Learning with Structured Tensor Decompositions

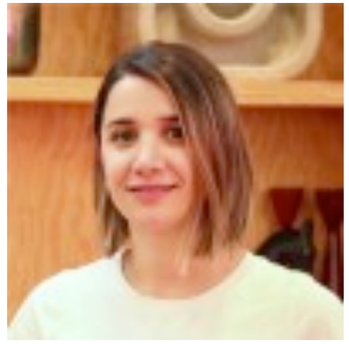
Anand D. Sarwate, Rutgers University

20 January 2025

IEEE Information Theory Society Distinguished Lecture
Indian Institute of Technology-Madras (Chennai)

Waheed U. Bajwa

Batoul Taki



Zahra Shakeri

Jose Hoyos Sanchez

Mohsen Ghassemi

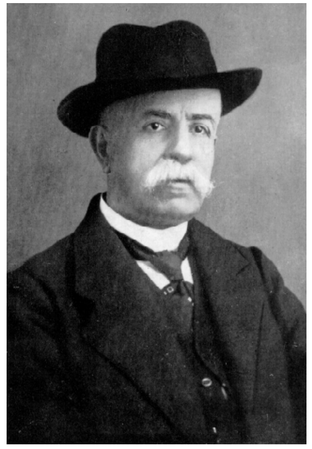
National Institutes of Health

IEEE INFORMATION THEORY SOCIETY

Tensors: what are they good for?

The history of the word “tensor”

Let’s meet some 19th century physicists



All images: Wikipedia

- 1848: William Rowan Hamilton used the word “tensor” to mean the absolute value (norm) of a quaternion. His “tensor” is actually a scalar (!)
- 1898: Woldemar Voigt used “tensor” in his paper *Die fundamentalen physikalischen Eigenschaften der Krystalle in elementarer Darstellung*
- 1892: Gregorio Ricci-Curbastro developed the theory of tensors. In 1900 he and his student Tullio Levi-Civita write a book on it called *Méthodes de calcul différentiel absolu et leurs applications*

From 1900 to the present

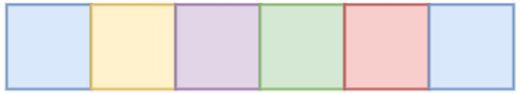
A relatively general timeline

All images: Wikipedia

- 1913: Albert Einstein and Marcel Grossman used tensor calculus extensively in their work on general relativity: *Entwurf einer verallgemeinerten Relativitätstheorie und einer Theorie der Gravitation*
- 1915–17: Levi-Civita and Einstein have a correspondence where the former helped fix the mistakes Einstein made in using tensor analysis.
- 1922: H. L. Brose's English translation of Weyl's book *Raum, Zeit, Materie* (*Space-Time-Matte*r) uses "tensor analysis."

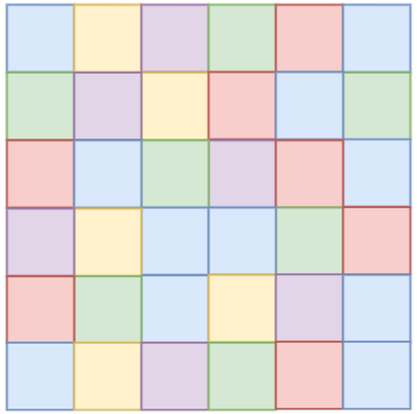
So what is a “tensor” anyway?

Tensors are many different things to many different people



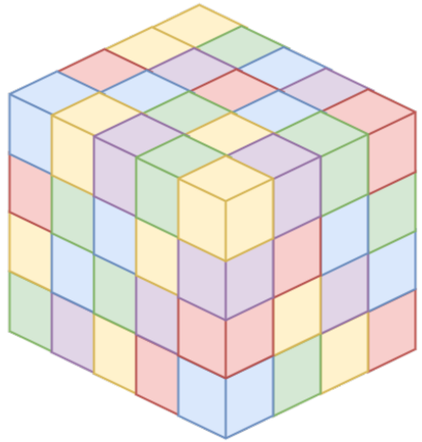
$\mathbf{x} \in \mathrm{R}^m$

First-Order Tensor (Vector)



$\mathbf{X} \in \mathrm{R}^{m_1 \times m_2}$

Second-Order Tensor (Matrix)



$\underline{\mathbf{X}} \in \mathrm{R}^{m_1 \times m_2 \times m_3}$

Third-Order Tensor

For this talk, treat tensors “computationally” as **multidimensional arrays**:

$$\underline{\mathbf{X}} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$$

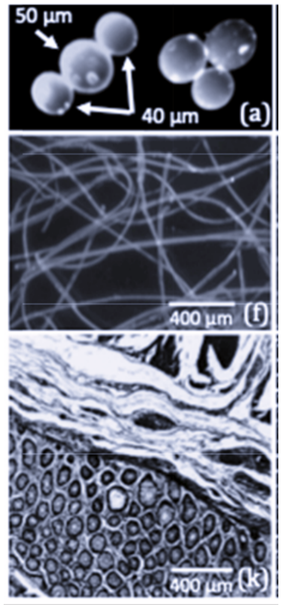
There are other (richer?) perspectives:

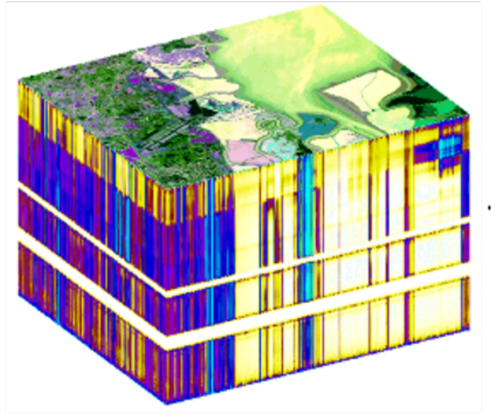
- Point in the tensor product of vector spaces
- Multilinear operator
- Tensor representation of $GL(n)$

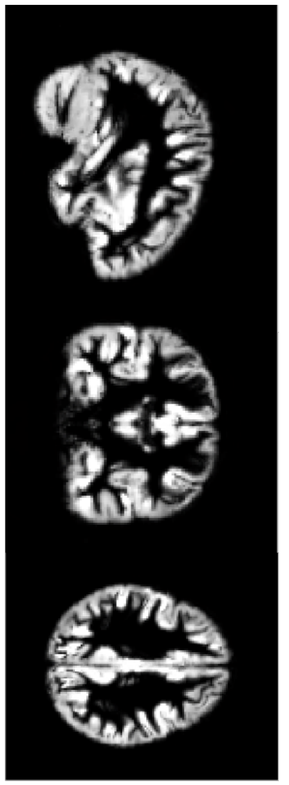
Where do we see tensors?

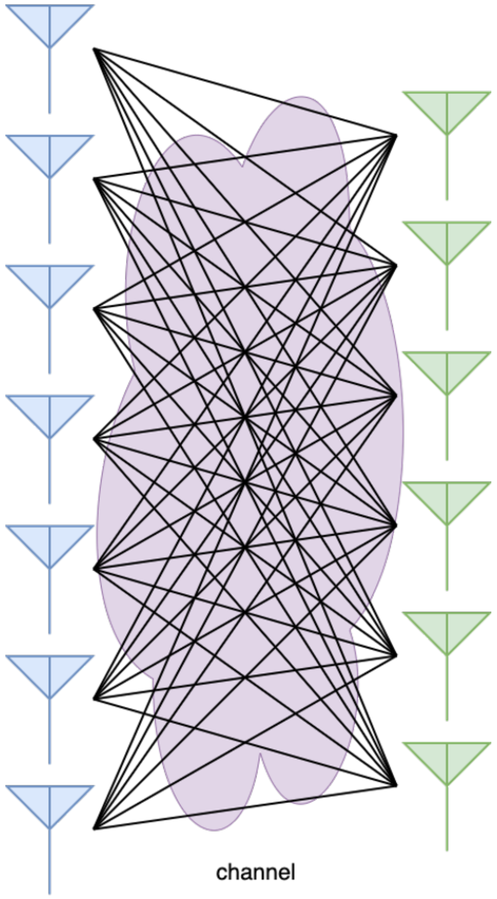
Multidimensional arrays are everywhere!

- **Medicine:** Neuroimaging and other medical imaging
- **Geosensing:** Hyperspectral imaging
- **Communications:** Massive MIMO
- **Probability:** Joint PMFs on multiple variables
- **Network science:** Time-varying graphs
- Also chemometrics, numerical linear algebra, psychometrics, theoretical computer science…



