

Adaptive Tensor Learning with Tensor Networks

Guillaume Rabusseau

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

November 20, 2020

Huawei Montreal research centre

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 $\{(x_1, y_1), \dots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- We often 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 $\{(x_1, y_1), \dots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- We often 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, ...



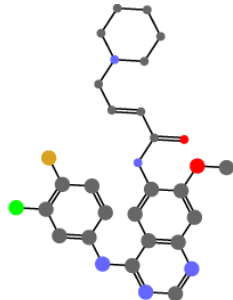
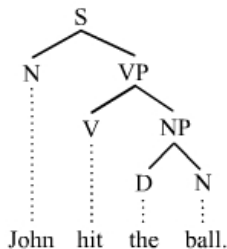
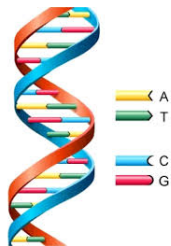
$$\in \mathbb{R}^{32 \times 32 \times 3} \simeq \mathbb{R}^{3072}$$

Learning with Structured Data

Supervised Learning:

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

- We often 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:

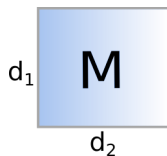
Learn $f : \mathcal{X} \rightarrow \mathcal{Y}$ from a sample $\{(x_1, y_1), \dots, (x_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$.

- We often 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

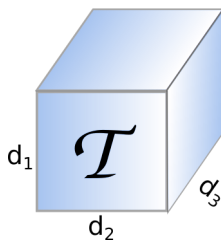
- 1 An Introduction to Tensors and Tensor Networks
- 2 Adaptive Learning of Tensor Decomposition Models

Tensors



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

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

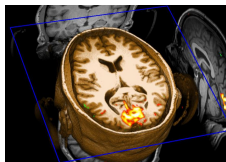
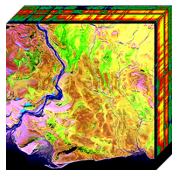


$$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$$

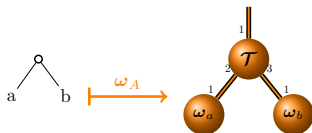
$$(\mathcal{T}_{ijk}) \in \mathbb{R} \text{ for } i \in [d_1], j \in [d_2], k \in [d_3]$$

Tensors and Machine Learning

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

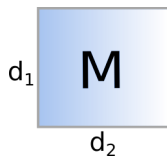


- (ii) Tensors as **parameters** of a model: polynomial regression, higher-order RNNs, weighted automata on trees and graphs...



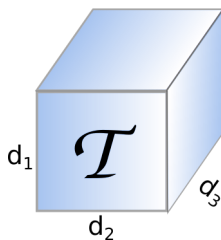
- (iii) Tensors as **tools**: tensor method of moments [Anandkumar et al., 2014], layer compression in neural networks [Novikov et al., 2015], deep learning theoretical analysis [Cohen et al., 2015]...

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**¹:

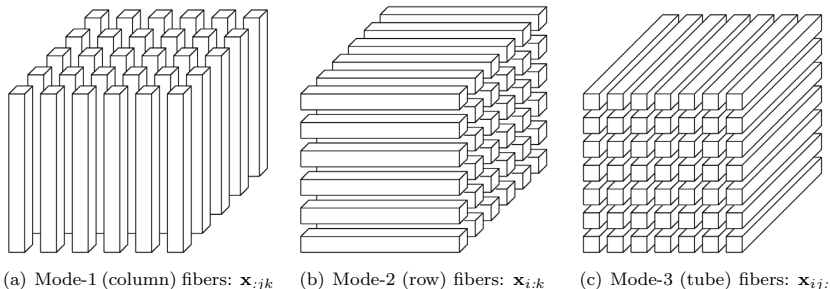
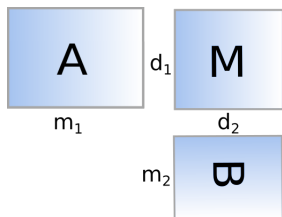


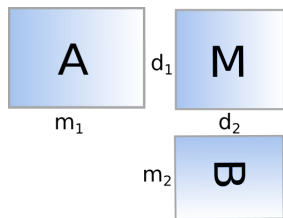
Fig. 2.1 *Fibers of a 3rd-order tensor.*

¹fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].

