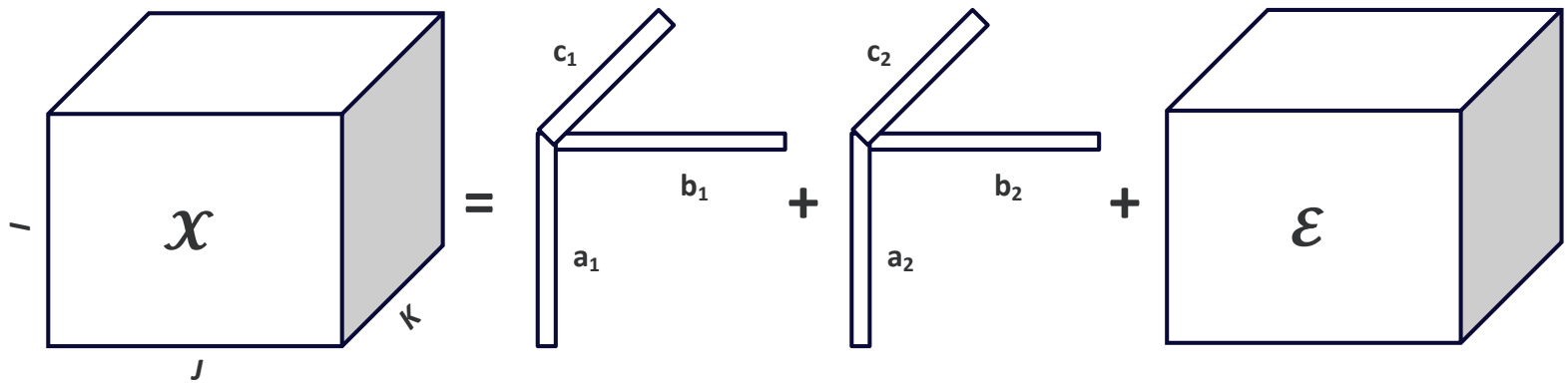
Adaptive Sketching for Fast and Convergent Canonical Polyadic Decomposition

Alex Gittens, Kareem S. Aggour, Bulent Yener

Rensselaer Polytechnic Institute, Troy, NY

Problem

$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$ is a *huge* tensor (multidimensional array). *Quickly* find an *accurate* low-rank approximation (LRA) to $\boldsymbol{\mathcal{X}}$.



Motivation/Applications

As a generalization of the SVD to higher-order relations in data:

- data mining and compression
- video/time-series analysis
- latent variable models (clustering, GMMs, HMMs, LDA, etc.)
- natural language processing (word embeddings)
- link prediction in hypergraphs
- ... many, *many* more

Canonical Polyadic Decomposition

For tensors, define the outer product of three vectors:

$$(\mathbf{a} \circ \mathbf{b} \circ \mathbf{c})_{\ell,m,p} = a_\ell b_m c_p.$$

Tensor LRA: Given a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, learn factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}, \mathbf{B} \in \mathbb{R}^{J \times R}, \mathbf{C} \in \mathbb{R}^{K \times R}$ that explain each of its modes, by minimizing the sum-of-squares error

$$\arg\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \left\| \boldsymbol{\mathcal{X}} - \sum_{i=1}^{R} \mathbf{a}_i \circ \mathbf{b}_i \circ \mathbf{c}_i \right\|_F^2 =$$

$$\arg\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \| \boldsymbol{\mathcal{X}} - [\![\mathbf{A}; \mathbf{B}; \mathbf{C}]\!] \|_F^2$$

Called a Canonical Polyadic decomposition (CPD) of rank R.

Tensor LRA is non-convex, and non-trivial. Even determining rank is NP-hard.

We relax our goal. No longer try to find globally *best* factors $\mathbf{A}, \mathbf{B}, \mathbf{C}$, but to find local optima of objective

$$F(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}; \mathbf{B}; \mathbf{C}]\!]\|_F .$$

All approaches are iterative.

