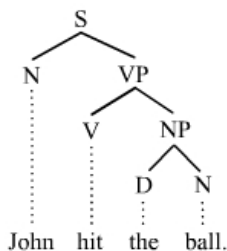
$$\in \mathbb{R}^{32 \times 32 \times 3} \simeq \mathbb{R}^{3072}$$

# Learning with Structured Data

## Supervised Learning:

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

- 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:

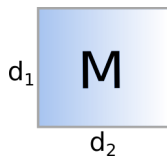
Learn  $f : \mathcal{X} \rightarrow \mathcal{Y}$  from a sample  $\{(x_1, y_1), \dots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$ .

- 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.

# Outline

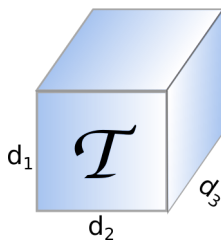
- 1 Preliminaries: Tensors and Multilinear Algebra
- 2 Tensorized Random Projections
- 3 Adaptive Learning of Tensor Decomposition Models

# Tensors



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

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



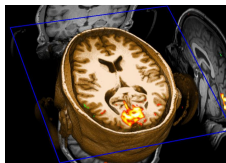
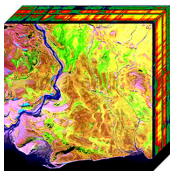
$$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$$

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

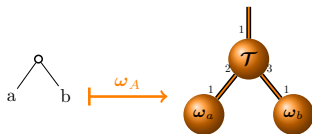


# Tensors and Machine Learning

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

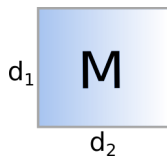


- (ii) Tensors as **parameters** of a model: polynomial regression, higher-order RNNs, weighted automata on trees and graphs...



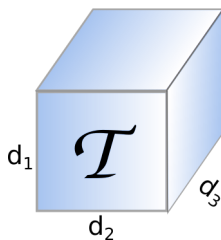
- (iii) Tensors as **tools**: tensor method of moments [Anandkumar et al., 2014], layer compression in neural networks [Novikov et al., 2015], deep learning theoretical analysis [Cohen et al., 2015]...

# Tensors



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

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



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

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

# Tensors are not easy...

## MOST TENSOR PROBLEMS ARE NP HARD

CHRISTOPHER J. HILLAR AND LEK-HENG LIM

ABSTRACT. The idea that one might extend numerical linear algebra, the collection of matrix computational methods that form the workhorse of scientific and engineering computing, to *numerical multilinear algebra*, an analogous collection of tools involving hypermatrices/tensors, appears very promising and has attracted a lot of attention recently. We examine here the computational tractability of some core problems in numerical multilinear algebra. We show that tensor analogues of several standard problems that are readily computable in the matrix (i.e. 2-tensor) case are NP hard. Our list here includes: determining the feasibility of a system of bilinear equations, determining an eigenvalue, a singular value, or the spectral norm of a 3-tensor, determining a best rank-1 approximation to a 3-tensor, determining the rank of a 3-tensor over  $\mathbb{R}$  or  $\mathbb{C}$ . Hence making tensor computations feasible is likely to be a challenge.

[Hillar and Lim, **Most tensor problems are NP-hard**, Journal of the ACM, 2013.]

# Tensors are not easy...

## MOST TENSOR PROBLEMS ARE NP HARD

CHRISTOPHER J. HILLAR AND LEK-HENG LIM

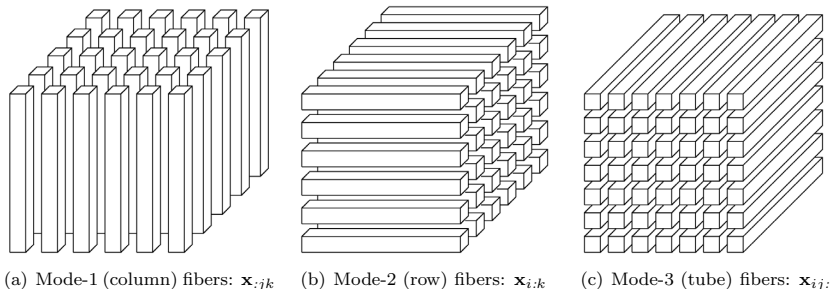
ABSTRACT. The idea that one might extend numerical linear algebra, the collection of matrix computational methods that form the workhorse of scientific and engineering computing, to *numerical multilinear algebra*, an analogous collection of tools involving hypermatrices/tensors, appears very promising and has attracted a lot of attention recently. We examine here the computational tractability of some core problems in numerical multilinear algebra. We show that tensor analogues of several standard problems that are readily computable in the matrix (i.e. 2-tensor) case are NP hard. Our list here includes: determining the feasibility of a system of bilinear equations, determining an eigenvalue, a singular value, or the spectral norm of a 3-tensor, determining a best rank-1 approximation to a 3-tensor, determining the rank of a 3-tensor over  $\mathbb{R}$  or  $\mathbb{C}$ . Hence making tensor computations feasible is likely to be a challenge.

[Hillar and Lim, **Most tensor problems are NP-hard**, Journal of the ACM, 2013.]

... but training a neural network with 3 nodes is also NP hard [Blum and Rivest, NIPS '89]

# Forget rows and columns... Now we have **fibers**!

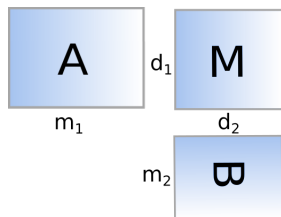
- Matrices have rows and columns, tensors have **fibers**<sup>1</sup>:



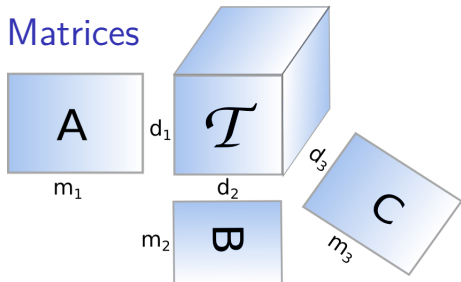
**Fig. 2.1** *Fibers of a 3rd-order tensor.*

<sup>1</sup>fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].

