Constrained Tensor Factorization with Accelerated AO-ADMM

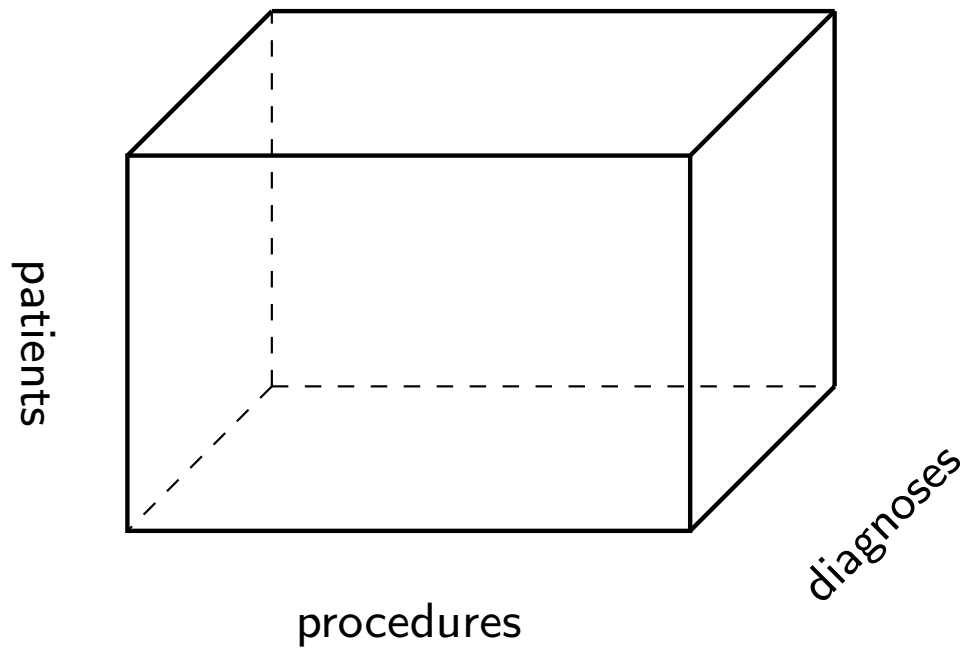
Shaden Smith[1*], Alec Beri[2], and George Karypis[1]

[1]Department of Computer Science & Engineering, University of Minnesota
[2]Department of Computer Science, University of Maryland
[*]shaden@cs.umn.edu

Table of Contents

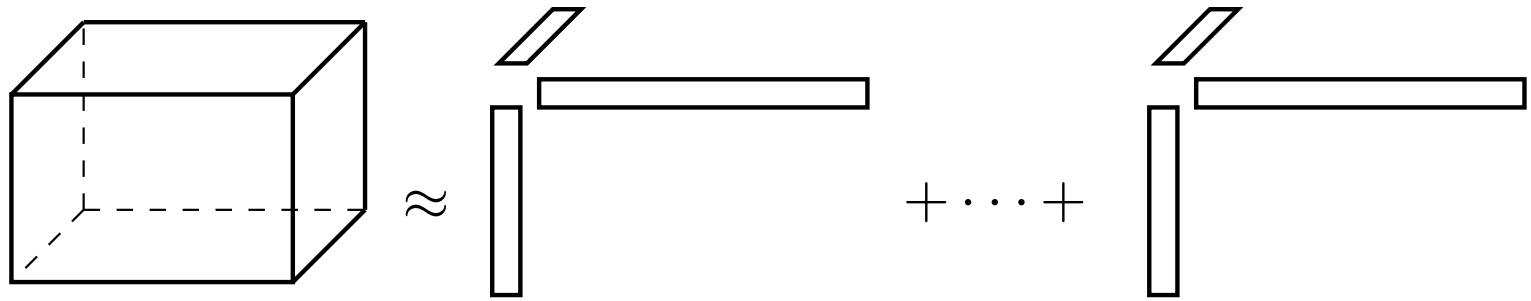
Tensor Introduction

- Tensors are the generalization of matrices to higher dimensions.
- Allow us to represent and analyze multi-dimensional data.
- Applications in precision healthcare, cybersecurity, recommender systems, ...



Canonical polyadic decomposition (CPD)

The CPD models a tensor as the summation of rank-1 tensors.



$$\underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\text{minimize}} \quad \mathcal{L}(\boldsymbol{\mathcal{X}}, \mathbf{A}, \mathbf{B}, \mathbf{C}) = \left\| \boldsymbol{\mathcal{X}} - \sum_{f=1}^{F} \mathbf{A}(:, f) \circ \mathbf{B}(:, f) \circ \mathbf{C}(:, f) \right\|_F^2$$

Notation
$\mathbf{A} \in \mathbb{R}^{I \times F}$, $\mathbf{B} \in \mathbb{R}^{J \times F}$, and $\mathbf{C} \in \mathbb{R}^{K \times F}$ denote the factor matrices for a 3D tensor.

Alternating least squares (ALS)

The CPD is most commonly computed with ALS:

Algorithm 1 CPD-ALS

1: **while** not converged **do**

2: $\quad \mathbf{A}^T \leftarrow (\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B})^{-1} \left(\mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})\right)^T$

3: $\quad \mathbf{B}^T \leftarrow (\mathbf{C}^T\mathbf{C} * \mathbf{A}^T\mathbf{A})^{-1} \left(\mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})\right)^T$

4: $\quad \mathbf{C}^T \leftarrow \underbrace{(\mathbf{B}^T\mathbf{B} * \mathbf{A}^T\mathbf{A})^{-1}}_{\text{Normal equations}} \underbrace{\left(\mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})\right)^T}_{\text{MTTKRP}}$

5: **end while**

Notation

$*$ denotes the Hadamard (elementwise) product.

Constrained factorization

We often want to impose some constraints or regularizations on the factorization:

$$\underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\text{minimize}} \quad \underbrace{\mathcal{L}(\mathcal{X}, \mathbf{A}, \mathbf{B}, \mathbf{C})}_{\text{Loss}} + \underbrace{r(\mathbf{A}) + r(\mathbf{B}) + r(\mathbf{C})}_{\text{Constraints/Regularizations}}$$

Example

Non-negative factorizations use an indicator function for $\mathbb{R}_+$:

$$r(\mathbf{A}) = \begin{cases} 0 & \text{if } \mathbf{A} \geq \mathbf{0} \\ \infty & \text{otherwise} \end{cases}$$

AO-ADMM [Huang & Sidiropoulos '15]

AO-ADMM combines alternating optimization (AO) with alternating direction method of multipliers (ADMM).