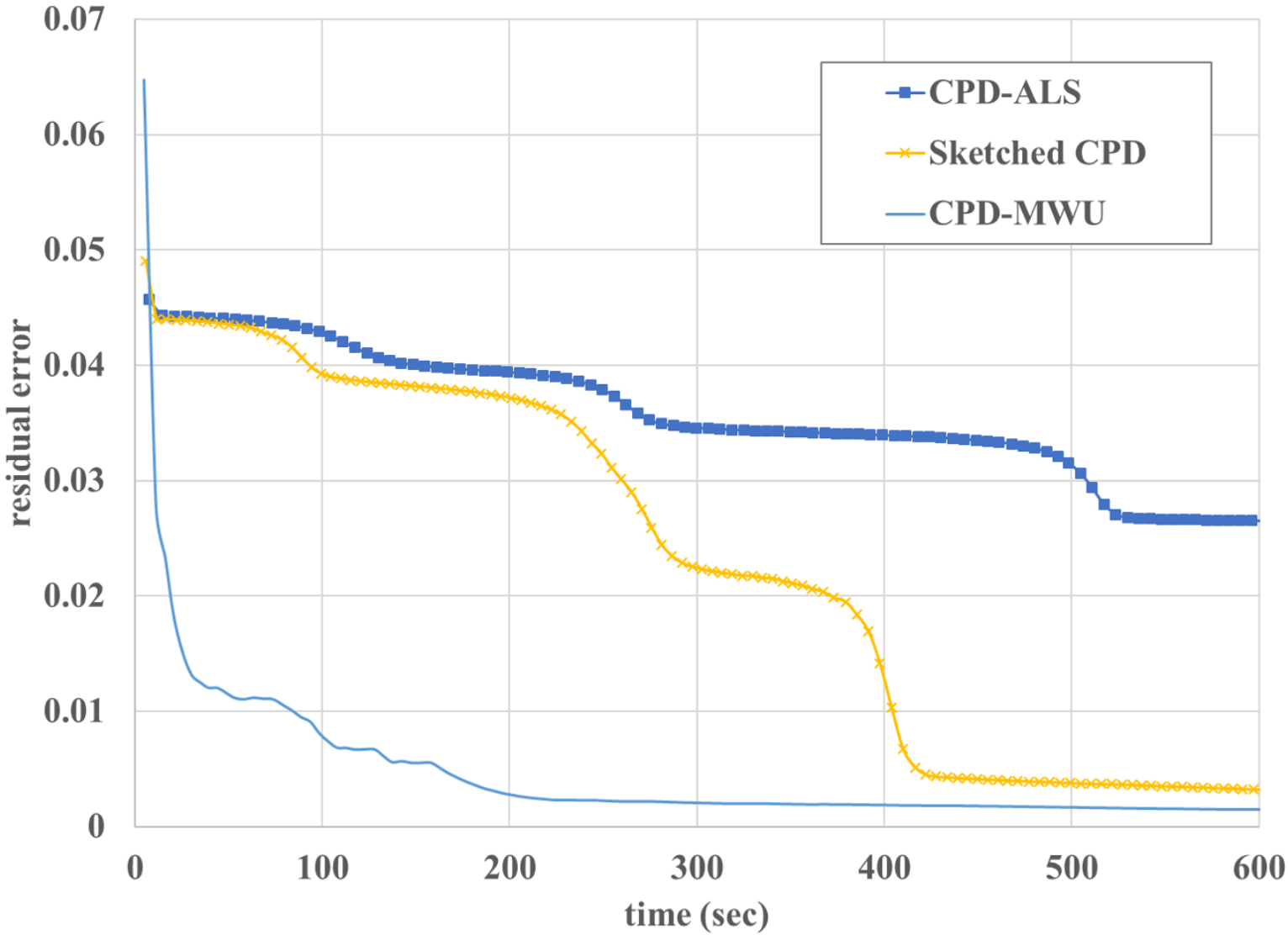
Our Contributions

We consider the use of sketching and regularization to obtain faster CPD approximations to tensors.

- We prove for the first time that sketched, regularized CPD approximation converges to an approximate critical point if the sketching rates are chosen appropriately at each step.
- We introduce a heuristic that selects the sketching rate adaptively and in practice has superior error-time tradeoffs to prior state-of-the-art sketched CPD heuristics. It greatly ameliorates the hyperparameter selection problem for sketched CPD.

Example error-time tradeoff

100GB rank 5 synthetic tensor with ill-conditioned factors.
CPD-MWU uses five rates: four from $[10^{-6}, 10^{-4}]$ and 1.
Sketched CPD uses hand-tuned rate.



Classic CPD-ALS

The classical iterative algorithm for finding CPDs is ALS, a Gauss-Siedel/block coordinate descent algorithm:

$$\mathbf{A}_{t+1} = \arg\min_{\mathbf{A}} \|\mathcal{X} - [\![\mathbf{A}; \mathbf{B}_t; \mathbf{C}_t]\!]\|_F^2$$

$$\mathbf{B}_{t+1} = \arg\min_{\mathbf{B}} \|\mathcal{X} - [\![\mathbf{A}_{t+1}; \mathbf{B}; \mathbf{C}_t]\!]\|_F^2$$

$$\mathbf{C}_{t+1} = \arg\min_{\mathbf{C}} \|\mathcal{X} - [\![\mathbf{A}_{t+1}; \mathbf{B}_{t+1}; \mathbf{C}]\!]\|_F^2$$

This constructs a sequence of LRAs whose approximation error is non-increasing. Under reasonable conditions these approximations converge to a critical point.

The sum-of-squares error is invariant to the shape of the tensor, so we solve these subproblems as matrix problems.

$$\mathbf{A}_{t+1} = \arg\min_{\mathbf{A}} \left\| \mathbf{X}_{(1)} - \mathbf{A}(\mathbf{B}_t \odot \mathbf{C}_t)^T \right\|_F^2$$

$$\mathbf{B}_{t+1} = \arg\min_{\mathbf{B}} \left\| \mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C}_t \odot \mathbf{A}_{t+1})^T \right\|_F^2$$

$$\mathbf{C}_{t+1} = \arg\min_{\mathbf{C}} \left\| \mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B}_{t+1} \odot \mathbf{A}_{t+1})^T \right\|_F^2$$

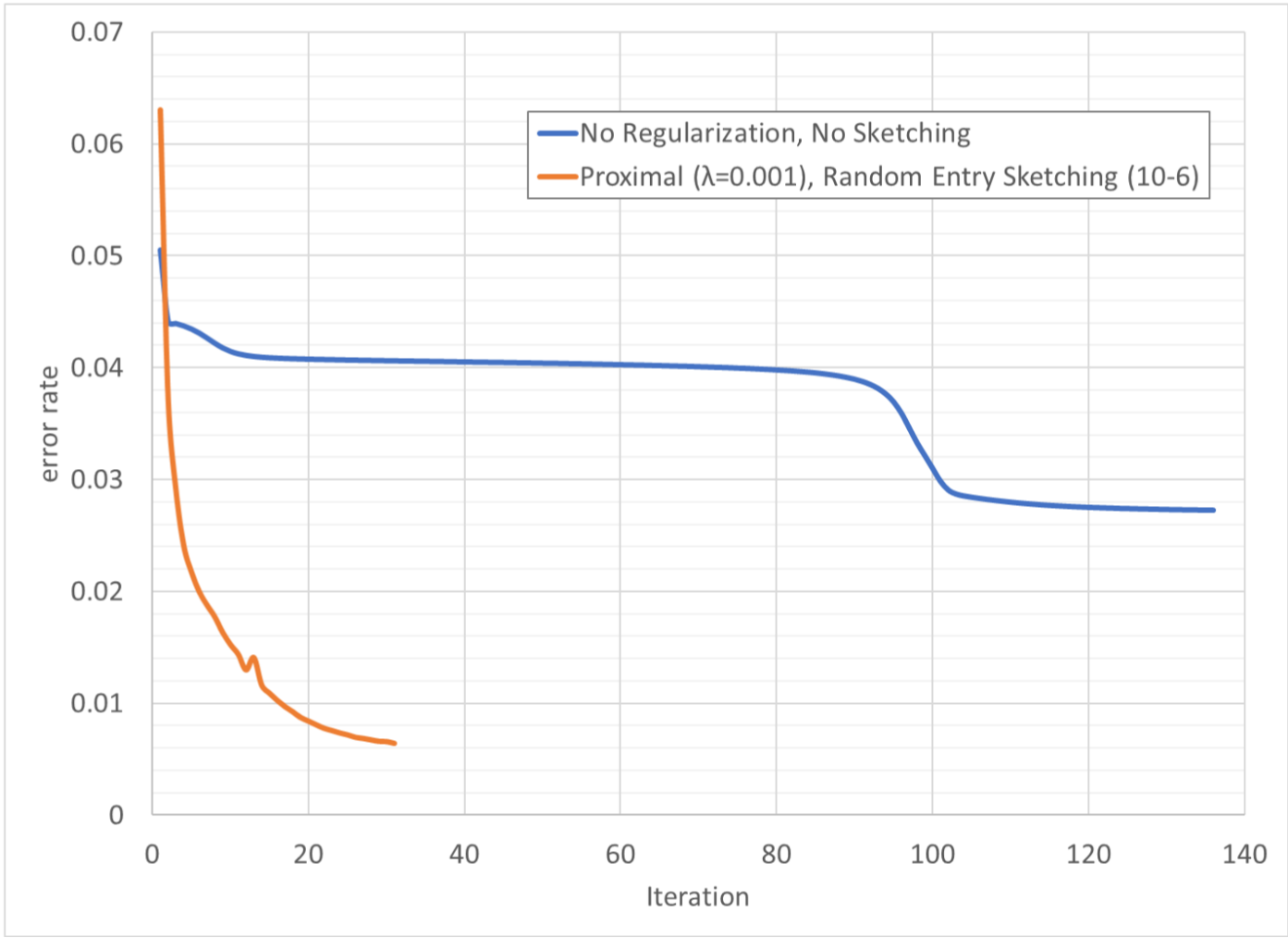
Classic CPD-ALS consists of a series of matrix least-squares problems.

