Tensor Networks for Machine Learning

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

January 29, 2021
Séminaire Signal et Apprentissage, Groupe ALEA
Learning with Structured Data

Supervised Learning:

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

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.
Learning with Structured Data

Supervised Learning:

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

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

- 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, ...
Learning with Structured Data

Supervised Learning:

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

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

- 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. Tensor Networks for ML
   - Adaptive Learning of Tensor Decomposition Models
   - VC dimension of Tensor Network Models
   - Tensorized Random Projections
Tensors

\[ M \in \mathbb{R}^{d_1 \times d_2} \]
\[ M_{ij} \in \mathbb{R} \text{ for } i \in [d_1], j \in [d_2] \]

\[ T \in \mathbb{R}^{d_1 \times d_2 \times d_3} \]
\[ (T_{ijk}) \in \mathbb{R} \text{ for } i \in [d_1], j \in [d_2], k \in [d_3] \]
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

\[ M \in \mathbb{R}^{d_1 \times d_2} \]
\[ M_{ij} \in \mathbb{R} \text{ for } i \in [d_1], j \in [d_2] \]

\[ T \in \mathbb{R}^{d_1 \times d_2 \times d_3} \]
\[ (T_{ijk}) \in \mathbb{R} \text{ for } i \in [d_1], j \in [d_2], k \in [d_3] \]
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 R or 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\(^1\):

  (a) Mode-1 (column) fibers: \(x_{jk}\)
  (b) Mode-2 (row) fibers: \(x_{ik}\)
  (c) Mode-3 (tube) fibers: \(x_{ij}\)

---

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

---

\(^1\)fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].
Tensors: Multiplication with Matrices

\[
\begin{align*}
A & \quad \quad d_1 \\
m_1 & \\
M & \quad \quad d_2 \\
B & \quad \quad m_2
\end{align*}
\]

Ex: If \( T \in \mathbb{R}^{d_1 \times d_2 \times d_3} \) and \( A \in \mathbb{R}^{m_1 \times d_1} \), \( B \in \mathbb{R}^{m_2 \times d_2} \), \( C \in \mathbb{R}^{m_3 \times d_3} \), then

\[
(T \times A \times B \times C)_{i_1, i_2, i_3} = \sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \sum_{k_3=1}^{n_3} T_{k_1} A_{i_1 k_1} B_{i_2 k_2} C_{i_3 k_3}
\]

for all \( i_1 \in [d_1], i_2 \in [m_2], i_3 \in [d_3] \).
Tensors: Multiplication with Matrices

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

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

\[ \text{AMB}^\top \in \mathbb{R}^{m_1 \times m_2} \]

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

**ex:** If \( \mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3} \) and \( A \in \mathbb{R}^{m_1 \times d_1} \), \( B \in \mathbb{R}^{m_2 \times d_2} \), \( C \in \mathbb{R}^{m_3 \times d_3} \), then

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

for all \( i_1 \in [d_1], i_2 \in [m_2], i_3 \in [d_3] \).
Degree of a node $\equiv$ order of tensor

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

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

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

Degree of a node $\equiv$ order of tensor

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

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

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

Edge $\equiv$ contraction

Matrix product:

$(\mathbf{A}\mathbf{B})_{i_1,i_2} = \sum_{k=1}^{n} \mathbf{A}_{i_1k} \mathbf{B}_{ki_2}$
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:

$\mathbf{u} \mathbf{v}^\top = \sum_{k=1}^{n} u_k v_k$
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 between tensors:

$\langle \mathbf{S}, \mathbf{V} \rangle = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \sum_{i_3=1}^{d_3} \mathbf{S}_{i_1i_2i_3} \mathcal{T}_{i_1i_2i_3}$
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

Frobenius norm of a tensor:

$$\| \mathbf{S} \|_F^2 = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \sum_{i_3=1}^{d_3} (\mathbf{S}_{i_1i_2i_3})^2$$
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

Trace of an $n \times n$ matrix:

$\text{Tr}(\mathbf{M}) = \sum_{i=1}^{n} M_{ii}$
**Tensor Networks**

Degree of a node $\equiv$ order of tensor

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

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

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

Edge $\equiv$ contraction

Tensor times matrices:

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

Degree of a node $\equiv$ order of tensor

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

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

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

Edge $\equiv$ contraction

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

$\sum_{i=1}^{n} u_i v_i w_i$
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 $d \times d \times \cdots \times d$: $d^{10}$ parameters
  - ...
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 $\mathbf{M} = \mathbf{A}\mathbf{B}$ with $\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times n}$
Simple idea: decompose a tensor into product of small factors.