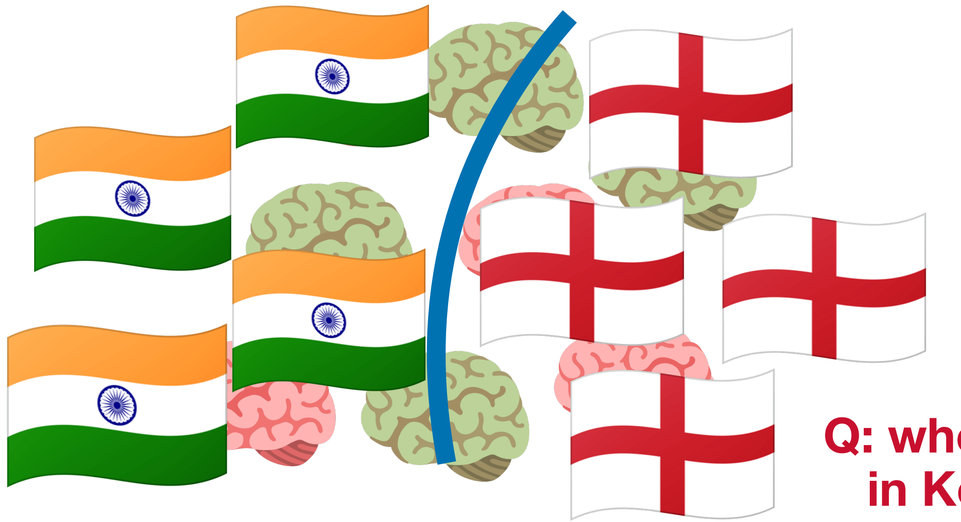


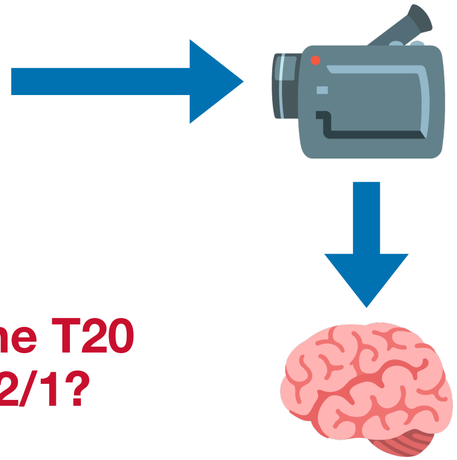




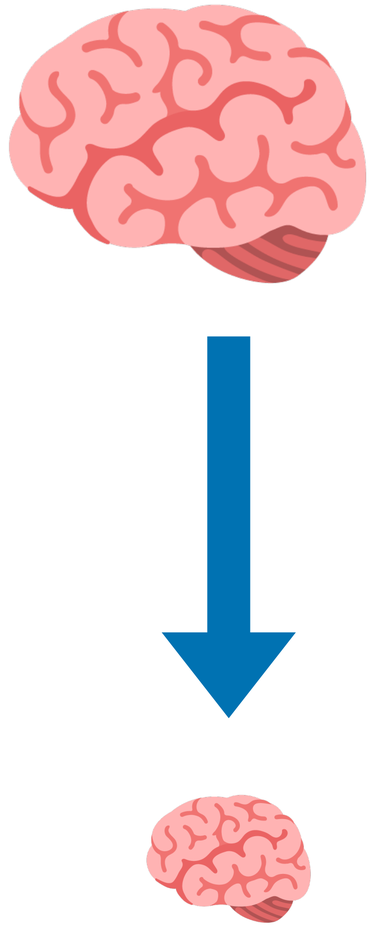
What do we want to do with tensor data?

All the regular things we do with data…





Q: who will win the T20 in Kolkata on 22/1?



- Signal recovery
- Supervised learning (prediction)
- Representation learning (compression)

Unsupervised learning with tensors

Example: dictionary learning and sparse representations

Task: given a collection of tensors $\underline{\mathbf{Y}}_1, \underline{\mathbf{Y}}_2, \ldots, \underline{\mathbf{Y}}_n \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}$, find a *dictionary* $\underline{\mathbf{d}}_1, \underline{\mathbf{d}}_2, \ldots, \underline{\mathbf{d}}_p$ such that

$$\underline{\mathbf{Y}}_i \approx \sum_{j=1}^{p} x_{ij} \underline{\mathbf{d}}_j,$$

where each vector of coefficients $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^\top$ is s-sparse.

Application: processing or storing hyperspectral images acquired from a drone.

Supervised learning with tensors

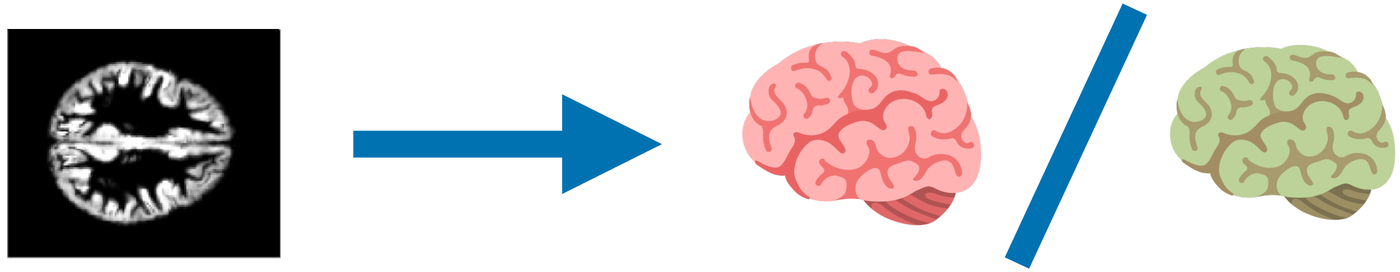
Exampled: regression with tensor-valued covariates

Task: given a collection of tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\} \subset \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K} \times \mathbb{R}$, find a *regression tensor* $\underline{\mathbf{B}}$ such that

$$y_i \approx \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + \text{noise},$$

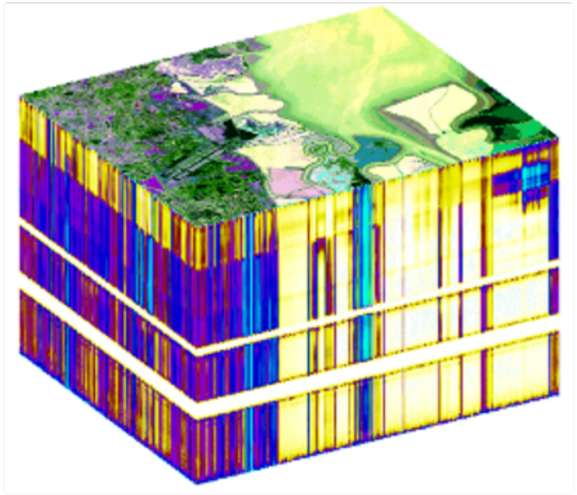
where $\langle \cdot\,, \cdot \rangle$ is the element-wise inner product.

Application: predicting a brain health condition from an MRI scan.



A baseline approach: reuse existing tools

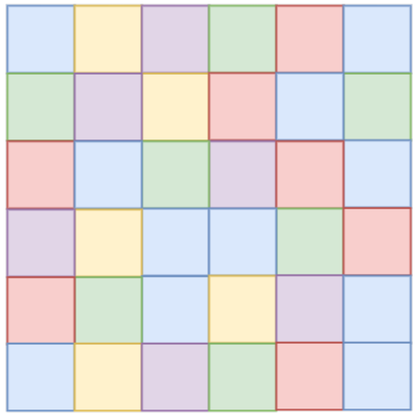
We can always use `reshape( )`



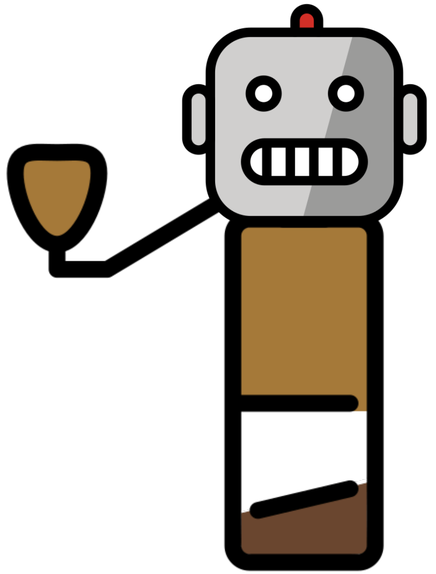
Hyperspectral Image

$m_1 \times m_2 \times m_3$
100 x 50 x 110

1 x 550,000



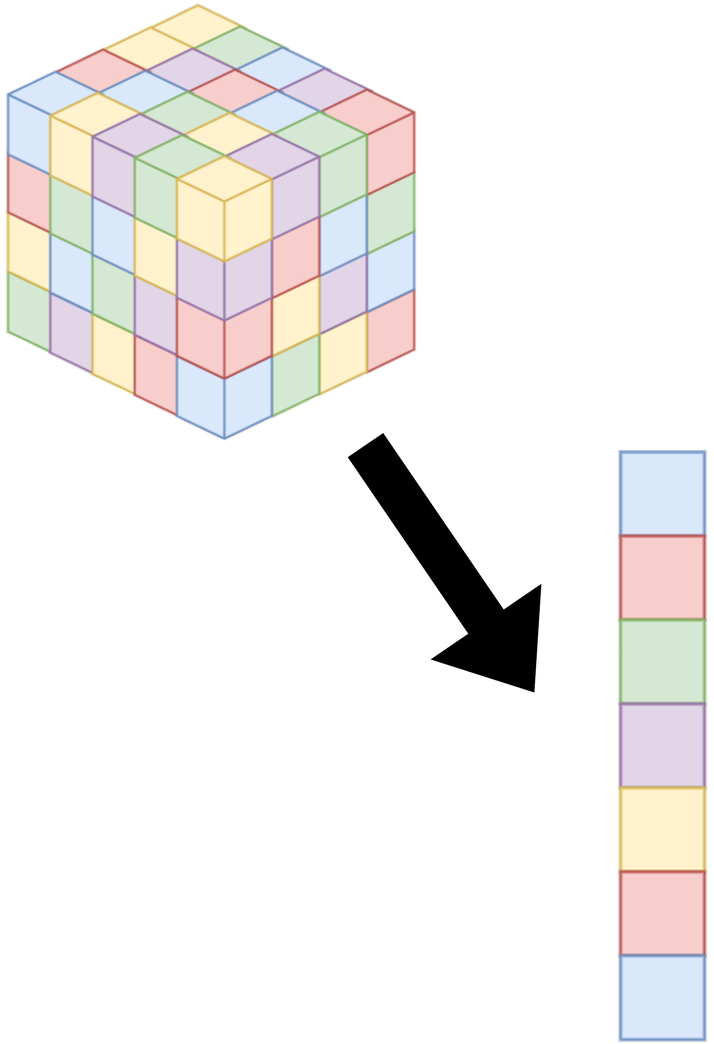
100 x 5500



Regression: 550,001
ViT-Huge: 632m

Why (and why not) vectorize?