Drawbacks of classical CPD-ALS: these are *huge*, potentially *ill-conditioned* least-squares problems.

- Expensive Iterations: each round of ALS takes $O((JK + IK + IJ)R^2 + JKI)$ time
- Many Iterations: The number of rounds to convergence depends on the conditioning of the linear-systems.

Two (separate, until our work) remedies:

- Add regularization to improve the conditioning of the linear solves (scientific computing community)
- Use *sketching* to reduce the size of the linear systems (theoretical computer science community)



Proximal regularization requires that the factor matrices stay close to their previous values.

$$\mathbf{A}_{t+1} = \arg\min_{\mathbf{A}} \left\| \mathbf{X}_{(1)} - \mathbf{A}(\mathbf{B}_t \odot \mathbf{C}_t)^T \right\|_F^2 + \lambda \|\mathbf{A} - \mathbf{A}_t\|_F^2$$

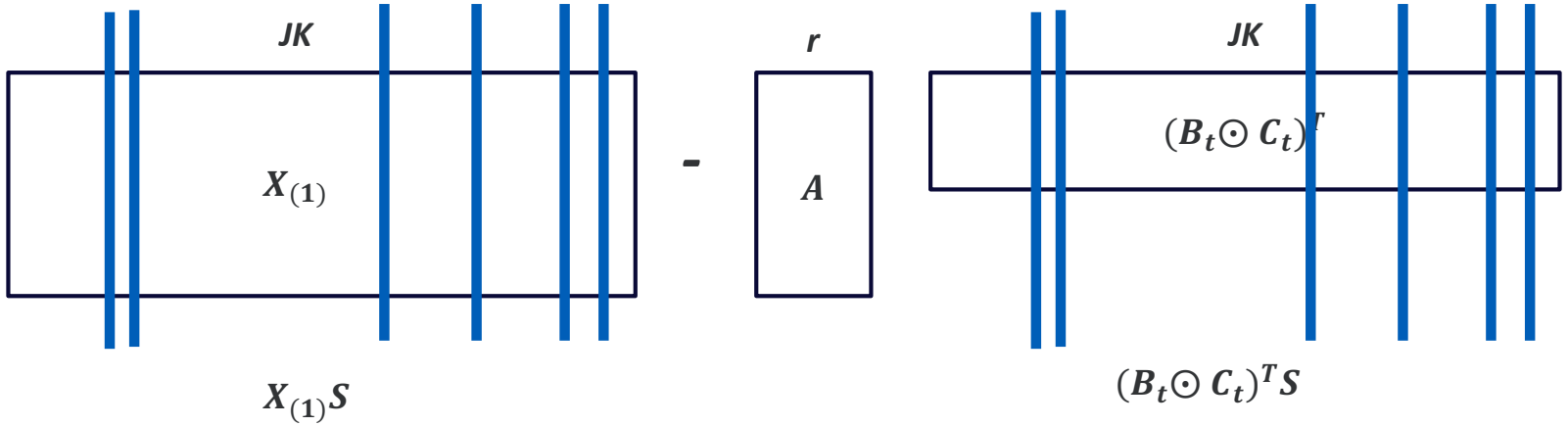
$$\mathbf{B}_{t+1} = \arg\min_{\mathbf{B}} \left\| \mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C}_t \odot \mathbf{A}_{t+1})^T \right\|_F^2 + \lambda \|\mathbf{B} - \mathbf{B}_t\|_F^2$$

$$\mathbf{C}_{t+1} = \arg\min_{\mathbf{C}} \left\| \mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B}_{t+1} \odot \mathbf{A}_{t+1})^T \right\|_F^2 + \lambda \|\mathbf{C} - \mathbf{C}_t\|_F^2$$

This Regularized ALS (RALS) algorithm is known to have the same critical points as the original CPD-ALS formulation, in the deterministic case, and to help avoid swamping.

Sketching for CPD

Natural to think of sketching: sample the constraints to reduce the size of the problem. Runtime will decrease, but accuracy should not be too affected.



Prior work has considered sketched CPD-ALS heuristics:

1. From the scientific computing community: Battaglino, Ballard, Kolda. A Practical Randomized CP Tensor Decomposition. SIMAX 2018
2. From the TCS/ML community: Cheng, Peng, Liu, Perros. SPALS: Fast Alternating Least Squares via Implicit Leverage Scores Sampling. NIPS 2016.

Prior sketched CPD-ALS heuristics:

1. Provide guarantees on each individual least squares problem, e.g.

$$\|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}_{t+1}; \mathbf{B}_t; \mathbf{C}_t]\!]\|_F^2 \leq (1+\varepsilon)\|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}_{t+1}^*; \mathbf{B}_t; \mathbf{C}_t]\!]\|_F^2,$$

   so *potentially the error can increase at each iteration*.

2. Use fixed sketching rates. Hyperparameter selection is a problem.
3. Remain vulnerable to 'swamping' caused by ill-conditioned linear systems.

It is important to have guarantees on the behavior of these algorithms:

- CPD is a non-convex problem, so it's possible for intuitively reasonable heuristics to fail
- **HYPERPARAMETER SELECTION IS IMPORTANT AND EXPENSIVE**: how should we choose the sketching rates? Why should there be a good fixed sketching rate?
- Stopping criteria implicitly assume convergence, otherwise they do not make sense

Questions:

- how to ensure monotonic decrease of approximation error?
- how to ensure convergence to a critical point?
- how to choose sketching rates and regularization parameter?