Similar to matrix factorization:

- If \( M \in \mathbb{R}^{m \times n} \) and \( M = AB \) with \( A \in \mathbb{R}^{m \times r} \) and \( B \in \mathbb{R}^{r \times n} \)
  \( \Rightarrow r(m + n) \) parameters instead of \( mn \)...
Tensor Decomposition Techniques

- **Tucker decomposition** [Tucker, 1966]:

$$\mathcal{T} = \sum_{i=1}^{d_1} U_1^i \mathcal{G} R_1^i + \sum_{i=1}^{d_2} U_2^i \mathcal{G} R_2^i + \sum_{i=1}^{d_3} U_3^i \mathcal{G} R_3^i$$

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

Guillaume Rabusseau

Tensor networks for ML

January 29, 2021
Tensor Decomposition Techniques

- **Tucker decomposition** [Tucker, 1966]:

\[
\mathbf{T} = \mathbf{G} \mathbf{U}_1 \mathbf{U}_2 \mathbf{U}_3
\]

\[
R_1 R_2 R_3 + d_1 R_1 + d_2 R_2 + d_2 R_2 \text{ parameters instead of } d_1 d_2 d_3.
\]
Tensor Decomposition Techniques

- **CP decomposition** [Hitchcock, 1927]²:

  \[ x \approx \sum_{r=1}^{R} \left( a_1^{r} \cdot c_1^{r} b_1^{r} + a_2^{r} \cdot c_2^{r} b_2^{r} + \cdots + a_R^{r} \cdot c_R^{r} b_R^{r} \right) \]

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

- **CP decomposition** [Hitchcock, 1927]$^2$:

  \[
  x \approx d_1 b_1 + d_2 b_2 + \cdots + d_R b_R
  \]

  \[
  \Rightarrow R(d_1 + d_2 + d_3) \text{ parameters instead of } d_1 d_2 d_3.
  \]

$^2$fig. from [Kolda and Bader, *Tensor decompositions and applications*, 2009].
Tensor Decomposition Techniques

- **Tensor Train decomposition** [Oseledets, 2011]:

\[
\mathcal{T} = G_1 \circ R_1 + G_2 \circ R_2 + G_3 \circ R_3 + G_4 \circ R_4
\]

If the ranks are all the same (\(R_1 = R_2 = \cdots = R_n\)), can represent a vector of size \(2^n\) with \(O(nR^2)\) 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]:

\[ \mathcal{T} = G_1 \otimes \cdots \otimes G_4 \]

\[ \Rightarrow d_1 R_1 + R_1 d_2 R_2 + R_2 d_2 R_3 + R_3 d_4 \]

parameters instead of \( d_1 d_2 d_3 d_4 \).
Tensor Decomposition Techniques

- **Tensor Train decomposition** [Oseledets, 2011]:

  \[
  \tau = G_1 \circ G_2 \circ G_3 \circ G_4
  \]

  ⇒ \(d_1 R_1 + R_1 d_2 R_2 + R_2 d_2 R_3 + R_3 d_4\) parameters instead of \(d_1 d_2 d_3 d_4\).

  - If the ranks are all the same \((R_1 = R_2 = \cdots = R)\), can represent a vector of size \(2^n\) with \(O(nR^2)\) parameters!
Tensor Decomposition Techniques

- **Tensor Train decomposition** [Oseledets, 2011]:

\[
\mathcal{T} = G_1 \underbrace{d_1 \ldots d_4}_{d_1 R_1 + R_1 d_2 R_2 + R_2 d_2 R_3 + R_3 d_4 \text{ parameters instead of } d_1 d_2 d_3 d_4.}
\]

- If the ranks are all the same \( (R_1 = R_2 = \cdots = R) \), can represent a vector of size \( 2^n \) with \( O(nR^2) \) 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 Ring decomposition** [Zhao et al., 2016]:

\[
\mathcal{T} = \mathcal{G}_1 \mathcal{G}_2 \mathcal{G}_3 \mathcal{G}_4
\]

\[
\Rightarrow \mathcal{G} = \mathcal{G}_1 \mathcal{R}_1 \mathcal{G}_2 + \mathcal{G}_2 \mathcal{R}_2 \mathcal{G}_3 + \mathcal{G}_3 \mathcal{R}_3 \mathcal{G}_4 + \mathcal{G}_4 \mathcal{R}_4 \mathcal{G}_1
\]
Tensor Decomposition Techniques