## Tensors: Multiplication with Matrices

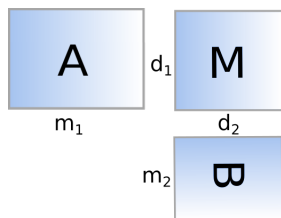


$$\mathbf{A}\mathbf{M}\mathbf{B}^T \in \mathbb{R}^{m_1 \times m_2}$$

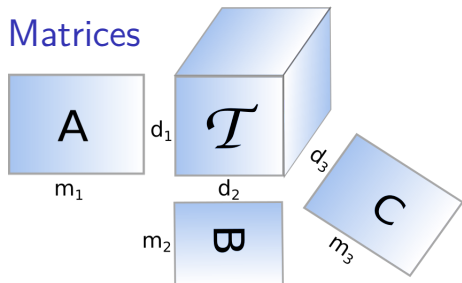


$$\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$$

# Tensors: Multiplication with Matrices



$$\mathbf{AMB}^T \in \mathbb{R}^{m_1 \times m_2}$$




$$\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$$

ex: If  $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$  and  $\mathbf{B} \in \mathbb{R}^{m_2 \times d_2}$ , then  $\mathcal{T} \times_2 \mathbf{B} \in \mathbb{R}^{d_1 \times m_2 \times d_3}$  and


$$(\mathcal{T} \times_2 \mathbf{B})_{i_1, i_2, i_3} = \sum_{k=1}^{d_2} \mathcal{T}_{i_1, k, i_3} \mathbf{B}_{i_2, k} \quad \text{for all } i_1 \in [d_1], i_2 \in [m_2], i_3 \in [d_3].$$

# Tensor Networks

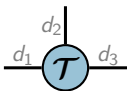
Degree of a node  $\equiv$  order of tensor



$\mathbf{v} \in \mathbb{R}^d$



$\mathbf{M} \in \mathbb{R}^{m \times n}$

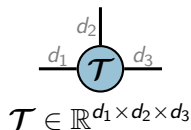
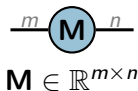
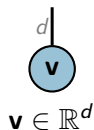


$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$



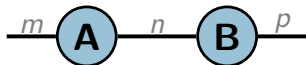
# Tensor Networks

Degree of a node  $\equiv$  order of tensor



Edge  $\equiv$  contraction


Matrix product:




$$(\mathbf{AB})_{i_1, i_2} = \sum_{k=1}^n \mathbf{A}_{i_1 k} \mathbf{B}_{k i_2}$$

# Tensor Networks

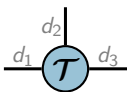
Degree of a node  $\equiv$  order of tensor



$\mathbf{v} \in \mathbb{R}^d$



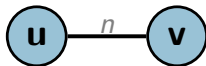
$\mathbf{M} \in \mathbb{R}^{m \times n}$



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

Edge  $\equiv$  contraction

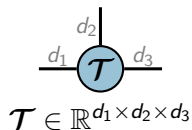
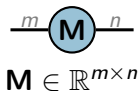
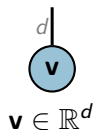
Inner product:



$$\mathbf{u}^\top \mathbf{v} = \sum_{k=1}^n \mathbf{u}_k \mathbf{v}_k$$

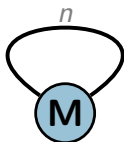
# Tensor Networks

Degree of a node  $\equiv$  order of tensor



Edge  $\equiv$  contraction

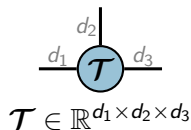
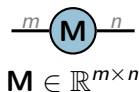
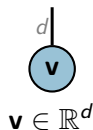
Trace of an  $n \times n$  matrix:



$$\text{Tr}(\mathbf{M}) = \sum_{i=1}^n \mathbf{M}_{ii}$$

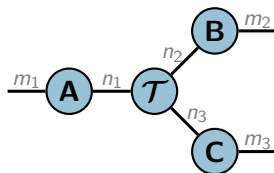
# Tensor Networks

Degree of a node  $\equiv$  order of tensor



Edge  $\equiv$  contraction

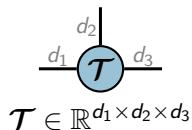
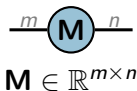
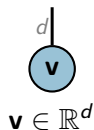
Tensor times matrices:



$$(\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C})_{i_1, i_2, i_3} = \sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \sum_{k_3=1}^{n_3} \mathcal{T}_{k_1 k_2 k_3} \mathbf{A}_{i_1 k_1} \mathbf{B}_{i_2 k_2} \mathbf{C}_{i_3 k_3}$$

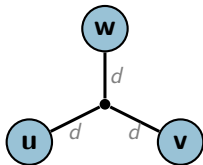
# Tensor Networks

Degree of a node  $\equiv$  order of tensor



Edge  $\equiv$  contraction

Hyperedge  $\equiv$  contraction between more than 2 indices:



$$\sum_{i=1}^n \mathbf{u}_i \mathbf{v}_i \mathbf{w}_i$$

# Tensor Decomposition Techniques

- Tensors can get huge quickly:
  - ▶ 3rd order tensor of shape  $d \times d \times d$ :  $d^3$  parameters
  - ▶ 4th order tensor of shape  $d \times d \times d \times d$ :  $d^4$  parameters
  - ▶ 10th order tensor of shape  $d \times d \times \dots \times d$ :  $d^{10}$  parameters
  - ▶ ...

# Tensor Decomposition Techniques

Simple idea: decompose a tensor into product of small factors.

# Tensor Decomposition Techniques

Simple idea: decompose a tensor into product of small factors.

- Similar to matrix factorization:

- ▶ If  $\mathbf{M} \in \mathbb{R}^{m \times n}$  and  $\mathbf{M} = \mathbf{AB}$  with  $\mathbf{A} \in \mathbb{R}^{m \times r}$  and  $\mathbf{B} \in \mathbb{R}^{r \times n}$



# Tensor Decomposition Techniques

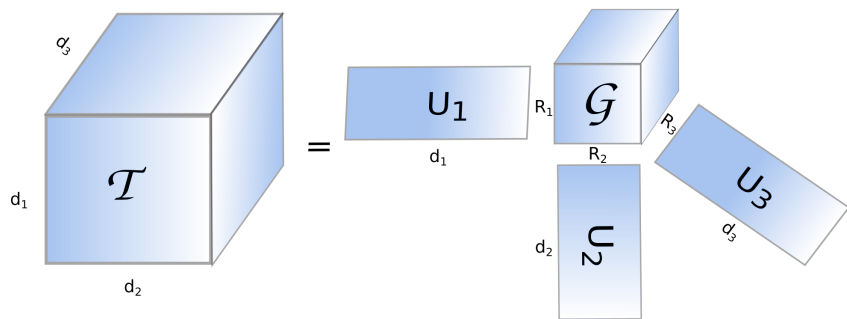
Simple idea: decompose a tensor into product of small factors.

- Similar to matrix factorization:

- ▶ If  $\mathbf{M} \in \mathbb{R}^{m \times n}$  and  $\mathbf{M} = \mathbf{A}\mathbf{B}$  with  $\mathbf{A} \in \mathbb{R}^{m \times r}$  and  $\mathbf{B} \in \mathbb{R}^{r \times n}$   
⇒  $r(m + n)$  parameters instead of  $mn...$

# Tensor Decomposition Techniques

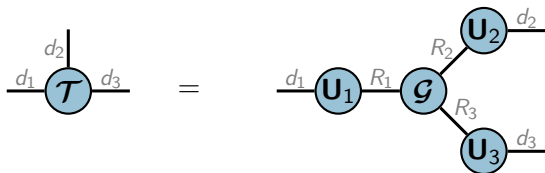
- Tucker decomposition [Tucker, 1966]:



$\Rightarrow R_1 R_2 R_3 + d_1 R_1 + d_2 R_2 + d_2 R_2$  parameters instead of  $d_1 d_2 d_3$ .