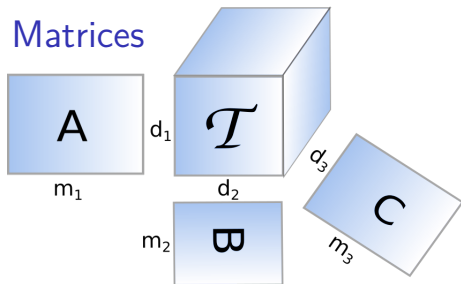
Tensors: Multiplication with Matrices



Tensors: Multiplication with Matrices

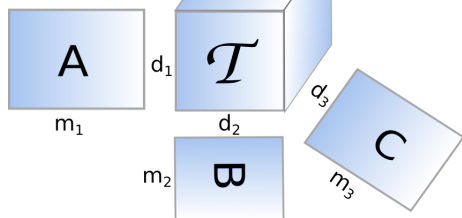
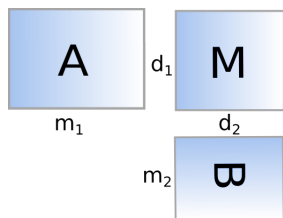


$$\mathbf{A}\mathbf{M}\mathbf{B}^T \in \mathbb{R}^{m_1 \times m_2}$$



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

Tensors: Multiplication with Matrices

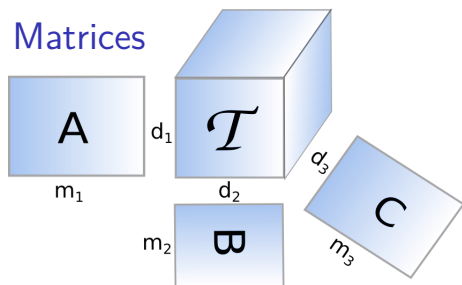
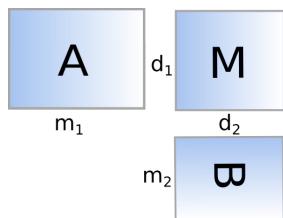


$$\mathbf{A}\mathbf{M}\mathbf{B}^T \in \mathbb{R}^{m_1 \times m_2}$$

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

ex: If $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $\mathbf{A} \in \mathbb{R}^{m_1 \times d_1}$, $\mathbf{B} \in \mathbb{R}^{m_2 \times d_2}$, $\mathbf{C} \in \mathbb{R}^{m_3 \times d_3}$, then $\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ is defined by

Tensors: Multiplication with Matrices



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

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


ex: If $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $\mathbf{A} \in \mathbb{R}^{m_1 \times d_1}$, $\mathbf{B} \in \mathbb{R}^{m_2 \times d_2}$, $\mathbf{C} \in \mathbb{R}^{m_3 \times d_3}$, then $\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ is defined by


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

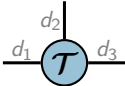
for all $i_1 \in [d_1]$, $i_2 \in [m_2]$, $i_3 \in [d_3]$.

Tensor Networks

Degree of a node \equiv order of tensor

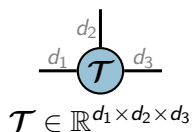
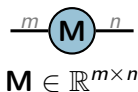
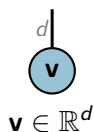

$$\mathbf{v} \in \mathbb{R}^d$$


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


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

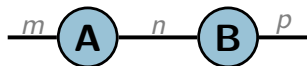
Tensor Networks

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



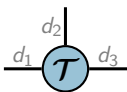
A light blue circle labeled \mathbf{v} with a vertical line extending upwards from its top, labeled d .

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



A light blue circle labeled \mathbf{M} with a horizontal line extending to the left from its center, labeled m , and a horizontal line extending to the right from its center, labeled n .

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

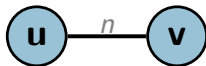


A light blue circle labeled \mathcal{T} with a vertical line extending upwards from its top, labeled d_2 , a horizontal line extending to the left from its center, labeled d_1 , and a horizontal line extending to the right from its center, labeled d_3 .

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

Edge \equiv contraction

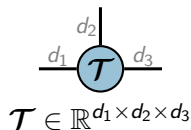
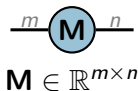
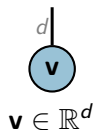
Inner product:



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

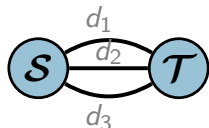
Tensor Networks

Degree of a node \equiv order of tensor



Edge \equiv contraction

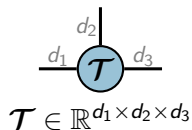
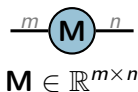
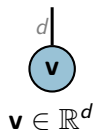
Inner product between tensors:



$$\langle S, T \rangle = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \sum_{i_3=1}^{d_3} S_{i_1 i_2 i_3} T_{i_1 i_2 i_3}$$