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

\[
\mathcal{T} = R_4 d_1 R_1 + R_1 d_2 R_2 + R_2 d_2 R_3 + R_3 d_4 R_4 \quad \text{parameters instead of} \quad d_1 d_2 d_3 d_4.
\]
Tensor Networks: Summary

- Tensor networks ≡ graphical notation to describe complex operations on tensors
Tensor Networks: Summary

- Tensor networks ≡ graphical notation to describe complex operations on tensors
- Tensor decomposition ≡ efficient way to compress high dimensional objects

\( \rightarrow \) 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

$\rightarrow$ 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
  $\leftrightarrow$ 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
  $\leftrightarrow$ 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. Tensor Networks for ML
   - Adaptive Learning of Tensor Decomposition Models
   - VC dimension of Tensor Network Models
   - Tensorized Random Projections

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

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

where \( L \) is a loss function and \( \text{rank} \) is some notion of tensor rank (e.g. TT, TR, CP, ...).
Tensor based optimization problems

- A lot of tensor problems can be formulated as

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

where \( L \) is a loss function and \( \text{rank} \) 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

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

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

- Tensor Classification

$$L(\mathcal{W}) = \sum_{i=1}^{N} CCE(y_i, f(x_i)) \quad \text{where} \quad f(x_i) = \text{sign}(\langle \mathcal{W}, x_i \rangle)$$
Tensor based optimization problems

- A lot of tensor problems can be formulated as

\[
\min_{\mathbf{W} \in \mathbb{R}^{d_1 \times \cdots \times d_p}} L(\mathbf{W}) \quad \text{s.t. } \text{rank}(\mathbf{W}) \leq R
\]

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

- Tensor Completion
Tensor based optimization problems

- A lot of tensor problems can be formulated as

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

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

- Tensor Completion

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

where $\Omega$ is the set of observed entries.
A greedy algorithm for adaptive learning of TN structures

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

- 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

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

- 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.

\[\rightarrow\] We optimize the loss both with respect to the TN structure and the core tensors of the TN:

\[
\min_{\text{Tensor Network Structure \ TN}} \min_{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(p)}} L(TN(\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(p)})) \\
\quad \text{s.t.} \quad \text{size}(\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(p)}) \leq C
\]
A greedy algorithm for adaptive learning of TN structures

\[
\min_{\text{Tensor Network Structure } TN} \min_{\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(p)}} L(TN(\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(p)}))
\]
\[
\text{s.t. } \text{size}(\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(p)}) \leq C
\]

- **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.

\[ \mathcal{L}(\mathcal{W}) = 0.9 \]
Greedy Algorithm Overview

- Consider all possible rank one increments on internal edges.

\[ \mathcal{L}(\mathcal{W}) = 0.9 \]
Greedy Algorithm Overview

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

\[ V = d \]
\[ L(W) = 0.9 \]
\[ L(W) = 0.2 \]
\[ L(W) = 0.4 \]
\[ L(W) = 0.5 \]
Greedy Algorithm Overview

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

![Diagram of greedy algorithm iterations with rank increments and corresponding loss values.]

- $\mathcal{L}(\mathcal{W}) = 0.9$
- $\mathcal{L}(\mathcal{W}) = 0.2$
- $\mathcal{L}(\mathcal{W}) = 0.4$
- $\mathcal{L}(\mathcal{W}) = 0.5$
- $\mathcal{L}(\mathcal{W}) = 0.9$
Greedy Algorithm Overview

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

\[
L(W) = 0.9 \rightarrow L(W) = 0.2 \rightarrow L(W) = 0.4
\]

\[
L(W) = 0.5 \rightarrow L(W) = 0.12 \rightarrow L(W) = 0.18 \rightarrow L(W) = 0.15
\]
Greedy Algorithm Overview

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

\[ \mathcal{L}(\mathbf{W}) = 0.5 \]

\[ \mathcal{L}(\mathbf{W}) = 0.9 \]

\[ \mathcal{L}(\mathbf{W}) = 0.2 \]

\[ \mathcal{L}(\mathbf{W}) = 0.12 \]

\[ \mathcal{L}(\mathbf{W}) = 0.18 \]

\[ \mathcal{L}(\mathbf{W}) = 0.15 \]

\[ \mathcal{L}(\mathbf{W}) = 0.4 \]
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 $7 \times 7 \times 7 \times 7 \times 7$:

```
TT target tensor
```

```
TR target tensor
```

```
“Triangle” target tensor
```
Experiment: Tensor decomposition

