Non-negative Tensor Decompositions for Unsupervised Learning

Joint work w/ Naomi Graham[1], Joel Saylor[2], & Michael Friedlander[1,3]

2023 SIAM PNW Conference at Western Washington University, Bellingham, WA

Nicholas Richardson[3]

October 14, 2023

[1]UBC Computer Science
[2]UBC Earth, Ocean, and Atmospheric Sciences
[3]UBC Mathematics

THE UNIVERSITY OF BRITISH COLUMBIA

Table of Contents

Overview

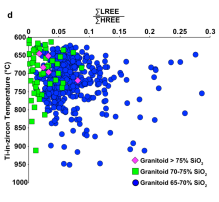
The Setting

Overview

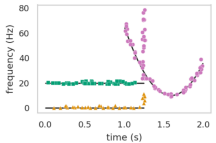
- Do you have a lot of data?
- A function that depends on many independent variables?
- High-dimensional samples?
- Want to compress the data?
- Or find relationships between samples/variables?

Tensor Decompositions are for you!

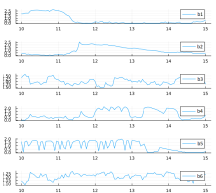
Suvanjanprasai n.d.



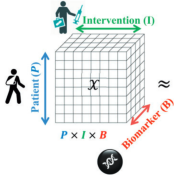
Sundell et al. 2022



Richardson, Schaeffer, and Tran 2023



Richardson n.d.



Luo, Wang, and Szolovits 2017

Benefits

Overview

- Less storage than original data
- Interpretable results
- There exist easy to implement algorithms with convergence guarantees
- Unlike supervised learning, no “training” required
- None or only a few hyperparameters to tune

Table of Contents

Tensor Models

The Data Tensor

Tensor Models

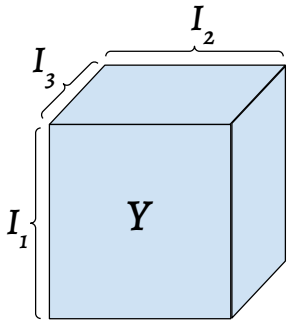


Figure: Order-3 Tensor

- Order-N tensor[1]

$$Y \in \mathbb{R}^{I_1 \times \cdots \times I_N}$$

- Often consider additional constraints
  - non-negative: $Y_{i_1 \ldots i_N} \in \mathbb{R}_+$
  - probability values: $0 \leq Y_{i_1 \ldots i_N} \leq 1$
  - distributions: $1 = \sum_j Y_{i_1 \ldots j \ldots i_N}$
  - binary: $Y_{i_1 \ldots i_N} \in \{0, 1\}$

[1] Kolda and Bader 2009

Tucker Decomposition[1]

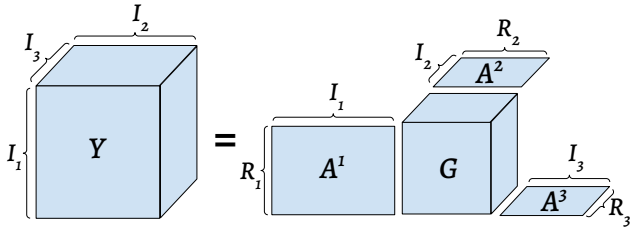
Tensor Models

- Factorize Y into a core tensor $G \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ and matrices $A^n \in \mathbb{R}^{I_n \times R_n}$

$$Y = G \times_1 A^1 \times_2 \cdots \times_N A^N$$

$$Y_{i_1 \ldots i_N} = \sum_{r_1 \ldots r_N} G_{r_1 \ldots r_N} A^1_{i_1 r_1} \cdots A^N_{i_N r_N}$$

- e.g. compress 1000^3 (4 GB) to $100^3 + 3 \cdot 100 \cdot 1000$ (5.2 MB)



[1]Tucker 1966

Special Cases of Tucker Decomposition

Tensor Models

- Canonical polyadic decomposition[1] (CP): $G = I$ and $R_1, \ldots, R_N = R$
- Smallest R is the tensor rank

$$Y = I \times_1 A^1 \times_2 \cdots \times_N A^N$$

$$Y_{i_1 \ldots i_N} = \sum_r A^1_{i_1 r} \cdots A^N_{i_N r}$$

e.g. low-rank decomp: $Y = AB^\top$

- Tucker-n[1]: $A^{n+1}, \ldots, A^N = I$ (possibly different sizes!)