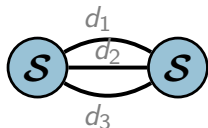
Tensor Networks

Degree of a node \equiv order of tensor



Edge \equiv contraction

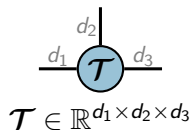
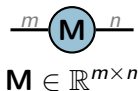
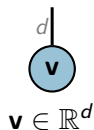
Frobenius norm of a tensor:



$$\|\mathbf{S}\|_F^2 = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \sum_{i_3=1}^{d_3} (\mathbf{s}_{i_1 i_2 i_3})^2$$

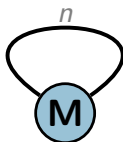
Tensor Networks

Degree of a node \equiv order of tensor



Edge \equiv contraction

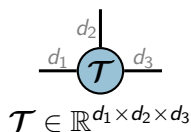
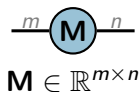
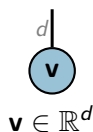
Trace of an $n \times n$ matrix:



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

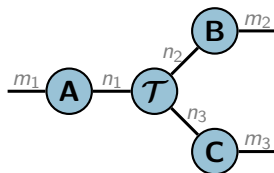
Tensor Networks

Degree of a node \equiv order of tensor



Edge \equiv contraction

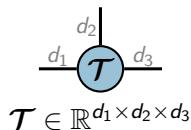
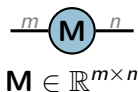
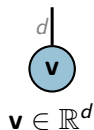
Tensor times matrices:



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

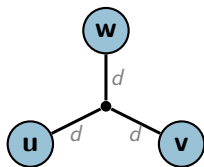
Tensor Networks

Degree of a node \equiv order of tensor



Edge \equiv contraction

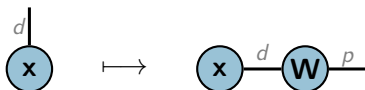
Hyperedge \equiv contraction between more than 2 indices:



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

Multilinear Maps

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

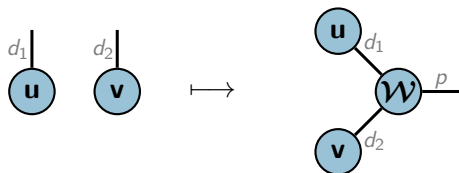


Multilinear Maps

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



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

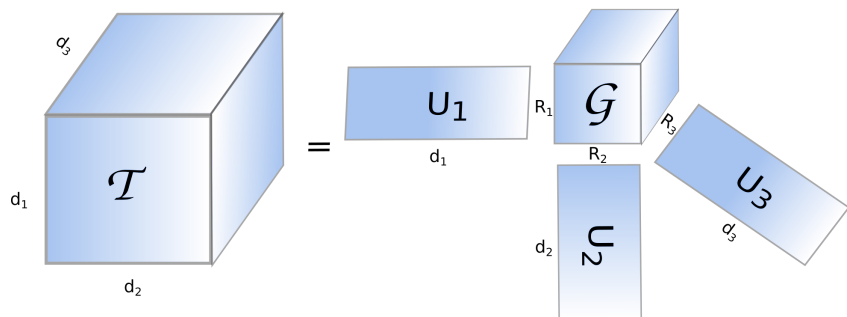
Simple idea: decompose a tensor into product of small factors.

- 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

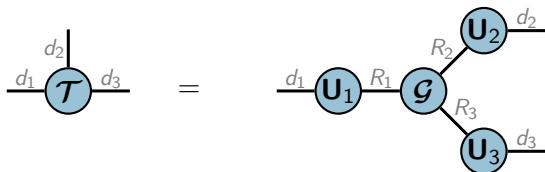
- Tucker decomposition [Tucker, 1966]:



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

Tensor Decomposition Techniques

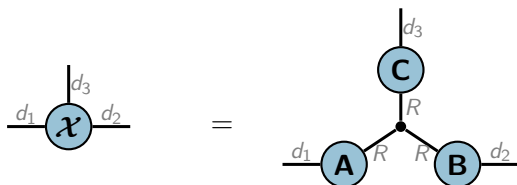
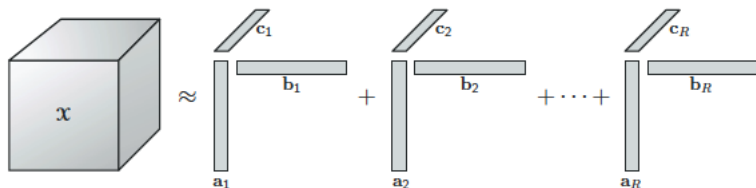
- Tucker decomposition [Tucker, 1966]:



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

Tensor Decomposition Techniques

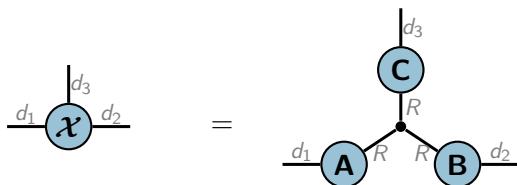
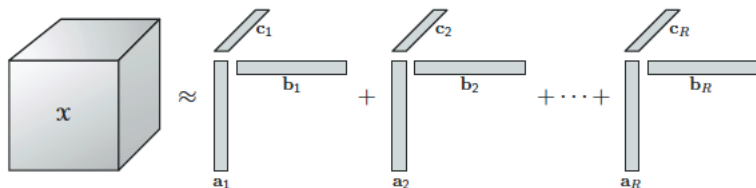
- CP decomposition [Hitchcock, 1927]²:



²fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].

Tensor Decomposition Techniques

- CP decomposition [Hitchcock, 1927]²:

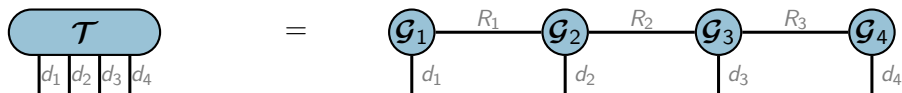


$\Rightarrow R(d_1 + d_2 + d_3)$ parameters instead of $d_1 d_2 d_3$.

²fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].

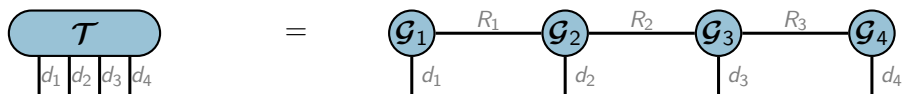
Tensor Decomposition Techniques

- Tensor Train decomposition [Oseledets, 2011]:



Tensor Decomposition Techniques

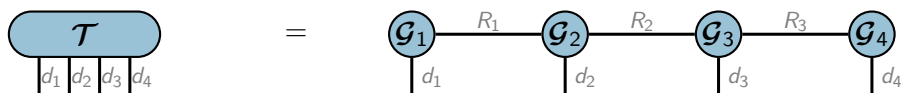
- Tensor Train decomposition [Oseledets, 2011]:



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

Tensor Decomposition Techniques

- Tensor Train decomposition [Oseledets, 2011]:

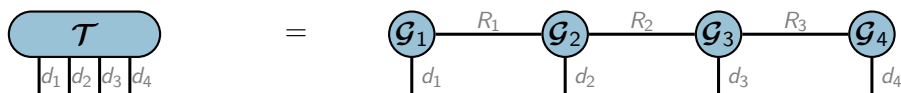


$\Rightarrow d_1 R_1 + R_1 d_2 R_2 + R_2 d_3 R_3 + R_3 d_4$ parameters instead of $d_1 d_2 d_3 d_4$.

- 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 Train decomposition [Oseledets, 2011]:

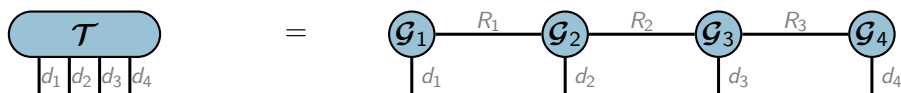


$\Rightarrow d_1 R_1 + R_1 d_2 R_2 + R_2 d_2 R_3 + R_3 d_3$ parameters instead of $d_1 d_2 d_3 d_4$.

- 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!
- We can also efficiently perform operations on TT tensors:
 - ▶ Inner product, sum, component-wise product, ... all in time linear in n for vectors of size d^n .

Tensor Decomposition Techniques

- Tensor Train decomposition [Oseledets, 2011]:

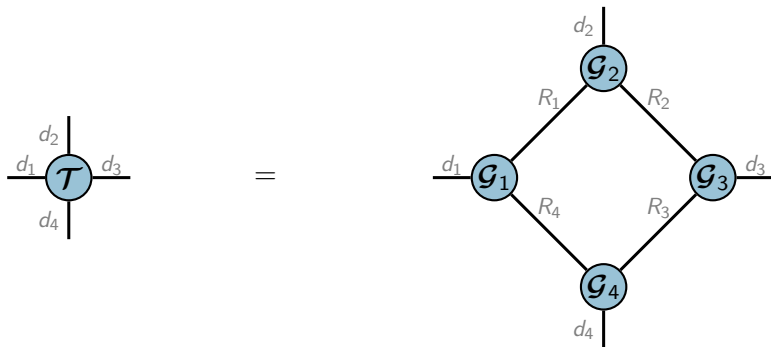


$\Rightarrow d_1 R_1 + R_1 d_2 R_2 + R_2 d_2 R_3 + R_3 d_3$ parameters instead of $d_1 d_2 d_3 d_4$.

