# Model-driven Sparse CP Decomposition for High-Order Tensors

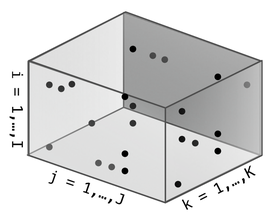
Jiajia Li, Ioakeim Perros, Jimeng Sun, Richard Vuduc
Georgia Institute of Technology

## Background

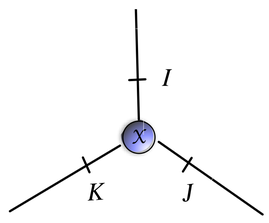
Tensor decomposition is a set of unsupervised methods to analyze and extract knowledge from tensors, which is widely used in healthcare analytics, image processing, machine learning, and social network analytics.

**Sparse tensors**

Many real-world tensors are hyper-sparse and have specific features. To discover useful knowledge, efficient sparse algorithms are critical to performance and scalability.



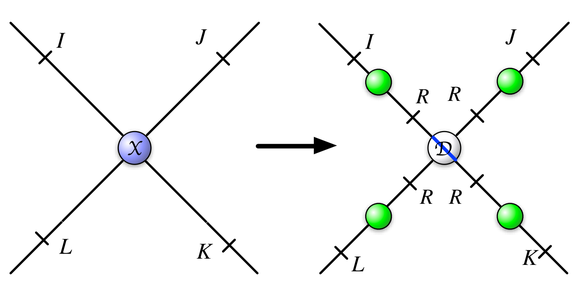

A 3rd-order sparse tensor




Tensor network diagram

**CPD**

CANDECOMP/PARAFAC decomposition (CPD) approximates an input tensor as a sum of component rank-one tensors, given the number of desired components. CPD is scalable in time and space compared to other low-rank methods.




CPD on a 4th-order tensor

## Contributions

- This work proposes an adaptive and efficient algorithm for the computational kernel, Matricized Tensor Times Khatri-Rao Product operation (MTTKRP) of the classical CANDECOMP/PARAFAC decomposition (CPD), by minimizing redundant computations.
- We also build a model-driven procedure to determine the adaptable algorithmic parameters for different input sparse tensors and enable the trade-off between time and space based on the user need and memory resource.
- Our adaptive algorithm shows up to 10x speedup compared to state of the art on real-world high order tensors. Our method also shows near constant scalability with respect to the tensor order, while using acceptable storage space.

## Algorithms

- Decouple an MTTKRP operation into a TTM and a TMHR.

**Algorithm 3** Memorized Full MTTKRP algorithm (memFM).

**Input:** A fourth-order sparse tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K \times L}$, dense factor matrices $\mathbf{B} \in \mathbb{R}^{J \times R}$, $\mathbf{C} \in \mathbb{R}^{K \times R}$, $\mathbf{D} \in \mathbb{R}^{L \times R}$, the product order $p_o$;
**Output:** Updated dense factor matrix $\tilde{\mathbf{A}} \in R^{I \times R}$;

▷ $\tilde{\mathbf{A}} \leftarrow \mathcal{X}_{(1)}(\mathbf{D} \odot \mathbf{C} \odot \mathbf{B})$

// Compute as $p_o = [4, 3, 2]$.
1: Save $\mathcal{Y}^{(1)}$: $\mathcal{Y}^{(1)} \leftarrow \mathcal{X} \times_4 \mathbf{D}$
2: Save $\mathcal{Y}^{(2)}$: $\mathcal{Y}^{(2)} = \text{TMHR}\ (\mathcal{Y}^{(1)}, \mathbf{C})$
3: $\tilde{\mathbf{A}} = \text{TMHR}\ (\mathcal{Y}^{(2)}, \mathbf{B})$ **return** $\tilde{\mathbf{A}}$;

TTM ●
TMHR ▮

1st MTTKRP $\tilde{\mathbf{A}} \leftarrow \mathcal{X}_{(1)}(\mathbf{D} \odot \mathbf{C} \odot \mathbf{B}) \Leftrightarrow \tilde{A} \leftarrow \mathcal{X}$ ● D ▮ C ▮ B, with $\mathcal{Y}^{(1)}$ = $\mathcal{X}$ ● D and $\mathcal{Y}^{(2)}$ = $\mathcal{Y}^{(1)}$ ▮ C

## Algorithms

- Design a simple memoization algorithm for MTTKRP sequence.

