IFT 6760A

Matrix and Tensor Factorization for Machine Learning

Winter 2020

Instructor: Guillaume Rabusseau

email: grabus@iro.umontreal.ca

Today

- What is this class about? Linear and multilinear algebra for ML...
- Class goals: refresh linear algebra, discover and have fun with tensors and do research!
- Class logistics.
- Content overview.
- In-class quiz (not graded): for me to assess your background and adjust material accordingly.
- Questions.

Material and discussion

- All material will be posted on my website
- We will use piazza for announcements and discussions, don't forget to sign-up (link on the course webpage and on Studium)
  - **access code is** `tensor`

Class high-level overview

- Objectives
  - End goal: for you to know what tensors and tensor decomposition techniques are and how they can be used in ML.
  - Along the way: learn useful theoretical tools from linear algebra and matrix decomposition techniques.
- We will start back from linear algebra and build up from there:
  1. Linear algebra refresher with focus on matrix decomposition
  2. Applications of Linear algebra in ML
  3. Introduction to tensors and tensor decomposition techniques
  4. Tensors in ML
  5. Seminar part of the class $\rightarrow$ your turn! (more on that later)

  (rough timeline: (1,2) in Jan., (3,4) in Feb, (5) in Mar.)
- I am very open to feedbacks and suggestions, and if there is a topic you would like to see covered in class come see me / send a mail!

About this class

- Not a math course but still a lot of maths
- I will assume a reasonable background in
  - probability / statistics
  - machine learning (methodology, bias-variance, regularization, etc.)
  - basic theoretical CS (big O notation, what is a graph, etc.)
  - linear algebra (even though we will review main results)

- We will do a lot of proofs...
  - ↪ getting comfortable with writing / reading formal proofs can be seen as a side-objective of the class.
- Have a look at last year's class webpage to get an idea:
  `https://www-labs.iro.umontreal.ca/~grabus/courses/ift6760a-w19.html`

Who is this class for?

- Advanced grad students
- If you're a first year MSc student the class may be too difficult or not given your background: come discuss with me or send me a mail with your background and list of relevant courses you followed.
- You should have followed at least one ML course (e.g. IFT3395/6390, COMP-551, COMP-652).

Class goals

- Get a good grasp on linear algebra and matrix decomposition techniques in the context of ML
- Learn theoretical tools potentially relevant to many aspects of ML
- Learn about tensors and how they can be used in ML

- Read research papers, get familiar with the literature
- Engage in research, connect the tools we'll learn with your own research
- Work on your presentation and writing skills
- Work on a semester research project

Class logistics

- Language: everything will be English, come see me or send me a mail if this is a concern.
- Grading:
  - scribing (10%): most lectures will be on the board, each lecture a group of students is responsible for taking notes and write them (in latex) for the following week.
    Each student has to scribe at least once (likely twice) during the semester.
  - Assignment(s) (20%): one or two assignments in the beginning of the semester.
  - paper presentation (30%)...
  - research project (40%)...

↪ no midterm but a project proposal due for middle of semester.

↪ no final but a project presentation (poster session) taking place at the end of the semester.

Paper presentation

- A few classes (likely starting late Feb./early March) will be devoted to paper presentation by students.
- Goal is for you to work on your presentation skills and potentially start elaborating your project.
  - read a research paper from the literature (I will post references but you can come up with your own suggestion, to be validated).
  - present the work in class either with slides or on the blackboard in a talk.
  - graded on the quality of the presentation.
- Specifics to be set up later (size of groups, length of presentations, etc.).

Research project

- Groups of 2-3 students
- Proposal due middle of semester
- One lecture will be devoted to projects presentation / progress report
- Project final presentation (either talks or poster session, TBD) at the end of the semester (date TBD)
- Project final written report due end of semester (date TBD)

Research project

- Topic chosen based on
  - your own research interests (aligned with the class)
  - a bibliography I will make available as we progress in the class.
- Each project should be grounded on at least one research paper.
- Only requirement: the project should be related to the class content.

Research project

- Types of projects:
  - Literature review: choose a topic/problem and present the existing approaches to handle it, comparing them and analyzing their drawbacks / advantages / limits, perform some experimental comparison.
  - Application: apply ideas/algorithms seen in class to a new problem you find interesting or related to your research.
  - Theory: focus on theoretical analysis of an algorithm or a problem (e.g. robustness analysis, complexity analysis), propose a new method to tackle a problem (existing or a new problem), ideally still perform experiments (e.g. on synthetic data).
- Best case scenario: project $\Rightarrow$ research paper

Times and dates

- Class on Tuesdays 12:30-14:15 and Thursdays 11:30-13:15 here at the Mila auditorium.
- Schedule conflicts (COMP-551/767)?
- Office hours: Tuesdays after class in my office (D15 @ 6666) or one of the meeting rooms at Mila
- Reading week: March 3-8
- If I understood correctly:
  - Deadline to switch/drop the course without fees: January 22nd
  - Deadline to drop the course with fees: March 15th

Auditing the class

- Everyone's welcome (as long as there are enough room, but this should be ok!)
- Participating to research projects may be doable for auditing students as well, come see me.

