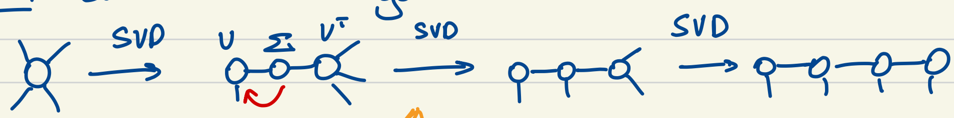
[Speaker Note]: Will use TT in place of MPS sometimes.

HSDC : Constructing Tensor Trains / MPS

- **Given**: Tensor A in D dimensions, general fmt.
  - Giant table (dense)
  - Sparse (mostly zero)
  - Black-box, on-demand access

Source Paper: TT-Cross Approximation for multidimensional Arrays, Oseledets & Tyrtyshnikov.

- **Produce**: MPS approximation T w/ cores $(T_1, T_2, \ldots T_D)$
- **Recall**: Successive SVD algorithm:

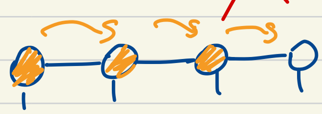


Replace w/ R-SVD [if ranks low], but still "see" whole tensor at step 1.

- No sub-exp. runtime algs w/ guaranteed decomposition!

Heuristics:

1) Start w/ random MPS



2) for $i = 1 \ldots D$:
   optimize core i
   for $i = D \ldots 1$:
   optimize core i

optimize objective 1 core at a time.

3) Repeat step 2 until convergence.

Two perspectives

- Volume Maximization
- L2 error minimization

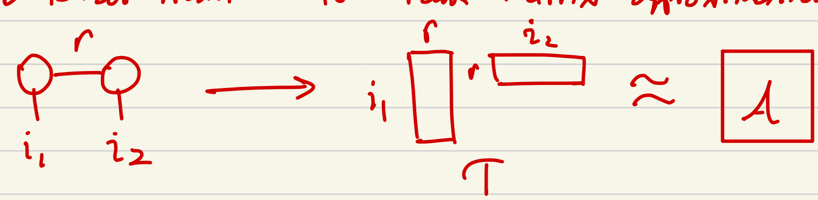
A volume maximization perspective

TT-Cross Heuristic:
- Simple & Fast (iterative)
- Few guarantees
- Difficult to beat (when A has sufficient structure)

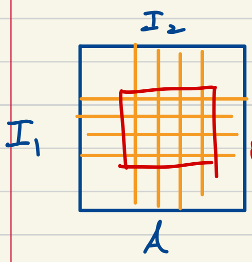
Backbone of functional tensor train algorithms.

Begin w/ special case: 2D tensor train = low-rank matrix approximation problem.

Briefly pretend that $N \approx 100$.



- CUR decomposition (aka cross / skeleton decomposition):



Want "most" lin. indep. set of rows & cols.

- Select $I_1 \subseteq [N]$ rows of A, $I_2 \subseteq [N]$ columns.

Define
$$C = A[:, I_2]$$
$$U = A[I_1, I_2]$$
$$R = A[I_1, :]$$
,
$$\underline{\underline{A \approx CU^{+}R}}$$

Moore-Penrose pseudo-inverse.

- If A exactly rank-R, then select any lin. indep. subset of rows I_1, cols I_2.

Equality holds: $A = CU^{+}R$.

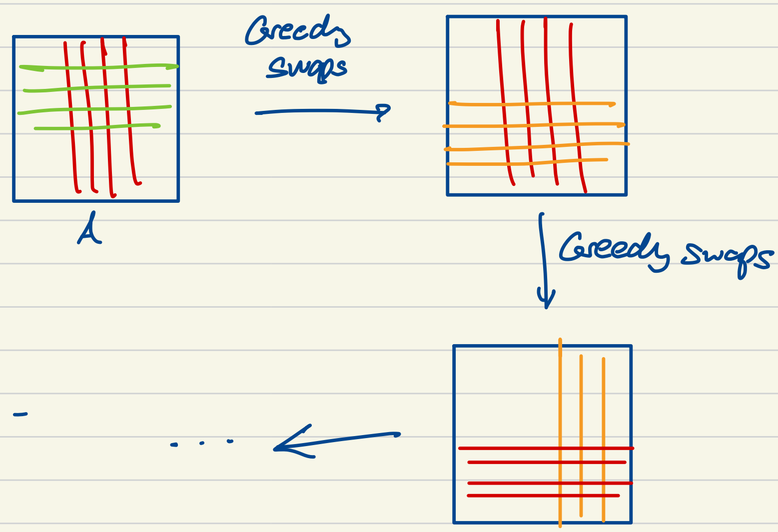
Motivation: Suppose singular values beyond R are small.