The problems with vectorization



1. Vectorization *ignores the tensor structure*.
2. Resulting problems have very high dimension.

Example: ADHD200 data set has fMRI images of children's brains.

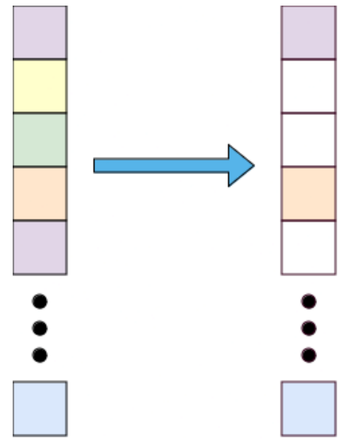
- fMRI data: 121 x 141 x 121 tensor
- After vectorizing: 2,122,945 dimensional vector
- Sample size: 959 total images

Dealing with overparameterization

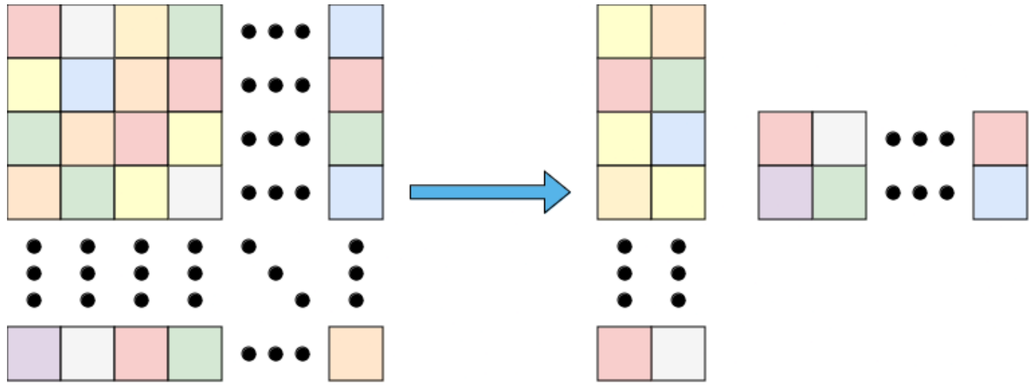
This is not just a problem with tensors!

We usually make models more tractable by assuming that our parameters have more structure. For example, for a regression model:

$$y_i = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}}_i \rangle + z_i$$



- **Vectors:** model $\underline{\mathbf{B}}$ as *sparse.*
- **Matrices:** model $\underline{\mathbf{B}}$ as *low rank.*



How do we impose structure on tensors?

What's in this talk

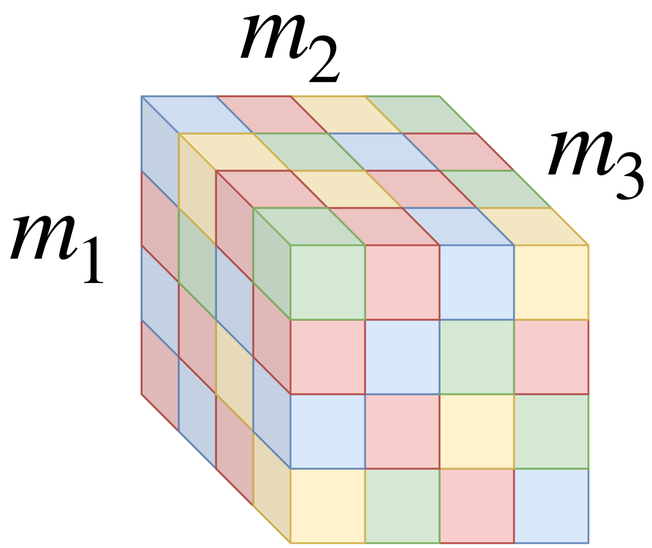
A preview of the rest of the talk

1. Tensor decompositions and where to find them
2. Regression with tensor-valued data and parameters
3. Other directions, old and new

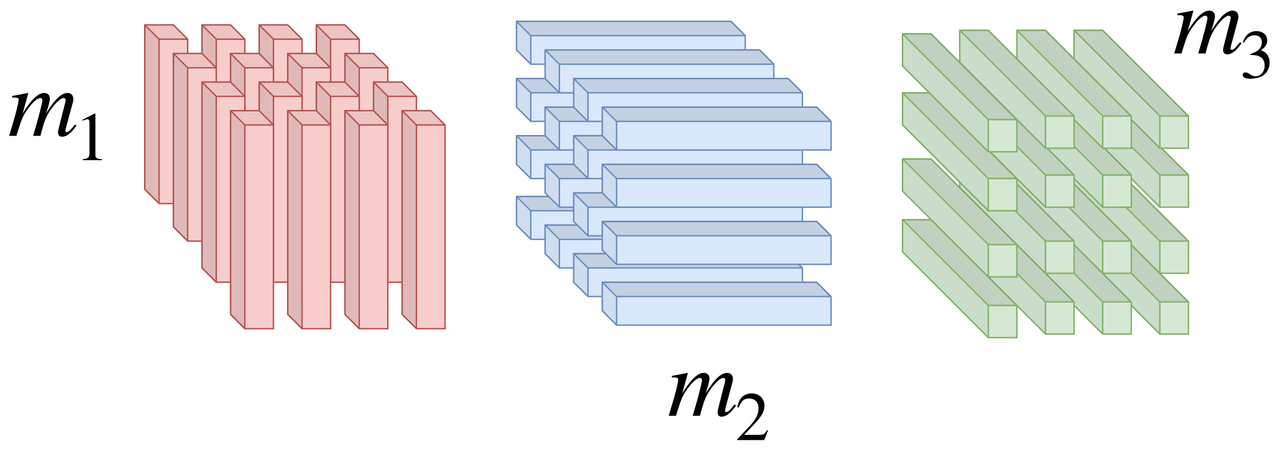
Tensor decompositions (old and “new”)

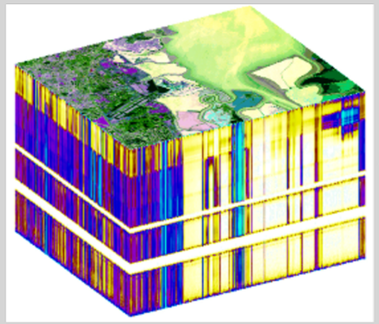
Some tensor terminology

A little jargon is unavoidable…



- **Mode:** each coordinate index
- **Order:** the number of modes of the tensor
- **Fibers:** 1-D vectors along each mode



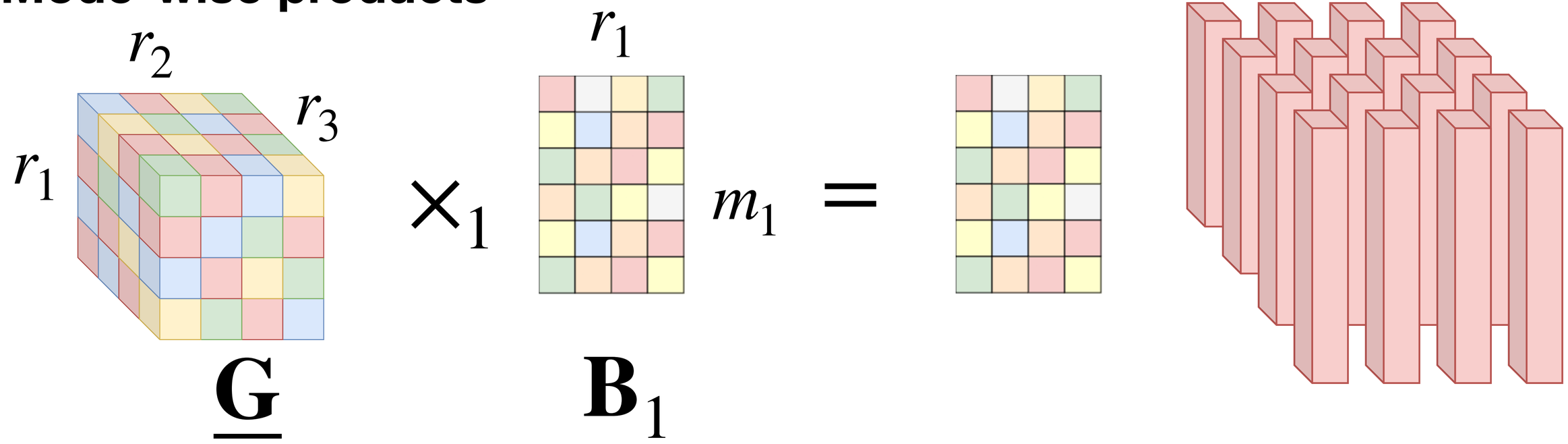


- Mode 1 = spectrum
- Mode 2 = longitude
- Mode 3 = latitude

Kolda and Bader (2009)
Sidiropolous et al. (2017)

Matrix-tensor products

Mode-wise products



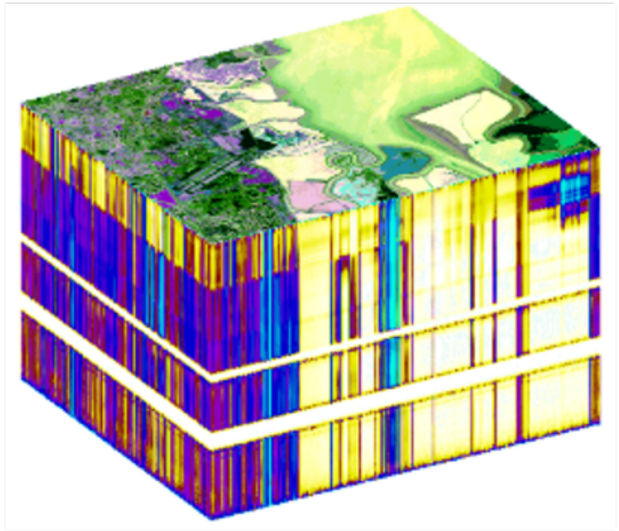
Multiply a tensor $\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$ by a matrix $\mathbf{B}_k \in \mathbb{R}^{m_k \times r_k}$ along mode k:

$$\underline{\mathbf{G}} \times_k \mathbf{B}_k$$

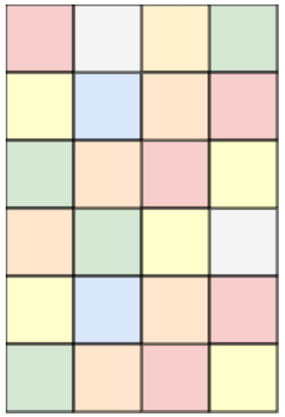
The result is a order-K tensor whose k-th mode is m_k dimensional.