Questions?

Class high-level overview

- Overall objective: for you to know what tensors and tensor decomposition techniques are and how they can be used in ML.
- We will start back from linear algebra and build up from there:
  1. Linear algebra refresher with focus on matrix decomposition
  2. Linear algebra and ML / linear algebra and discrete maths
  3. Introduction to tensors and tensor decomposition techniques
  4. Tensors in ML
  5. Seminar part of the class $\rightarrow$ your turn! (more on that later)

  (rough timeline: (1,2) in Jan., (3,4) in Feb, (5) in Mar.)
- I am very open to feedbacks and suggestions, if there is a topic you would like to see covered in class come see me or send a mail!

Linear Algebra road map

- Start by briefly recalling the basics: span, linear independence, dimension, rank of a matrix, rank nullity theorem, orthogonality, subspaces, projection, etc.
- Give some special care to eigenvalues/eigenvectors, diagonalizability, etc. and present common factorization (low rank factorization, QR, eigendecomposition, SVD), a bit of optimization (mainly the power method).
- Present a few fundamental theorems and tools (and prove some of them):
  - Spectral theorem
  - Jordan canonical form
  - Eckart-Young-Mirsky theorem
  - Min-max theorem (Courant-Fischer-Weyl theorem)
  - Generalized eigenvalue problems
  - Perturbation bounds on eigen-vectors/values
- Illustrate with ML applications.

Linear Algebra and ML: Teasers

Let's look at a few examples of connections between algebra and ML / CS...

Linear Algebra and ML: Linear Regression

- We want to find $f : \mathbb{R}^d \to \mathbb{R}$ linear (i.e. $f : \mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$) minimizing the squared error loss on the training data $\mathcal{L} = \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2$.

Linear Algebra and ML: Linear Regression

- We want to find $f : \mathbb{R}^d \to \mathbb{R}$ linear (i.e. $f : \mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$) minimizing the squared error loss on the training data $\mathcal{L} = \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2$.
- Solution: $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ where $\mathbf{X} \in \mathbb{R}^{N \times d}$ and $\mathbf{y} \in \mathbb{R}^N$.
- The vector of prediction is given by $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}^* = \underbrace{\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top}_{\text{orthogonal projection}} \mathbf{y}$

Linear Algebra and ML: Linear Regression

- We want to find $f : \mathbb{R}^d \to \mathbb{R}$ linear (i.e. $f : \mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$) minimizing the squared error loss on the training data $\mathcal{L} = \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2$.
- Solution: $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ where $\mathbf{X} \in \mathbb{R}^{N \times d}$ and $\mathbf{y} \in \mathbb{R}^N$.
- The vector of prediction is given by $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}^* = \underbrace{\mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top}_{\text{orthogonal projection}} \mathbf{y}$

> $\hat{\mathbf{y}}$ is the orthogonal projection of $\mathbf{y}$ onto the subspace of $\mathbb{R}^N$ spanned by the columns of $\mathbf{X}$.

Linear Algebra and ML: Principal Component Analysis

- Given a set of points $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbb{R}^d$ (assume centered) we want to find the k-dimensional subspace of $\mathbb{R}^d$ such that he projections of the points onto this subspace
  - have maximal variance
  - stay as close as possible to the original points (in ℓ_2 distance).

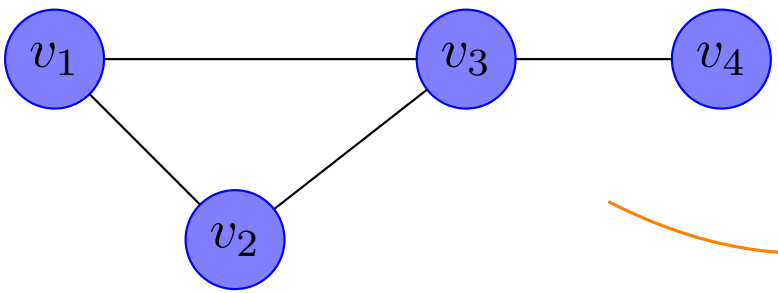
Linear Algebra and ML: Principal Component Analysis

- Given a set of points $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbb{R}^d$ (assume centered) we want to find the k-dimensional subspace of $\mathbb{R}^d$ such that he projections of the points onto this subspace
  - have maximal variance
  - stay as close as possible to the original points (in ℓ_2 distance).

> The solution is given by the subspace spanned by the top k eigenvectors of the covariance matrix $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{d \times d}$.

Linear Algebra and ML: Spectral Graph Clustering

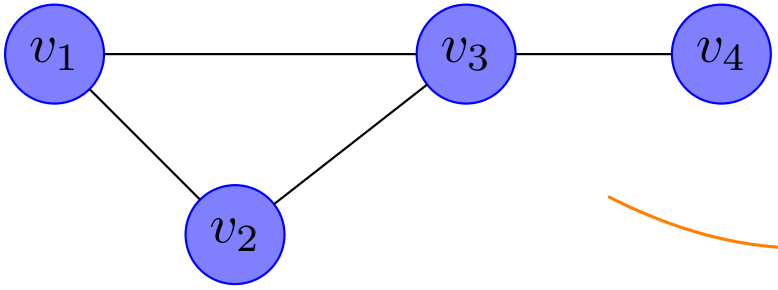
- The Laplacian of a graph is the difference between its degree matrix and its adjacency matrix: $\mathbf{L} = \mathbf{D} - \mathbf{A}$



$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Linear Algebra and ML: Spectral Graph Clustering

- The Laplacian of a graph is the difference between its degree matrix and its adjacency matrix: $\mathbf{L} = \mathbf{D} - \mathbf{A}$



$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Zero is an eigenvalue of the Laplacian, and its multiplicity is equal to the number of connected components of G.