- **A**, **B**, and **C** are updated in sequence using ADMM.

AO-ADMM [Huang & Sidiropoulos '15]

AO-ADMM combines alternating optimization (AO) with alternating direction method of multipliers (ADMM).

- $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are updated in sequence using ADMM.

ADMM formulation for the update of $\mathbf{A}$:

$$\begin{aligned} \underset{\mathbf{A},\tilde{\mathbf{A}}}{\text{minimize}} \quad & \frac{1}{2}\left\|\mathbf{X}_{(1)} - \tilde{\mathbf{A}}^T(\mathbf{C}\odot\mathbf{B})^T\right\|_F^2 + r(\mathbf{A}) \\ \text{subject to} \quad & \mathbf{A} = \tilde{\mathbf{A}}^T. \end{aligned}$$

Alternating optimization step (outer iterations)

1: Initialize primal variables $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ randomly.
2: Initialize dual variables $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$, and $\hat{\mathbf{C}}$ with $\mathbf{0}$.
3: **repeat**
4: $\quad \mathbf{G} \leftarrow \mathbf{B}^T\mathbf{B} * \mathbf{C}^T\mathbf{C}$
5: $\quad \mathbf{K} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$
6: $\quad \mathbf{A}, \hat{\mathbf{A}} \leftarrow \text{ADMM}(\mathbf{A}, \hat{\mathbf{A}}, \mathbf{K}, \mathbf{G})$
7:
8: $\quad \mathbf{G} \leftarrow \mathbf{A}^T\mathbf{A} * \mathbf{C}^T\mathbf{C}$
9: $\quad \mathbf{K} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})$
10: $\quad \mathbf{B}, \hat{\mathbf{B}} \leftarrow \text{ADMM}(\mathbf{B}, \hat{\mathbf{B}}, \mathbf{K}, \mathbf{G})$
11:
12: $\quad \mathbf{G} \leftarrow \mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}$
13: $\quad \mathbf{K} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})$
14: $\quad \mathbf{C}, \hat{\mathbf{C}} \leftarrow \text{ADMM}(\mathbf{C}, \hat{\mathbf{C}}, \mathbf{K}, \mathbf{G})$
15: **until** $\mathcal{L}(\mathcal{X}, \mathbf{A}, \mathbf{B}, \mathbf{C})$ ceases to improve.

ADMM step (inner iterations)

ADMM to update one factor matrix:

1: **Input:** $\mathbf{H}$, $\mathbf{U}$, $\mathbf{K}$, $\mathbf{G}$
2: **Output:** $\mathbf{H}$, $\mathbf{U}$
3: $\rho \leftarrow \text{trace}(\mathbf{G})/F$
4: $\mathbf{L} \leftarrow \text{Cholesky}(\mathbf{G} + \rho\mathbf{I})$
5: **repeat**
6: $\quad \mathbf{H}_0 \leftarrow \mathbf{H}$
7: $\quad \tilde{\mathbf{H}} \leftarrow \mathbf{L}^{-T}\mathbf{L}^{-1}\left(\mathbf{K} + \rho(\mathbf{H} + \mathbf{U})\right)^T$
8: $\quad \mathbf{H} \leftarrow \text{argmin}_{\mathbf{H}}\, r(\mathbf{H}) + \frac{\rho}{2}||\mathbf{H} - \tilde{\mathbf{H}}^T + \mathbf{U}||_F^2$
9: $\quad \mathbf{U} \leftarrow \mathbf{U} + \mathbf{H} - \tilde{\mathbf{H}}^T$
10: $\quad r \leftarrow ||\mathbf{H} - \tilde{\mathbf{H}}^T||_F^2 / ||\mathbf{H}||_F^2$
11: $\quad s \leftarrow ||\mathbf{H} - \mathbf{H}_0||_F^2 / ||\mathbf{U}||_F^2$
12: **until** $r < \epsilon$ and $s < \epsilon$

Table of Contents

Parallelization opportunities

All steps but Line 8 are either element-wise or row-wise independent.

1: **Input:** $\mathbf{H}$, $\mathbf{U}$, $\mathbf{K}$, $\mathbf{G}$
2: **Output:** $\mathbf{H}$, $\mathbf{U}$
3: $\rho \leftarrow \text{trace}(\mathbf{G})/F$
4: $\mathbf{L} \leftarrow \text{Cholesky}(\mathbf{G} + \rho\mathbf{I})$
5: **repeat**
6: $\quad \mathbf{H}_0 \leftarrow \mathbf{H}$
7: $\quad \tilde{\mathbf{H}} \leftarrow \mathbf{L}^{-T}\mathbf{L}^{-1}\left(\mathbf{K} + \rho(\mathbf{H} + \mathbf{U})\right)^T$
8: $\quad \mathbf{H} \leftarrow \text{argmin}_{\mathbf{H}}\, r(\mathbf{H}) + \frac{\rho}{2}||\mathbf{H} - \tilde{\mathbf{H}}^T + \mathbf{U}||_F^2$
9: $\quad \mathbf{U} \leftarrow \mathbf{U} + \mathbf{H} - \tilde{\mathbf{H}}^T$
10: $\quad r \leftarrow ||\mathbf{H} - \tilde{\mathbf{H}}^T||_F^2 / ||\mathbf{H}||_F^2$
11: $\quad s \leftarrow ||\mathbf{H} - \mathbf{H}_0||_F^2 / ||\mathbf{U}||_F^2$
12: **until** $r < \epsilon$ and $s < \epsilon$

Performance opportunities

1. The factor matrices are tall-skinny (e.g., $10^6 \times 50$).
   - The ADMM step will be bound by memory bandwidth.
2. Real-world tensors have non-uniform distributions of non-zeros.
   - This may lead to non-uniform convergence of the factor rows during ADMM.
3. Many constraints and regularizations naturally invoke sparsity in the factors.
   - We can exploit this sparsity during MTTKRP (*in paper*).