- 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!
- We can also efficiently perform operations on TT tensors:
 - ▶ Inner product, sum, component-wise product, ... all in time linear in n for vectors of size d^n .
- Limitations:
 - ▶ not all tensors have low TT rank
 - ▶ not possible to apply component-wise non-linear functions in the TT format...

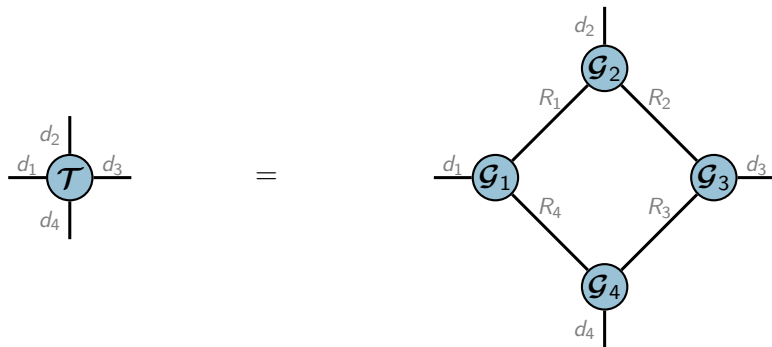
Tensor Decomposition Techniques

- 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): $\mathcal{O}(NdR^2)$ parameters

where the rank $R = \max_j R_j$.

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): $\mathcal{O}(NdR^2)$ parameters

where the rank $R = \max_j R_j$.

- Finding the exact low rank decomposition of a tensor is NP hard for CP but can be done in polynomial time for Tucker and TR
- Low rank approximation problem is NP hard for all decomposition models
- Efficient approximation algorithm exists for the low rank approximation problem

Tensor Networks: Summary

- Tensor networks \equiv graphical notation to describe complex operations on tensors

Tensor Networks: Summary

- Tensor networks \equiv graphical notation to describe complex operations on tensors
 - Tensor decomposition \equiv efficient way to compress high dimensional objects
- \hookrightarrow can be used to compress neural networks (e.g., [Novikov et al., 2015])

Tensor Networks: Summary

- Tensor networks \equiv graphical notation to describe complex operations on tensors
- Tensor decomposition \equiv efficient way to compress high dimensional objects
- \hookrightarrow can be used to compress neural networks (e.g., [Novikov et al., 2015])
- Tensor network methods \equiv algorithms to efficiently perform operations on (or optimize) very high dimensional objects

Tensor Networks: Summary

- Tensor networks \equiv graphical notation to describe complex operations on tensors
- Tensor decomposition \equiv efficient way to compress high dimensional objects
- \hookrightarrow can be used to compress neural networks (e.g., [Novikov et al., 2015])
- Tensor network methods \equiv algorithms to efficiently perform operations on (or optimize) very high dimensional objects
- \Rightarrow Lots of interesting open problems and connections with quantum physics and formal languages.

Tensor Networks: Summary

- Tensor networks \equiv graphical notation to describe complex operations on tensors
- Tensor decomposition \equiv efficient way to compress high dimensional objects
- \hookrightarrow can be used to compress neural networks (e.g., [Novikov et al., 2015])
- Tensor network methods \equiv algorithms to efficiently perform operations on (or optimize) very high dimensional objects
- \Rightarrow Lots of interesting open problems and connections with quantum physics and formal languages.
- \Rightarrow **Tensors are the new matrices** (linear \rightarrow multilinear) and tensor networks make it "easy" to reason about tensors, tensor decomposition and multi-linear algebra.

Outline

- 1 An Introduction to Tensors and Tensor Networks
- 2 Adaptive Learning of Tensor Decomposition Models

Joint work with Meraj Hashemizadeh, Michelle Liu and Jacob Miller



Tensor Decomposition Techniques

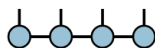
- Lots of ways to decompose a tensor:



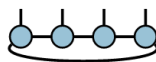
CP



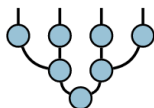
Tucker



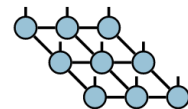
Tensor Train



Tensor Ring



Hierarchical Tucker



PEPS

- ⇒ 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

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

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

Tensor based optimization problems

- A lot of tensor problems can be formulated as

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

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 based optimization problems