**TT target tensor**

- Greedy
- Random walk
- CP
- Tucker
- TT

**TR target tensor**

- Greedy
- Random walk
- CP
- Tucker
- TT

**Triangle target tensor**

- Greedy
- Random walk
- CP
- Tucker
- TT
Tensor structures recovered by Greedy

**TT target tensor**

\[
\begin{array}{cccccc}
2 & 3 & 6 & 5 & 7 \\
\end{array}
\]

**Greedy-TL recovery**

\[
\begin{array}{cccccc}
2 & 3 & 7 & 4 & 7 \\
\end{array}
\]

**TR target tensor**

\[
\begin{array}{cccccc}
2 & 3 & 4 & 5 & 7 \\
\end{array}
\]

**“Triangle” target tensor**

\[
\begin{array}{cccccc}
5 & 2 & 2 & 5 & 2 \\
\end{array}
\]
Experiment: Tensor completion

- Initial image is reshaped into a $6 \times 10 \times 10 \times 6 \times 10 \times 10 \times 3$ tensor.
### Experiment: Tensor completion

<table>
<thead>
<tr>
<th>Method</th>
<th>Rank</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT (rank=2)</td>
<td>2</td>
<td>202 param.</td>
</tr>
<tr>
<td>TR (rank=2)</td>
<td>2</td>
<td>220 param.</td>
</tr>
<tr>
<td>TT (rank=10)</td>
<td>10</td>
<td>3547 param.</td>
</tr>
<tr>
<td>TR (rank=10)</td>
<td>10</td>
<td>5500 param.</td>
</tr>
<tr>
<td>TT (rank=18)</td>
<td>18</td>
<td>10093 param.</td>
</tr>
<tr>
<td>TR (rank=18)</td>
<td>18</td>
<td>17820 param.</td>
</tr>
<tr>
<td>TT (rank=26)</td>
<td>26</td>
<td>19967 param.</td>
</tr>
<tr>
<td>TR (rank=26)</td>
<td>26</td>
<td>37180 param.</td>
</tr>
<tr>
<td>Greedy (iter=4)</td>
<td>4</td>
<td>295 param.</td>
</tr>
<tr>
<td>Greedy (iter=6)</td>
<td>6</td>
<td>1041 param.</td>
</tr>
<tr>
<td>Greedy (iter=10)</td>
<td>10</td>
<td>3273 param.</td>
</tr>
<tr>
<td>Greedy (iter=12)</td>
<td>12</td>
<td>4905 param.</td>
</tr>
<tr>
<td>Greedy (iter=17)</td>
<td>17</td>
<td>10635 param.</td>
</tr>
<tr>
<td>Greedy (iter=23)</td>
<td>23</td>
<td>20175 param.</td>
</tr>
<tr>
<td>Greedy (iter=26)</td>
<td>26</td>
<td>26085 param.</td>
</tr>
<tr>
<td>Greedy (iter=31)</td>
<td>31</td>
<td>37695 param.</td>
</tr>
</tbody>
</table>
Experiment: Tensor completion

Einstein Image Completion

Greedy
TR-ALS
TT-ALS
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
Outline

1. An Introduction to Tensors and Tensor Networks

2. Tensor Networks for ML
   - Adaptive Learning of Tensor Decomposition Models
   - VC dimension of Tensor Network Models
   - Tensorized Random Projections

Joint work with Behnoush Khavari
Classification problem: Given

- a sample \( S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathcal{X} \times \{-1, +1\} \) drawn i.i.d. from an unknown distribution \( D \)
- a class of hypothesis \( \mathcal{H} \subset \{-1, 1\}^\mathcal{X} \)
- a loss function \( \ell \)

we want to find \( h \in \mathcal{H} \) which minimizes the risk

\[
R(h) = \mathbb{E}_{(x, y) \sim D} \ell(h(x), y)
\]
Statistical Framework of Learning

- **Classification problem**: Given
  - a sample $S = \{(x_1, y_1), \cdots, (x_n, y_n)\} \subset X \times \{-1, +1\}$ drawn i.i.d. from an unknown distribution $D$
  - a class of hypothesis $\mathcal{H} \subset \{-1, 1\}^X$
  - a loss function $\ell$

  we want to find $h \in \mathcal{H}$ which minimizes the risk

  $$R(h) = \mathbb{E}_{(x,y) \sim D} \ell(h(x), y)$$

- **Empirical Risk Minimization**: $D$ is unknown, instead we minimize the empirical risk on $S$

  $$\hat{R}_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i)$$
Statistical Framework of Learning