Linear Algebra and ML: Spectral Learning of HMMs/WFAs

- Let Σ be a finite alphabet (e.g. $\Sigma = \{a, b\}$).
- Let Σ^* be the set of all finite sequences of symbols in Σ (e.g. $a, b, ab, aab, bbba, \ldots$).
- Given a real-valued function $f : \Sigma^* \to \mathbb{R}$, its Hankel matrix $\mathbf{H} \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$ is a bi-infinite matrix whose entries are given by $\mathbf{H}_{u,v} = f(uv)$.

Linear Algebra and ML: Spectral Learning of HMMs/WFAs

- Let Σ be a finite alphabet (e.g. $\Sigma = \{a, b\}$).
- Let Σ^* be the set of all finite sequences of symbols in Σ (e.g. $a, b, ab, aab, bbba, \ldots$).
- Given a real-valued function $f : \Sigma^* \to \mathbb{R}$, its Hankel matrix $\mathbf{H} \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$ is a bi-infinite matrix whose entries are given by $\mathbf{H}_{u,v} = f(uv)$.

> The rank of $\mathbf{H}$ is finite if and only if $\mathbf{H}$ can be computed by a weighted automaton.

(if you don't know what a weighted automaton is, think some kind of RNN with linear activation functions)

Linear Algebra and ML: Method of Moments

- Consider a Gaussian mixture model with k components, where the ith Gaussian has mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and all Gaussians have the same diagonal covariance $\sigma^2\mathbf{I}$, i.e. the pdf of $\mathbf{x}$ is

$$f(\mathbf{x}) = \sum_{i=1}^{k} p_i \mathcal{N}(\boldsymbol{\mu}_i, \sigma^2\mathbf{I})$$

Linear Algebra and ML: Method of Moments

- Consider a Gaussian mixture model with k components, where the ith Gaussian has mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and all Gaussians have the same diagonal covariance $\sigma^2\mathbf{I}$, i.e. the pdf of $\mathbf{x}$ is

$$f(\mathbf{x}) = \sum_{i=1}^{k} p_i \mathcal{N}(\boldsymbol{\mu}_i, \sigma^2\mathbf{I})$$

> The rank of the (modified) second-order moment
>
> $$\mathbf{M} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \sigma^2\mathbf{I}$$
>
> is at most k.

(we actually have $\mathbf{M} = \sum_i p_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$)

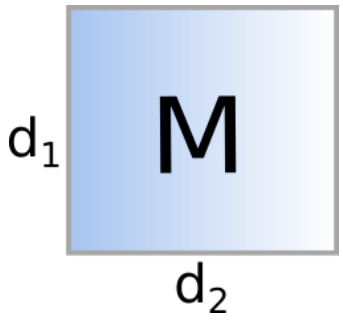
Spectral Methods (high-level view)

- Spectral methods usually achieve learning by extracting structure from observable quantities through eigen-decompositions/tensor decompositions.
- Spectral methods often constitute an alternative to EM to learn latent variable models (e.g. HMMs, single-topic/Gaussian mixtures models).
- Advantages of spectral methods:
  - computationally efficient,
  - consistent,
  - no local optima.

Tensors

What about tensors?

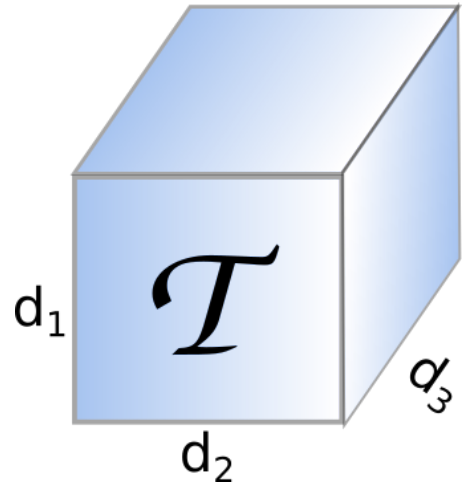
Tensors vs Matrices



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

$\mathbf{M}_{ij} \in \mathbb{R}$ for

$i \in [d_1], j \in [d_2]$

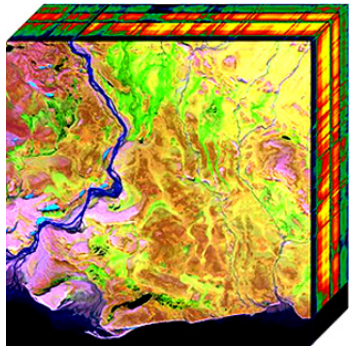


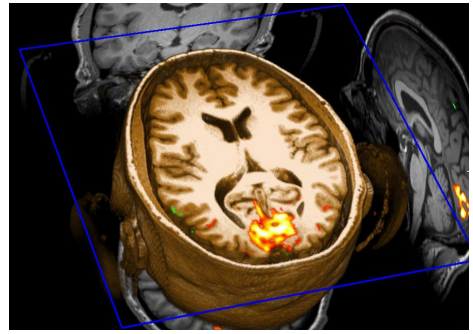
$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$