Matrix-tensor product example

Filtering hyperspectral images



$\underline{\mathbf{X}} \quad \times_1 \quad \mathbf{L}$



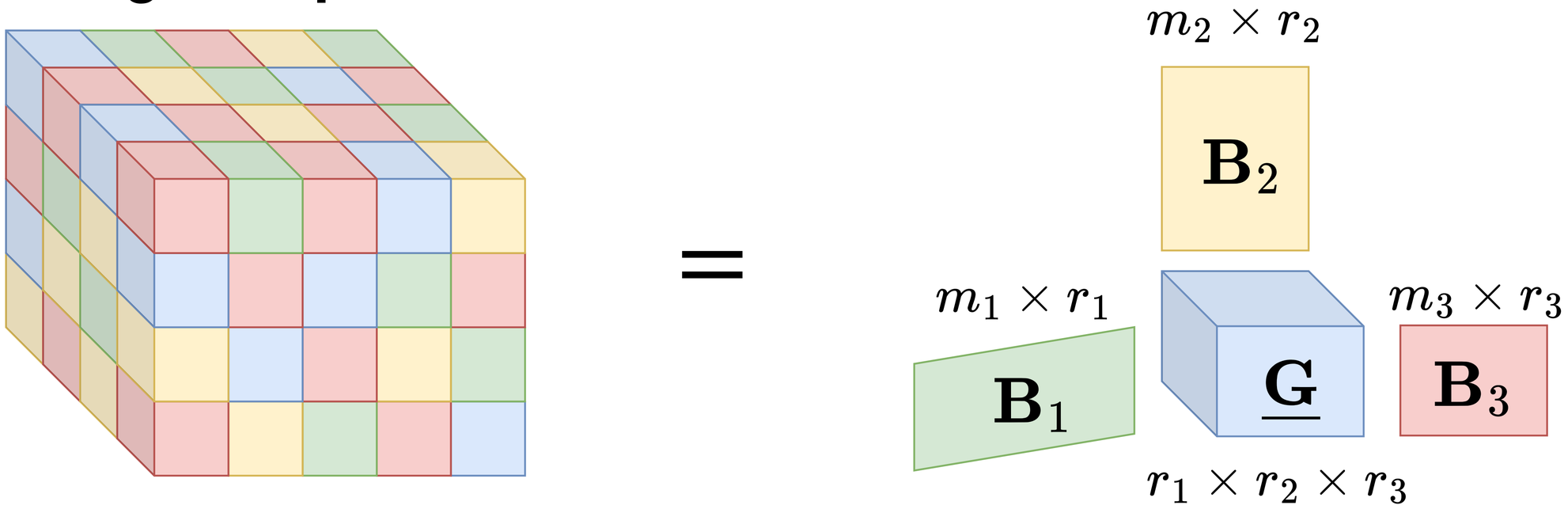
If $\underline{\mathbf{X}}$ is a hyperspectral image and $\mathbf{L}$ is a Discrete Fourier Transform (DFT) matrix corresponding to a lowpass filter, then:

$$\underline{\mathbf{X}} \times_1 \mathbf{L}_1$$

Applies the lowpass filter to the fiber (spectrum) at each physical location in space.

Chaining matrix-tensor products

Processing multiple modes



We can change the shape of a tensor with repeated matrix-tensor products

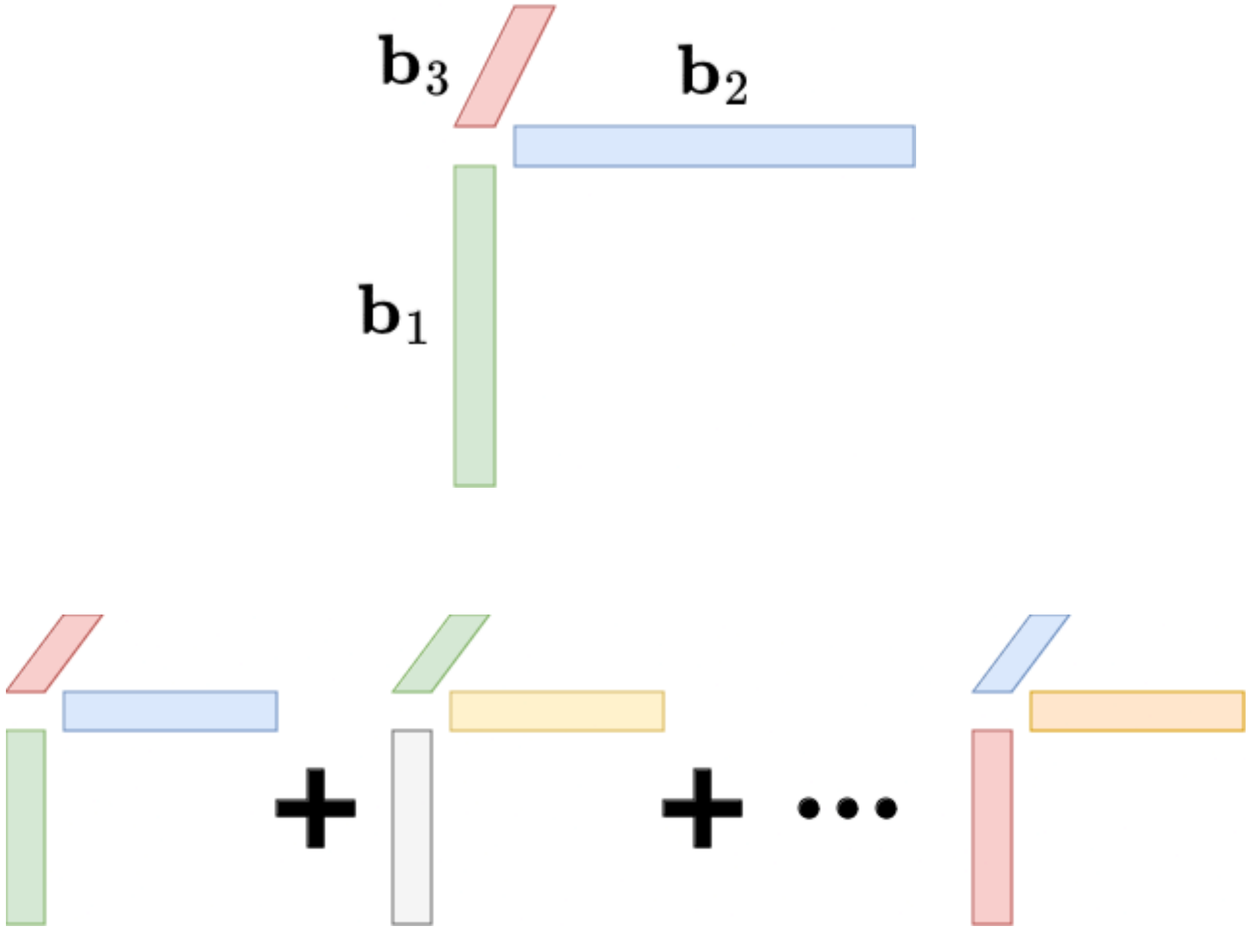
$$\underline{\mathbf{G}} \times_1 \mathbf{B}_1 \times_2 \mathbf{B}_2 \cdots \times_K \mathbf{B}_K = \underline{\mathbf{X}} \in \mathbb{R}^{m_1 \times m_2 \cdots \times m_K}$$

Tensor Rank(s) and Tensor Decompositions/Factorizations

Rank-1 tensors are outer products

Trying to get a handle on rank

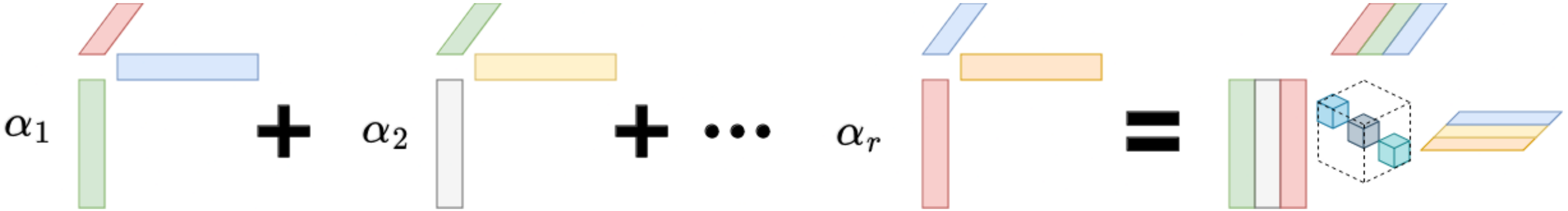
- 2D: a rank-1 *matrix*
- rank-r matrix can be written as the sum of r rank-1 matrices.
- A matrix has a **CANDECOMP/ PARAFAC (CP)** representation of order r if we can write it as a sum of r rank-1 outer products.