# Tensor Decomposition Techniques

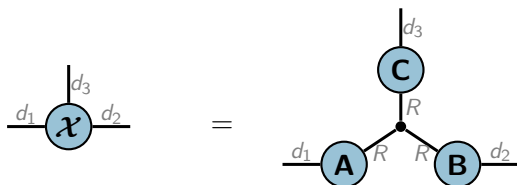
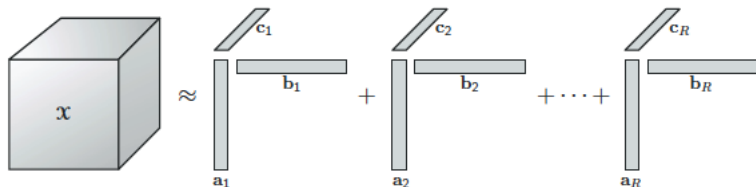
- Tucker decomposition [Tucker, 1966]:



$\Rightarrow R_1 R_2 R_3 + d_1 R_1 + d_2 R_2 + d_3 R_3$  parameters instead of  $d_1 d_2 d_3$ .

# Tensor Decomposition Techniques

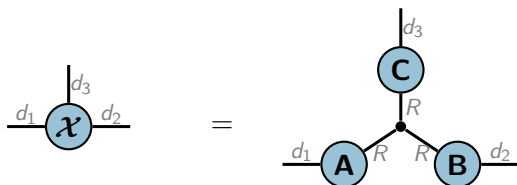
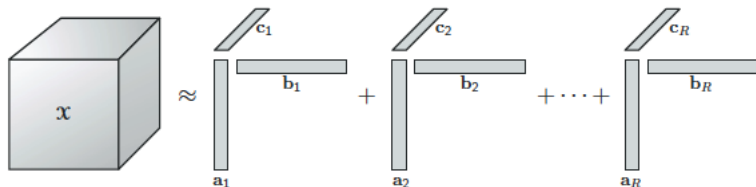
- CP decomposition [Hitchcock, 1927]<sup>2</sup>:



<sup>2</sup>fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].

# Tensor Decomposition Techniques

- CP decomposition [Hitchcock, 1927]<sup>2</sup>:

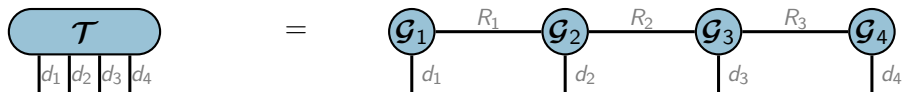


$\Rightarrow R(d_1 + d_2 + d_3)$  parameters instead of  $d_1 d_2 d_3$ .

<sup>2</sup>fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].

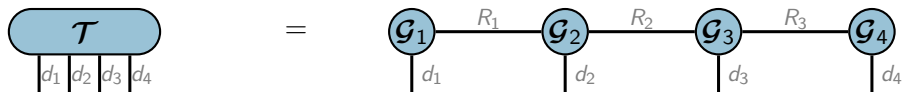
# Tensor Decomposition Techniques

- Tensor Train decomposition [Oseledets, 2011]:



# Tensor Decomposition Techniques

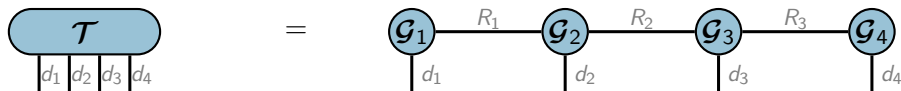
- Tensor Train decomposition [Oseledets, 2011]:



$\Rightarrow d_1 R_1 + R_1 d_2 R_2 + R_2 d_2 R_3 + R_3 d_3 d_4$  parameters instead of  $d_1 d_2 d_3 d_4$ .

# Tensor Decomposition Techniques

- Tensor Train decomposition [Oseledets, 2011]:



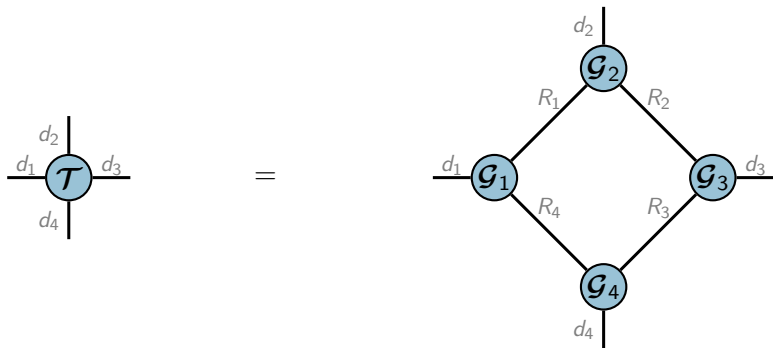
$\Rightarrow d_1 R_1 + R_1 d_2 R_2 + R_2 d_2 R_3 + R_3 d_3$  parameters instead of  $d_1 d_2 d_3 d_4$ .

- If the ranks are all the same ( $R_1 = R_2 = \dots = R$ ), can represent a vector of size  $2^n$  with  $\mathcal{O}(nR^2)$  parameters!



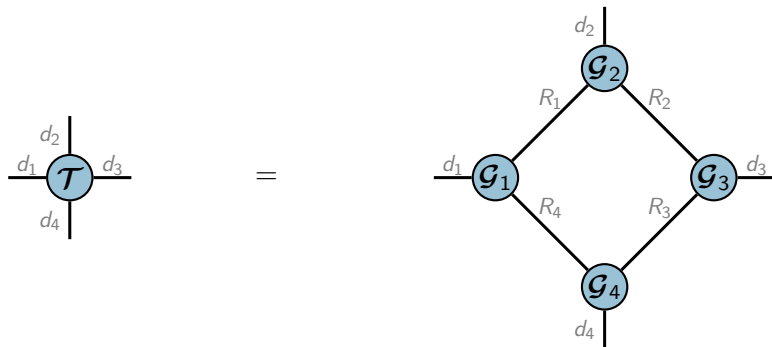
# Tensor Decomposition Techniques

- Tensor Ring decomposition [Zhao et al., 2016]:



# Tensor Decomposition Techniques

- Tensor Ring decomposition [Zhao et al., 2016]:



$\Rightarrow R_4 d_1 R_1 + R_1 d_2 R_2 + R_2 d_3 R_3 + R_3 d_4 R_4$  parameters instead of  $d_1 d_2 d_3 d_4$ .

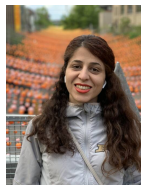
# Summary of Common Tensor Decomposition Models

- For an  $N$ th order tensor of size  $d \times d \times d \times \dots \times d$ , instead of  $d^N$  parameters we have
  - ▶ Tucker:  $\mathcal{O}(R^N + NdR)$  parameters
  - ▶ CP:  $\mathcal{O}(NdR)$  parameters
  - ▶ Tensor train (TT):  $\mathcal{O}(NdR^2)$  parameters
  - ▶ Tensor ring (TR):  $\mathcal{O}(NdR^2)$  parameterswhere the rank  $R = \max_j R_j$ .

# Outline

- 1 Preliminaries: Tensors and Multilinear Algebra
- 2 Tensorized Random Projections
- 3 Adaptive Learning of Tensor Decomposition Models

Joint work with Beheshteh T. Rakhshan, published at AISTATS 2020.