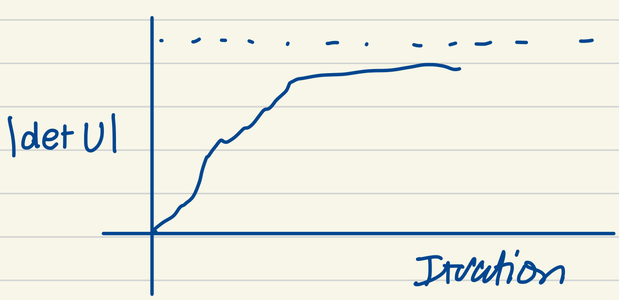
- NOT EXACTLY low-rank: select I_1, I_2 to maximize $|\det U|$.

Intuition: Higher determinant $\longleftrightarrow$ Higher diversity of sampled rows & cols

NP-Hard! Settle for a greedy algorithm:

Init: Random I_1, I_2.
Repeat: • Greedily exchange rows to increase $|\det U|$
• Greedily exchange cols to increase $|\det U|$
Repeat.





- $|\det U|$ increases monotonically, may not converge to global maximum, but will converge.

[Other ways to do this:
LU w/ column-pivoting
If A p.s.d : Ridge leverage scores
Determinental Point Process Sampling.]

• In practice: Use QR decomposition. Numerical stability in case rank overestimated.

$$R = A(:, J)$$

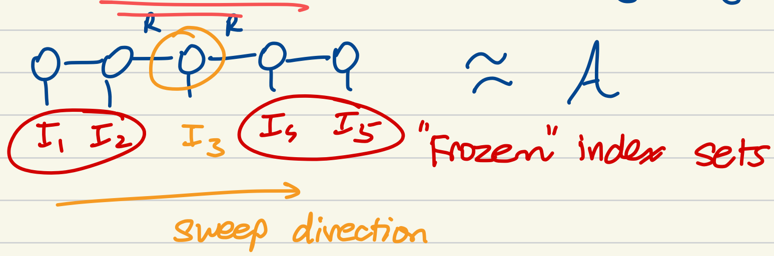
$$C = A(I, :), \text{ compute } C = QT$$

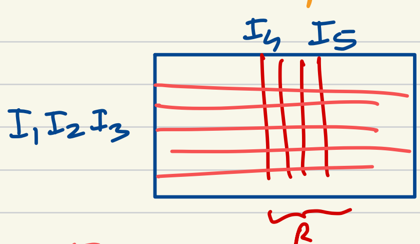
then $A_{K+1} = A(:, J)$

Adapt to general MPS: greedy volume maximization on the matricizations of the tensor.

Each set has cardinality R.

1) Start w/ random $I_1, I_2, \dots I_D$. Sweep left to right, right to left:





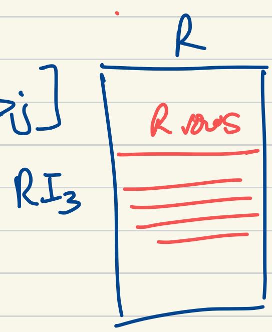
Update Rows (Too many rows to select from!)

Restrict to a candidate set of left-nested indices:

I_1	I_2	I_3
0	3	?
5	4	?
2	7	?
1	2	?
⋮	⋮	

max # of choices: RI_3.

$A[I_{<j}, I_j, I_{>j}]$



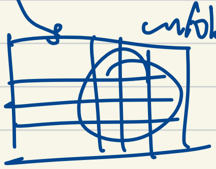
Run greedy maxvol row swap algorithm on this restricted matrix to update I_3

1) Caveat: In practice, run maxvol on the QR decomposition of the selected matrix; prevents the case where the ranks are over-estimated.