- A lot of tensor problems can be formulated as

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

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

- ▶ Tensor Classification

$$L(\mathcal{W}) = \sum_{i=1}^N \text{CCE}(y_i, f(\mathbf{x}_i)) \quad \text{where } f(\mathbf{x}_i) = \text{sign}(\langle \mathcal{W}, \mathbf{x}_i \rangle)$$

Tensor based optimization problems

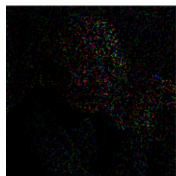
- A lot of tensor problems can be formulated as

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

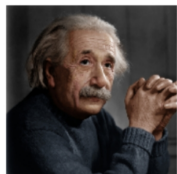
where L is a loss function and rank is some notion of tensor rank (e.g. TT, TR, CP, ...).

- ▶ Tensor Completion

Observed pixels



Original image



X
 $n \times m$

4	3		7	5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

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 Completion

$$L(\mathcal{W}) = \sum_{(i,j,k) \in \Omega} (\mathcal{W}_{ijk} - \mathbf{x}_{ijk})^2$$

where Ω is the set of observed entries

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:

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

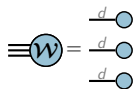
A greedy algorithm for adaptive learning of TN structures

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

- **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.

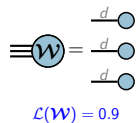
Greedy Algorithm Overview

- Start with a random rank one tensor.



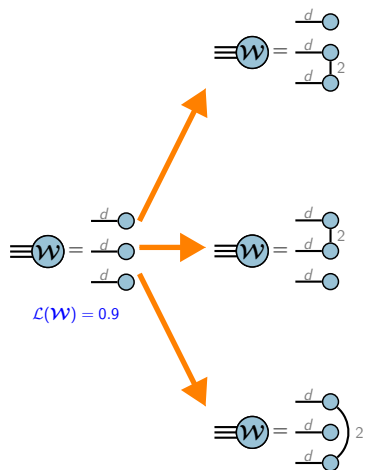
Greedy Algorithm Overview

- Optimize the loss wrt the core tensors.



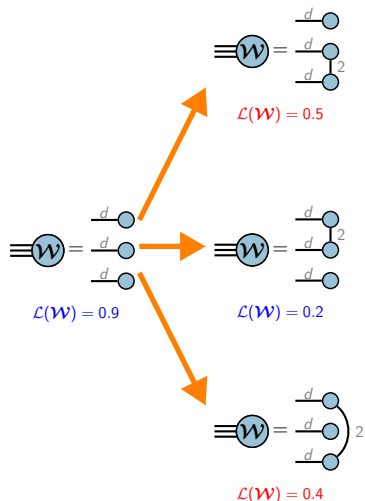
Greedy Algorithm Overview

- Consider all possible rank one increments on internal edges.



