Faster Accurate Sketching for Tensor Networks

Linjian Ma and Edgar Solomonik

LPNA @CS@Illinois

Department of Computer Science
University of Illinois at Urbana-Champaign

SIAM CSE
Amsterdam, NL

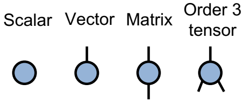
Talk Overview

- Linjian Ma and ES. *Fast and accurate randomized algorithms for low-rank tensor decompositions*, NeurIPS 2021, arXiv:2104.01101.
  - problem: efficiently sketch the (standard) HOOI algorithm for low-rank Tucker decomposition of sparse tensors
  - results: algorithms based on randomized range finding, leverage score sampling, and TensorSketch; error bounds and experimental analysis
- Linjian Ma and ES. *Cost-efficient Gaussian tensor network embeddings for tensor-structured inputs*, NeurIPS 2022, arXiv:2205.13163.
  - problem: if X is represented by a tensor network, choose a tensor network sketch S to minimize cost of sketching (computing SX)
  - results: sufficient condition for JL lemma for any tensor network graph, cost-optimal tensor network sketch under this condition

`https://linjianma.github.io/`

Tensor Diagrams

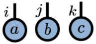
Tensor diagram: each tensor represented by a vertex, joining edges means contraction



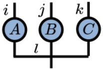
Examples:

Inner product: $\sum_i a_i b_i$

Matrix product : $C_{ik} = \sum_j A_{ij} B_{jk}$



Kronecker/outer product : $T_{ijk} = a_i b_j c_k$

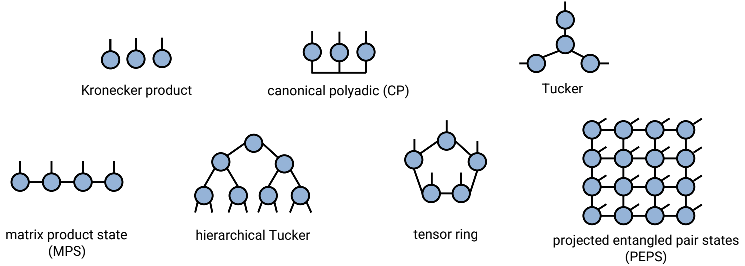


Khatri-Rao product: $T_{ijkl} = A_{il} B_{jl} C_{kl}$

Tensor Decompositions and Tensor Networks

Tensor network: a set of tensors contracted according to a (hyper)graph

Tensor decomposition: represent a (high-dimensional) tensor with a tensor network

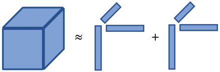


Applications: addressing curse of dimensionality, useful for many tasks in signal processing, machine learning, quantum simulation

Alternating Optimization

CP decomposition

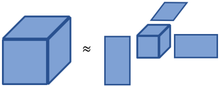
$$\boldsymbol{\mathcal{T}} \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r$$



- $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n\times n\times n}$,
  $A = [a_1, \ldots, a_R] \in \mathbb{R}^{n\times R}$

Tucker decomposition

$$\boldsymbol{\mathcal{T}} \approx \boldsymbol{\mathcal{X}} \times_1 A \times_2 B \times_3 C$$



- $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n\times n\times n}$, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{R\times R\times R}$
- $A, B, C \in \mathbb{R}^{n\times R}$ orthogonal

CP-Alternating least squares (CP-ALS)

$$\min_{A} \left\| (C \odot B) A^T - T_{(1)}^T \right\|_F$$

Higher order orthogonal iteration (HOOI)

$$\min_{A, \boldsymbol{\mathcal{X}}} \left\| (C \otimes B) X_{(1)}^T A^T - T_{(1)}^T \right\|_F$$

HOOI interpretation: solve a rank-constrained linear least squares problem

$$\min_{X, \text{rank}(X)=R} \|QX - B\|_F$$

Amenable to sketching: rank-constraint :(, $Q = C \otimes B$ is orthogonal :)