Theorem: Given cores $C_1, C_2, \ldots C_D$ from the index sets where

$T[I_{<j}, I_{>j}]$ nonsingular for $\forall j$,

Nonsingular for each unfolding



Construct

$$\hat{C}_K = C_K \times_3 \hat{A}_K^{-1}$$
$$\hat{C}_1 = C_1 \hat{A}_1$$

Max evaluations of tensor: $R^2 I$ per core update, NIR^2 per sweep.

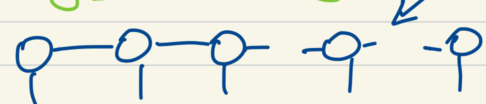
Guarantees
- Converges to best fit TT? ✗
- Converges? ✗

Notice: no monotonically increasing objective.

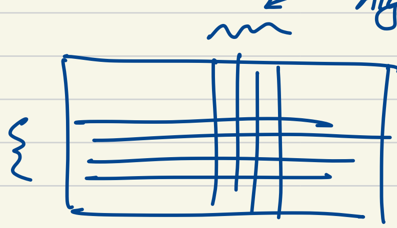
But in practice, usually does converge.

- "Fixed point" property? ✓

perturb single core



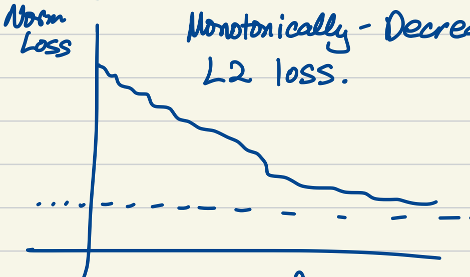
Algorithm finds a lin. indep. set of columns



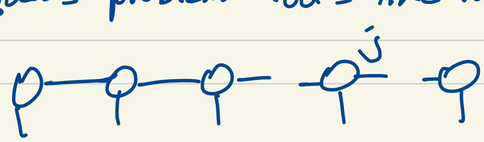
Alternative view: Alternating Least-Squares