# Motivation

- **Random projection (RP)** and **tensor decomposition** : Two tools to deal with high-dimensional data
- But **RP** cannot scale to very high-dimensional inputs (e.g. high-order tensors)
- We use tensor decomposition to scale Gaussian RP to high-order tensors

# Random Projections (RP)

- Goal: find a low-dimensional projection  $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$  ( $k \ll d$ ) that preserves distances (with high proba.).

## Random Projections (RP)

- Goal: find a low-dimensional projection  $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$  ( $k \ll d$ ) that preserves distances (with high proba.).
- Johnson-Lindenstrauss Transform (or Gaussian RP).

$$f : \mathbf{x} \mapsto \frac{1}{\sqrt{k}} \mathbf{M} \mathbf{x} \quad \text{where } \mathbf{M}_{ij} \sim_{iid} \mathcal{N}(0, 1) \text{ for each } i, j$$

### Theorem (JL, 1984)

Let  $\varepsilon > 0$  and  $\mathbf{x}_1, \dots, \mathbf{x}_m \in \mathbb{R}^d$ .

If  $k \gtrsim \varepsilon^{-2} \log m$ , then, with high proba.,  $\|f(\mathbf{x}_i)\| = (1 \pm \varepsilon) \|\mathbf{x}_i\|$  for all  $i = 1, \dots, m$ .

- Applications: sketched linear regression, randomized SVD, pre-processing step in ML pipeline, ...

# Random Projections (RP)

- Goal: find a low-dimensional projection  $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$  ( $k \ll d$ ) that preserves distances (with high proba.).
- Johnson-Lindenstrauss Transform (or Gaussian RP).

$$f : \mathbf{x} \mapsto \frac{1}{\sqrt{k}} \mathbf{M} \mathbf{x} \quad \text{where } \mathbf{M}_{ij} \sim_{iid} \mathcal{N}(0, 1) \text{ for each } i, j$$

## Theorem (JL, 1984)

Let  $\varepsilon > 0$  and  $\mathbf{x}_1, \dots, \mathbf{x}_m \in \mathbb{R}^d$ .

If  $k \gtrsim \varepsilon^{-2} \log m$ , then, with high proba.,  $\|f(\mathbf{x}_i)\| = (1 \pm \varepsilon) \|\mathbf{x}_i\|$  for all  $i = 1, \dots, m$ .

- Applications: sketched linear regression, randomized SVD, pre-processing step in ML pipeline, ...
- **Pbm**: if  $\mathbf{x} \in \mathbb{R}^{d^N}$  is a high-order tensor represented in CP/TT format, the Gaussian RP has  $d^N k$  parameters...



# Objective

- We want to find a RP map  $f : \mathbb{R}^{d^N} \rightarrow \mathbb{R}^k$  such that:
  - ▶ the number of parameters is linear in  $N$
  - ▶ computing  $f(\mathbf{x})$  is efficient when  $\mathbf{x}$  is in the CP or TT format
  - ▶  $f$  preserves distances with high probability.

# Objective

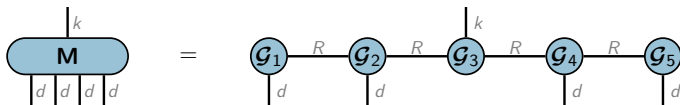
- We want to find a RP map  $f : \mathbb{R}^{d^N} \rightarrow \mathbb{R}^k$  such that:
  - ▶ the number of parameters is linear in  $N$
  - ▶ computing  $f(\mathbf{x})$  is efficient when  $\mathbf{x}$  is in the CP or TT format
  - ▶  $f$  preserves distances with high probability.
- Two important properties that a RP must satisfy:
  - ▶  $\mathbb{E}[\|f(\mathbf{x})\|^2] = \|\mathbf{x}\|^2$  for all  $\mathbf{x}$
  - ▶  $\lim_{k \rightarrow \infty} \mathbb{V}[\|f(\mathbf{x})\|^2] = 0$

# Objective

- We want to find a RP map  $f : \mathbb{R}^{d^N} \rightarrow \mathbb{R}^k$  such that:
    - ▶ the number of parameters is linear in  $N$
    - ▶ computing  $f(\mathbf{x})$  is efficient when  $\mathbf{x}$  is in the CP or TT format
    - ▶  $f$  preserves distances with high probability.
  - Two important properties that a RP must satisfy:
    - ▶  $\mathbb{E}[\|f(\mathbf{x})\|^2] = \|\mathbf{x}\|^2$  for all  $\mathbf{x}$
    - ▶  $\lim_{k \rightarrow \infty} \mathbb{V}[\|f(\mathbf{x})\|^2] = 0$
- ↪ the rate at which  $\mathbb{V}[\|f(\mathbf{x})\|^2]$  converges to 0 captures the **quality** of a RP.

# Tensor Train RP: First Attempt

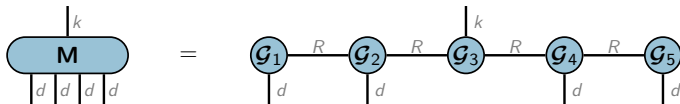
- We build a Gaussian RP  $f : \mathbf{x} \mapsto \frac{1}{Z} \mathbf{M} \mathbf{x}$  where  $\mathbf{M} \in \mathbb{R}^{k \times d^N}$  is represented using the TT format:



where the entries of each core tensor  $\mathcal{G}_n$  are drawn iid from  $\mathcal{N}(0, 1)$ .

# Tensor Train RP: First Attempt

- We build a Gaussian RP  $f : \mathbf{x} \mapsto \frac{1}{Z} \mathbf{M} \mathbf{x}$  where  $\mathbf{M} \in \mathbb{R}^{k \times d^N}$  is represented using the TT format:



where the entries of each core tensor  $\mathcal{G}_n$  are drawn iid from  $\mathcal{N}(0, 1)$ .

- ☺  $\mathcal{O}(R^2 N d + R^2 k)$  parameters instead of  $d^N k$ .
- ☺ Efficient computation of  $\mathbf{M} \mathbf{x}$  when  $\mathbf{x}$  is in the CP/TT format.
- ☺ We have  $\mathbb{E}[\|f(\mathbf{x})\|^2] = \|\mathbf{x}\|^2$ .
- ☹ We can show that  $\lim_{k \rightarrow \infty} \mathbb{V}[\|f(\mathbf{x})\|^2] > 0 \dots$

# Tensor Train Random Projection (TT-RP): Second attempt

- Tensor Train RP:

$$f_{TT(R)} : \mathbf{x} \mapsto \frac{1}{\sqrt{kR^N}} \mathbf{M}\mathbf{x}$$

where each row of  $\mathbf{M} = \begin{pmatrix} -\mathbf{m}_1^T \\ -\mathbf{m}_2^T \\ \vdots \\ -\mathbf{m}_k^T \end{pmatrix} \in \mathbb{R}^{k \times d^N}$  is in the TT format:

$\mathbf{m}_i = \mathcal{G}_1^i \text{---} \mathcal{G}_2^i \text{---} \mathcal{G}_3^i \text{---} \mathcal{G}_4^i$  for each  $i = 1, \dots, k$

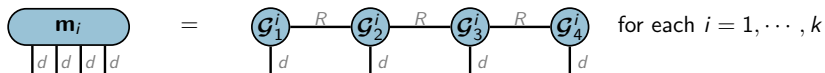
and the entries of each core tensor  $\mathcal{G}_n^i$  are drawn iid from  $\mathcal{N}(0, 1)$ .