Accurately Sketching an Orthogonal Matrix

- Seek random matrix $S \in \mathbb{R}^{m\times n}$ so that solution to sketched problem

$$\min_{X,\text{rank}(X)=R} \|SQX - SB\|_F$$

$$\hat{X} = (SQ)^{+}SB$$

satisfies $\|Q\hat{X} - B\|_F \leq (1+\epsilon)\|QX^* - B\|_F$ relative to the optimal X^* with probability $1-\delta$

- Using known error bounds on sketching of matrix products, we show
  - leverage score sampling satisfies above with

$$m = \tilde{O}(R^{N-1}/(\epsilon^2\delta))$$

  - TensorSketch[1] satisfies this with

$$m = O((3R)^{N-1}(R^{N-1} + 1/\epsilon^2)/\delta)$$

[1] O. Malik and S. Becker, 2018

Efficient Sketching for HOOI

Leverage score sampling

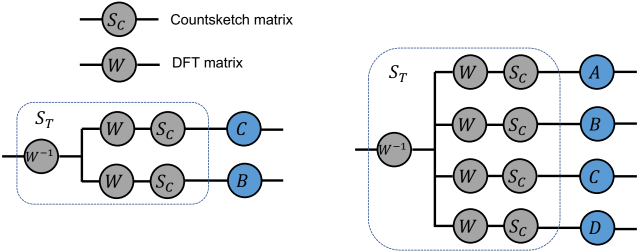
- Since $Q = C \otimes B$, leverage scores satisfy

$$l_{(i-1)n+j}(Q) = \|q_{(i-1)n+j}\|_2^2 = \|c_i\|_2^2 \|b_j\|_2^2 = l_i(C) l_j(B)$$

  hence we can take products of independent samples of rows of A and B to obtain the leverage-score based distribution of columns of Q

- Since A, B, C are changing, we must sample the tensor for each optimization step

TensorSketch reduces the amount of necessary sampling to 1 round



Sketched HOOI algorithm

Input: Input order N tensor $\boldsymbol{\mathcal{T}}$, Tucker rank R, number of sweeps I_{max}
Output: $\{\boldsymbol{\mathcal{X}}, A^{(1)}, \ldots, A^{(N)}\}$
For $n \in \{2, ..., N\}$ **do**
  $A^{(n)} \leftarrow$ `Init-RRF`$(T_{(n)}, R, \epsilon)$ // Randomized range finder with composite sketch (Gaussian + CountSketch)
Endfor
For $i \in \{1, ..., I_{max}\}$ **do**
  For $n \in \{1, ..., N\}$ **do**
    Build the sketching matrix S
    $Y \leftarrow ST_{(n)}$ // Can be done outside i loop for TensorSketch
    $Z \leftarrow S^{(n)}(A^{(1)} \otimes \cdots \otimes A^{(n-1)} \otimes A^{(n+1)} \otimes \cdots \otimes A^{(N)})$
    $X_{(n)}^T, A^{(n)} \leftarrow$ `Solve-truncate`(Z, Y, R)
  Endfor
Endfor
Return $\{\boldsymbol{\mathcal{X}}, A^{(1)}, \ldots, A^{(N)}\}$

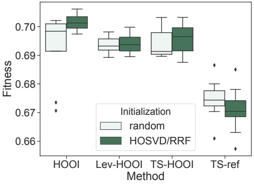
Cost comparison for order 3 tensor

ALS + TensorSketch (Malik and Becker, NeurIPS 2018)

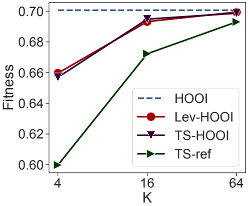
- Solving for each factor matrix or the core tensor at a time
- $\min_A \frac{1}{2} \left\| (C \otimes B) X_{(1)}^T A^T - T_{(1)}^T \right\|_F^2$ or
  $\min_{\boldsymbol{\mathcal{X}}} \frac{1}{2} \left\| (C \otimes B \otimes A) \mathrm{vec}(X) - \mathrm{vec}(T) \right\|_F^2$