Classification problem: Given
- a sample $S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathcal{X} \times \{-1, +1\}$ drawn i.i.d. from an unknown distribution $D$
- a class of hypothesis $\mathcal{H} \subset \{-1, 1\}^\mathcal{X}$
- a loss function $\ell$

we want to find $h \in \mathcal{H}$ which minimizes the risk

$$R(h) = \mathbb{E}_{(x, y) \sim D} \ell(h(x), y)$$

Empirical Risk Minimization: $D$ is unknown, instead we minimize the empirical risk on $S$

$$\hat{R}_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i)$$

$\hat{R}_S(h)$ is an unbiased estimator of $R(h)$. The quality of this estimator depends on
- the sample size $n$
- the complexity of the hypothesis class $\mathcal{H}$. 
Statistical Framework of Learning

- We want to find \( h \in \mathcal{H} \) which minimizes the risk

\[
R(h) = \mathbb{E}_{(x,y) \sim D} \ell(h(x), y)
\]

- **Empirical Risk Minimization:** \( D \) is unknown, instead we minimize the empirical risk on \( S \)

\( \hat{R}_S(h) \) is an unbiased estimator of \( R(h) \). The quality of this estimator depends on
  - the sample size \( n \)
  - the complexity of the hypothesis class \( \mathcal{H} \).

- Many measures of complexity, here we focus on the VC dimension
  - \( d_{VC}(\mathcal{H}) \): maximum number of points that hypothesis in \( \mathcal{H} \) can separate in all \( 2^n \) ways.
Problem

- We consider a classification problem with tensors:

\[ R(h) = \sum_{i=1}^{n} \ell(y_i, h(x_i)) \text{ where } h(x_i) = \text{sign}(\langle \mathcal{W}, x_i \rangle) \]

and \( \mathcal{W} \) is parameterized as a tensor network with an arbitrary structure

- Goal: Characterize the complexity of the hypothesis class \( \mathcal{H}_G \) for an arbitrary tensor network structure \( G \).
Main Result

Theorem

Let $G = (V, E, \text{dim})$ be a tensor network structure, let $\mathcal{H}_G$ be the corresponding hypothesis class. The following hold:

- $d_{\text{VC}}(\mathcal{H}_G) \leq 2N_G \log(12|V|)$

where

- $N_G$ is the number of parameters in the tensor network
- $|V|$ is the number of vertices in the tensor network structure
Main Result

Theorem

Let $G = (V, E, \text{dim})$ be a tensor network structure, let $\mathcal{H}_G$ be the corresponding hypothesis class. The following hold:

- $d_{VC}(\mathcal{H}_G) \leq 2N_G \log(12|V|)$
- For any $\delta > 0$, with probability at least $1 - \delta$ over the choice of $S$, we have for any $h \in \mathcal{H}_G$

$$R(h) < \hat{R}_S(h) + 2\sqrt{\frac{2}{n} \left( N_G \log \frac{8en|V|}{N_G} + \log \frac{4}{\delta} \right)}.$$ 

where

- $N_G$ is the number of parameters in the tensor network
- $|V|$ is the number of vertices in the tensor network structure
Discussion

- Our bound can be applied to any family of multilinear low rank classifiers, e.g. low rank matrix classifiers.
- Our bound improves previous results in special cases (e.g. low rank matrices).
- Future work:
  - lower bounds
  - structured risk minimization
Outline

1. An Introduction to Tensors and Tensor Networks

2. Tensor Networks for ML
   - Adaptive Learning of Tensor Decomposition Models
   - VC dimension of Tensor Network Models
   - Tensorized Random Projections

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.).

Johnson-Lindenstrauss Transform (or Gaussian RP).

$f : x \mapsto \sqrt{\frac{1}{k}} M x$ where $M_{ij} \sim iid \mathcal{N}(0, 1)$ for each $i, j$.

Theorem (JL, 1984)

Let $\varepsilon > 0$ and $x_1, \ldots, x_m \in \mathbb{R}^d$.

If $k \gg \varepsilon^{-2} \log m$, then, with high proba.,

$\|f(x_i)\| = (1 \pm \varepsilon) \|x_i\|$ for all $i = 1, \ldots, m$.

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

Problem: if $x \in \mathbb{R}^d$ is a high-order tensor represented in CP/TT format, the Gaussian RP has $d N k$ parameters...
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).