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

Tensors and Machine Learning

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



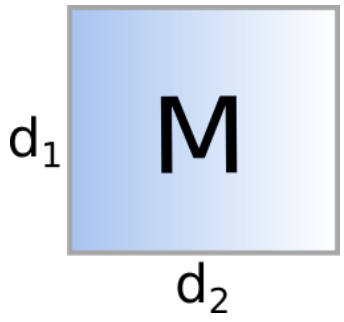


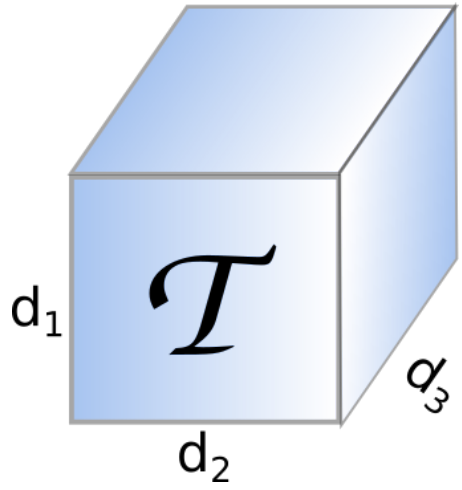
(ii) Tensor as parameters of a model: neural networks, polynomial regression, weighted tree automata...

$$f(\mathbf{x}) = \sum_{i,j,k} \mathcal{T}_{ijk} \mathbf{x}_i \mathbf{x}_j \mathbf{x}_k$$

(iii) Tensors as tools: tensor method of moments, system of polynomial equations, layer compression in neural networks, theoretical analysis of expressiveness of neural networks...

Tensors





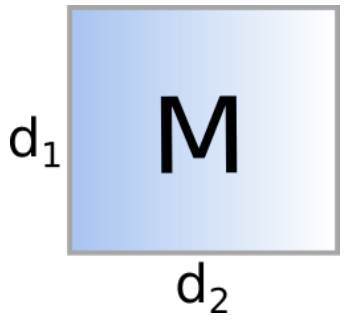
$\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$

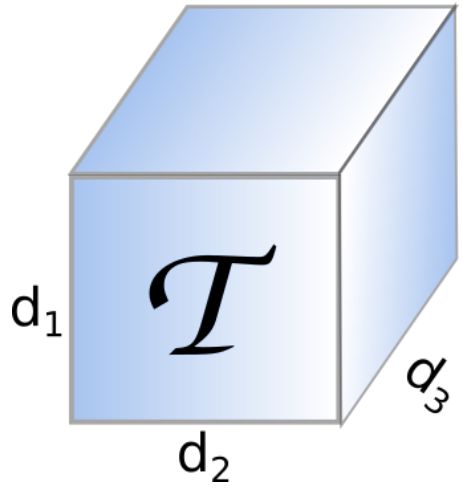
$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$

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

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

Tensors





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

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

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

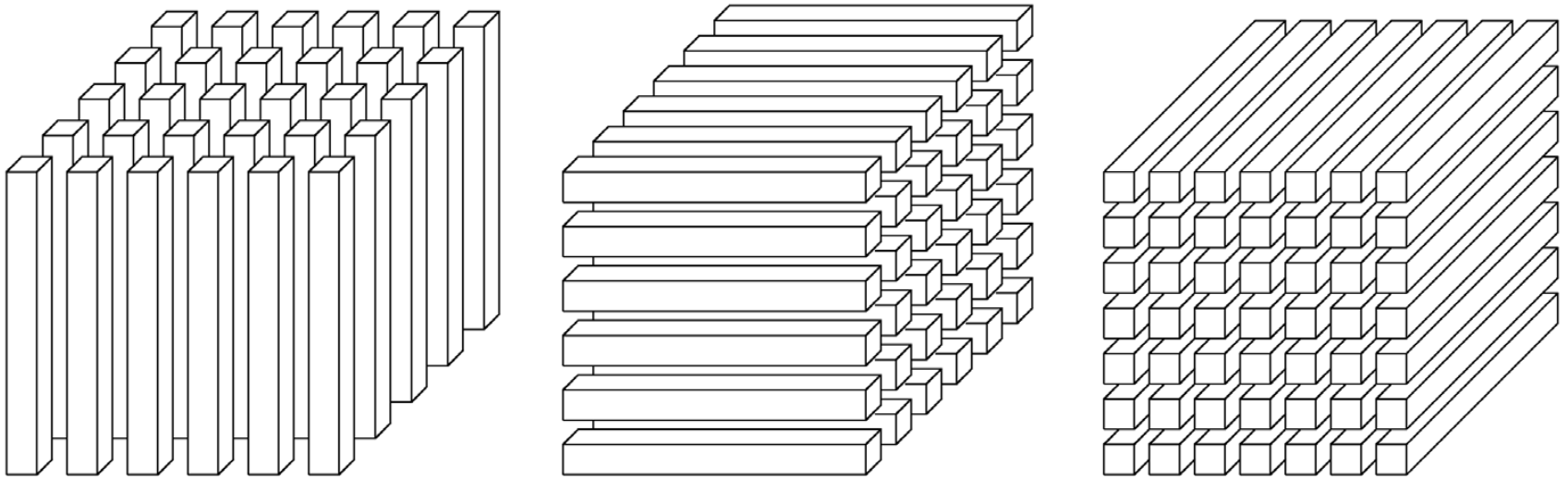
- Outer product. If $\mathbf{u} \in \mathbb{R}^{d_1}$, $\mathbf{v} \in \mathbb{R}^{d_2}$, $\mathbf{w} \in \mathbb{R}^{d_3}$:

$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^\top \in \mathbb{R}^{d_1 \times d_2}$ $\qquad (\mathbf{u} \otimes \mathbf{v})_{i,j} = \mathbf{u}_i \mathbf{v}_j$

$\mathbf{u} \otimes \mathbf{v} \otimes \mathbf{w} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ $\qquad (\mathbf{u} \otimes \mathbf{v} \otimes \mathbf{w})_{i,j,k} = \mathbf{u}_i \mathbf{v}_j \mathbf{w}_k$

Tensors: mode-n fibers

- Matrices have rows and columns, tensors have fibers[1]:



(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

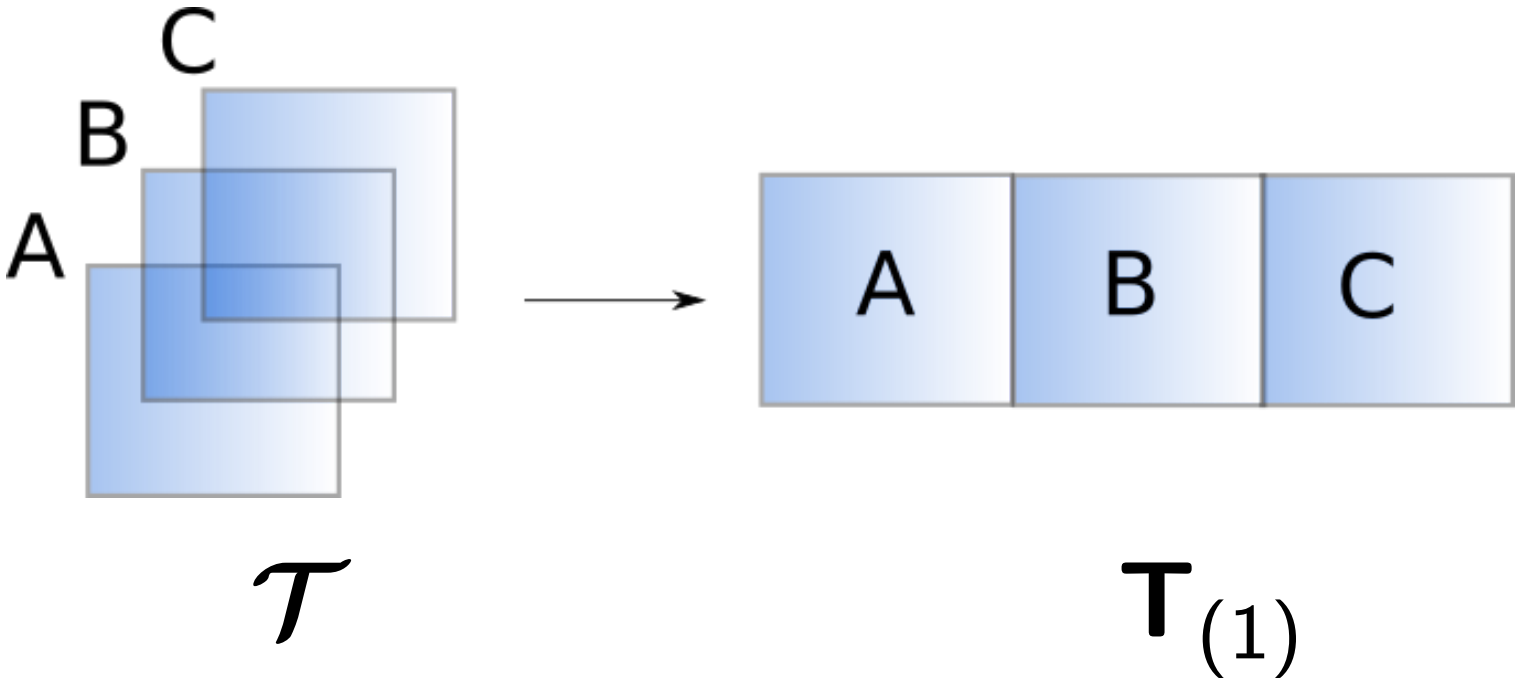
Fig. 2.1 *Fibers of a 3rd-order tensor.*