$$Y = G \times_1 A^1 \times_2 \cdots \times_n A^n$$

e.g. Tucker-1/matrix-tensor factor[2]: $Y = G \times_1 A = AG$

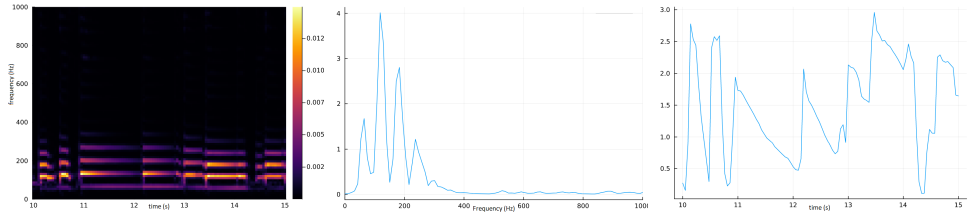
[1] Kolda and Bader 2009

[2] Richardson, Graham, et al. n.d.

CP Decomposition Example 1

Tensor Models

- Decompose one bass line (STFT) into Fourier Transform & Amplitude Envelope[1]
  - Order $N = 2$, rank $R = 1$

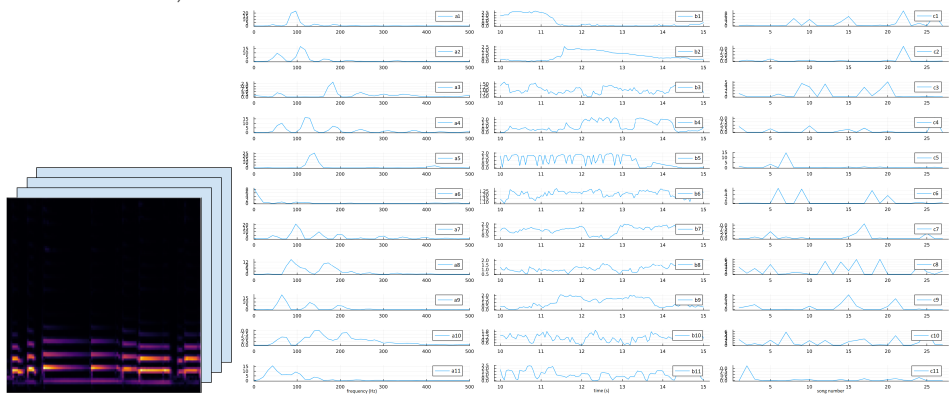


[1] Richardson n.d.

CP Decomposition Example 2

Tensor Models

- Decompose multiple bass lines into notes & envelopes for each song[1]
  - Order $N = 3$, rank $R = 11$

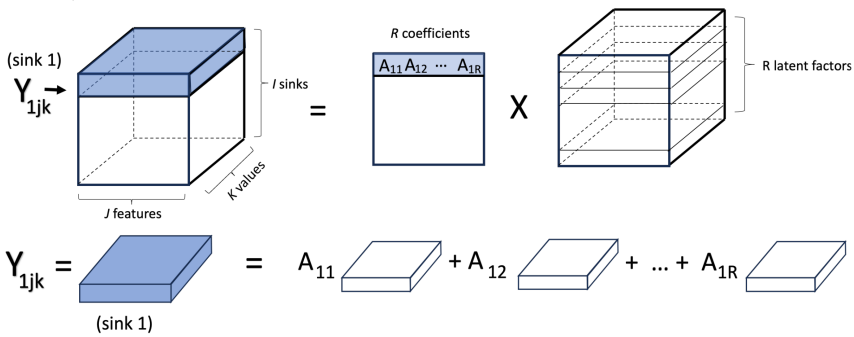


[1]Richardson n.d.

Tucker-1 Example

Tensor Models

- Decompose multiple sink's feature densities into linear combination of latent source densities[1]
  - Order 3, rank 3



[1]Richardson, Graham, et al. n.d.