Theoretical Contribution

We look at proximally regularized sketched least squares algorithms and argue that:

- Each sketched least squares solve *decreases* the objective almost as much as a full least squares solve (must assume sketching rates are high enough)
- This decrease can be related to the size of the gradient of the CPD objective
- Proximal regularization ensures that the gradient is bounded away from zero
- Thus progress is made at each step, obtaining a sublinear rate of convergence to an approximate critical point

Guaranteed Decrease

Fix a failure probability $\delta \in (0,1)$ and a precision $\varepsilon \in (0,1)$. Let $\mathbf{S}$ be a random sketching matrix that samples at least $\ell = O\left(\frac{1}{\nu\varepsilon^2\delta} R \log(\frac{R}{\delta})\right)$ columns. Update

$$\mathbf{A}_{t+1} = \arg\min_{\mathbf{A}} \|(\mathbf{X}_{(1)} - \mathbf{A}\mathbf{M})\mathbf{S}\|_F^2 + \lambda_{t+1}\|\mathbf{A} - \mathbf{A}_t\|_F^2,$$

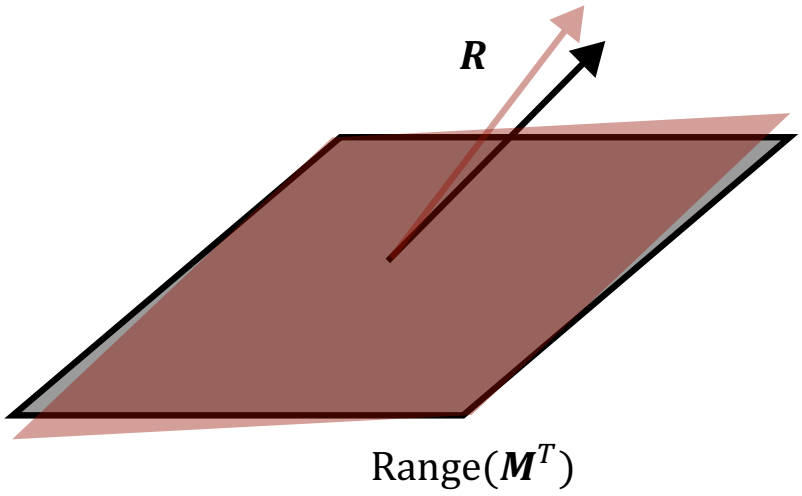
with $\lambda_{t+1} = o(\sigma_{\min}^2(\mathbf{M}))$. The sum-of-squares error F of $\mathbf{A}_{t+1}$ satisfies

$$F(\mathbf{A}_{t+1}, \mathbf{B}_t, \mathbf{C}_t) \leq F(\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t) - (1 - \varepsilon_{t+1})\|\mathbf{R}\mathbf{P}_{\mathbf{M}^T}\|_F^2,$$

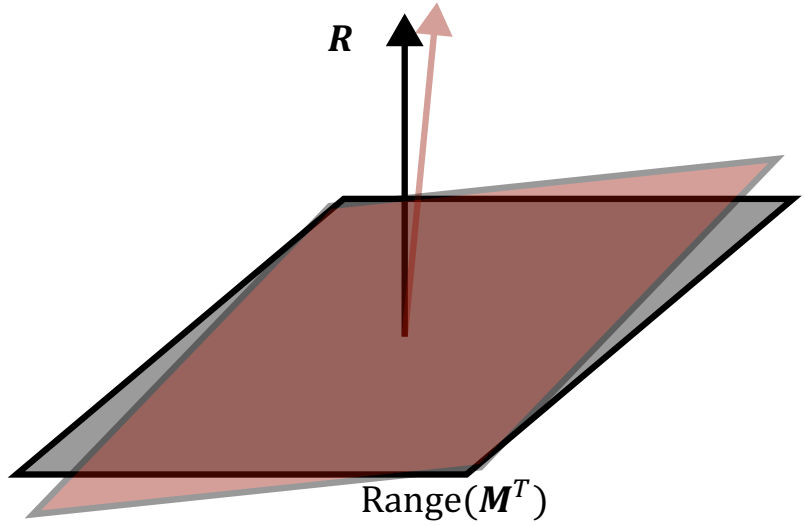
with probabilty at least $1 - \delta$.

Consequence for sketching rate

ν is related to an 'angle' between **R** and **M**.



Initially **R** and **M** have a small angle, so even aggressive sketching preserves the angle.



Near convergence **R** and **M** have a large angle, so preserving the angle requires more expensive sketching.

We do not expect convergence if a static sketching rate is used throughout!

Adaptation of standard results now leads to a convergence guarantee.

Sublinear convergence

If the sketching rates are selected to ensure sufficient decrease at each iteration with probability at least $(1-\delta)$, and the precisions ε_{t+1} are bounded away from one, then regularized sketched CPD-ALS visits a $O(T^{-1/2})$-approximate critical point in T iterations with probability at least $(1-\delta)^T$:

$$\min_{1\leq i\leq T} \|\nabla F(\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i)\|_F = O\left(\sqrt{\frac{F(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)}{T}}\right).$$

Important takeways:

- Running the algorithm for more time continues to increase the accuracy of the solution
- Gradient-based termination conditions can be used, because eventually the gradient will be small.

Note that prior sketched CPD-ALS algorithms did not come with these guarantees (indeed, more time does not continue to increase accuracy for them, empirically)

But ... *in practice, how to choose the sketching rate?* We can't realistically compute ν.

A new heuristic: online sketching rate selection

Key observation: low-rank approximation is an *iterative* process.

1. As in SGD, when closer to convergence, more constraints need to be sampled to ensure progress.
2. The performance of a given sketching rate historically is predictive of future performance.

This suggests an online approach to learning the performance of the sketching rates.

Adaptive sketching rate selection: choose the best of N sketching rates, to maximize reductions in the error, while minimizing runtime.

We employ label efficient multiplicative weights update. Given sketching rates $\{s_1, \ldots, s_N\}$,

1. Quality of sketching rate i at iteration t is

$$\ell_{i,t} = \frac{\|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}_{t+1}; \mathbf{B}_{t+1}; \mathbf{C}_{t+1}]\!]\|_F - \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}_t; \mathbf{B}_t; \mathbf{C}_t]\!]\|_F}{\text{runtime}(i)\|\boldsymbol{\mathcal{X}}\|_F}$$

   where the factor updates are computed using sketching rate s_i and take time runtime(i).

2. At $t = 0$, $w_{i,0} = 1$ for $i = 1, \ldots, N$. At each subsequent iteration, with probability ε, update all the weights

$$w_{i,t+1} = w_{i,t} \exp\left(-\eta \frac{\ell_{i,t}}{\varepsilon}\right).$$

3. At each iteration use a sketching rate selected with probability proportional to $w_{i,t}$ to compute $\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_{t+1}$ using column sampling.

Notes:

1. $\epsilon > 0$ determines update frequency
2. $\eta > 0$ determines aggressiveness of weight updates
3. Computing $\ell_{i,t}$ for all N arms requires N CPD solves, so take $\varepsilon \approx \frac{1}{N}$ in practice
4. Take one arm to be fully constrained CPD-ALS to ensure that convergence is possible

Park bench video decomposition

Rank 250 decomposition of a 5GB tensor. Five rates for CPD-MWU: four in $[10^{-6}, 10^{-4}]$ and 1.