Blocked ADMM

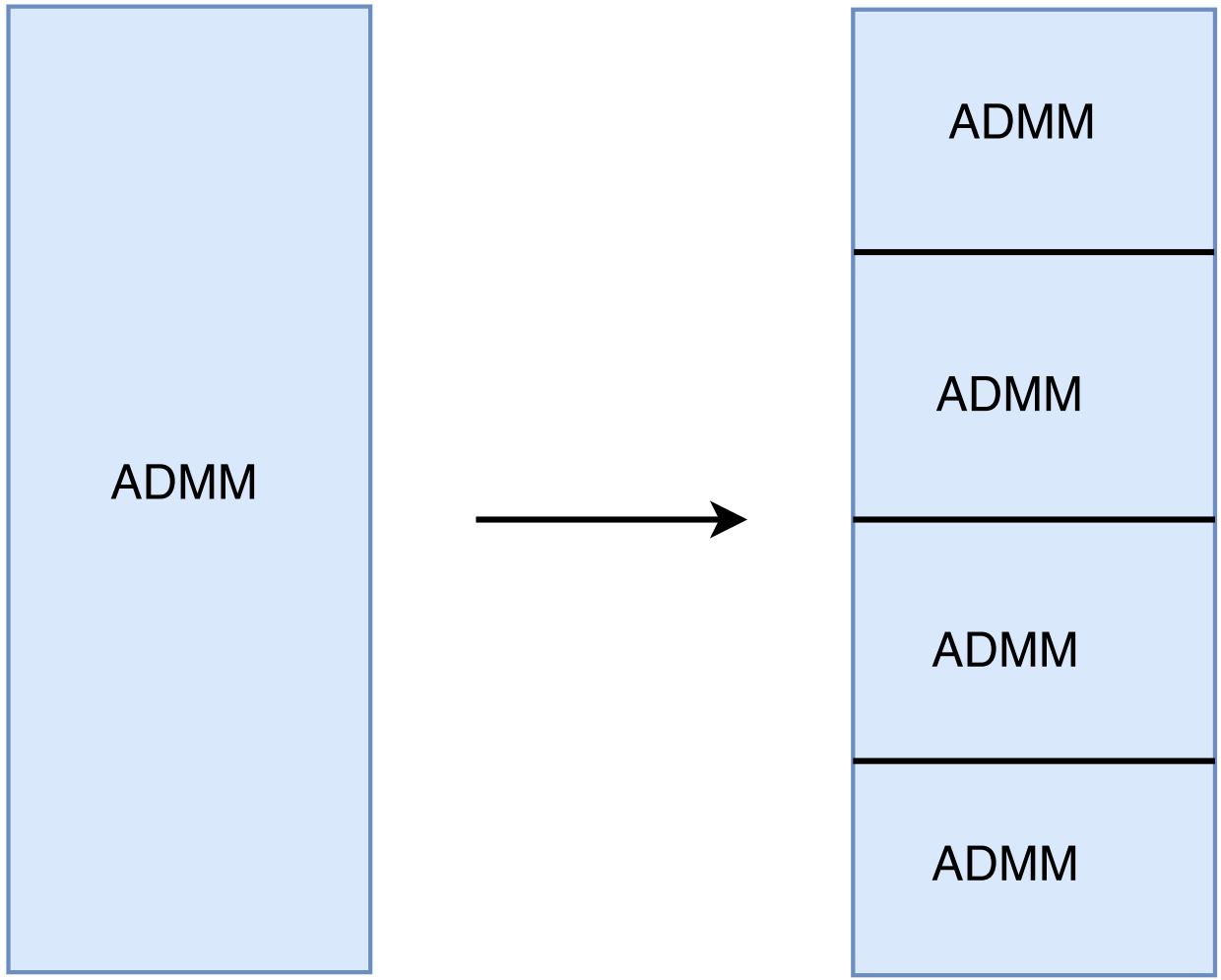
If the proximity operator coming from $r(\cdot)$ is row-separable, reformulate the ADMM problem to work on B blocks of rows:

$$\begin{aligned} \underset{(\mathbf{A}_1,\tilde{\mathbf{A}}_1),\ldots,(\mathbf{A}_B,\tilde{\mathbf{A}}_B)}{\text{minimize}} \quad & \sum_{b=1}^{B} \frac{1}{2} \left\| (\mathbf{X}_{(1)})_b - \tilde{\mathbf{A}}_b^T (\mathbf{C} \odot \mathbf{B})_b^T \right\|_F^2 + r(\mathbf{A}_b) \\ \text{subject to} \quad & \mathbf{A}_1 = \tilde{\mathbf{A}}_1, \ldots, \mathbf{A}_B = \tilde{\mathbf{A}}_B. \end{aligned}$$

Optimizing each block separately allows for them to converge at different rates, while acting as a form of cache tiling.

Blocked ADMM

More simply:



Effects of block size

The block size affects both convergence rate and computational efficiency:

- A block size of 1 optimizes each row of $\mathbf{H}$ independently.
- Larger block sizes better utilize hardware resources, but should be chosen to fit in cache.

Our evaluation uses $F{=}50$, and we experimentally found a block size of 50 rows to be a good balance between convergence rate and performance.

Table of Contents

Experimental Setup

Source code:

- Modified from SPLATT [1]
- Written in C and parallelized with OpenMP
- Compiled with `icc` v17.0.1 and linked with Intel MKL

Machine specifications:

- 2× 10-core Intel Xeon E5-2650v3 (Haswell)
- 396GB RAM

[1] `https://github.com/ShadenSmith/splatt`

Convergence measurement

We measure convergence based on the relative reconstruction error:

$$\text{relative error} = \frac{\mathcal{L}(\boldsymbol{\mathcal{X}}, \mathbf{A}, \mathbf{B}, \mathbf{C})}{\|\boldsymbol{\mathcal{X}}\|_F^2}.$$

Termination:

- Convergence is detected when the relative error improves less than 10^{-6} or if we exceed 200 outer iterations.
- ADMM is limited to 50 iterations and $\epsilon = 10^{-2}$.

Datasets

We selected four tensors from the FROSTT [2] collection based on non-negative factorization performance:

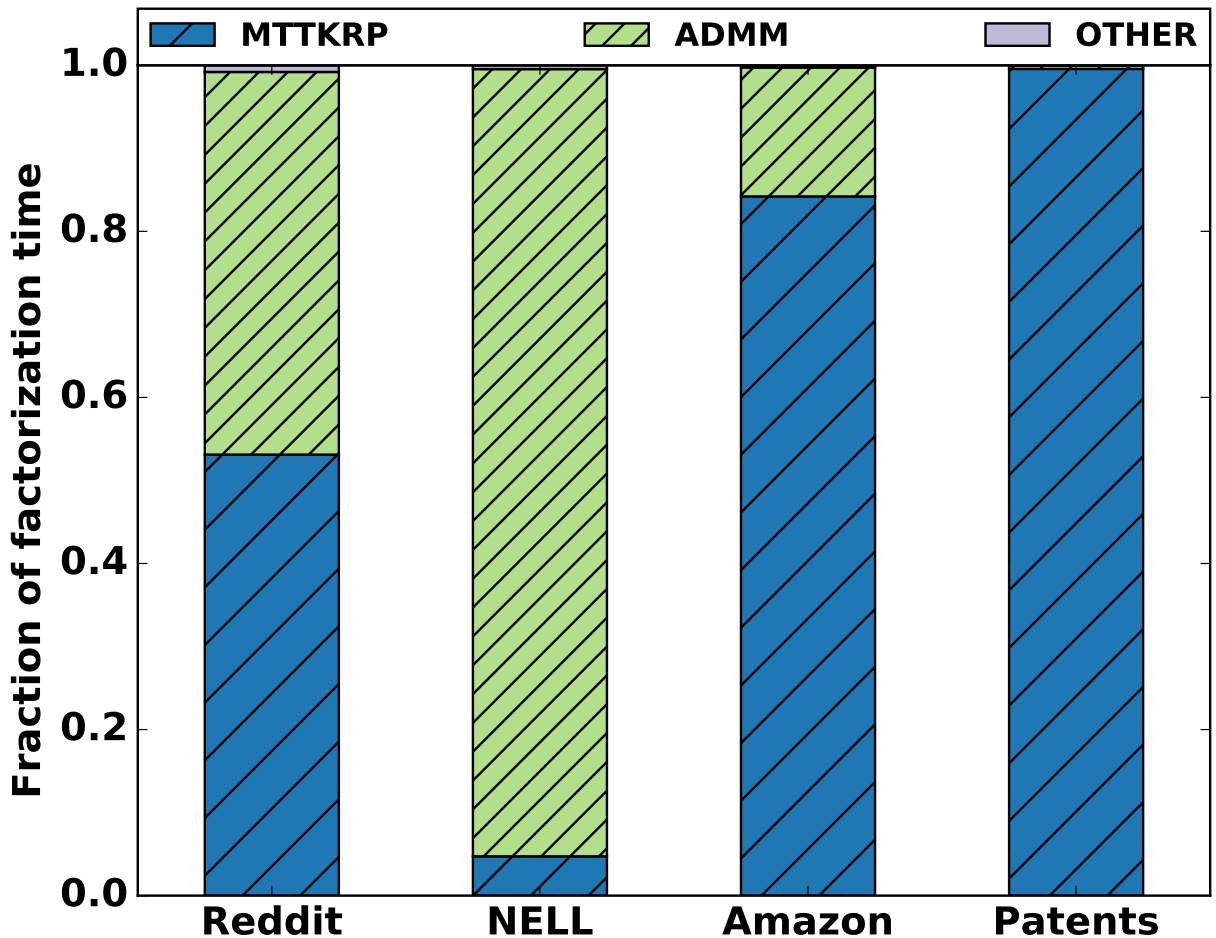
- require a non-trivial number of iterations
- have a factorization quality that suggests a non-negative CPD is appropriate

Dataset	NNZ	I	J	K
Reddit	95M	310K	6K	510K
NELL	143M	3M	2M	25M
Amazon	1.7B	5M	18M	2M
Patents	3.5B	46	240K	240K

[2] http://frostt.io/

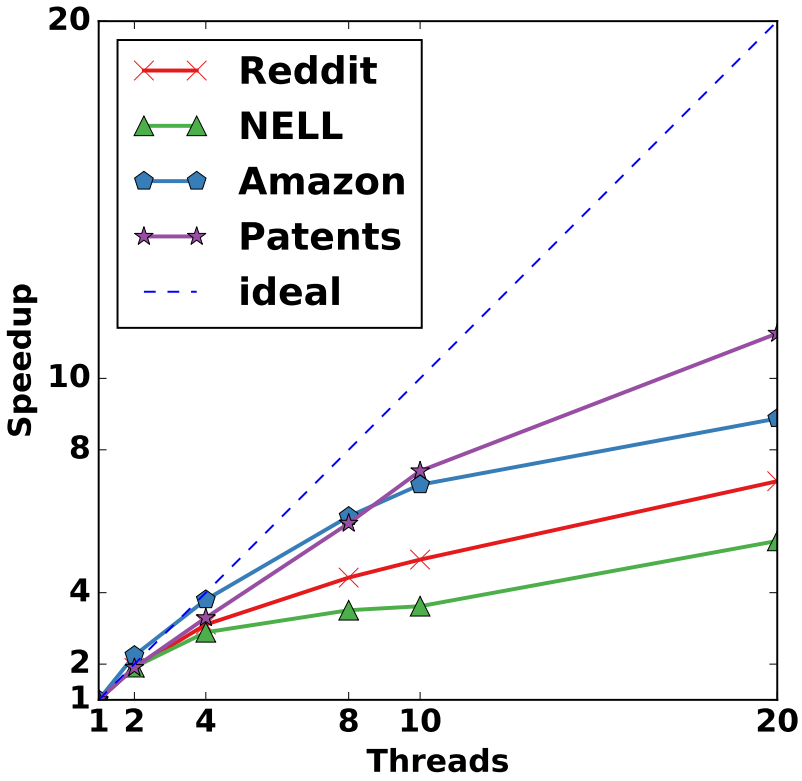
Relative Factorization Costs

Fraction of time spent in MTTKRP and ADMM during a rank-50 non-negative factorization:

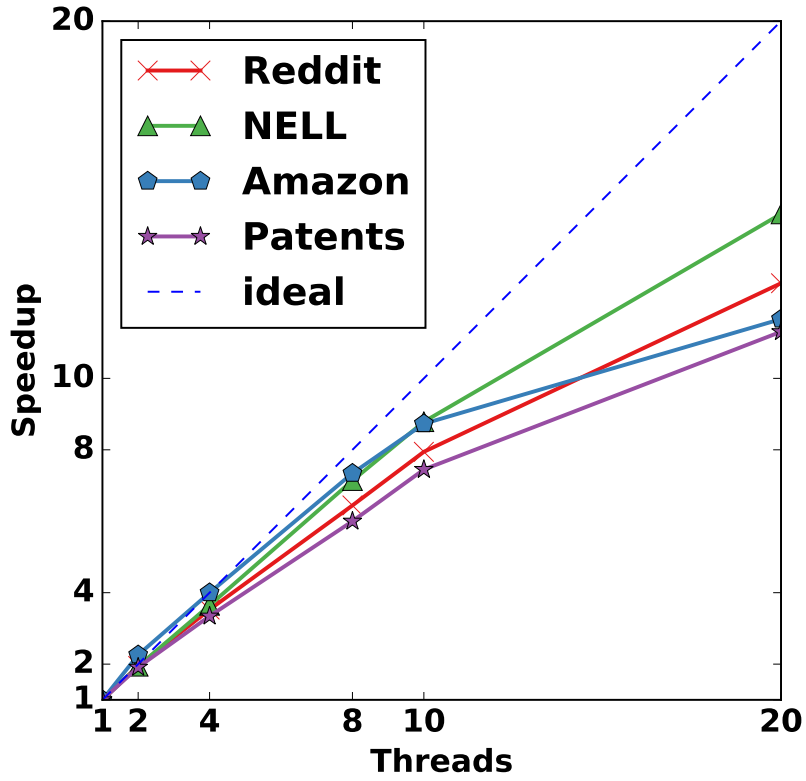


Parallel Scalability

Blocked ADMM improves speedup when the factorization is dominated by ADMM:



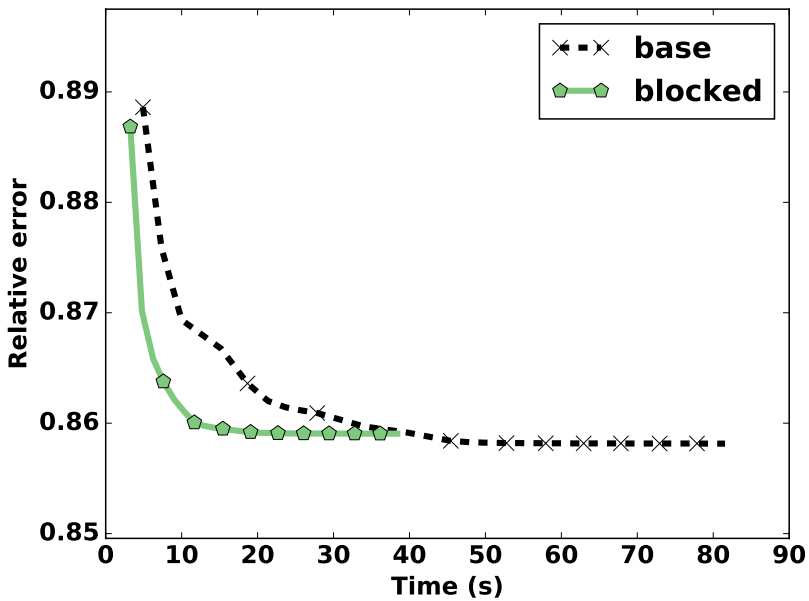
Baseline

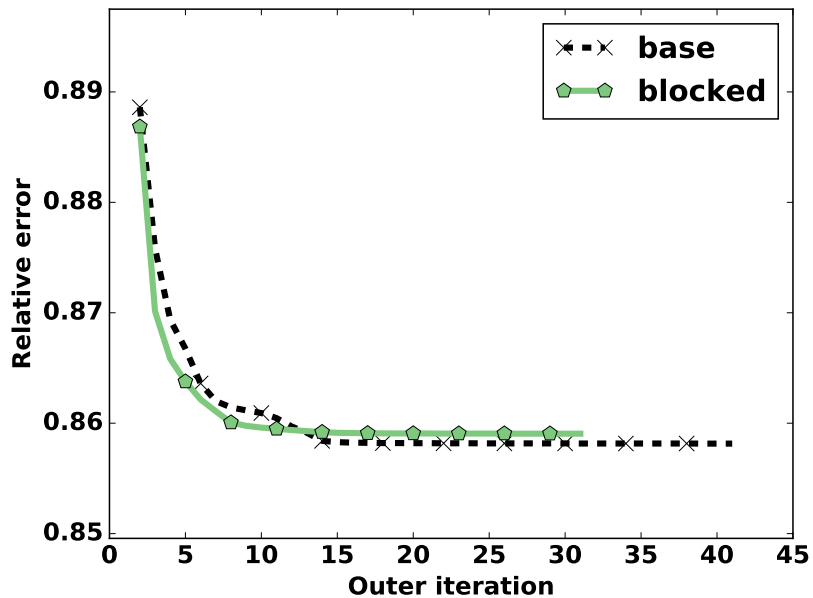


Blocked

Convergence: `Reddit`

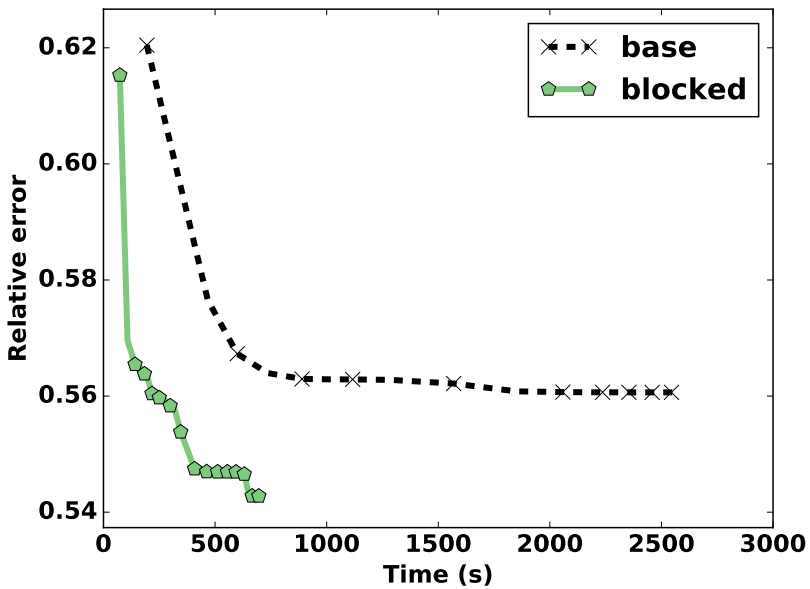
Blocking results in faster per-iteration runtimes and also converges in fewer iterations.