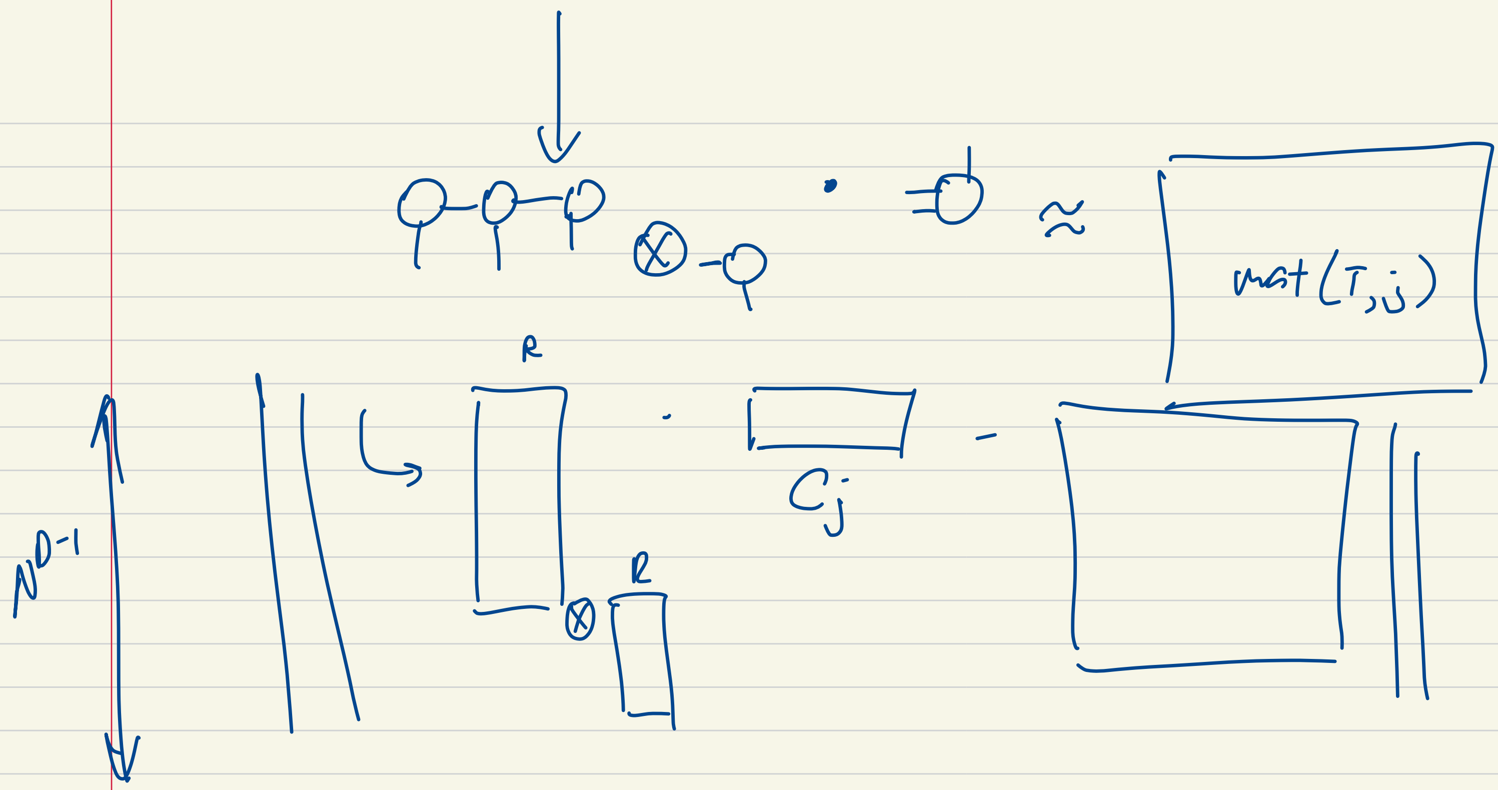
for $i = 1 \ldots D$
optimize core i to minimize $\|T - A\|_F$.

for $i = D-1 \ldots 1$
" "



That linear least-squares problem looks like this:





<u>Thm</u>: There exist sketching algorithms to solve the LSTSQ problem to residual accuracy ε whp $(1-\delta)$ in <u>poly</u>-time.

<u>Proof</u>: Apply a Johnson-Lindenstrauss transform matrix to both sides.

[Note: MPS random projections are useful for a bunch of things].

HSDC Seminar Round 2

Agenda

- Recap: MPS-Cross & Data Plots
- TT-ALS & Modifications to TT-ALS
- Random projections for MPS & new results.

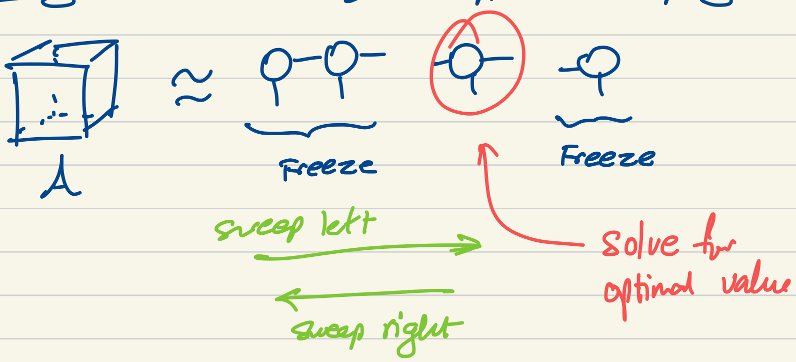
- Last time: Volume maximization to construct TT-tensor.

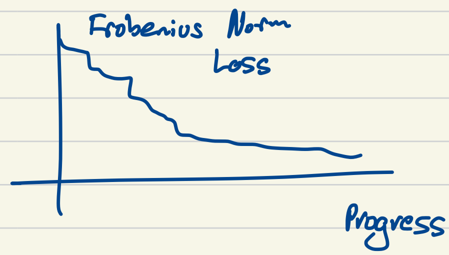
  - Maxvol on each matricization of tensor T, use indices to construct tensor. Start w/ random index sets from tensor and then refine.