	ϵ	Std(ϵ)	Time	Std(Time)
CPD-ALS	0.5153	1.74*10^{-3}	618.84	9.69
Sketched CPD	0.5148	1.54*10^{-3}	564.53	22.20
CPD-MWU	**0.5069**	6.57*10^{-3}	**444.58**	70.79

Knowledge Base Mining (NELL dataset)

Rank 30 approximation of 302MB database. Target stopping error set by running exact ALS for 30 minutes. All algorithms allowed to run for up to 30 minutes.

	ϵ	Std(ϵ)	Time	Std(Time)
SPALS	0.104	0.0061	1829.36	14.84
CPRAND	0.072	0.0046	1806.70	3.50
CPD-ALS	0.060	0.0002	1044.75	386.03
CPD-MWU	0.058	0.0015	**354.55**	224.59

Even accounting for standard deviations, CPD-MWU is around 2x faster and as accurate as CPD-ALS.

Knowledge Base Mining (NELL dataset)

Rank 30 approximation of 302MB database. Target stopping error set by running exact ALS for 30 minutes. All algorithms allowed to run for up to 2 hours.

	ϵ	Std(ϵ)	Time	Std(Time)
SPALS	0.098	0.0045	7224.28	18.50
CPRAND-MIX	0.066	0.0039	7205.37	4.03
CPD-ALS + MIX	0.060	0.0002	1007.48	372.58
CPD-MWU + MIX	0.058	0.0015	**337.16**	204.28

CPD-MWU finishes in same amount of time. The other sketched CPD-ALS algorithms still do not coverge in over 2 hours.

Takeaway: you need to select the sketching rate appropriately at each iteration. Static sketching is inappropriate.

To recap:

- Established convergence of regularized sketched CPD-ALS algorithms when sketching rate is appropriately chosen
- Introduced CPD-MWU, a heuristic for choosing the sketching rates
- Demonstrated empirically the superior performance of CPD-MWU to prior sketched CPD-ALS algorithms

Future directions:

- Is the convergence rate truly only sublinear?
- Jointly choose regularization and sketching rates to accelerate convergence?
- Remove the requirement that finite sketching rates be selected for CPD-MWU
- Apply adaptive sketching to constrained tensor factorizations

Thank you!

Here's an example of matricizations in the different modes:

$$\mathcal{X} = \left[\begin{array}{ccc|ccc} 1 & 5 & 9 & 13 & 17 & 21 \\ 2 & 6 & 10 & 14 & 18 & 22 \\ 3 & 7 & 11 & 15 & 19 & 23 \\ 4 & 8 & 12 & 16 & 20 & 24 \end{array}\right]$$

$$\boldsymbol{X}_{(1)} = \left[\begin{array}{ccc|ccc} 1 & 5 & 9 & 13 & 17 & 21 \\ 2 & 6 & 10 & 14 & 18 & 22 \\ 3 & 7 & 11 & 15 & 19 & 23 \\ 4 & 8 & 12 & 16 & 20 & 24 \end{array}\right]$$

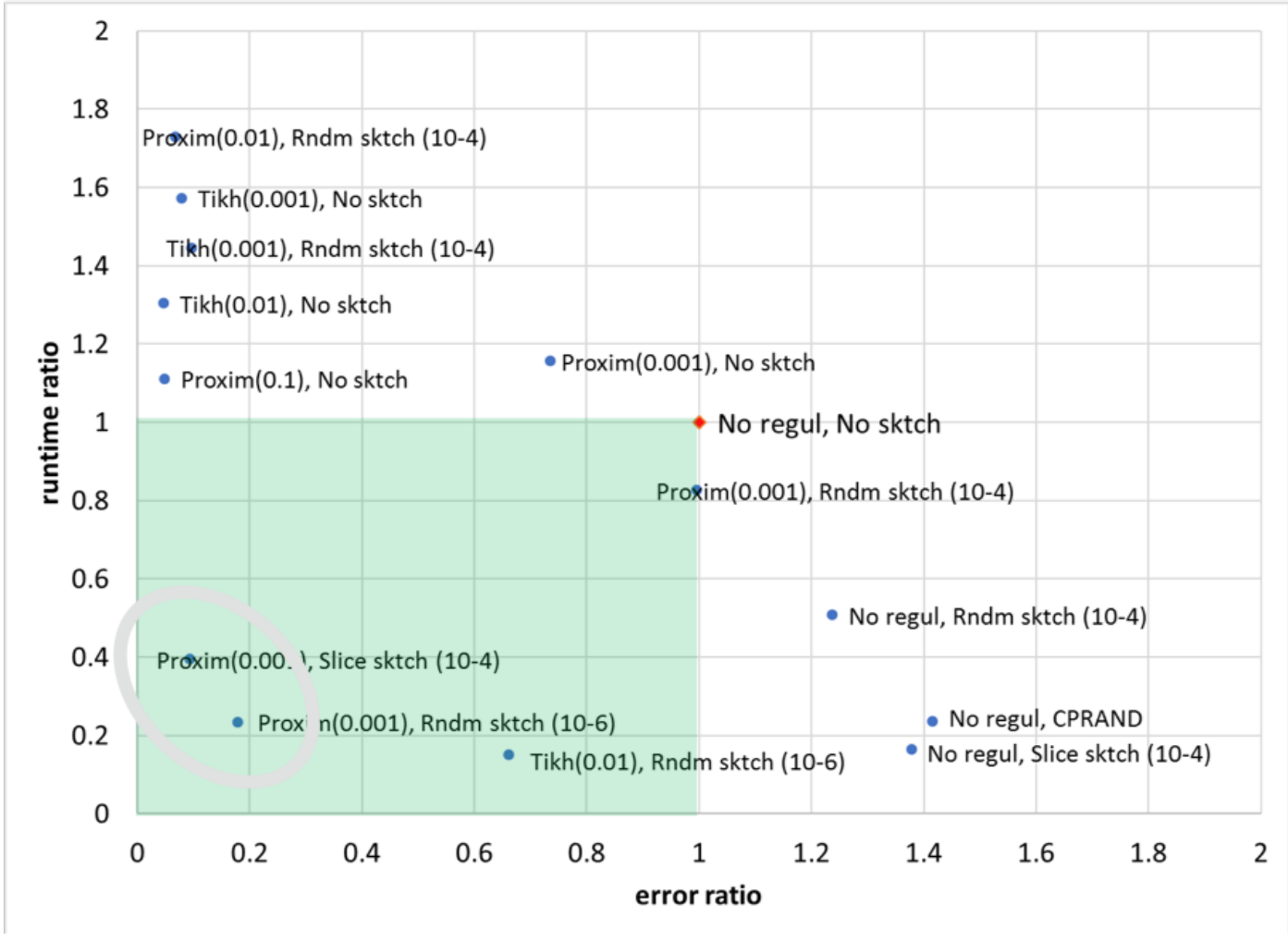
$$\boldsymbol{X}_{(2)} = \left[\begin{array}{cccc|cccc} 1 & 2 & 3 & 4 & 13 & 14 & 15 & 16 \\ 5 & 6 & 7 & 8 & 17 & 18 & 19 & 20 \\ 9 & 10 & 11 & 12 & 21 & 22 & 23 & 24 \end{array}\right]$$

$$\boldsymbol{X}_{(3)} = \left[\begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \end{array}\right]$$