Table of Contents

Computing Tensor Decompositions

The Optimization Problem

Computing Tensor Decompositions

- May try to solve the model exactly

$$Y = f(A^1, \ldots, A^N) \quad \text{(e.g. NMF: } Y = WH\text{)}$$

- With noise, or imperfect modelling, better to find "closest" fit

$$\min_{A^n \in C^n} D(Y, \hat{Y}) \quad \text{s.t.} \quad \hat{Y} = f(A^1, \ldots, A^N)$$

- Define objective function $D(Y, \hat{Y}) = F(A^1, \ldots, A^N)$
  - e.g. Least Squares: $\|Y - \hat{Y}\|_F^2$
  - e.g. KL-Divergence: $\sum_{i_1 \ldots i_N} Y_{i_1 \ldots i_N} \log \left(\frac{Y_{i_1 \ldots i_N}}{\hat{Y}_{i_1 \ldots i_N}} \right)$

Problems With Naive Methods

Computing Tensor Decompositions

- (Projected) Full Gradient Descent:

$$\mathcal{A} \leftarrow \arg\min_{\mathcal{B} \in \mathcal{C}} \langle \nabla F(\mathcal{A}), \mathcal{B} - \mathcal{A} \rangle + \frac{1}{2\alpha} \|\mathcal{B} - \mathcal{A}\|_F^2$$

$$(A^1, \ldots, A^N) \leftarrow P_{\mathcal{C}} \left((A^1, \ldots, A^N) - \alpha \nabla F(A^1, \ldots, A^N) \right)$$

  - Non-expensive updates, but F not convex, ∇F not Lipschitz

- Alternating Least-Squares:

$$A^n \leftarrow \arg\min_{A \in \mathcal{C}} \|Y - f(A^1, \ldots, A^{n-1}, A, A^{n+1}, \ldots, A^N)\|_F^2$$

  - f linear in A^n, but waste time fully optimizing A^n every step

Block Coordinate Descent (BCD) Algorithm[2]

Computing Tensor Decompositions

- Combine into BCD[1]:

$$A^n \leftarrow \arg\min_{A \in \mathcal{C}} \langle \nabla_{A^n} F, A - A^n \rangle + \frac{1}{2\alpha} \|A - A^n\|_F^2$$

$$A^n \leftarrow P_{\mathcal{C}} \left(A^n - \alpha \nabla_{A^n} F \right)$$

- Alternately update the factors, but don't fully optimize every iteration
- Can choose $\alpha = 1/L_n$
  - L is the Lipshitz constant of $\nabla F(A^1, \ldots, A^{n-1}, \cdot, A^{n+1}, \ldots, A^N)$

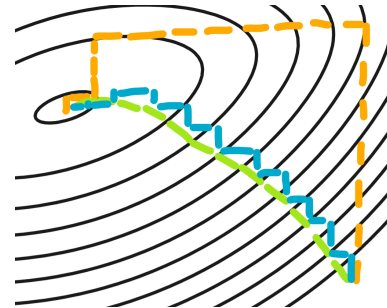


Figure: Example minimization with BCD (blue), Gradient Descent (green), Alternating Least Squares (orange)

[1] Xu and Yin 2013

[2] `https://github.com/njericha/Sediment-Source-Analysis.jl`

Table of Contents

Special Considerations

Convergence

Special Considerations

- BCD converges to (global) Nash point[1] $(A^1, \ldots, A^N)$:

$$A^n = \arg\min_{A \in C^n} F(A^1, \ldots, A^{n-1}, A, A^{n+1}, \ldots, A^N), \quad n = 1, \ldots, N$$

- "Cannot improve objective by updating one block"

Nash point condition[2]

Let $F(\cdot, B), F(A, \cdot)$ be convex, and $(A_0, B_0) \in C = C_A \times C_B$. Then,

$$\mathbf{0} \in \partial(F + \delta_C)(A_0, B_0) \iff \begin{array}{l} F(A_0, B_0) \leq F(A, B_0)\ \forall A \in C_A \\ F(A_0, B_0) \leq F(A_0, B)\ \forall B \in C_B \end{array}.$$

- When F is differentiable and C is convex:

$$\mathbf{0} \in \partial(F + \delta_C)(A, B) \iff -\nabla F \in N_C(A, B)$$

[1] Xu and Yin 2013

[2] Richardson, Graham, et al. n.d.

Selecting the Rank

Special Considerations

- Option 1: Use information about your physical system
  - e.g. decompose piano audio into notes and amplitudes
  - $R = 88$ since there are 88 keys
- Option 2: compute the "best bank for your buck"[1]
  - Solve the model for all ranks $r = 1, \ldots, I$
  - Compute final objective value $F(R)$
  - Select point of maximum curvature

$$R = \arg\max_r \kappa(r) := \frac{F''(r)}{(1 + F'(r)^2)^{3/2}}$$

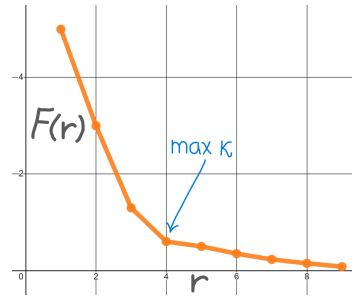


Figure: Typical final error $F(r)$ vs rank r plot

[1] Satopaa et al. 2011

Uniqueness & Scaling

Special Considerations

- These models are not (usually) unique
- Ex. $Y = AB^\top = (AC)(C^{-1}B^\top)$ for invertable C
- Fix a scaling on factors: set $\|A^n\| = c_n$ or $\sum_j A_{\dots j \dots} = c_n$ for…