CP Decomposition

CP factorization

Writing the decomposition with matrix-tensor products



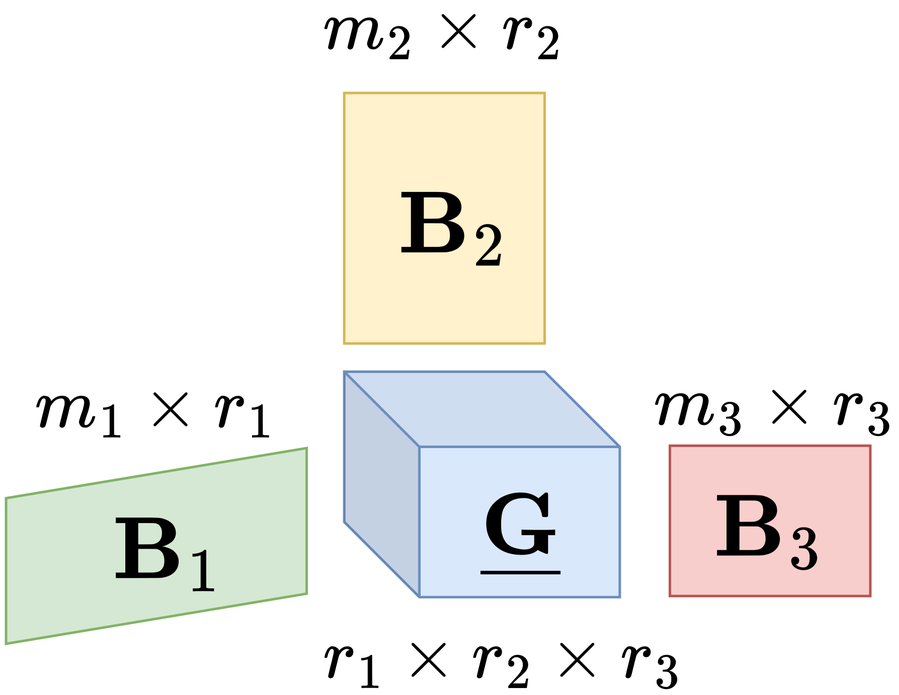
Gather the factors from each mode into matrices and define an $r \times r \times \cdots \times r$ **diagonal core tensor $\underline{\mathbf{G}}$**:

$$\underline{\mathbf{B}}_{\text{CP}} = \underline{\mathbf{G}} \times_1 \mathbf{B}_1 \times_2 \mathbf{B}_2 \cdots \times_K \mathbf{B}_K$$

The total number of parameters is $r\left(1 + \sum_{k=1}^{K} m_k\right)$ as opposed to $\prod_{k=1}^{K} m_k$.

Tucker decomposition

Filling out the core tensor



Suppose we have a **core tensor**

$$\underline{\mathbf{G}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_K}$$

and expand the dimensions using matrix-tensor products. This is the **Tucker decomposition**:

$$\underline{\mathbf{B}}_{\text{Tucker}} = \underline{\mathbf{G}} \times_1 \mathbf{B}_1 \times_2 \mathbf{B} \times_3 \mathbf{B}_3$$

The total number of parameters is

$$\prod_{k=1}^{K} r_k + \sum_{k=1}^{K} m_k r_k$$

Block tensor decompositions

Yet more generality

More recent work has studied **block tensor decompositions** (Section 5.7, Kolda and Bader 2009), which can written as a **mixture of Tucker models**:

$$\underline{\mathbf{B}}_{\text{BTD}} = \sum_{s=1}^{S} \underline{\mathbf{G}}_s \times_1 \mathbf{B}_{1,s} \times_2 \mathbf{B}_{2,s} \cdots \times_K \mathbf{B}_{K,s},$$

Here we have a lot more generality: each term is an independent Tucker model so each $\underline{\mathbf{G}}_s$ can have a different size.

Need to choose S *and* $\{m_{k,s}, r_{k,s}\}$ for each $s \in [S]$.

Issues with decompositions

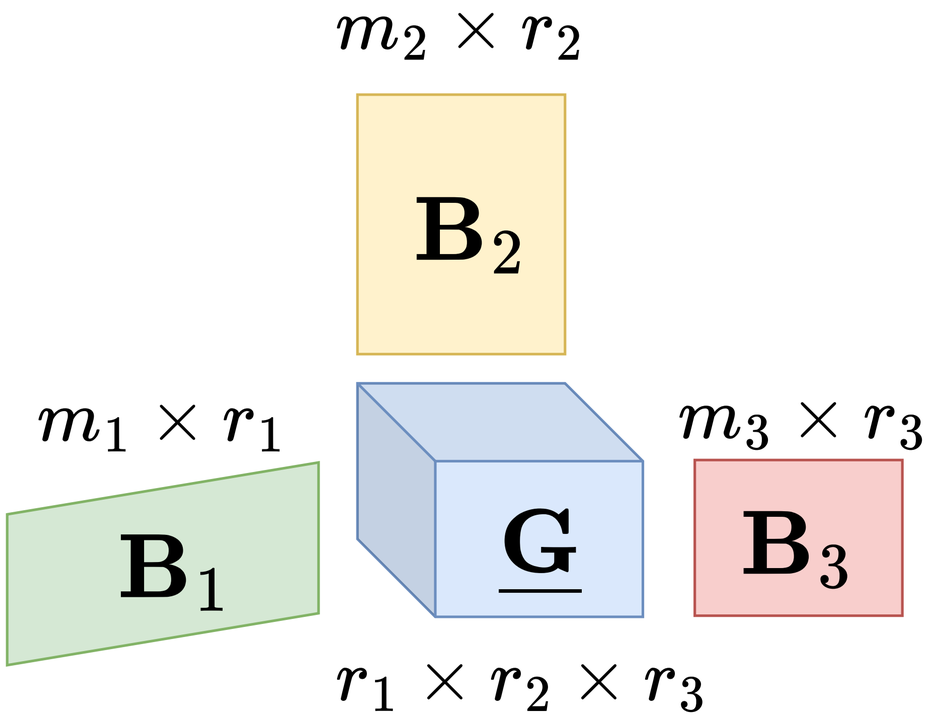
There are many different definitions of “rank” for tensors

- **CP rank** of $\underline{\mathbf{B}}$ = smallest number of terms in a CP decomposition (Hitchcock 1927, Kruskal 1977).
  - 👍🏽 The decomposition is (often) unique.
  - 👎🏽 Computing the rank is NP-complete for finite fields and NP-hard for ℚ (Håstad 1990, resolving a conjecture of Gonzalez and Ja’Ja’ 1980).
- **Tucker rank** is a **vector**. Decomposition can be computed using the higher-order SVD [HOSVD] or other algorithms (De Lathauwer et al. 2000, also others).
  - Tucker rank is **not** unique.

Matrix Equivalents of Tensor Factorizations

A different kind of vectorization

Matrix-tensor products as matrix vector products



Start with a Tucker factorization:

$$\underline{\mathbf{B}}_{\text{Tucker}} = \underline{\mathbf{G}} \times_1 \mathbf{B}_1 \times_2 \mathbf{B}_2 \cdots \times_K \mathbf{B}_K$$

If we vectorzize $\underline{\mathbf{B}}_{\text{Tucker}}$, we get get the following equivalent model:

$$\text{vec}(\underline{\mathbf{B}}_{\text{Tucker}}) = \left(\mathbf{B}_K \otimes \cdots \otimes \mathbf{B}_1\right) \text{vec}(\underline{\mathbf{G}})$$

where $\otimes$ is the **Kronecker product**.

The Kronecker product

Matrix-tensor products as a matrix vector product

The Kronecker product makes “copies” of one matrix inside the other:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

Vectorizing shows that the Tucker decomposition

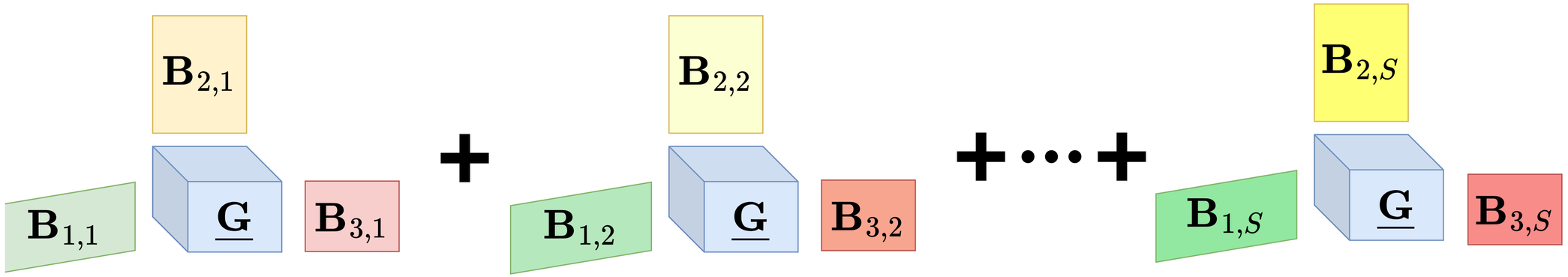
$$\mathrm{vec}(\underline{\mathbf{B}}_{\text{Tucker}}) = \left(\mathbf{B}_K \otimes \cdots \otimes \mathbf{B}_2 \otimes \mathbf{B}_1\right) \mathrm{vec}(\underline{\mathbf{G}})$$

Is somewhat restrictive.

Matrix Equivalents of Tensor Factorizations and “Low Separation Rank” Tensors

Proposal: low separation rank (LSR) tensors

BTD with a common core tensor



Special case of the BTD is a **low separation rank (LSR)** decomposition:

$$\underline{\mathbf{B}}_{\text{LSR}} = \sum_{s=1}^{S} \underline{\mathbf{G}} \times_1 \underline{\mathbf{B}}_{1,s} \times_2 \underline{\mathbf{B}}_{2,s} \cdots \times_K \underline{\mathbf{B}}_{K,s}$$

We use the *same core tensor* $\underline{\mathbf{G}}$ for each term. We also assume that the factor matrices $\{\underline{\mathbf{B}}_{k,s}\}$ have orthonormal columns.

What does separation rank mean?