# Tensor Train Random Projection (TT-RP): Second attempt

- Tensor Train RP:

$$f_{TT(R)} : \mathbf{x} \mapsto \frac{1}{\sqrt{kR^N}} \mathbf{M}\mathbf{x}$$

where each row of  $\mathbf{M} = \begin{pmatrix} -\mathbf{m}_1^\top - \\ -\mathbf{m}_2^\top - \\ \vdots \\ -\mathbf{m}_k^\top - \end{pmatrix} \in \mathbb{R}^{k \times d^N}$  is in the TT format:



and the entries of each core tensor  $\mathcal{G}_n^i$  are drawn iid from  $\mathcal{N}(0, 1)$ .

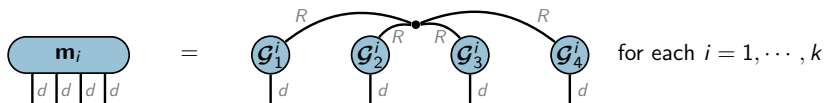
- ☺  $\mathcal{O}(kNdR^2)$  parameters instead of  $d^N k$ .
- ☺ Efficient computation of  $\mathbf{M}\mathbf{x}$  when  $\mathbf{x}$  is in the CP/TT format.
- ☺ We have  $\mathbb{E}[\|f_{TT(R)}(\mathbf{x})\|^2] = \|\mathbf{x}\|^2$ .
- ☺ We have  $\lim_{k \rightarrow \infty} \mathbb{V}[\|f_{TT(R)}(\mathbf{x})\|^2] = 0 \dots$

# CP Random projection (CP-RP)

- CP Random Projection:

$$f_{\text{CP}(R)} : \mathbf{x} \mapsto \frac{1}{\sqrt{kR^N}} \mathbf{M}\mathbf{x}$$

where each row of  $\mathbf{M} = \begin{pmatrix} -\mathbf{m}_1^\top \\ -\mathbf{m}_2^\top \\ \vdots \\ -\mathbf{m}_k^\top \end{pmatrix} \in \mathbb{R}^{k \times d^N}$  is in the CP format:



and the entries of each core tensor  $\mathcal{G}_n^i$  are drawn iid from  $\mathcal{N}(0, 1)$ .

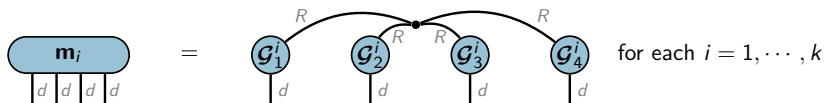


# CP Random projection (CP-RP)

- CP Random Projection:

$$f_{CP(R)} : \mathbf{x} \mapsto \frac{1}{\sqrt{kR^N}} \mathbf{M}\mathbf{x}$$

where each row of  $\mathbf{M} = \begin{pmatrix} -\mathbf{m}_1^\top \\ -\mathbf{m}_2^\top \\ \vdots \\ -\mathbf{m}_k^\top \end{pmatrix} \in \mathbb{R}^{k \times d^N}$  is in the CP format:



and the entries of each core tensor  $\mathcal{G}_n^i$  are drawn iid from  $\mathcal{N}(0, 1)$ .

- ☺  $\mathcal{O}(kNdR)$  parameters instead of  $d^N k$ .
- ☺ Efficient computation of  $\mathbf{M}\mathbf{x}$  when  $\mathbf{x}$  is in the CP/TT format.
- ☺ We have  $\mathbb{E}[\|f_{CP(R)}\|^2] = \|\mathbf{x}\|^2$ .
- ☺ We have  $\lim_{k \rightarrow \infty} \mathbb{V}[\|f_{CP(R)}(\mathbf{x})\|^2] = 0 \dots$

# Main Result

## Theorem

Let  $\mathbf{x} \in \mathbb{R}^{d^N}$  and  $R \in \mathbb{N}$ .

The RP maps  $f_{TT(R)}$  and  $f_{CP(R)}$  satisfy the following properties:

- $\mathbb{E}[\|f_{CP(R)}(\mathbf{x})\|^2] = \mathbb{E}[\|f_{TT(R)}(\mathbf{x})\|^2] = \|\mathbf{x}\|^2$
- $\mathbb{V}[\|f_{TT(R)}(\mathbf{x})\|^2] \leq \frac{1}{k} \left( 3 \left( 1 + \frac{2}{R} \right)^{N-1} - 1 \right) \|\mathbf{x}\|^4$
- $\mathbb{V}[\|f_{CP(R)}(\mathbf{x})\|^2] \leq \frac{1}{k} \left( 3^{N-1} \left( 1 + \frac{2}{R} \right) - 1 \right) \|\mathbf{x}\|^4$

# Main Result

## Theorem

Let  $\mathbf{x} \in \mathbb{R}^{d^N}$  and  $R \in \mathcal{Ncal}$ .

The RP maps  $f_{TT(R)}$  and  $f_{CP(R)}$  satisfy the following properties:

- $\mathbb{E}[\|f_{CP(R)}(\mathbf{x})\|^2] = \mathbb{E}[\|f_{TT(R)}(\mathbf{x})\|^2] = \|\mathbf{x}\|^2$
- $\mathbb{V}[\|f_{TT(R)}(\mathbf{x})\|^2] \leq \frac{1}{k} \left( 3 \left( 1 + \frac{2}{R} \right)^{N-1} - 1 \right) \|\mathbf{x}\|^4$
- $\mathbb{V}[\|f_{CP(R)}(\mathbf{x})\|^2] \leq \frac{1}{k} \left( 3^{N-1} \left( 1 + \frac{2}{R} \right) - 1 \right) \|\mathbf{x}\|^4$

↪ The bounds on the variances are substantially different...