… all but one factor

e.g.: $Y_{ij} = \sum_r A_{ir}B_{jr}$ s.t. $\|B_{:r}\|_2 = 1$

… all factors & add scaling parameter λ

e.g.: $Y_{ij} = \sum_r \lambda_r A_{ir}B_{jr}$ s.t. $\|A_{:r}\|_2, \|B_{:r}\|_2 = 1$

- Enforce through:
  - constraint (projection)
  - rescale at the end/each iteration
- Still only unique up to permutations of rows/columns/fibres

Table of Contents

Conclusion

Summary

Conclusion

- Looked at various tensor decomposition models
- Optimization methods to solve them
- Practical considerations

Future Directions for Additional Compression

Conclusion

- Factorize the core symmetrically[1] (Extend Tucker)

$$Y_{ijk} = \sum_{r_1,r_2,r_3} \underbrace{\sum_{p,q,s=1}^{R} B^1_{r_1 q s} B^2_{p r_2 s} B^3_{p q r_3}}_{G_{r_1 r_2 r_3}} A^1_{i r_1} A^2_{j r_2} A^3_{k r_3}$$

- Tensor Trains[2] (Extend Tucker-2)

$$Y_{i_1 \dots i_N} = \sum_{r_1, r_N} A^1_{i_1 r_1} \underbrace{\sum_{j_2 \dots j_{N-2}} A^2_{r_1 i_2 j_2} A^3_{j_2 i_3 j_3} \dots A^{n-1}_{j_{N-2} i_{N-1} r_2}}_{G_{r_1 i_2 \dots i_{N-1} r_N}} A^N_{i_N r_N}$$

- Factorize the matrices (Extend CP)

$$Y_{ij} = \sum_r A_{ir} \underbrace{\sum_k T_{jrk} b_k}_{B_{jr}}$$

[1] Qi et al. 2020

[2] Oseledets 2011

References

Kolda, Tamara G. and Brett W. Bader (Aug. 2009). "Tensor Decompositions and Applications". In: *SIAM Review* 51.3, pp. 455–500.

Luo, Yuan, Fei Wang, and Peter Szolovits (May 2017). "Tensor Factorization toward Precision Medicine". In: *Briefings in Bioinformatics* 18.3, pp. 511–514.

Oseledets, I. V. (Jan. 2011). "Tensor-Train Decomposition". In: *SIAM Journal on Scientific Computing* 33.5, pp. 2295–2317.

Qi, Liqun et al. (Mar. 2020). *Triple Decomposition and Tensor Recovery of Third Order Tensors*.

Richardson, Nicholas (n.d.). *A Consistant Framework for Non-negative Tensor Models and Algorithms with Applications*.

Richardson, Nicholas, Naomi Graham, et al. (n.d.). *Non-Negative Matrix-Tensor Factorization for Sediment Source Analysis*.

Richardson, Nicholas, Hayden Schaeffer, and Giang Tran (June 2023). "SRMD: Sparse Random Mode Decomposition". In: *Communications on Applied Mathematics and Computation*.

Satopaa, Ville et al. (June 2011). "Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior". In: *2011 31st International Conference on Distributed Computing Systems Workshops*, pp. 166–171.

Sundell, Kurt E. et al. (2022). "Crustal Thickening of the Northern Central Andean Plateau Inferred From Trace Elements in Zircon". In: *Geophysical Research Letters* 49.3, e2021GL096443.

Suvanjanprasai (n.d.). *Sample Images from MNIST Test Dataset*.

Tucker, Ledyard R. (Sept. 1966). "Some Mathematical Notes on Three-Mode Factor Analysis". In: *Psychometrika* 31.3, pp. 279–311.

Xu, Yangyang and Wotao Yin (Jan. 2013). "A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion". In: *SIAM Journal on Imaging Sciences* 6.3, pp. 1758–1789.

Thank you for listening!
Any questions?