Algorithm for Tucker	LS subproblem cost	Sketch size (k)
HOOI	$\Omega(\mathrm{nnz}(\boldsymbol{\mathcal{T}})R)$	/
ALS + TensorSketch	$\tilde{O}(knR + kR^3)$	$O((R^2/\delta) \cdot (R^2 + 1/\epsilon))$
HOOI + TensorSketch	$O(knR + kR^4)$	$O((R^2/\delta) \cdot (R^2 + 1/\epsilon^2))$
HOOI + leverage scores	$O(knR + kR^4)$	$O(R^2/(\epsilon^2\delta))$

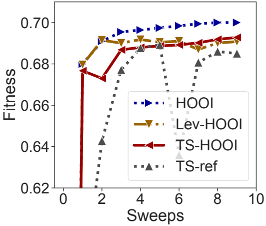
Experiments: Tensors with Spiked Signal



(a) 5 sweeps, sample size $16R^2$



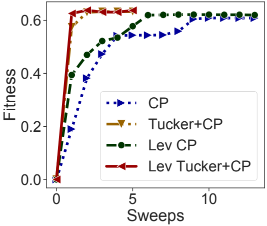
(b) 5 sweeps, sample size KR^2



(c) sample size $16R^2$

- $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{T}}_0 + \sum_{i=1}^{5} \lambda_i a_i \circ b_i \circ c_i$, each a_i, b_i, c_i has unit 2-norm, $\lambda_i = 3\frac{\|\boldsymbol{\mathcal{T}}_0\|_F}{i^{1.5}}$
- Leading low-rank components obey the power-law distribution
- Tensor size $200 \times 200 \times 200$, $R = 5$
- TS-ref: (Malik and Becker, NeurIPS 2018)

Experiments: CP decomposition

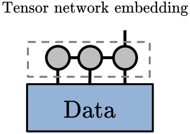


- $\boldsymbol{\mathcal{T}} = \sum_{i=1}^{R_{\text{true}}} a_i \circ b_i \circ c_i,\ R_{\text{true}}/R = 1.2$
- Tensor size $2000 \times 2000 \times 2000$, $R = 10$, sample size $16R^2$
- Lev CP: leverage score sampling for CP-ALS (Larsen and Kolda, arXiv:2006.16438)
- Tucker+CP: Run Tucker HOOI first, then run CP-ALS on the Tucker core
- Run Tucker HOOI with 5 sweeps, CP-ALS with 25 sweeps

Sketching General Tensor Networks

Problem: Given a tensor network input data, x, find a **Gaussian** tensor network embedding, S, such that the embedding is (ϵ, δ)-accurate and

- The number of rows of S (sketch size m) is low
- Asymptotic cost to compute Sx is minimized



An (oblivious) embedding $S \in \mathbb{R}^{m \times s}$ is (ϵ, δ)-accurate if[1]

$$\Pr\left[\left|\frac{\|Sx\|_2 - \|x\|_2}{\|x\|_2}\right| > \epsilon\right] \leq \delta \quad \text{for any } x$$

[1]Woodruff, Sketching as a tool for numerical linear algebra, 2014

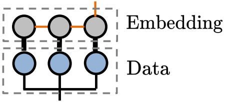
Sketching Tensor Network Data

Previous work:

- Kronecker product embedding[1]: inefficient in computational cost
- Tree embedding (e.g. MPS)[2]: efficient for specific data (Kronecker product, MPS), but efficiency unclear for general tensor network data

Assumptions throughout our analysis:

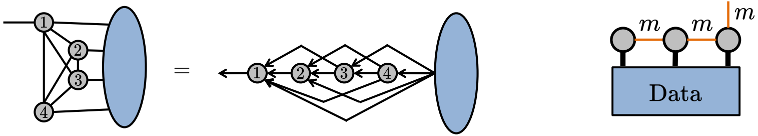
- Classical $O(n^3)$ matmul cost
- Consider embeddings defined on **graphs with no hyperedges**
- Each dimension to be sketched
  - has a **size lower bounded by the sketch size**
  - is only adjacent to one data tensor