- Problem: Local minima, only works for the low-error case.

[Show graphs of TT-cross heuristic]

- Today: Linear least-squares approach / Adapting the tensor-train rank.

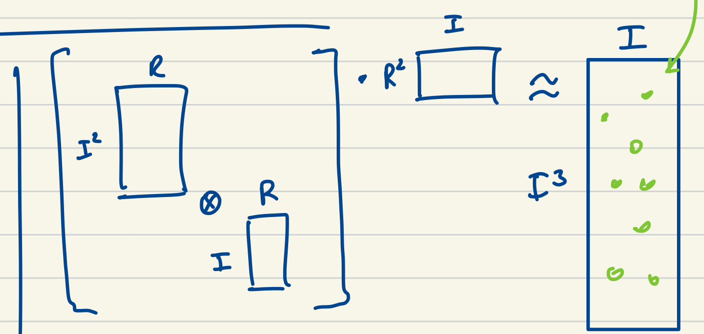




Advantages:
- Monotonically decreasing objective
- Parallel, simple
- Can be adapted to deal w/ missing entries.

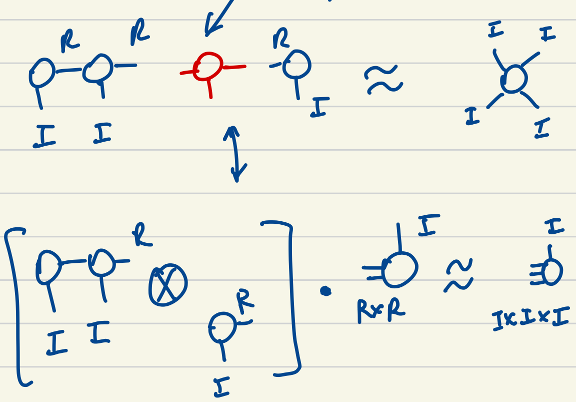
Disadvantages:
- Too expensive w/ out modifications
- Needs whole tensor* → Randomized algs address this

+ "Less chance" of local minimum.



- TT-ALS:

solve for this



Cost to solve this: $O(I^4R^2 + IR^4)$
{by QR decomposition of LHS, cost to multiply by Q, backsolve w/ Kronecker product R matrix).

Might as well perform TT-SVD, so this is not that useful.

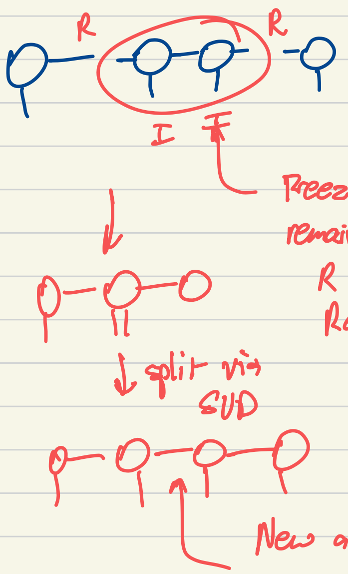
[Note: this alternating scheme also used for eigenvalue problems where both matrix & vector have MPO/MPS form: DMRG algorithm]

Modifications:

- Missing entries: Solve for $[T_1, \dots T_D]$ representation that minimizes loss w.r.t. sparse subset of known entries, representation generalizes to the unknown entries.

  Solve for each core in time $O(\mathrm{nnz}(A) R^2)$, system is smaller.

- Adaptive Rank (also applies to TT-cross):