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

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

Theorem (JL, 1984)

Let $\varepsilon > 0$ and $\mathbf{x}_1, \cdots, \mathbf{x}_m \in \mathbb{R}^d$.
If $k \gtrsim \varepsilon^{-2} \log m$, then, with high proba., $\|f(\mathbf{x}_i)\| = (1 \pm \varepsilon)\|\mathbf{x}_i\|$ for all $i = 1, \cdots, m$.

- 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).

\[ f : x \mapsto \frac{1}{\sqrt{k}} Mx \quad \text{where } M_{ij} \sim_{iid} \mathcal{N}(0, 1) \text{ for each } i, j \]

Theorem (JL, 1984)

Let $\varepsilon > 0$ and $x_1, \ldots, x_m \in \mathbb{R}^d$.

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

- Applications: sketched linear regression, randomized SVD, pre-processing step in ML pipeline, ...
- Pbm: if $x \in \mathbb{R}^{d^N}$ is a high-order tensor represented in CP/TT format, the Gaussian RP has $d^Nk$ 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 $f(x)$ is efficient when $x$ is in the CP or TT format  
  - $f$ preserves distances with high probability.
Objective

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

- Two important properties that a RP must satisfy:
  - $\mathbb{E}[\|f(x)\|^2] = \|x\|^2$ for all $x$
  - $\lim_{k \to \infty} \mathbb{V}[\|f(x)\|^2] = 0$
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 $f(x)$ is efficient when $x$ is in the CP or TT format
  - $f$ preserves distances with high probability.

- Two important properties that a RP must satisfy:
  - $\mathbb{E}[\|f(x)\|^2] = \|x\|^2$ for all $x$
  - $\lim_{k \rightarrow \infty} \nabla[\|f(x)\|^2] = 0$

$\Rightarrow$ the rate at which $\nabla[\|f(x)\|^2]$ converges to 0 captures the quality of a RP.
Tensor Train Random Projection (TT-RP)

- Tensor Train RP:

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

where each row of \( M = \begin{pmatrix} m_1^\top & m_2^\top & \cdots & m_k^\top \end{pmatrix} \in \mathbb{R}^{k \times d^N} \) is in the TT format:

\[
m_i \quad = \quad G^i_1 \quad R \quad G^i_2 \quad R \quad G^i_3 \quad R \quad G^i_4
\]

for each \( i = 1, \cdots, k \)

and the entries of each core tensor \( G^i_n \) are drawn iid from \( \mathcal{N}(0, 1) \).
Tensor Train Random Projection (TT-RP)

- 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 \( \mathbf{G}^i_n \) are drawn iid from \( \mathcal{N}(0, 1) \).

- \( O\left(kNdR^2\right) \) parameters instead of \( d^Nk \).
- 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 \( \lim_{k \to \infty} \mathbb{V}[^{\|f_{TT}(R)(\mathbf{x})\|^2}] = 0 \).
CP Random projection (CP-RP)

- CP Random Projection:

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

where each row of \( M = \begin{pmatrix} -m_1^T & -m_2^T & \cdots & -m_k^T \end{pmatrix} \in \mathbb{R}^{k \times d^N} \) is in the CP format:

\[
\begin{array}{l}
\begin{array}{l}
g_{i1}^i \\
g_{i2}^i \\
g_{i3}^i \\
g_{i4}^i \\
m_i
\end{array}
\end{array}
\]

and the entries of each core tensor \( g_{i}^n \) are drawn iid from \( \mathcal{N}(0,1) \).
**CP Random projection (CP-RP)**

- **CP Random Projection:**
  
  $f_{CP}(R) : x \mapsto \frac{1}{\sqrt{kR^N}} Mx$

  where each row of $M = \begin{pmatrix} m_1^T \\ m_2^T \\ \vdots \\ m_k^T \end{pmatrix} \in \mathbb{R}^{k \times d^N}$ is in the CP format:

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

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

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

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

→ The bounds on the variances are substantially different...
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:

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

The bounds on the variances are substantially different...

More details in paper on arXiv and Beheshteh’s talk at AISTATS 2020.
Experiment Results

[Graphs showing distortions for different embedding dimensions and tensor network types]
Discussion

- 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
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
Conclusion

- Forget matrices and linear algebra... **Tensors and multilinear algebra!**
- Tensor networks \(\equiv\) unifying language for tensor methods
Conclusion

- Forget matrices and linear algebra... **Tensors and multilinear algebra!**
- Tensor networks ≡ unifying language for tensor methods

**Thank you! Questions?**