[1] Ahle et al, Oblivious sketching of high-degree polynomial kernels, SODA 2020
[2] Rakhshan and Rabusseau, Tensorized random projections, AISTATS 2020

Sufficient condition for (ϵ, δ)-accurate embedding

The embedding $G = (V, E, w)$ is accurate if there exists a linear ordering of V such that in its induced DAG, the weighted sum of out-going edges adjacent to each $v \in V$ is $\Omega(m)$, where $m = N \log(1/\delta)/\epsilon^2$

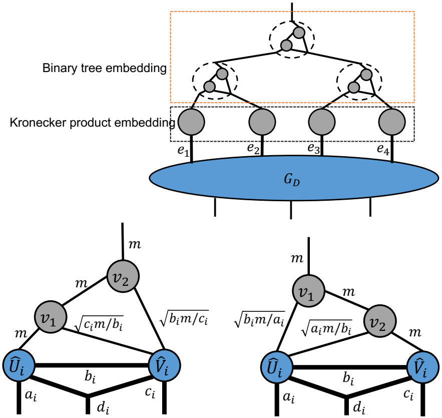


Proof of accuracy leverages two key prior results[1]

1. If S is (ϵ, δ)-accurate, so is $I \otimes S \otimes I$
2. If $S_1, \ldots, S_N$ are $(O(\epsilon/\sqrt{N}), \delta)$-accurate, $S_1 \cdots S_N$ is (ϵ, δ)-accurate

[1] Ahle et al, Oblivious sketching of high-degree polynomial kernels, SODA 2020

Efficient General Sketching



- Tensor network sketch contains
  1. Kronecker product embedding
  2. binary tree of small tensor network gadgets
- Each gadget sketches product of two tensors
  - chosen to minimize cost depending on connectivity
  - may or may not be a tree
- Can reduce cost by up to $O(\sqrt{m})$ relative to binary tree
- near-optimal under assumptions

Applications of Tensor Network Sketching

- If input data is Khatri-Rao product or tensor product
  - new gadgets reduce cost by $O(\sqrt{m})$ relative to Gaussian binary tree embedding
  - this allows acceleration of sketching for CP decomposition
  - tree-like sketch structure also allows intermediate reuse during optimization (dimension trees)
- When data is an MPS (tensor train)
  - plain tree sketch is efficient (sketch can be binary tree or MPS-like)
  - shows optimality (subject to our sufficient condition) of prior work[1]

[1] Al Daas, Hussam, et al. Randomized algorithms for rounding in the tensor-train format, SISC 2023.

Summary and Conclusions

- Sketching for Tucker decomposition
  - Sketching HOOI gives accurate decomposition with enough sketch size
  - TensorSketch permits 1-pass (streaming) Tucker and CP
  - High polynomial scaling in rank; for CP addressable by indirect leverage score sampling[1]
- Gaussian tensor network sketching
  - achieves linear cost relative to number of input tensors
  - limited analysis to Gaussian tensors, classical matrix multiplication cost
  - not considering hyperedges in sketch, e.g., Khatri-Rao product in TensorSketch

[1] Bharadwaj, Vivek, et al. Fast exact leverage score sampling from Khatri-Rao products with applications to tensor decomposition, 2023. arXiv:2301.12584

References

- Linjian Ma and ES. *Fast and accurate randomized algorithms for low-rank tensor decompositions*, NeurIPS 2021, arXiv:2104.01101.
- Linjian Ma and ES. *Cost-efficient Gaussian tensor network embeddings for tensor-structured inputs*, NeurIPS 2022, arXiv:2205.13163.

https://linjianma.github.io/

LPNA @CS@Illinois

https://lpna.cs.illinois.edu/