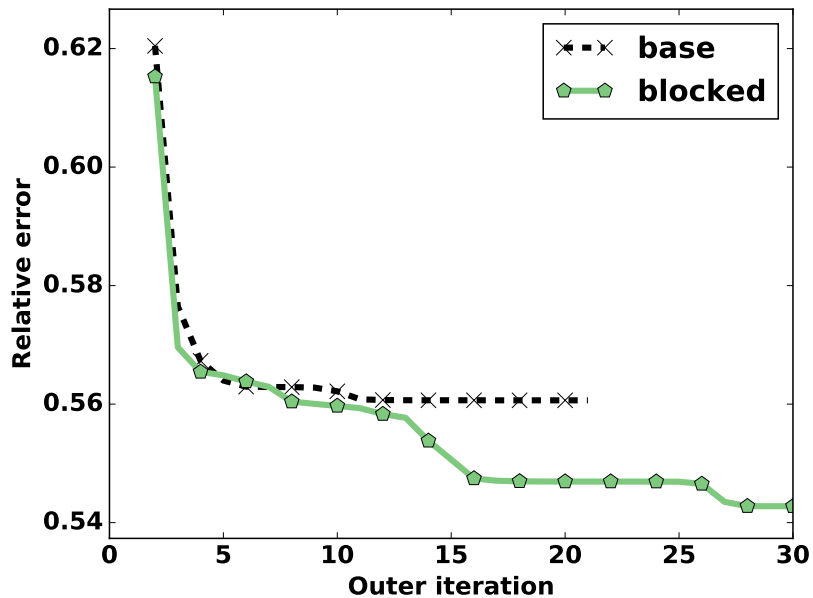




Convergence: NELL

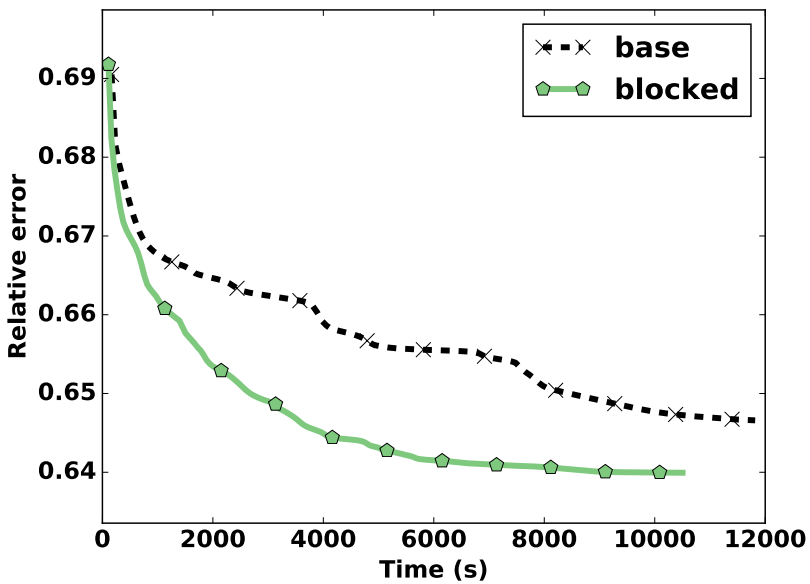
Convergence is 3.7× faster with blocking, despite using additional iterations to achieve a lower error.

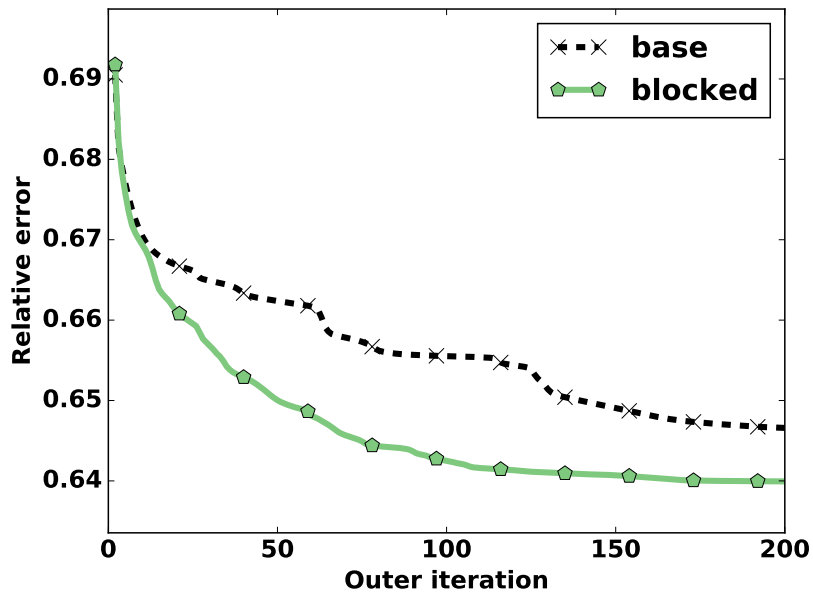




Convergence: Amazon

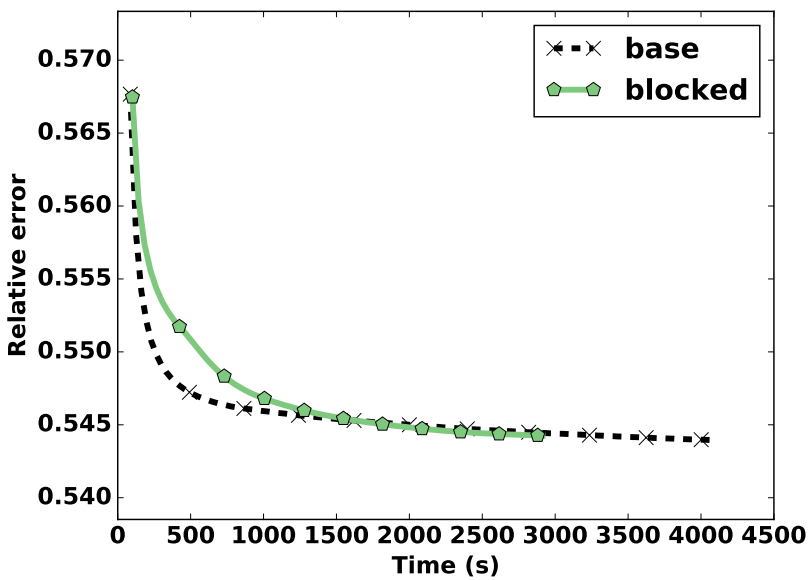
Both formulations exceed the maximum of 200 outer iterations, but the blocked formulation achieves a lower error in less time.





Convergence: Patents

Per-iteration runtimes are largely unaffected, as Patents is dominated by MTTKRP time. However, fewer iterations are required.