$\mathbf{X} \odot \mathbf{Y}$ denotes the column-wise Khatri-Rao product: if $\mathbf{X} \in \mathbb{R}^{I \times R}$ and $\mathbf{Y} \in \mathbb{R}^{J \times R}$, then $\mathbf{X} \odot \mathbf{Y} \in \mathbb{R}^{IJ \times R}$. An example:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \Rightarrow \mathbf{X} \odot \mathbf{Y} = \begin{bmatrix} a & 2b \\ c & 2d \\ e & 2f \\ 3a & 4b \\ 3c & 4d \\ 3e & 4f \\ 5a & 6b \\ 5c & 6d \\ 5e & 6f \end{bmatrix}$$

We found experimentally that proximal regularization and sketching *synergize*:



The first step is showing that the error decreases by a fraction of the maximum possible decrease at each iteration.

Consider the update for factor matrix $\mathbf{A}$. Let $\mathbf{M} = (\mathbf{B}_t \odot \mathbf{C}_t)^T$. The update

$$\mathbf{A}_{t+1} = \arg\min_{\mathbf{A}} \|(\mathbf{X}_{(1)} - \mathbf{A}\mathbf{M})\mathbf{S}\|_F^2 + \lambda_{t+1}\|\mathbf{A} - \mathbf{A}_t\|_F^2,$$

can be rewritten in terms of $\mathbf{\Delta} = \mathbf{A}_{t+1} - \mathbf{A}_t$ as

$$\mathbf{A}_{t+1} = \mathbf{A}_t + \arg\min_{\mathbf{\Delta}} \|(\mathbf{R} - \mathbf{\Delta}\mathbf{M})\mathbf{S}\|_F^2 + \lambda_{t+1}\|\mathbf{\Delta}\|_F^2,$$

where $\mathbf{R} = \mathbf{X}_{(1)} - \mathbf{A}_t\mathbf{M}$ is the residual from the previous $\mathbf{A}$ factor.

The only portion of the residual that can be captured is $\mathbf{R}\mathbf{P}_{\mathbf{M}^T}$, the projection of the residual onto the row span of $\mathbf{M}$.

Thus the level of the sketching needed depends on how much of the residual can be captured by the optimal $\mathbf{A}_{t+1}$:

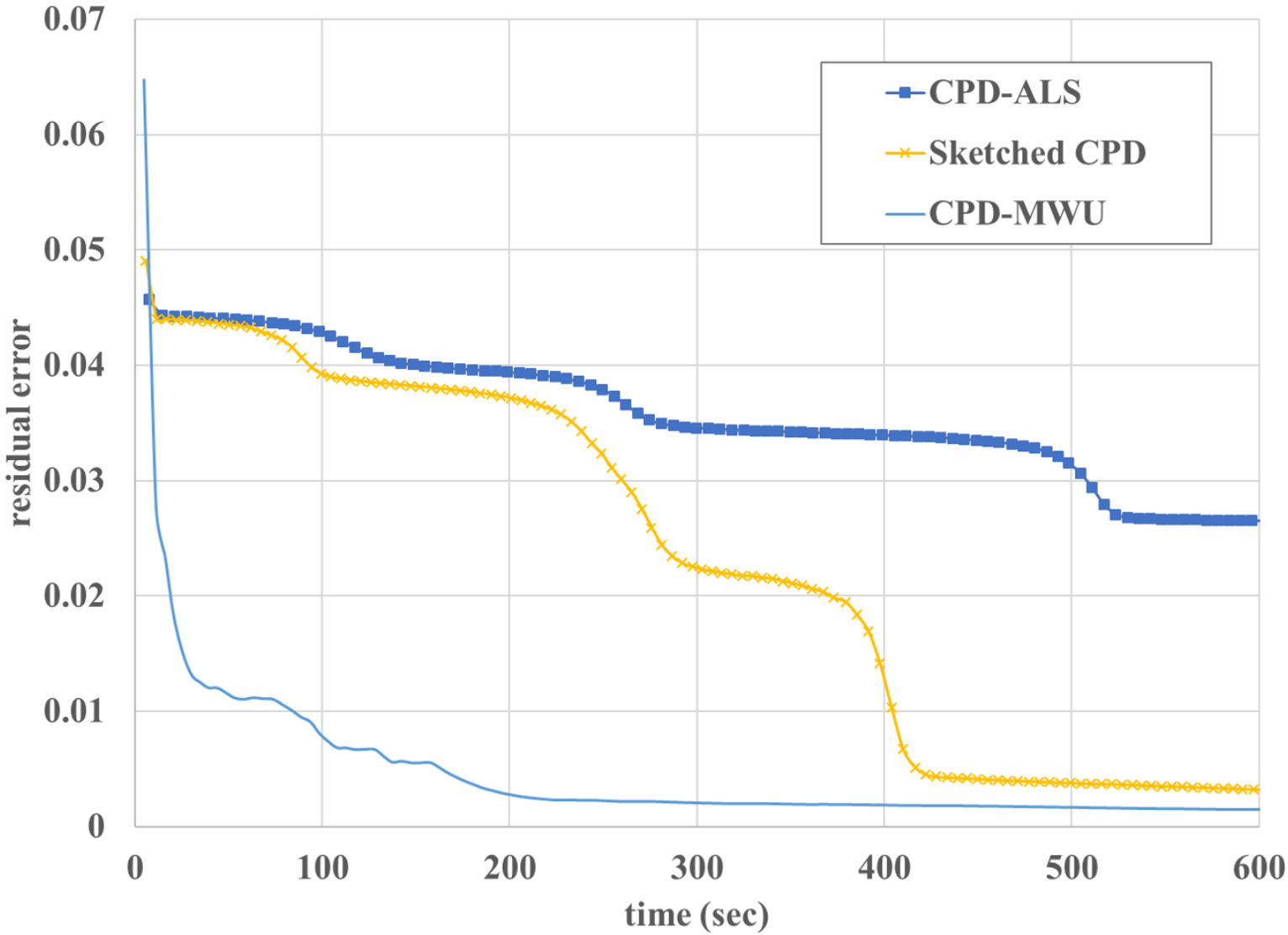
- $\|\mathbf{R}\mathbf{P}_{\mathbf{M}^T}\|_F^2$ is exactly the maximum decrease possible: this happens when $\mathbf{A}_{t+1}$ is chosen optimally.
- $\nu = \|\mathbf{R}\mathbf{P}_{\mathbf{M}^T}\|_F^2 / \|\mathbf{R}\|_F^2$ quantifies how much of the residual can be captured by the optimal $\mathbf{A}_{t+1}$. This quantity is small when the residual is orthogonal to $\mathbf{M}$.
- When $\nu \approx 1$, you can sketch aggressively, otherwise you need to sketch more conservatively.