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

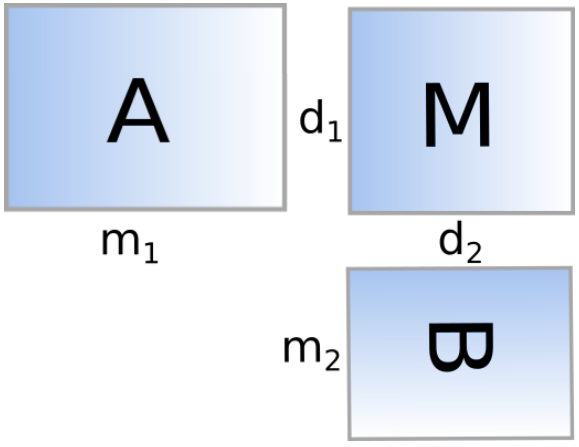
Tensors: Matricizations

- $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ can be reshaped into a matrix as

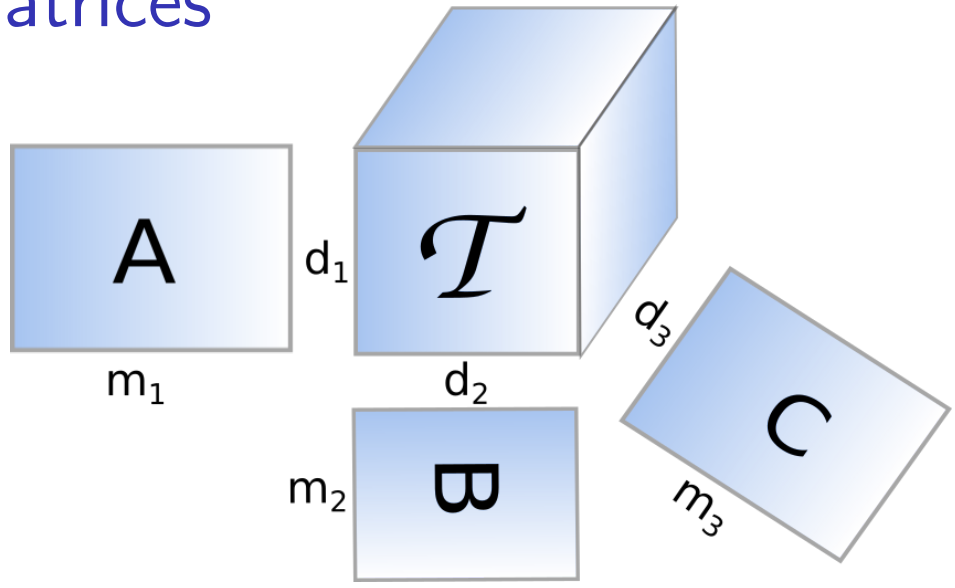
$$\mathbf{T}_{(1)} \in \mathbb{R}^{d_1 \times d_2 d_3}$$
$$\mathbf{T}_{(2)} \in \mathbb{R}^{d_2 \times d_1 d_3}$$
$$\mathbf{T}_{(3)} \in \mathbb{R}^{d_3 \times d_1 d_2}$$