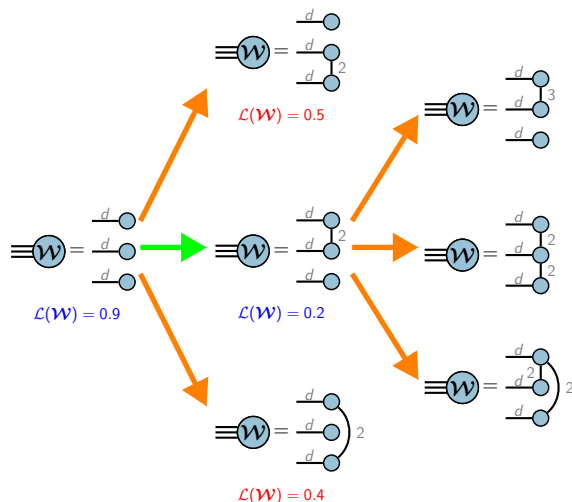
Greedy Algorithm Overview

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



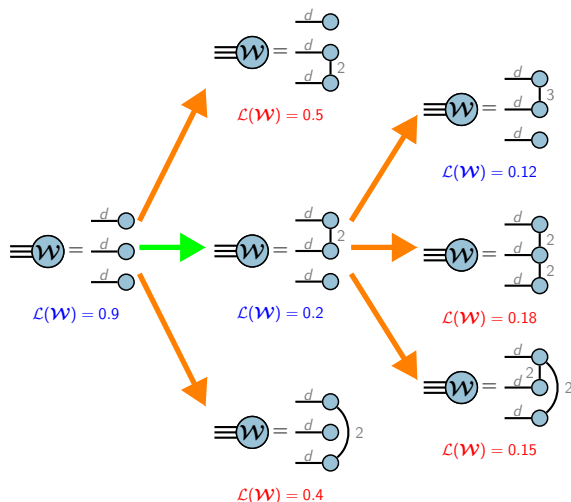
Greedy Algorithm Overview

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



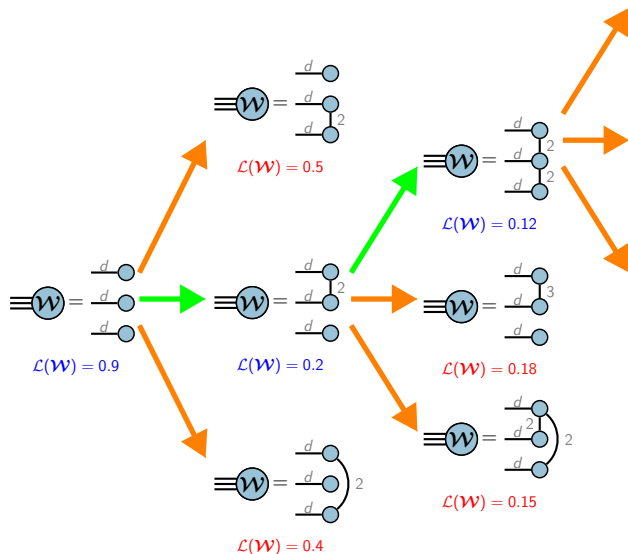
Greedy Algorithm Overview

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



Greedy Algorithm Overview

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

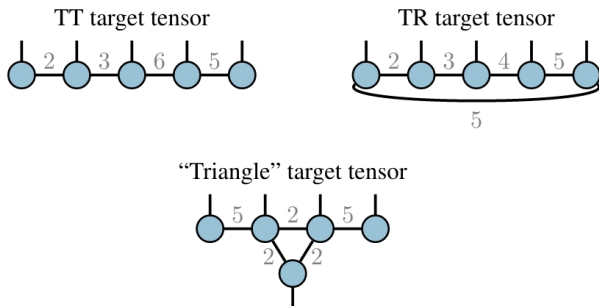


Implementation Details and Limitations

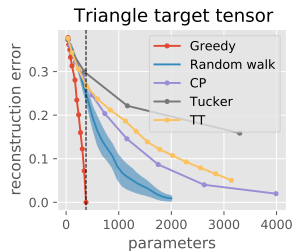
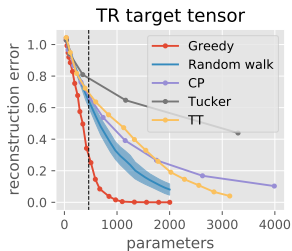
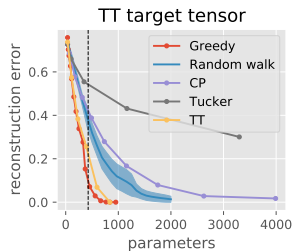
- At each iteration of greedy, we restart the optimization from the previous solution.
- No internal nodes are added to the initial TN structure (cannot represent Tucker).
- No hyperedge (cannot represent CP).
- Computationally expensive.

Experiment: Tensor decomposition

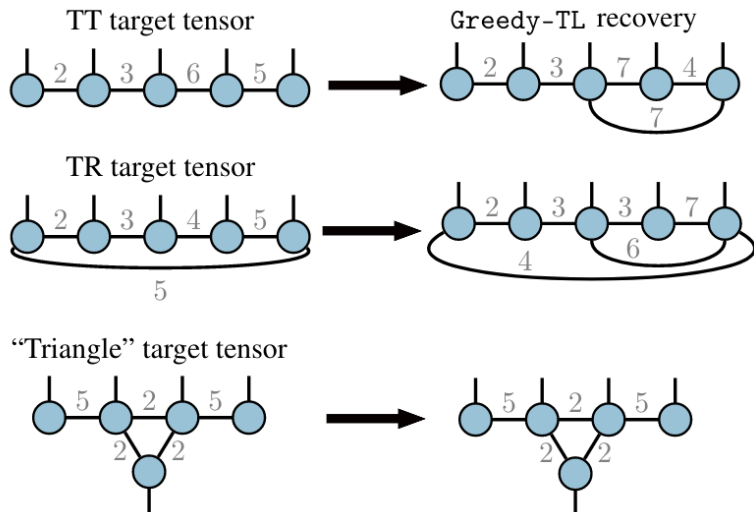
- Objective: compress a given tensor (with unknown tensor network structure) by decomposing it.
- Three target tensors of size $7 \times 7 \times 7 \times 7 \times 7$:



Experiment: Tensor decomposition

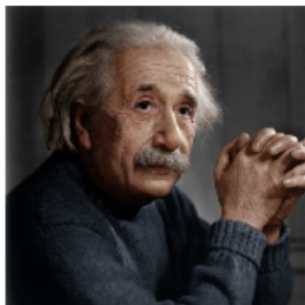


Tensor structures recovered by Greedy

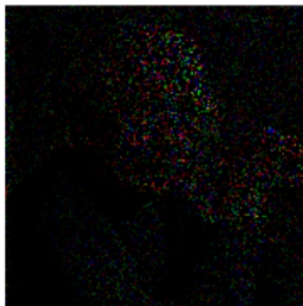


Experiment: Tensor completion

Original
image



Observed
pixels



- Initial image is reshaped into a $6 \times 10 \times 10 \times 6 \times 10 \times 10 \times 3$ tensor

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.



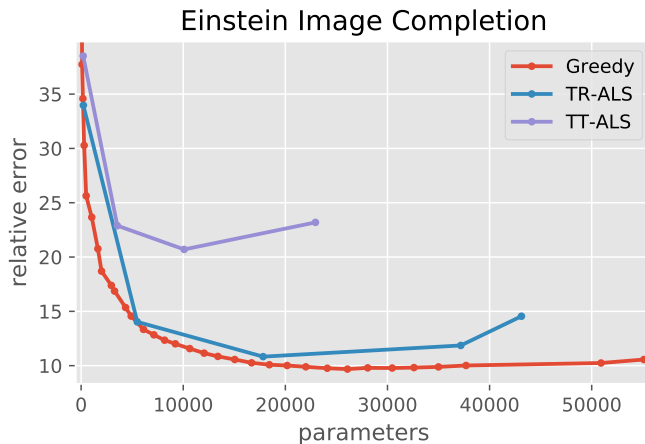
Greedy (iter=12)
4905 param.



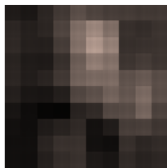
Greedy (iter=31)
37695 param.



Experiment: Tensor completion



Experiment: Tensor completion



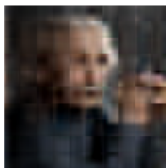
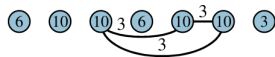
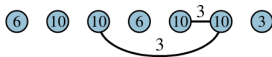
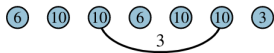
Greedy (iter 2)



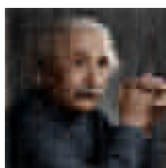
Greedy (iter 3)



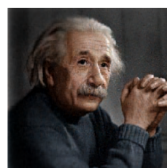
Greedy (iter 4)



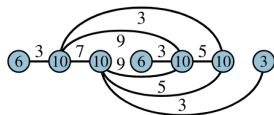
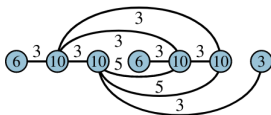
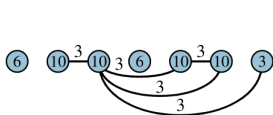
Greedy (iter 6)



Greedy (iter 12)



Greedy (iter 31)



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

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
- Future directions (ongoing):
 - ▶ Theory: convergence rate analysis
 - ▶ 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

Conclusion

- Forget matrices and linear algebra... **Tensors and multilinear algebra!**
- Tensor networks \equiv unifying language for tensor methods
- Lots of interesting open problems
- Promising direction: general tensor methods with tensor networks

Conclusion

- Forget matrices and linear algebra... **Tensors and multilinear algebra!**
- Tensor networks \equiv unifying language for tensor methods
- Lots of interesting open problems
- Promising direction: general tensor methods with tensor networks
- Other relevant recent work from my group at Mila:
 - ▶ Tensorized Random Projections with Beheshteh T. Rakhshan
 - ▶ VC dimension of Tensor Network models with Behnoush Khavari
 - ▶ Connections between tensor networks, RNNs and weighted automata with Tianyu Li, Maude Lizaire, Simon Verret
 - ▶ Tensor networks for sequence modeling with Jacob Miller

Conclusion

- Forget matrices and linear algebra... **Tensors and multilinear algebra!**
- Tensor networks \equiv unifying language for tensor methods
- Lots of interesting open problems
- Promising direction: general tensor methods with tensor networks
- Other relevant recent work from my group at Mila:
 - ▶ Tensorized Random Projections with Beheshteh T. Rakhshan
 - ▶ VC dimension of Tensor Network models with Behnoush Khavari
 - ▶ Connections between tensor networks, RNNs and weighted automata with Tianyu Li, Maude Lizaire, Simon Verret
 - ▶ Tensor networks for sequence modeling with Jacob Miller

Thank you! Questions?