Practical questions

- How does CPD-MWU perform relative to classical CPD-ALS and prior sketched CPD-ALS algorithms (CPRAND from the scientific computing community and SPALS from the TCS/ML community) in terms of runtime and accuracy?
- Does CPD-MWU ameliorate the hyperparameter selection problem for the sketching rate?
- Does CPD-MWU allow for convergence?

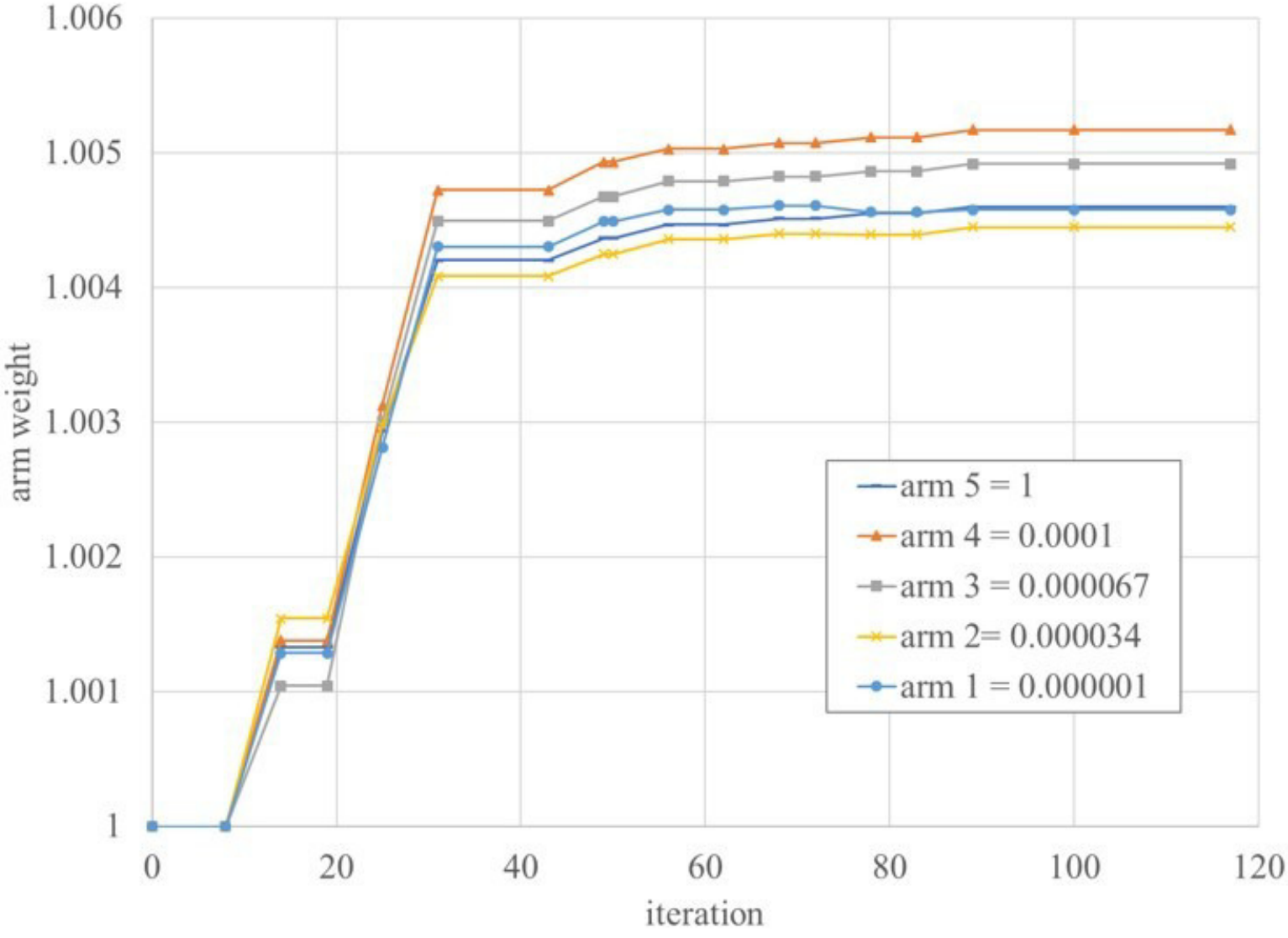
Error-time tradeoff

100GB rank 5 synthetic tensor with ill-conditioned factors.
CPD-MWU uses five rates: four from $[10^{-6}, 10^{-4}]$ and 1.
Sketched CPD uses hand-tuned rate.



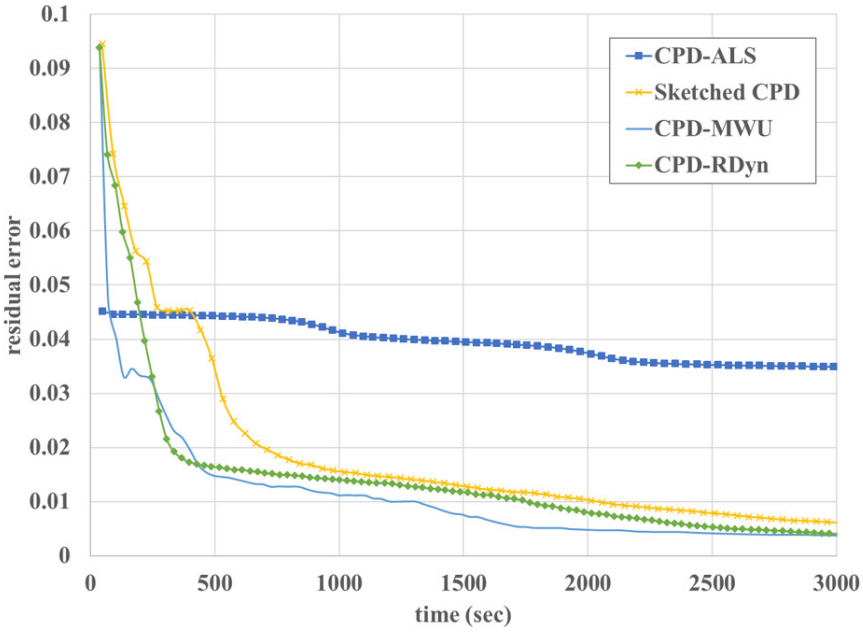
Evolution of sketching rates' weights over time

Same setup: starts off with aggressive sketching, becomes more conservative.