Writing matrices as sums of Kronecker products

The **separation rank** (Tsiligkaridis and Hero, 2013) of a matrix is the minimum number S of terms needed so that

$$\mathbf{M} = \sum_{s=1}^{S} \mathbf{A}_{K,s} \otimes \cdots \otimes \mathbf{A}_{2,s} \otimes \mathbf{A}_{1,s}$$

Our LSR model corresponds assuming the matrix-vector product has a matrix with low separation rank

$$\sum_{s=1}^{S} \underline{\mathbf{G}} \times_1 \underline{\mathbf{B}}_{1,s} \times_2 \underline{\mathbf{B}}_{2,s} \cdots \times_K \underline{\mathbf{B}}_{K,s} = \underline{\mathbf{B}}_{\text{LSR}} \Longrightarrow \left(\sum_{s} \bigotimes_{k} \mathbf{B}_{\mathbf{k}} \right) \mathbf{g}$$

Space of LSR models

Counting parameters

Suppose we are given $(r_1, r_2, \ldots, r_K, S)$. Then define

$$\mathscr{C}_{\text{LSR}} = \left\{ \underline{\mathbf{B}} : \underline{\mathbf{B}} = \sum_{s=1}^{S} \underline{\mathbf{G}} \times_1 \mathbf{B}_{(1,s)} \times_2 \cdots \times_K \mathbf{B}_{(K,s)} \right\},$$

where for each (k, s), the columns of $\mathbf{B}_{(k,s)}$ are orthonormal.

Statistical/ML problems boil down to finding a "good" $\underline{\mathbf{B}} \in \mathscr{C}_{\text{LSR}}$.

Question: does the # of parameters are $S \sum_k m_k r_k + \prod_k r_k$ capture the complexity?

Packing and covering LSR tensors

Statistical estimation and information theory

Packings: find a large set of points in $\mathscr{C}_{\text{LSR}}$ which are a packing in the Frobenius norm $\|\cdot\|_F$.

- Construction inspired by superposition codes (a bit) plus Gilbert-Varshamov coding.

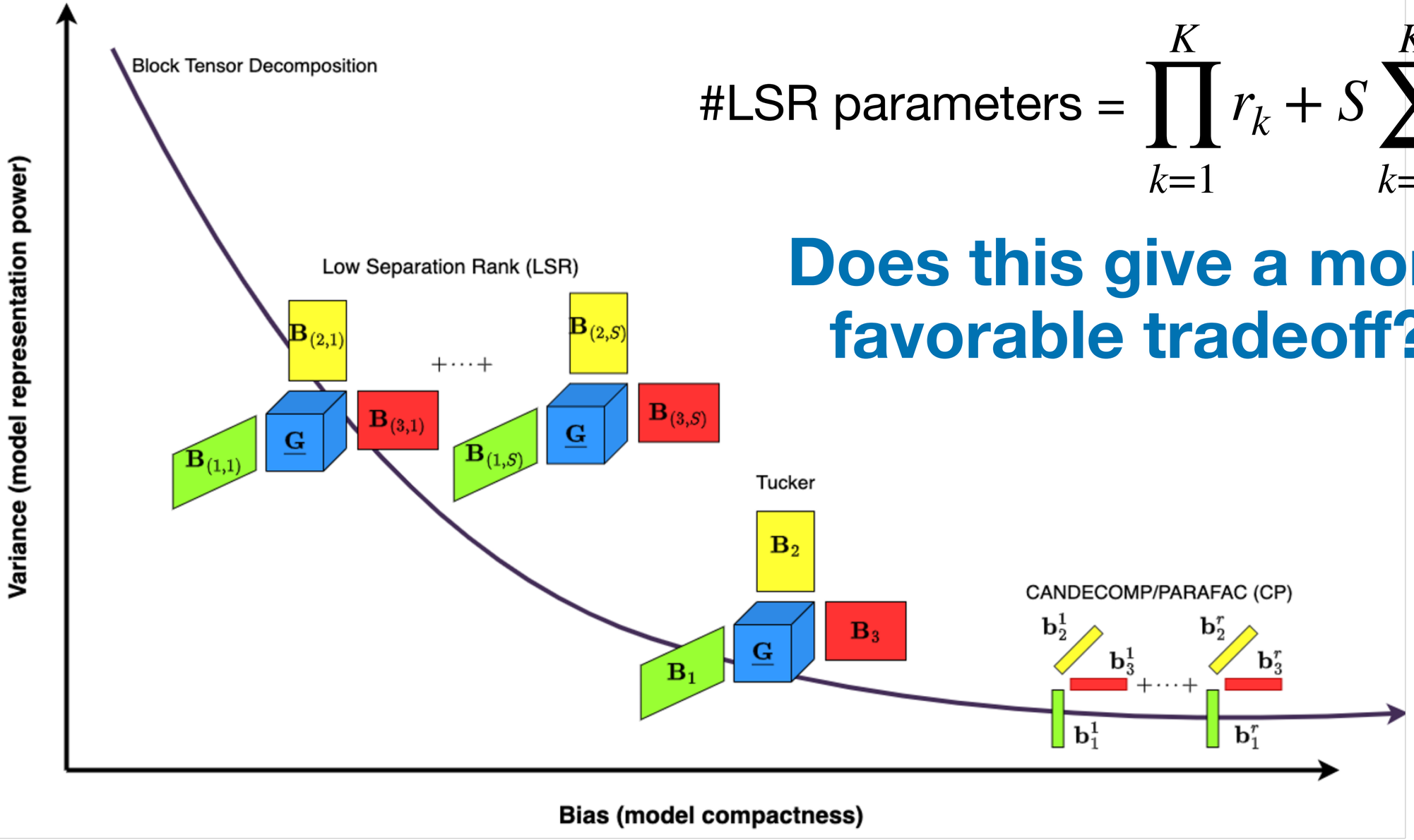
Coverings: find a small set of ϵ-balls in $\|\cdot\|_F$ which cover $\mathscr{C}_{\text{LSR}}$.

- Glue together packings for the factors $\underline{\mathbf{G}}$ and (orthogonal) $\{\underline{\mathbf{B}}_{(k,s)}\}$.

Results: we get sets of the right size…

$$\approx \exp\left(S\sum_k m_k r_k + \prod_k r_k\right)$$

Comparing different decompositions



$$\#\text{LSR parameters} = \prod_{k=1}^{K} r_k + S \sum_{k=1}^{K} m_k r_k$$

Does this give a more favorable tradeoff?

Regression and classification with LSR tensors

Generalized linear models for regression

Includes linear, logistic, Poisson, etc.

We have a *training set* of n tensor-scalar pairs $\{(\underline{\mathbf{X}}_i, y_i)\}$ following a **generalized linear model (GLM)**. Model the responses y as coming from an an *exponential family*:

$$p(y;\eta) = b(y)\exp\left(-\eta T(y) - a(\eta)\right).$$

Where the parameter $\eta = \langle \underline{\mathbf{B}}, \underline{\mathbf{X}} \rangle$. One example is *logistic regression*:

$$y \sim \text{Bernoulli}\left(\frac{1}{1+\exp(-\langle \underline{\mathbf{B}}, \underline{\mathbf{X}} \rangle)}\right)$$

Our goal: estimate $\underline{\mathbf{B}}$.

Prior work using CP and Tucker tensors

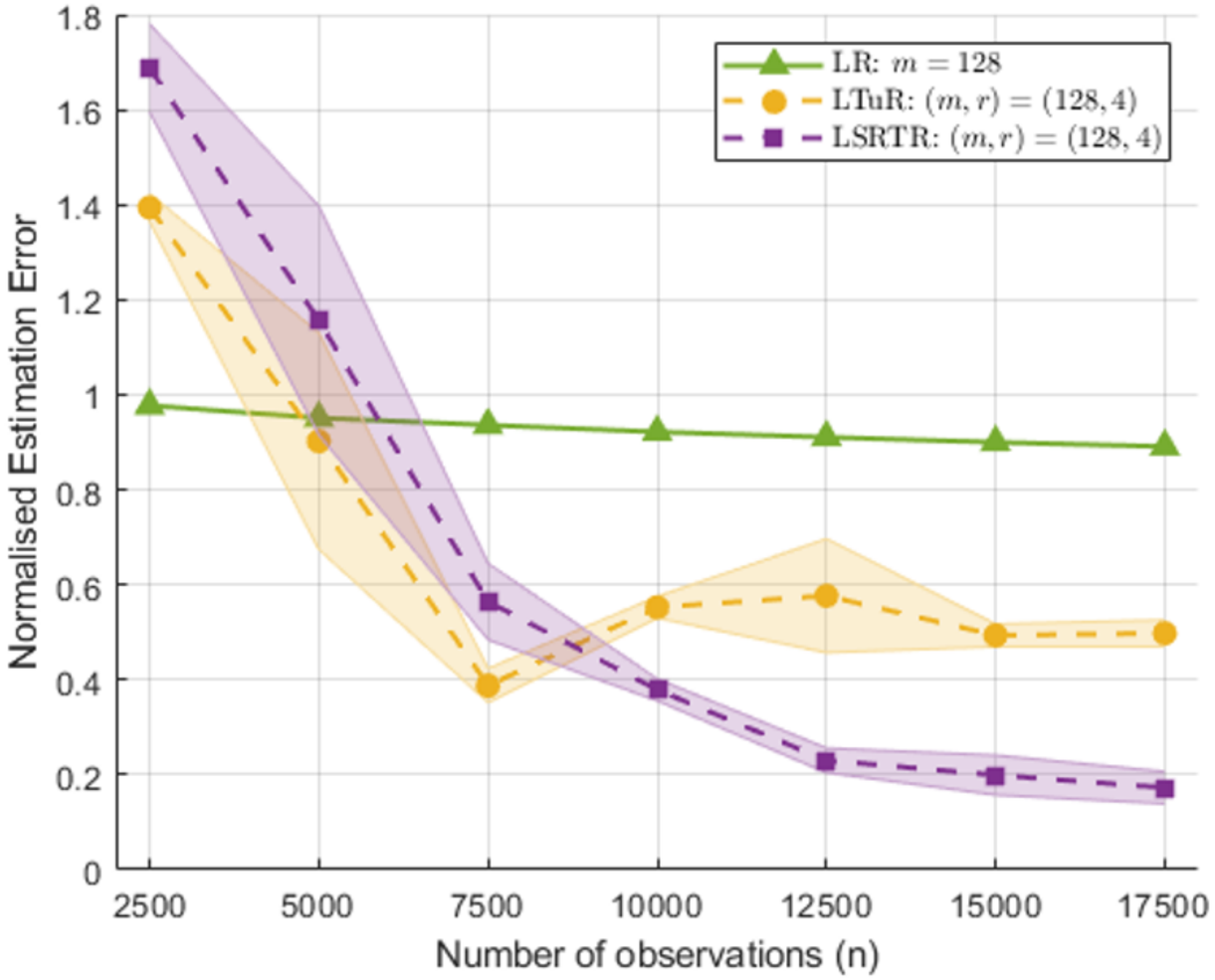
Generalized linear models

We look **LSR** models for **GLMs**:

- **CP** + **logistic regression** (Tan et al., 2012)
- **CP** + **GLMs** (Zhou et al. 2014)
- **Tucker** + **linear regression** (Zhang et al. 2020, Ahmed et al. 2020)
- **Tucker** + **logistic regression** (Zhang et al. 2016)
- **Tucker** + **GLMs** (Li et al., 2018; Zhou et al., 2013)

The benefits of more flexible modeling

Taking advantage of more data



LSR models let use scale the number of parameters to the data set size.

Synthetic data experiments show that with a modest number of samples, LSR models are better than vectorizing or using a Tucker model.

Estimation in GLMs using LSR Tensors

Mapping the tensor to a matrix

Using the LSR matrix in the vectorized problem

Under an LSR model, we have

$$\eta = \left\langle \sum_{s=1}^{S} \underline{\mathbf{G}} \times_1 \mathbf{B}_{(1,s)} \times_2 \mathbf{B}_{(2,s)} \times_3 \cdots \times_K \mathbf{B}_{(K,s)}, \underline{\mathbf{X}} \right\rangle$$

Vectorizing:

$$\eta = \left\langle \left(\sum_{s=1}^{S} \mathbf{B}_{(K,s)} \otimes \mathbf{B}_{(K-1,s)} \otimes \cdots \otimes \mathbf{B}_{(1,s)} \right) \mathbf{g}, \mathbf{x} \right\rangle$$

Maximum likelihood estimator (MLE)

Sorry, but it's a bit messy…

The MLE can is computed by minimizing

$$\sum_{i=1}^{n}\left[\left\langle\left(\sum_{s=1}^{S}\bigotimes_{k}\mathbf{B}_{(k,s)}\right)\mathbf{g},\mathbf{x}_i\right\rangle T(y_i)-a\left(\left\langle\left(\sum_{s=1}^{S}\bigotimes_{k}\mathbf{B}_{(k,s)}\right)\mathbf{g},\mathbf{x}_i\right\rangle\right)\right],$$

Over all $\mathbf{B}_{k,s}\in\mathbb{O}^{m_k\times r_k}$ and $\mathbf{g}\in\mathbb{R}^{r_1 r_2\cdots r_K}$.

Question: does the MLE work and is it optimal?

Identifiability using MLE

Sorry, but it's a bit messy…

Suppsse $\{(\underline{\mathbf{X}}_i, y_i) : i \in [n]\} \subset \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K} \times \mathbb{R}$ are generated from a GLM with an LSR-structured parameter $\underline{\mathbf{B}}^*$. Then if

$$n > \frac{C}{\epsilon^2}\left(\left(S\sum_k m_k r_k + \prod_k r_k\right)\log\left(\frac{C'}{\epsilon}\right) + \log\left(\frac{1}{\delta}\right)\right),$$

with probability $1-\delta$ the MLE will find a model $\underline{\hat{\mathbf{B}}}$ with excess risk no larger than ϵ.

A general lower bound for GLM + LSR

After much fun with algebra…

Suppose our data was generated with an LSR tensor $\underline{\mathbf{B}}^*$ We have a lower bound on the MSE for *any estimator* of $\underline{\mathbf{B}}^*$:

$$\mathbb{E}\left[\left\|\underline{\mathbf{B}}^* - \underline{\hat{\mathbf{B}}}\right\|_F^2\right] = \Omega\left(\frac{S\sum_k (m_k - 1)r_k + \prod_k (r_k - 1) - 1}{\left\|\mathbf{\Sigma}_x\right\|_2 n}\right)$$

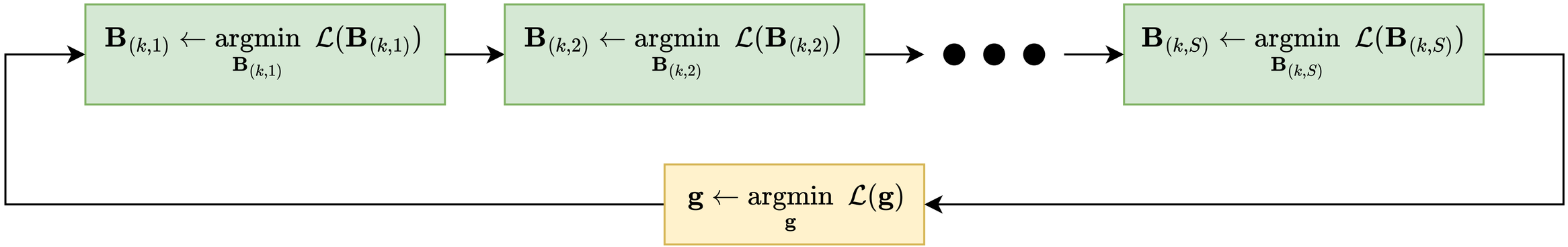
We can specialize this result to the Tucker and CP cases as well.

(Taki, Sarwate, Bajwa, 2023)

Regression	Structure of $\underline{\mathbf{B}}$			
	Unstructured	**CP**	**Tucker**	**LSR**
Linear	$\frac{\sigma_y^2 \widetilde{m}}{n}$ (Raskutti et al., 2011)	–	$\frac{\sigma_y^2 \left(\sum_{k \in [K]} m_k r_k - r_k^2 + \widetilde{r} \right)}{n}$ (Zhang et al., 2020)	–
Logistic	$\frac{\widetilde{m}}{n}$ (Abramovich & Grinshtein, 2016)	–	–	–
GLM	$\frac{\sigma_y^2 \widetilde{m}}{Dn}$ (Lee & Courtade, 2020)	$\frac{\sum_{k \in [K]} m_k r + r}{M \left\| \mathbf{\Sigma}_x \right\|_2 n}$ Corollary 2	$\frac{\sum_{k \in [K]} m_k r_k + \widetilde{r}}{M \left\| \mathbf{\Sigma}_x \right\|_2 n}$ Corollary 1	$\frac{S \sum_{k \in [K]} m_k r_k + \widetilde{r}}{M \left\| \mathbf{\Sigma}_x \right\|_2 n}$ Theorem 6

Alternating minimization: LSR-TR

Seems to work well in practice



Practically, MLE is intractable, so we use **alternating minimization** cycling through each $\mathbf{B}_{(k,s)}$ and then $\mathbf{g}$.

In particular, use projected gradient descent on each $\mathbf{B}_{(k,s)}$ and regular gradient descent on $\mathbf{g}$.

Experiments and applications

Experiments on medical imaging data

Data sets and algorithms

Data sets: ABIDE Autism [fMRI] (Craddock et al., 2013 2020), Vessel MNIST 3D [MRA] (Yang et al., 2020).

Other algorithms:

- **TTR**: **Tucker** + **GLMs** using a ‘block relaxation’ algorithm (Li et al., 2018)
- **LTuR**: **Tucker** + **logistic regression** with Frobenius norm regularization (Zhang & Jiang, 2016)
- **LR**: **Unstructured** + **logistic regression** (Seber & Lee, 2003)
- **LCPR**: **CP** + **logistic regression** (Tan et al., 2013)

ABIDE Autism data set

A tiny data set: $K = 2$, $\mathbf{m} = (111,116)$, $n = 80$

	SVM	LR	LCPR	LTuR	LSRTR
Sensitivity	0.71	0.71	0.71	0.71	1
Specificity	0.14	0.71	0.85	0.85	0.85
F1 score	0.55	0.71	0.77	0.77	**0.93**
AUC	0.42	0.51	0.84	0.84	**0.9**
Average Accuracy	0.43	0.71	0.78	0.78	**0.92**

- Chose ranks $r_1 = 6$ and $r_2 = 6$ with $S = 2$.
- Unstructured models are quite bad in the undersampled regime.
- Adding one more Tucker component can give significant improvements.