Tensors: Multiplication with Matrices

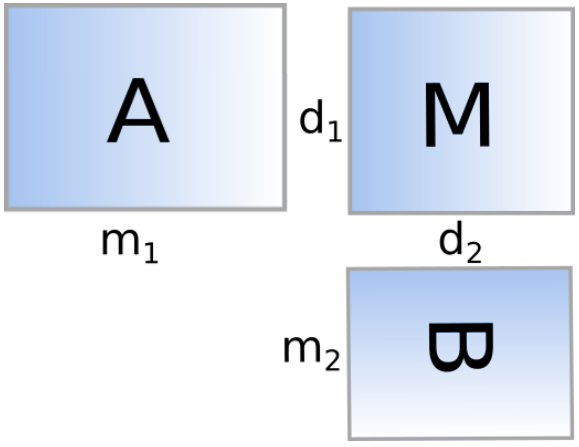


$\mathbf{AMB}^\top \in \mathbb{R}^{m_1 \times m_2}$

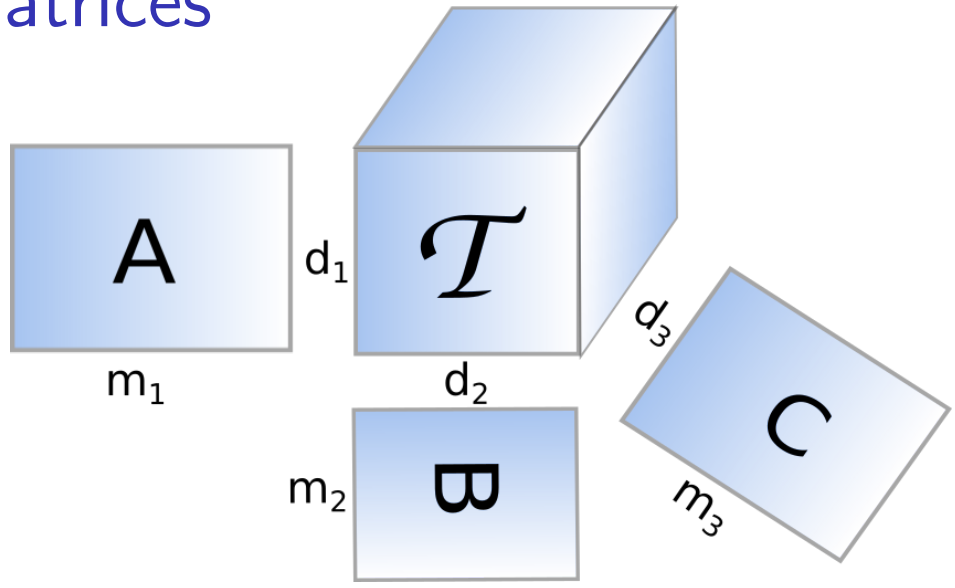


$\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$

Tensors: Multiplication with Matrices



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



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

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

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

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]

Tensors vs. Matrices: Rank

- The rank of a matrix $\mathbf{M}$ is:
  - the number of linearly independent columns of $\mathbf{M}$
  - the number of linearly independent rows of $\mathbf{M}$
  - the smallest integer R such that $\mathbf{M}$ can be written as a sum of R rank-one matrix:

$$\mathbf{M} = \sum_{i=1}^{R} \mathbf{u}_i \mathbf{v}_i^\top .$$

Tensors vs. Matrices: Rank

- The rank of a matrix $\mathbf{M}$ is:
  - the number of linearly independent columns of $\mathbf{M}$
  - the number of linearly independent rows of $\mathbf{M}$
  - the smallest integer R such that $\mathbf{M}$ can be written as a sum of R rank-one matrix:

$$\mathbf{M} = \sum_{i=1}^{R} \mathbf{u}_i \mathbf{v}_i^\top .$$

- The multilinear rank of a tensor $\mathcal{T}$ is a tuple of integers (R_1, R_2, R_3) where R_n is the number of linearly independent mode-n fibers of $\mathcal{T}$:

$$R_n = \operatorname{rank}(\mathbf{T}_{(n)})$$

Tensors vs. Matrices: Rank

- The rank of a matrix $\mathbf{M}$ is:
  - the number of linearly independent columns of $\mathbf{M}$
  - the number of linearly independent rows of $\mathbf{M}$
  - the smallest integer R such that $\mathbf{M}$ can be written as a sum of R rank-one matrix:

$$\mathbf{M} = \sum_{i=1}^{R} \mathbf{u}_i \mathbf{v}_i^\top.$$

- The multilinear rank of a tensor $\mathcal{T}$ is a tuple of integers (R_1, R_2, R_3) where R_n is the number of linearly independent mode-n fibers of $\mathcal{T}$:

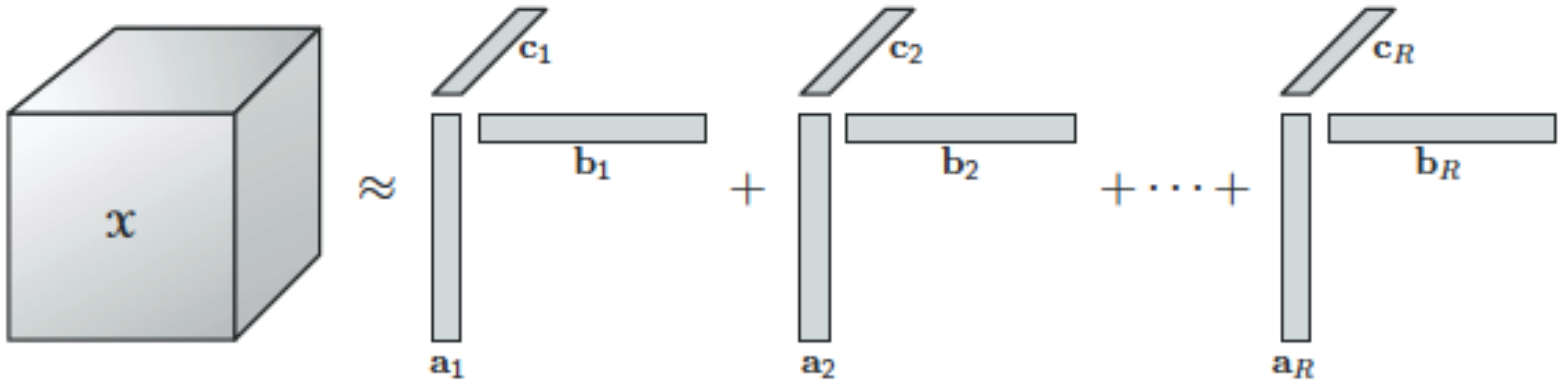
$$R_n = \operatorname{rank}(\mathbf{T}_{(n)})$$

- The CP rank of $\mathcal{T}$ is the smallest integer R such that $\mathcal{T}$ can be written as a sum of R rank-one tensors:

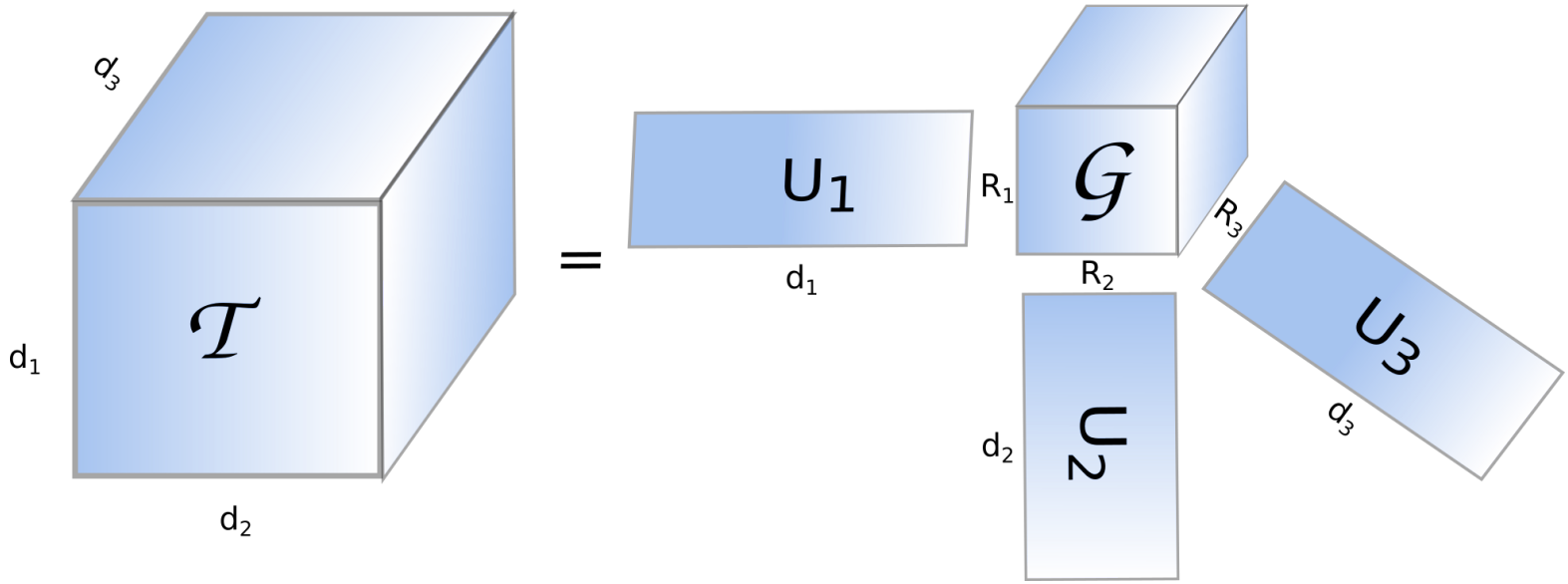
$$\mathcal{T} = \sum_{i=1}^{R} \mathbf{u}_i \otimes \mathbf{v}_i \otimes \mathbf{w}_i.$$

CP and Tucker decomposition

- CP decomposition[2]:



- Tucker decomposition:



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

Hardness results

- Those are all NP-hard for tensor of order ≥ 3 in general:
  - Compute the CP rank of a given tensor
  - Find the best approximation with CP rank R of a given tensor
  - Find the best approximation with multilinear rank $(R_1, \cdots, R_p)$ of a given tensor (*)
  - ...
- On the positive side:
  - Computing the multilinear rank is easy and efficient algorithms exist for (*).
  - Under mild conditions, the CP decomposition is unique (modulo scaling and permutations).

  $\Rightarrow$ Very relevant for model identifiability...

Back to the Method of Moments

- Consider a Gaussian mixture model with k components, where the ith Gaussian has mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and all Gaussians have the same diagonal covariance $\sigma^2\mathbf{I}$.

> The (modified) second-order moment $\mathbf{M} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \sigma^2\mathbf{I}$ is such that
>
> $$\mathbf{M} = \sum_{i=1}^{k} p_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$$

- Can we recover the mixing weights p_i and centers $\boldsymbol{\mu}_i$ from $\mathbf{M}$?

Back to the Method of Moments

- Consider a Gaussian mixture model with k components, where the ith Gaussian has mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and all Gaussians have the same diagonal covariance $\sigma^2\mathbf{I}$.

> The (modified) second-order moment $\mathbf{M} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \sigma^2\mathbf{I}$ is such that
>
> $$\mathbf{M} = \sum_{i=1}^{k} p_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top$$

- Can we recover the mixing weights p_i and centers $\boldsymbol{\mu}_i$ from $\mathbf{M}$?
  - No, except if the $\boldsymbol{\mu}_i$ are orthonormal, in which case they are the eigenvectors of $\mathbf{M}$ and the p_i are the corresponding eigenvalues.
- But we will see that if we know both the matrix $\mathbf{M}$ and the 3rd order tensor $\boldsymbol{\mathcal{T}} = \sum_{i=1}^{k} p_i \boldsymbol{\mu}_i \otimes \boldsymbol{\mu}_i \otimes \boldsymbol{\mu}_i$, then we can recover the weights and centers if the $\boldsymbol{\mu}_i$ are linearly independent.

Quiz

Quiz Time