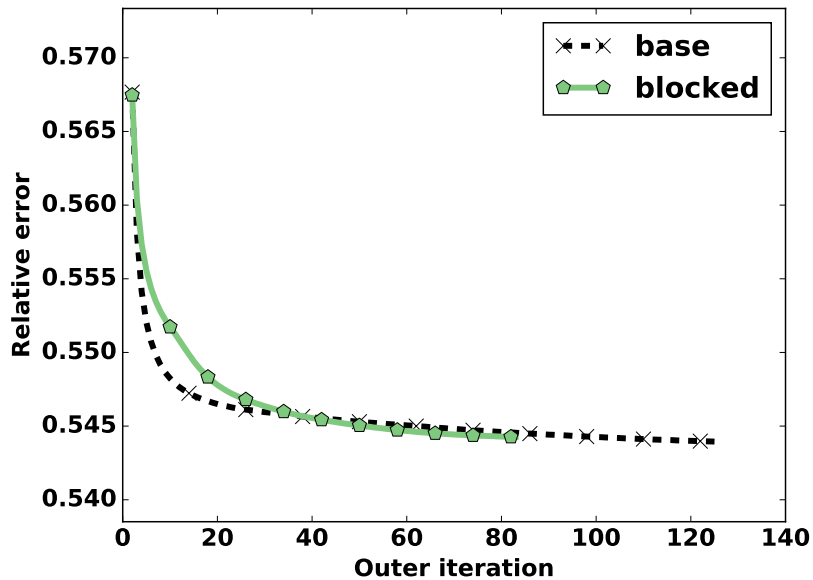


Table of Contents

Wrapping up

Blocked ADMM accelerates constrained tensor factorization in two ways:

- Optimizing blocks independently saves computation on the "simple" rows and better optimizes "hard" rows.
- Blocks can be kept in cache during ADMM, saving memory bandwidth.

Also in the paper:

- MTTKRP can be accelerated by exploiting the sparsity that dynamically evolves in the factors.
- An additional $\sim 2\times$ speedup is achieved.

Future work:

- Analytical model for selecting block sizes.
- Automatic runtime selection of data structure for sparse factors.

Reproducibility

All of our work is open source (in the `wip/ao-admm` branch for now):

`https://github.com/ShadenSmith/splatt`

Datasets are freely available:

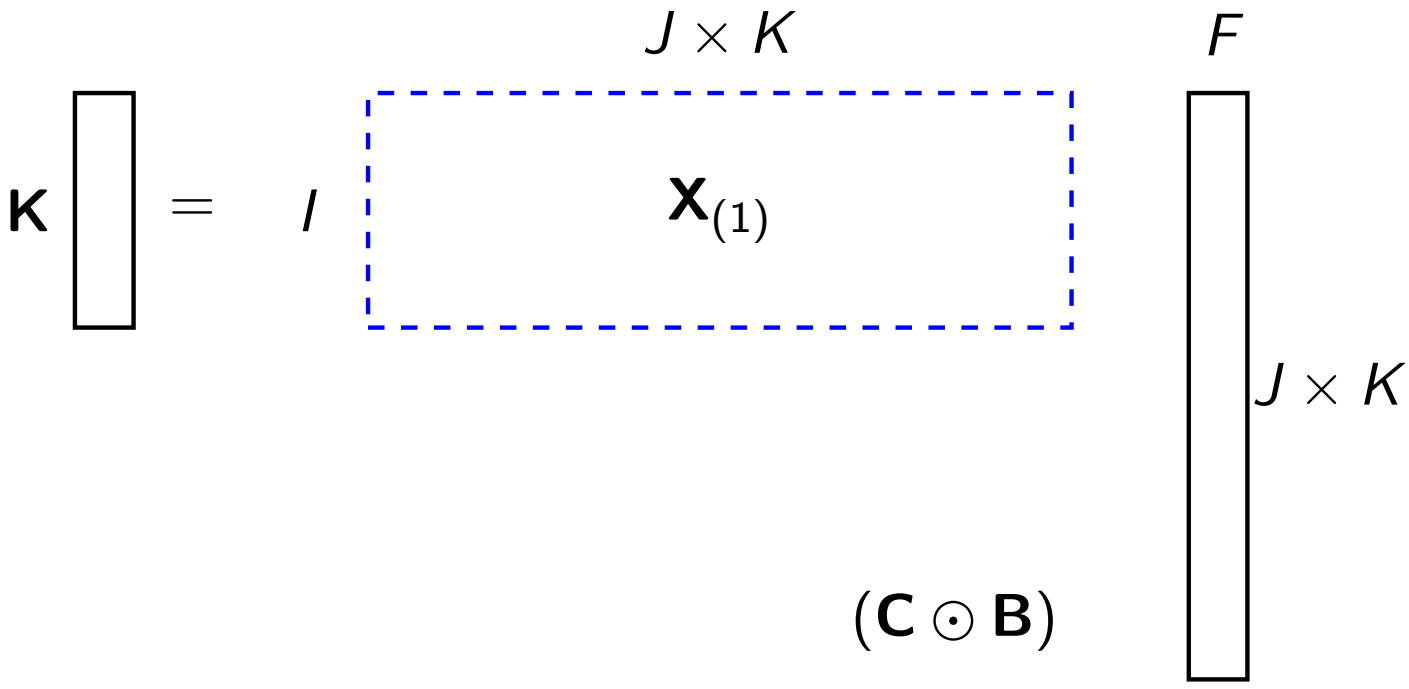
`http://frostt.io/`

Backup Slides

Matricized tensor times Khatri-Rao product

MTTKRP is a key kernel for computing the CPD:

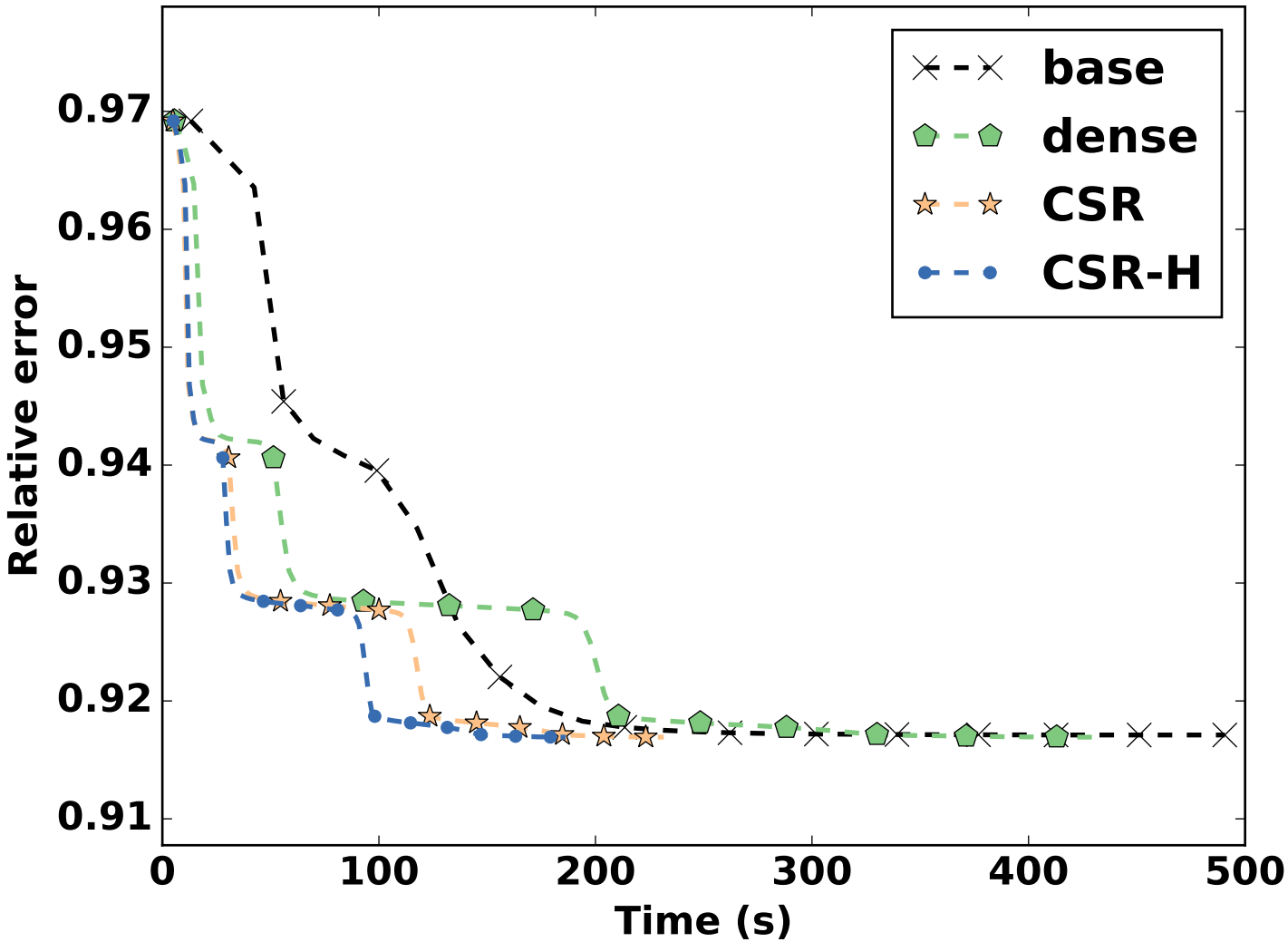
$$\mathbf{K} = \mathbf{X}_{(1)} (\mathbf{C} \odot \mathbf{B})$$



Notation
$\mathbf{X}_{(1)}$ *unfolds* a tensor.
$(\mathbf{C} \odot \mathbf{B})$ is the Khatri-Rao (columnwise Kronecker) product.

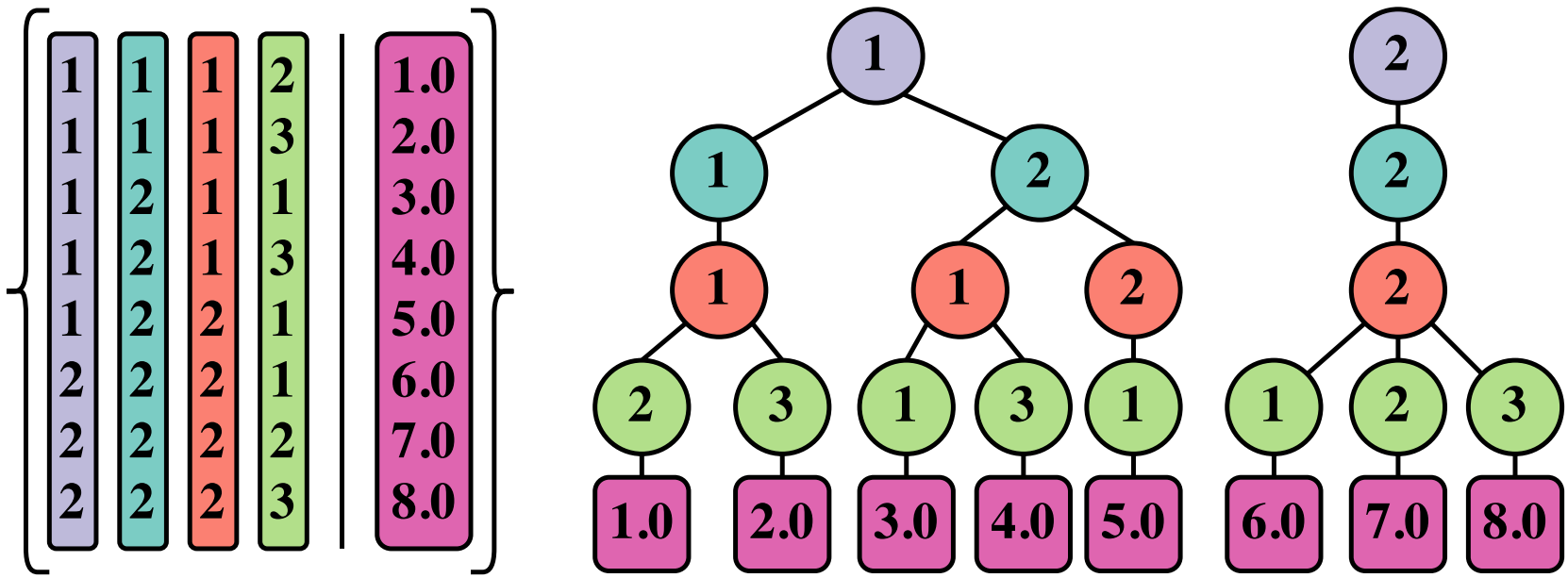
Sparse MTTKRP

Convergence on `Reddit` with $F = 100$ and $r(\cdot) = 10^{-1}||\cdot||_1$.



Compressed sparse fiber (CSF)

- Modes are recursively compressed.
- Paths from roots to leaves encode non-zeros.
- The tree structure encodes opportunities for savings.



MTTKRP with CSF

```
/* foreach outer slice */
for(int i=0; i < I; ++i) {
  /* foreach fiber in slice */
  for(int s = s_ptr[i]; s < s_ptr[i+1]; ++s) {
    accum[0:r] = 0;

    /* foreach  nnz in fiber */
    for(int nnz = f_ptr[s]; nnz < f_ptr[s+1]; ++nnz) {
      int k = f_ids[nnz];
      accum[0:r] += vals[nnz] * C[k][0:r];
    }

    int j = s_ids[s];
    A[i][0:r] += accum[0:r] * B[s][0:r];
  }
}
```