# Comparison between $f_{CP}$ and $f_{TT}$

	$f_{CP(R)} : \mathbb{R}^{d^N} \rightarrow \mathbb{R}^k$	$f_{TT(R)} : \mathbb{R}^{d^N} \rightarrow \mathbb{R}^k$
Number of parameters	$\mathcal{O}(kNdR)$	$\mathcal{O}(kNdR^2)$
Computing $f(\mathbf{x})$ $\mathbf{x}$ in CP with rank $\tilde{R}$	$\mathcal{O}(kNd \max(r, \tilde{R}^2))$	$\mathcal{O}(kNd \max(r, \tilde{R}^3))$
Computing $f(\mathbf{x})$ $\mathbf{x}$ in TT with rank $\tilde{R}$	$\mathcal{O}(kNd \max(r, \tilde{R}^3))$	$\mathcal{O}(kNd \max(r, \tilde{R}^3))$
With proba $\geq 1 - \delta$ , $\mathbb{P}(\ f(\mathbf{x})\ ^2 = (1 \pm \varepsilon) \ \mathbf{x}\ ^2)$ as soon as $k \gtrsim \dots$	$\frac{3^{N-1}(1+2/R)}{\varepsilon^2} \log^{2N} \left( \frac{1}{\delta} \right)$	$\frac{(1+2/R)^N}{\varepsilon^2} \log^{2N} \left( \frac{1}{\delta} \right)$

## Comparison between $f_{CP}$ and $f_{TT}$

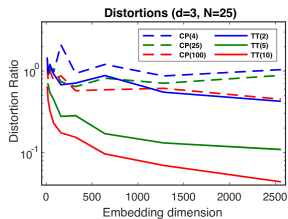
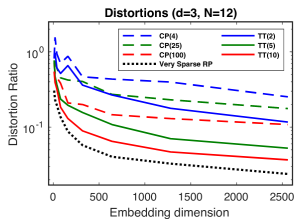
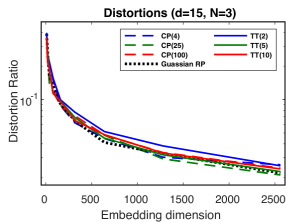
	$f_{CP(R)} : \mathbb{R}^{d^N} \rightarrow \mathbb{R}^k$	$f_{TT(R)} : \mathbb{R}^{d^N} \rightarrow \mathbb{R}^k$
Number of parameters	$\mathcal{O}(kNdR)$	$\mathcal{O}(kNdR^2)$
Computing $f(\mathbf{x})$ $\mathbf{x}$ in CP with rank $\tilde{R}$	$\mathcal{O}(kNd \max(r, \tilde{R}^2))$	$\mathcal{O}(kNd \max(r, \tilde{R}^3))$
Computing $f(\mathbf{x})$ $\mathbf{x}$ in TT with rank $\tilde{R}$	$\mathcal{O}(kNd \max(r, \tilde{R}^3))$	$\mathcal{O}(kNd \max(r, \tilde{R}^3))$
With proba $\geq 1 - \delta$ , $\mathbb{P}(\ f(\mathbf{x})\ ^2 = (1 \pm \varepsilon) \ \mathbf{x}\ ^2)$ as soon as $k \gtrsim \dots$	$\frac{3^{N-1}(1+2/R)}{\varepsilon^2} \log^{2N} \left( \frac{1}{\delta} \right)$	$\frac{(1+2/R)^N}{\varepsilon^2} \log^{2N} \left( \frac{1}{\delta} \right)$

- Lower bounds on  $k$  suggest that  $f_{TT}$  is a better RP than  $f_{CP}$ .
- Classical Gaussian RP needs  $k \gtrsim \frac{1}{\varepsilon^2} \log \left( \frac{1}{\delta} \right)$ .
- Comparisons with other approaches: see [paper on arXiv](#) and [Beheshteh Rakhshan's talk](#) at AISTATS 2020.

# Experiment Setup

- Compare  $f_{TT}$ ,  $f_{CP}$  and classical Gaussian RP to project  $d^N$  dimensional vectors
  - ▶ small order:  $d = 15, N = 3$
  - ▶ medium order:  $d = 3, N = 12$
  - ▶ higher order:  $d = 3, N = 25$
- Input  $\mathbf{x}$  is a random unit-norm TT vector with rank  $\tilde{R} = 10$ .
- Metric: distortion ratio  $\frac{\|f(\mathbf{x})\|^2}{\|\mathbf{x}\|^2} - 1$
- Report averages over 100 trials

# Experiment Results



# Conclusion

- We proposed an efficient way to tensorize classical Gaussian RP
- Theory and experiments suggest that TT is better suited than CP for very high dimensional RP



# Conclusion

- We proposed an efficient way to tensorize classical Gaussian RP
- Theory and experiments suggest that TT is better suited than CP for very high dimensional RP
- Future work:
  - ▶ Leverage results to design efficient linear regression and SVD algorithms
  - ▶ Beyond classical tensor decomposition: other TN structures better suited for RP?
  - ▶ Study of statistical properties of TT vectors with random Gaussian cores

# Outline

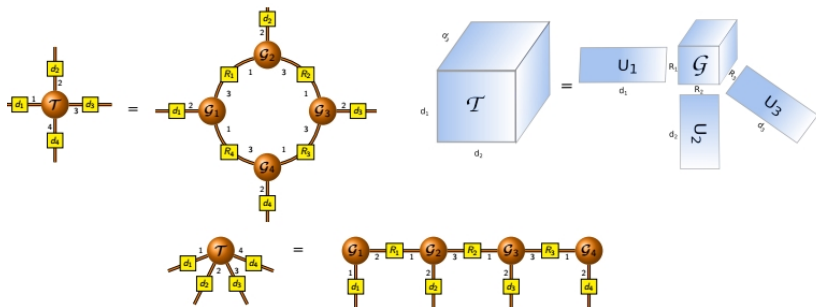
- 1 Preliminaries: Tensors and Multilinear Algebra
- 2 Tensorized Random Projections
- 3 Adaptive Learning of Tensor Decomposition Models

Joint work with Meraj Hashemizadeh, Michelle Liu and Jacob Miller



# Tensor Decomposition Techniques

- Lots of ways to decompose a tensor:



- ⇒ How to choose the *right* decomposition model for a given ML problem?
- ⇒ Can we design adaptive algorithms, learning the decomposition structure from data?
- ⇒ What are the different implicit bias encoded in each decomposition model?
- ⇒ ...

# Tensor based optimization problems

- A lot of tensor problems can be formulated as

$$\min_{\mathcal{W} \in \mathbb{R}^{d_1 \times \dots \times d_p}} L(\mathcal{W}) \quad \text{s.t.} \quad \text{rank}(\mathcal{W}) \leq R$$

where  $L$  is a loss function and rank is some notion of tensor rank (e.g. TT, TR, CP, ...).

- ▶ Tensor Decomposition

$$L(\mathcal{W}) = \|\mathcal{T} - \mathcal{W}\|_F^2$$

- ▶ Tensor Regression

$$L(\mathcal{W}) = \|\mathcal{W} \times_1 \mathbf{X} - \mathcal{Y}\|_F^2$$

- ▶ Tensor Completion
- ▶ ...