$\tilde{\mathbf{A}} \leftarrow \mathcal{X}_{(1)}(\mathbf{D} \odot \mathbf{C} \odot \mathbf{B}) \Leftrightarrow \tilde{A} \leftarrow \mathcal{X}$ ● $D$ ▮ $C$ ▮ $B$, with $\mathcal{Y}^{(1)}$ = $\mathcal{X}$ ● $D$ and $\mathcal{Y}^{(2)}$ = $\mathcal{Y}^{(1)}$ ▮ $C$ — Reuse degree: 2

$\Leftrightarrow \tilde{B} \leftarrow \mathcal{Y}^{(2)}$ ▮ $\tilde{A}$

$\tilde{\mathbf{C}} \leftarrow \mathcal{X}_{(3)}(\mathbf{D} \odot \tilde{\mathbf{B}} \odot \tilde{\mathbf{A}}) \Leftrightarrow \tilde{C} \leftarrow \mathcal{Y}^{(1)}$ ▮ $\tilde{B}$ ▮ $\tilde{A}$

$\tilde{\mathbf{D}} \leftarrow \mathcal{X}_{(4)}(\tilde{\mathbf{C}} \odot \tilde{\mathbf{B}} \odot \tilde{\mathbf{A}}) \Leftrightarrow \tilde{D} \leftarrow \mathcal{X}$ ● $\tilde{C}$ ▮ $\tilde{B}$ ▮ $\tilde{A}$

One Group

- Design an **adaptive** memoization algorithm for MTTKRP sequence, to flexibly choose number of memoized MTTKRPs and reuse degrees.

$\tilde{\mathbf{A}} \leftarrow \mathcal{X}_{(1)}(\mathbf{D} \odot \mathbf{C} \odot \mathbf{B}) \Leftrightarrow \tilde{A} \leftarrow \mathcal{X}$ ● $D$ ▮ $C$ ▮ $B$, with $\mathcal{Y}^{(2)}$ = $\mathcal{X}$ ● $D$ ▮ $C$ — Reuse degree: 1

$\tilde{\mathbf{B}} \leftarrow \mathcal{X}_{(2)}(\mathbf{D} \odot \mathbf{C} \odot \tilde{\mathbf{A}}) \Leftrightarrow \tilde{B} \leftarrow \mathcal{Y}^{(2)}$ ▮ $\tilde{A}$

Group 1

$\tilde{\mathbf{C}} \leftarrow \mathcal{X}_{(3)}(\tilde{\mathbf{B}} \odot \tilde{\mathbf{A}} \odot \mathbf{D}) \Leftrightarrow \tilde{C} \leftarrow \mathcal{X}$ ● $\tilde{B}$ ▮ $\tilde{A}$ ▮ $D$, with $\mathcal{Z}^{(2)}$ = $\mathcal{X}$ ● $\tilde{B}$ ▮ $\tilde{A}$ — Reuse degree: 1

$\tilde{\mathbf{D}} \leftarrow \mathcal{X}_{(4)}(\tilde{\mathbf{B}} \odot \tilde{\mathbf{A}} \odot \tilde{\mathbf{C}}) \Leftrightarrow \tilde{D} \leftarrow \mathcal{Z}^{(2)}$ ▮ $\tilde{C}$

Group 2

- Decrease the operation number.
- Probably increase the storage space.

Tradeoff

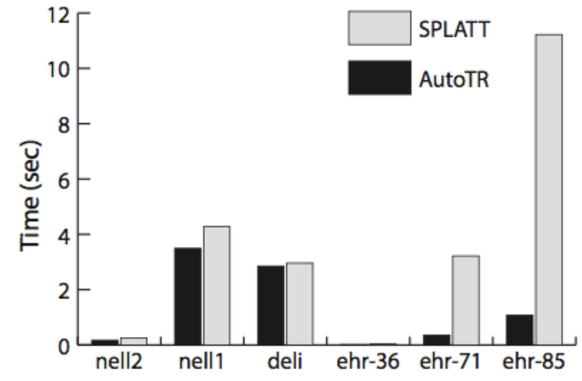
- Build a mode-driven auto-tuner to accurately predict the algorithm parameters for sparse tensors.

## Results

We test our algorithm on Intel Xeon E7-4820 platform using 16 threads.

| Dataset | Order | Max Mode Size | NNZ | Density |
|---|---|---|---|---|
| nell2 | 3 | 30K | 77M | 1.3e-05 |
| nell1 | 3 | 25M | 144M | 3.1e-13 |
| deli | 3 | 17M | 140M | 6.1e-12 |
| ehr-36 | 36 | 19 | 11K | 4.7e-26 |
| ehr-71 | 71 | 21 | 221K | 1.4e-55 |
| ehr-85 | 85 | 21 | 920K | 7.9e-68 |

Sparse tensor description




CP-ALS runtime using SPLATT and AutoTR

## Conclusion

Our optimization to increase the reuse of MTTKRP sequence is beneficial for high-order tensors. We use acceptable space to trade for higher performace.

## References

B. W. Bader, T. G. Kolda et al., “Matlab tensor toolbox (Version 2.6),” Available online, February 2015.

S. Smith, N. Ravindran, N. Sidiropoulos, and G. Karypis, “Splatt: Efficient and parallel sparse tensor-matrix mul- tiplication,” in Proceedings of the 29th IEEE Interna- tional Parallel & Distributed Processing Symposium, ser. IPDPS, 2015.

Acknowledgement: This project is supported by the U.S. National Science Foundation (NSF) Award Number 1533768.