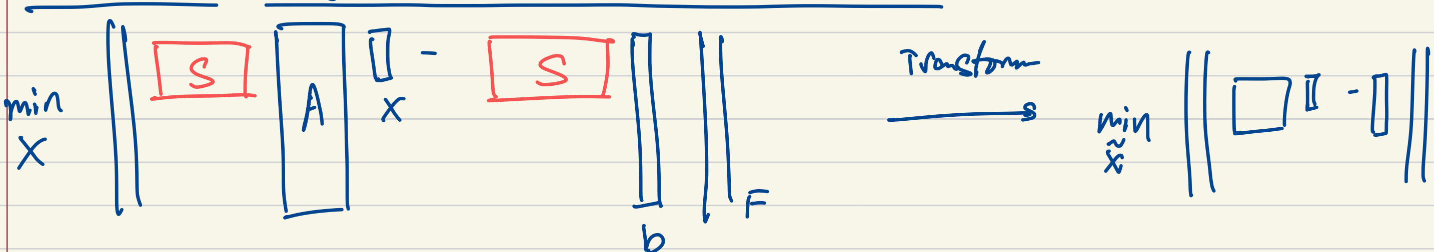


Freeze all but two cores, solve for remaining "supercore" of dimension $R \times I^2 \times R$, Reshape $\rightarrow RI \times RI$

Compute SVD, truncate at appropriate threshold.

New adaptively determined rank.

MPS Random Projections



$\min \|Ax - b\|_2^2$

Expensive, too many rows!

$\min_{\tilde{x}} \|SAx - Sb\|_2^2$

Soln: Apply sketching / sampling matrix to system; [S has far fewer rows than columns].

Choose S so that $\|A\tilde{x} - b\|^2 \le (1+\varepsilon) \min_x \|Ax - b\|^2$

How to do this? [Subject of active research w/ well-known results]

- Turns out S can be (works w/ high probability)
  - An i.i.d Gaussian matrix [but MM ruins speedup]
  - A sparse ± 1 random matrix w/ 1 nonzero per column [or several nonzeros]
  - An MPS where every core has [normalized] i.i.d Gaussian random entries
  - A sampling matrix where each row selected w/ probability $\propto$ $\ell_i = A_{i:} (A^T A)^+ A_{i:}^T$.

Statistical leverage scores

(subspace embedding, low distortion)

Key property: Let $A = U \Sigma V^T$ be SVD of A. Want

$\kappa(SU)$ as small as possible.

Intuition: vectors that are orthogonal in original space almost orthogonal in transformed space.

"Embeds" colspace of A into a smaller subspace while preserving distances between vectors.

[These sketches have other uses, e.g. MPS-rounding].

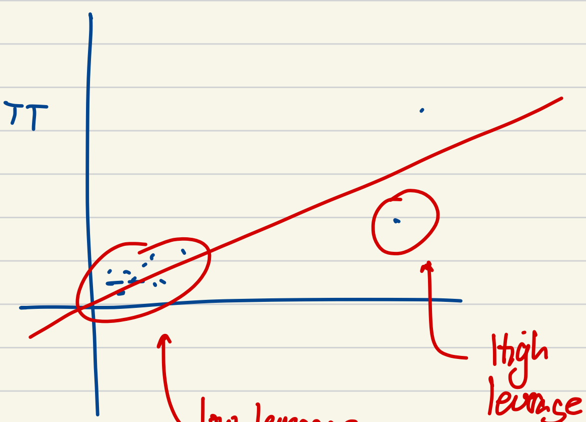
Key Challenge: How to sketch when A is in tensor train format, B is in general format (matricized tensor?)

Output of Sketch: Row count $O(R^2 \log(\dots) / \varepsilon\delta)$ if column count is R^2.

Results on Leverage Score Sampling

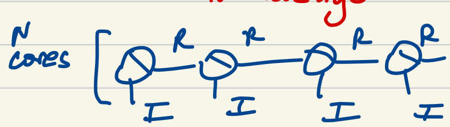
- Facts: Only $O\left(\frac{R^2}{\varepsilon} \log\left(\frac{R}{\delta}\right)\right)$ rows needed for TT problem.

  Leverage scores do not depend on obs. matrix B (!)



- Expensive to compute leverage scores. For A w/ I^N rows & R^2 columns,

  $O(I^N R^4)$ to compute scores.



New Result: Can sample 1 row according to leverage score distribution of a TT in canonical form in time $O(NR^2 \log I)$, which is time required to form that row.

Time Complexity of ALS: $O(N I^N R^N)$

$\downarrow$ Sketching

$\tilde{O}(N(NR^N + R^6))$

Hides log factors

ok when R is small, space usage of dense TT grows quadratically in rank R.

- Compromise: "See" whole tensor eventually, but make progress faster than TT-SVD
(Full-batch vs. stochastic gradient descent)