# Matrix and tensor completion

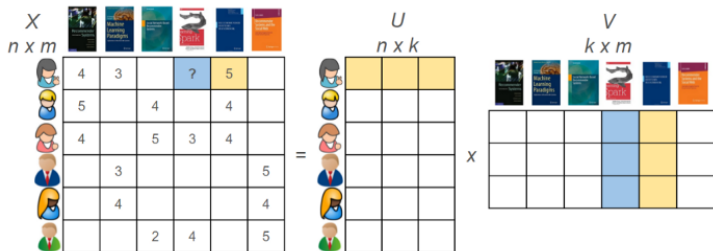
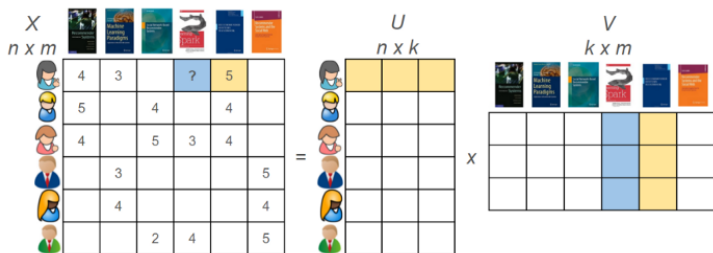


figure credits: Heartbeat Fritz AI: Recommender systems with Python

# Matrix and tensor completion

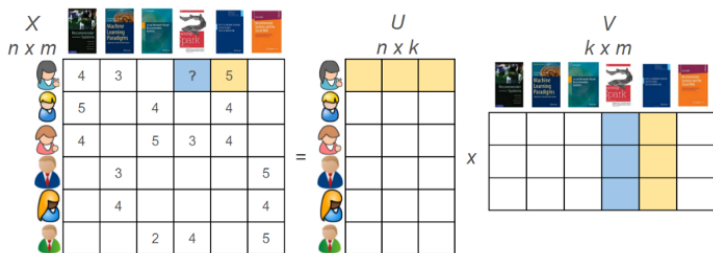


- Here  $L(\mathbf{W}) = \sum_{(i,j) \in \Omega} (\mathbf{W}_{i,j} - \mathbf{X}_{i,j})^2$  where  $\Omega$  is the set of observed entries and the minimization problem is

$$\min_{\mathbf{W} \in \mathbb{R}^{n \times m}} L(\mathbf{W}) \quad \text{s.t.} \quad \text{rank}(\mathbf{W}) \leq k$$

figure credits: Heartbeat Fritz AI: Recommender systems with Python

# Matrix and tensor completion



- Here  $L(\mathbf{W}) = \sum_{(i,j) \in \Omega} (\mathbf{W}_{i,j} - \mathbf{X}_{i,j})^2$  where  $\Omega$  is the set of observed entries and the minimization problem is

$$\min_{\mathbf{W} \in \mathbb{R}^{n \times m}} L(\mathbf{W}) \quad \text{s.t. } \text{rank}(\mathbf{W}) \leq k$$

which is equivalent to:

$$\min_{\mathbf{U} \in \mathbb{R}^{n \times k}, \mathbf{V} \in \mathbb{R}^{k \times m}} L(\mathbf{UV})$$

figure credits: Heartbeat Fritz AI: Recommender systems with Python

# A greedy algorithm for adaptive learning of TN structures

$$\min_{\mathcal{W} \in \mathbb{R}^{d_1 \times \dots \times d_p}} L(\mathcal{W}) \quad \text{s.t.} \quad \text{rank}(\mathcal{W}) \leq R$$

- We do not want to assume a fixed decomposition model.
- We want an algorithm that can adaptively find the best decomposition model for the task at hand.



# A greedy algorithm for adaptive learning of TN structures

$$\min_{\mathcal{W} \in \mathbb{R}^{d_1 \times \dots \times d_p}} L(\mathcal{W}) \quad \text{s.t.} \quad \text{rank}(\mathcal{W}) \leq R$$

- We do not want to assume a fixed decomposition model.
  - We want an algorithm that can adaptively find the best decomposition model for the task at hand.
- ↪ We optimize the loss both with respect to the TN structure and the core tensors of the TN:

$$\min_{\text{Tensor Network Structure}} \min_{\text{TN } \mathcal{G}^{(1)}, \dots, \mathcal{G}^{(p)}} L(\text{TN}(\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(p)}))$$
$$\text{s.t.} \quad \text{size}(\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(p)}) \leq C$$

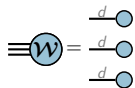
# A greedy algorithm for adaptive learning of TN structures

$$\begin{aligned} \min_{\text{Tensor Network Structure TN } \mathcal{G}^{(1), \dots, \mathcal{G}^{(p)}}} \min_{\mathcal{G}^{(1), \dots, \mathcal{G}^{(p)}}} L(TN(\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(p)})) \\ \text{s.t. } \text{size}(\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(p)}) \leq C \end{aligned}$$

- **Pbm**: the space of TN structures is exponentially large...
- We propose a simple greedy approach:
  - ▶ Start with a rank one tensor
  - ▶ Optimize the loss wrt the core tensors.
  - ▶ Greedily choose an edge to increment in the TN.
  - ▶ Repeat until the parameters budget is reached.

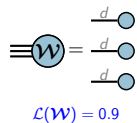
# Greedy Algorithm Overview

- Start with a random rank one tensor.



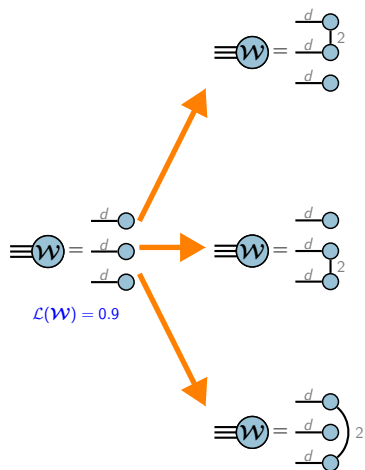
# Greedy Algorithm Overview

- Optimize the loss wrt the core tensors.



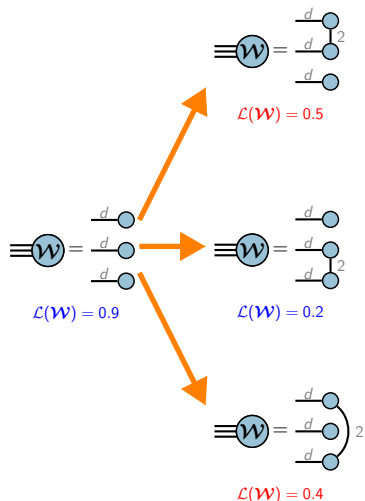
# Greedy Algorithm Overview

- Consider all possible rank one increments on internal edges.



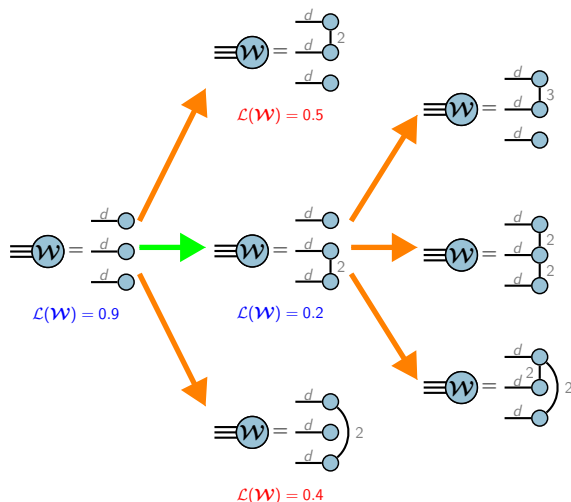
# Greedy Algorithm Overview

- Optimize the loss wrt core tensors for each possible increment.