VesselMNIST 3D

Comparing against a DNN too: $K = 3$, $\mathbf{r} = (28,28,28)$, $n = 1335$

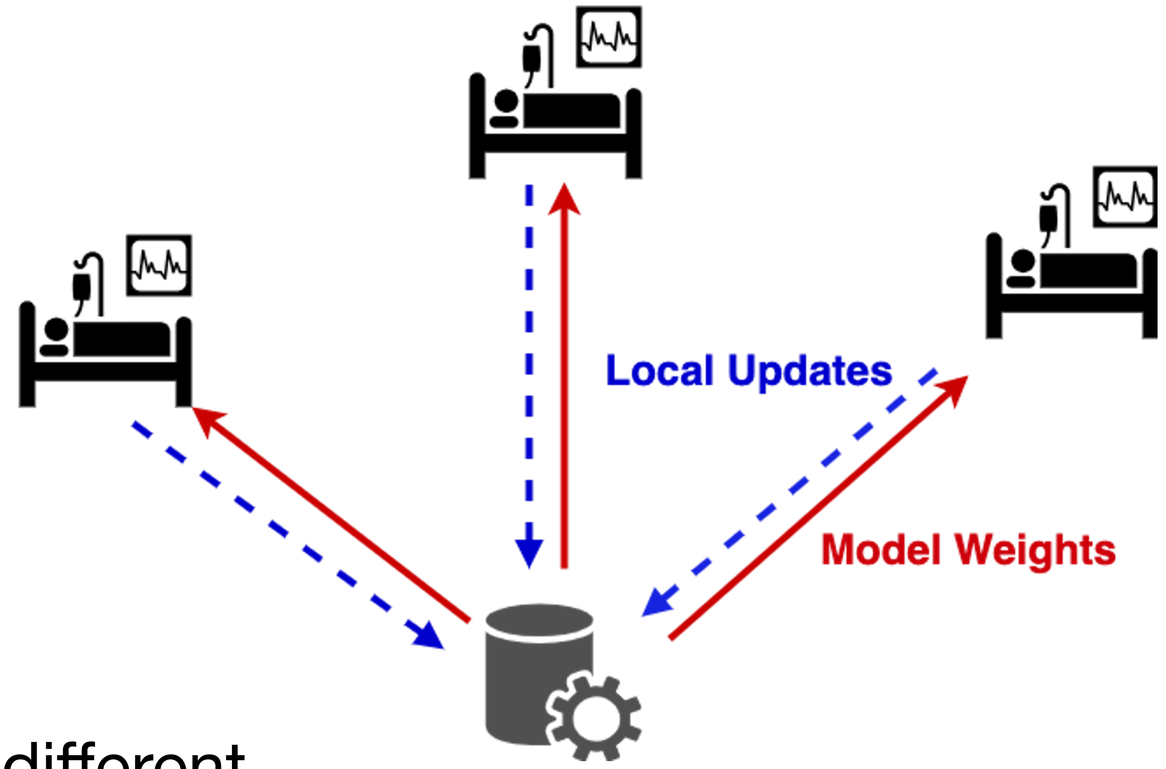
	SVM	LR	LCPR	LTuR	LSRTR	ResNet 50 + 3D
Sensitivity	0.39	0.53	0.26	0.32	0.47	0.85
Specificity	0.95	0.55	0.946	0.94	0.96	0.86
F1 score	0.44	0.21	0.3	0.37	0.55	**0.57**
AUC	0.84	0.52	0.6	0.66	0.81	**0.9**
Average Accuracy	0.89	0.55	0.869	0.87	**0.91**	0.85

- Chose ranks $r_1 = 3$, $r_2 = 3$, $r_3 = 3$, and $S = 2$
- LSRTR has better accuracy but worse F1 and AUC (see paper).
- Issues such as overfitting, interpretability, etc. are still open.

Federated learning from tensor valued data

Tensor data are often hard to acquire

In “federated learning” we want to efficiently learn from data which are held at different sites.

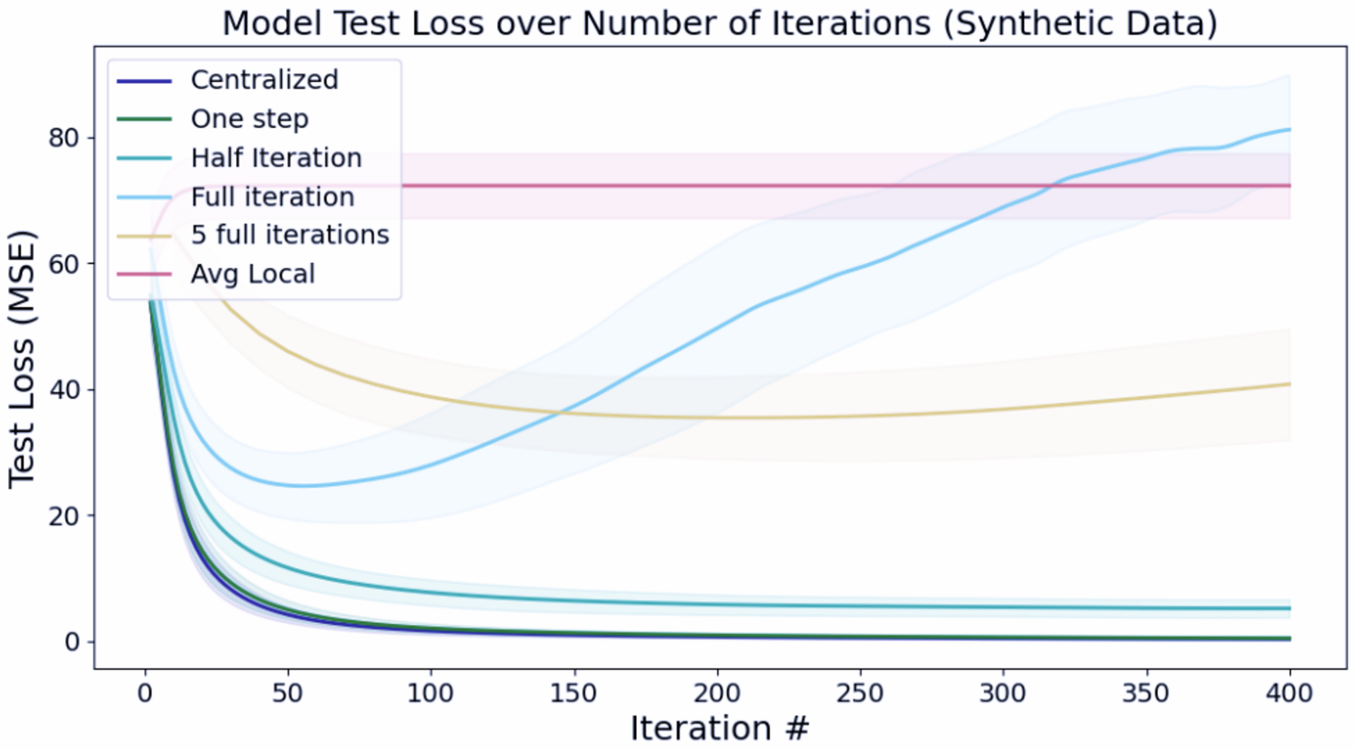


COINSTAC

Example: Given fMRI data collected by different research groups, learn a estimator of Alzheimer’s risk without sharing the “raw” data.

Balancing local and global updates

Empirical results are promising but preliminary



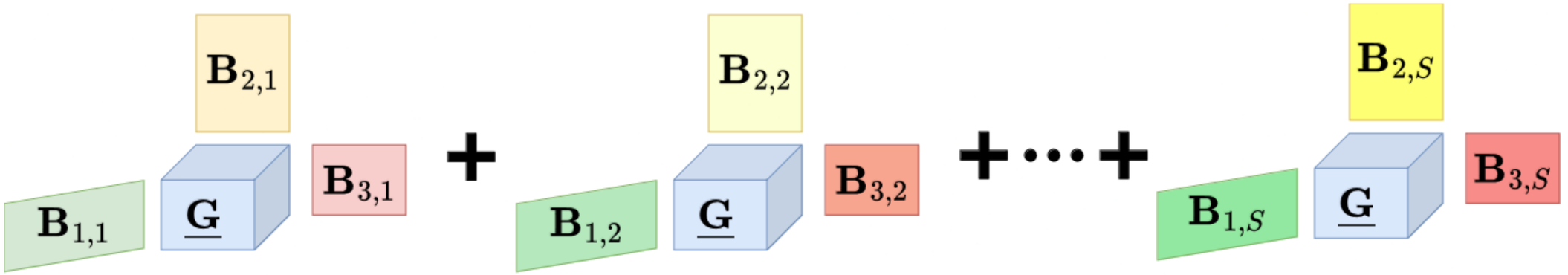
(Sanchez, Taki, Bajwa, S., 2024)

- Need tight coupling between local and centralized updates.
- Poses a challenge when communication reliability is a bottleneck.
- Lots of interesting work on the applications/engineering side!

Recap and looking forward

Recap of what we've seen

Structuring tensors using factorizations for simpler modeling



There is a whole continuum of tensor decompositions and **LSR structured tensors** can be very useful:

- Adapt parameterization to the data available.
- Efficiently (empirically) learnable/estimatable.

Dictionary learning with tensor factorizations

What if our dictionary has a Tucker structure?

A Tucker-structured dictionary:

$$\underbrace{\underline{\mathbf{Y}}}_{\in \mathbb{R}^{m_1 \times \cdots \times m_N}} = \overbrace{\underline{\mathbf{X}}}^{\text{Sparse core tensor}} \times_1 \underbrace{\mathbf{D}_1}_{\in \mathbb{R}^{m_1 \times p_1}} \times_2 \cdots \times_N \underbrace{\mathbf{D}_K}_{\in \mathbb{R}^{m_K \times p_K}} + \underline{\mathbf{W}}$$

What happens when we vectorize?

$$\underbrace{\mathbf{y}}_{\in \mathbb{R}^{m}} = \left(\mathbf{D}_K \otimes \mathbf{D}_{K-1} \otimes \cdots \otimes \mathbf{D}_1\right) \underbrace{\mathbf{x}}_{\in \mathbb{R}^{p}} + \mathbf{w}$$

Even a KS assumption can help

Even better results with LSR models (S > 1)

Original Image

Noisy Image

Unstructured DL:

147456 parameters

Separable DL:

265 parameters

Many questions remain!

Lots to understand on the theory and practical side

Theory

- Approximation and denoising guarantees.
- Topological pathologies vs. practical scenarios.
- Convex relaxations of LSR constraint for optimization
- Random tensor theory and spectral analysis.

Practice

- “Real” applications in neuroimaging and other domains.
- Other data domains: hyper spectral imaging, chemometrics, etc.
- Compression of neural network weights.
- Structuring embedding spaces for generative models.

Thank you!