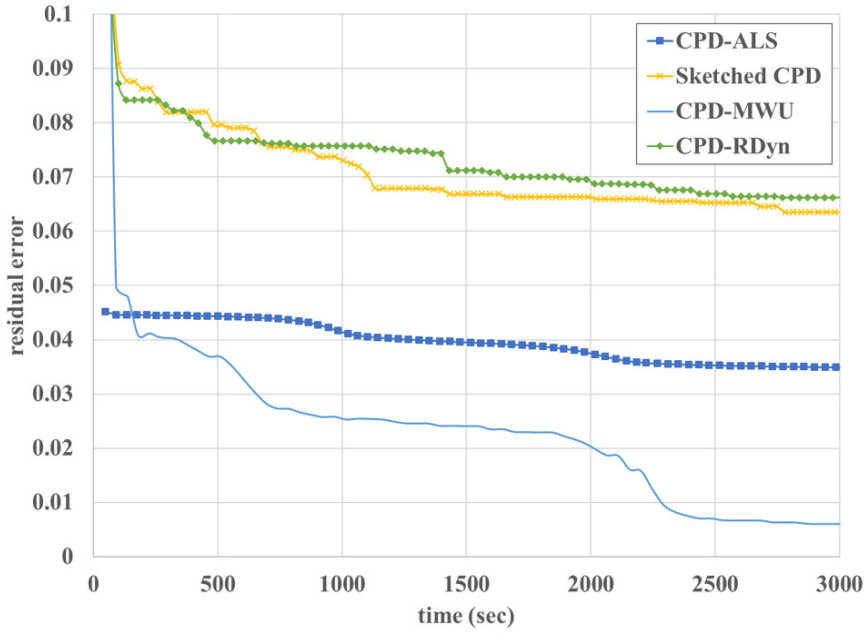


Impact of the sketching rate range

Decomposing a 1TB ill-conditioned rank 5 synthetic tensor.



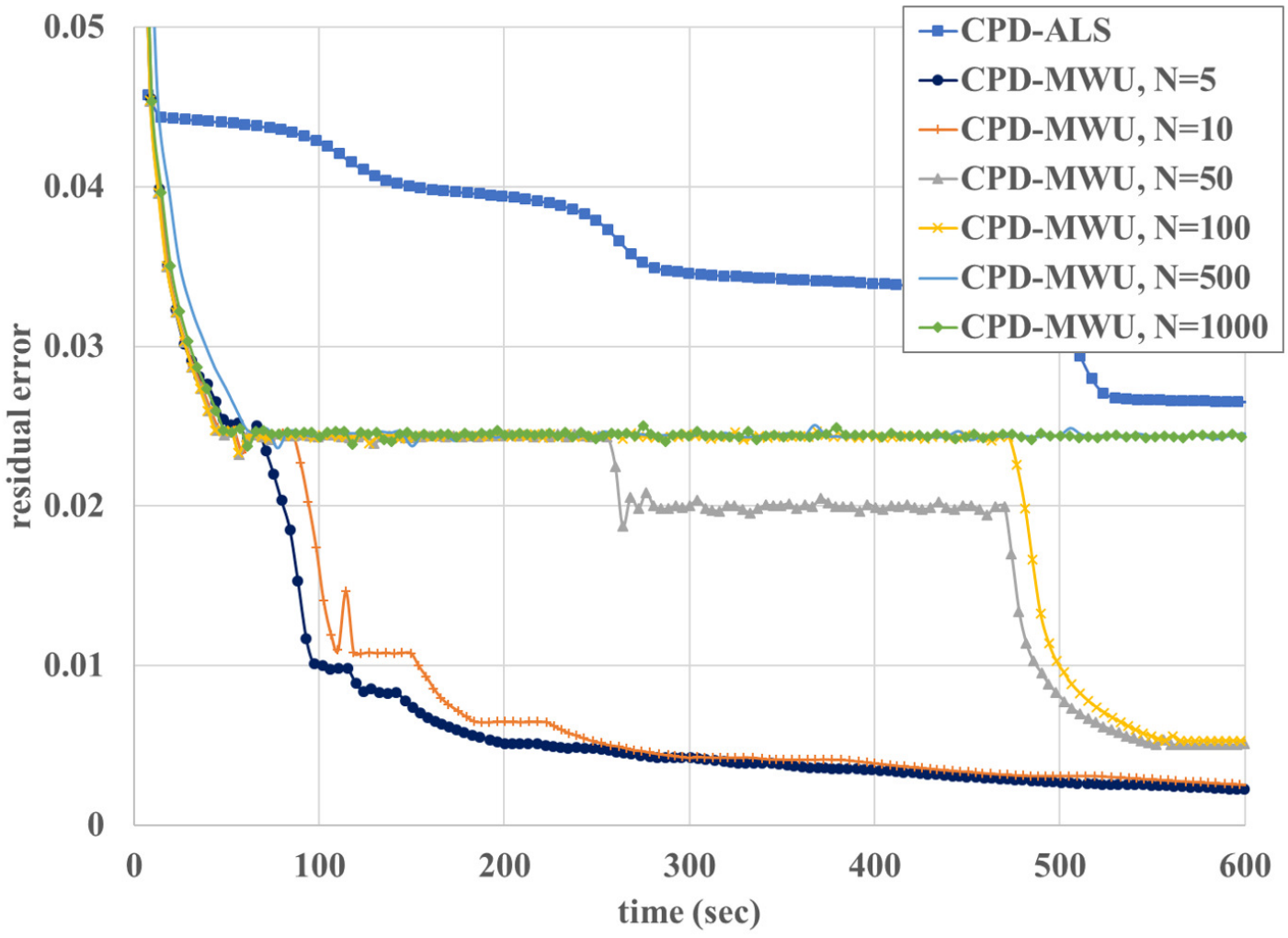
four rates in $[10^{-6}, 10^{-4}]$ and 1



four rates in $[10^{-9}, 10^{-6}]$ and 1

CPD-RDyn randomly selects rates from the five choices. Sketched CPD uses the best sketching rate (not equal to 1) from the five choices.

Impact of number of sketching rates



Residual error when decomposing an ill-conditioned tensor with increasing numbers of sketching rates,
$N \in \{5, 10, 50, 100, 500, 1000\}$.