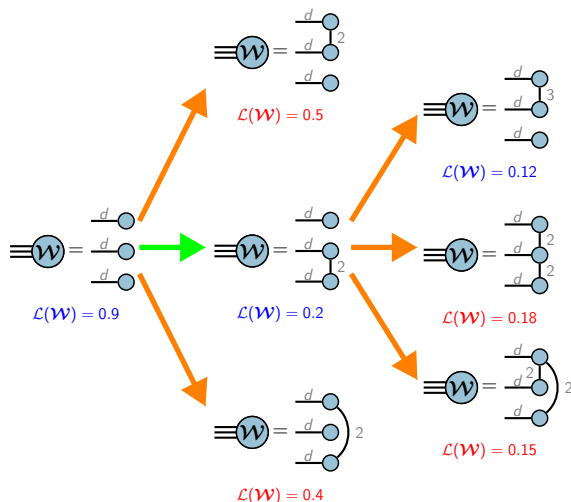
# Greedy Algorithm Overview

- Select the most promising rank increment and repeat...



# Greedy Algorithm Overview

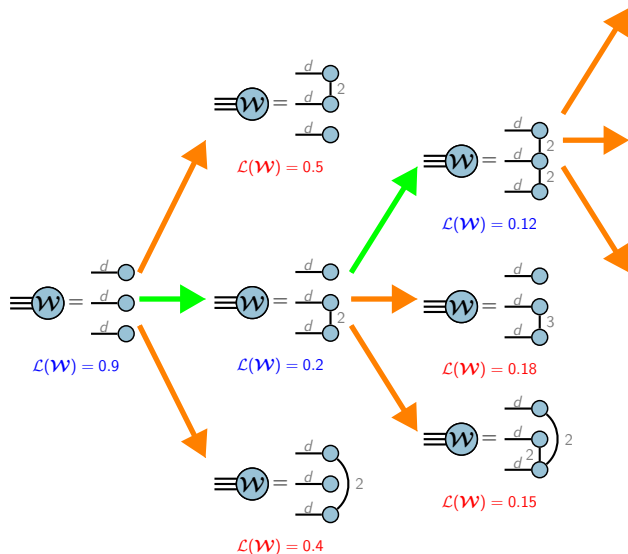
- Select the most promising rank increment and repeat...





# Greedy Algorithm Overview

- Select the most promising rank increment and repeat...

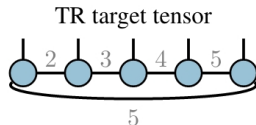
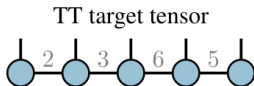


# Implementation Details and Limitations

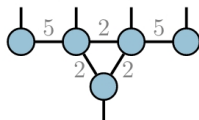
- At each iteration of greedy, we restart the optimization from the previous solution.
- No internal nodes are added to the initial TN structure (cannot represent Tucker).
- No hyperedge (cannot represent CP).
- Computationally expensive.

# Experiment: Tensor decomposition

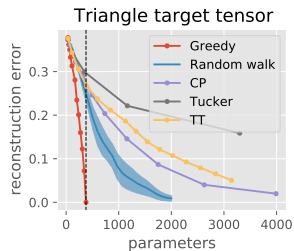
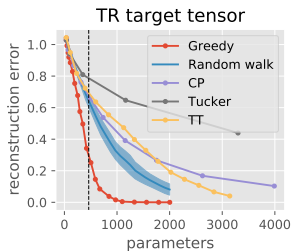
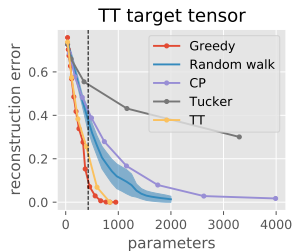
- Objective: compress a given tensor (with unknown tensor network structure) by decomposing it.
- Target tensors:



“Triangle” target tensor

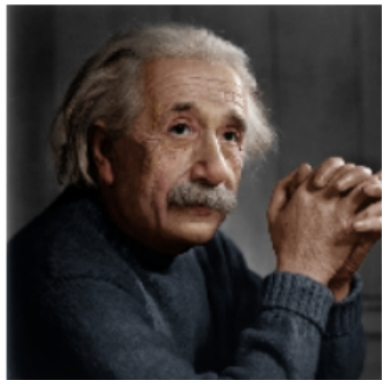


# Experiment: Tensor decomposition

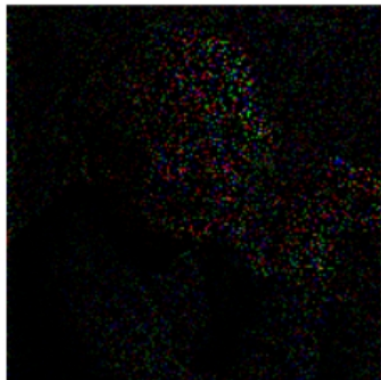


## Experiment: Tensor completion

Original  
image



Observed  
pixels



# Experiment: Tensor completion

TT (rank=2)  
202 param.



TT (rank=18)  
10093 param.



TR (rank=2)  
220 param.



TR (rank=18)  
17820 param.



TT (rank=10)  
3547 param.



TT (rank=26)  
19967 param.



TR (rank=10)  
5500 param.



TR (rank=26)  
37180 param.



Greedy (iter=4)  
295 param.



Greedy (iter=17)  
10635 param.



Greedy (iter=6)  
1041 param.



Greedy (iter=23)  
20175 param.



Greedy (iter=10)  
3273 param.



Greedy (iter=26)  
26085 param.



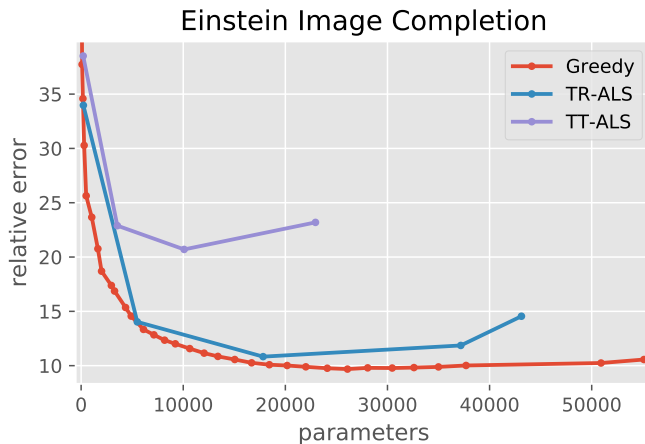
Greedy (iter=12)  
4905 param.



Greedy (iter=31)  
37695 param.



# Experiment: Tensor completion



# Conclusion

- We propose a general adaptive learning algorithm for tensor problem
- First step towards algorithms for general TN rather than specific tensor decomposition models
- Experimental results are very encouraging
- Related to the work of Chao Li and Zhun Sun at ICML 2020



# Conclusion

- We propose a general adaptive learning algorithm for tensor problem
- First step towards algorithms for general TN rather than specific tensor decomposition models
- Experimental results are very encouraging
- Related to the work of Chao Li and Zhun Sun at ICML 2020
- Future directions (ongoing):
  - ▶ Theory: we can show an exponential convergence rate to a solution achieving the optimal loss
  - ▶ Add support for internal nodes and hyperedges
  - ▶ Beyond Greedy:
    - ★ develop heuristics for more efficient search
    - ★ backtracking (e.g. A\* algorithm)
  - ▶ experiments on compressing neural networks

Thank you! Questions?