Towards efficient algorithms and systems for tensor decompositions and tensor networks

Linjian Ma

Department of Computer Science
University of Illinois Urbana-Champaign

Final exam

Doctoral committee: Edgar Solomonik, Chandra Chekuri, Luke Olson, Miles Stoudenmire

August 14th, 2023

Presentation overview

Thesis motivation: design and automate fast numerical algorithms for tensor computations in science and engineering

Outline of the presentation:

- Introduction to tensors
- An overview of thesis contributions
- Sketching for tensor decompositions and tensor networks
- Algorithms for approximate tensor network contractions

Tensor

Tensor: multi-dimensional array of data

- Order: number of dimensions of a tensor
- Dimension size: number of elements in each dimension

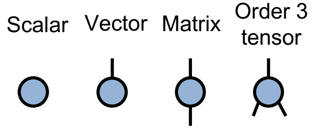
vector	matrix	third order tensor
$\begin{bmatrix} 4 \\ 5 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$

Tensors occur in

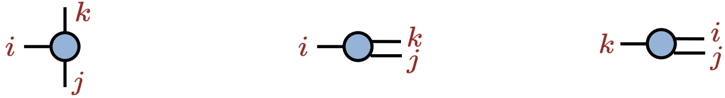
- Data science: image, video, medical data...
- Scientific computing: discretization of high-dimensional functions
- Quantum physics and quantum computing: wavefunction, Hamiltonian, quantum gate

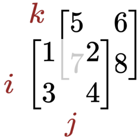
Tensor diagram notation

Tensor diagram: an order N tensor is represented by a vertex with N adjacent edges



Matricization: transform a tensor into a matrix





$$\begin{array}{cc} & k, j \\ i & \begin{bmatrix} 1 & 5 & 2 & 6 \\ 3 & 7 & 4 & 8 \end{bmatrix} \end{array}$$

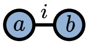
$$\begin{array}{cc} & i, j \\ k & \begin{bmatrix} 1 & 3 & 2 & 4 \\ 5 & 7 & 6 & 8 \end{bmatrix} \end{array}$$

Tensor contraction

Tensor contraction: summing element products from two tensors over contracted dimensions

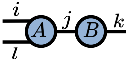
A dimension (edge) is contracted if it has no open end

Examples:

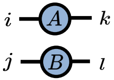


Inner product: $\sum_i a_i b_i$

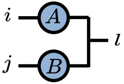
Matrix product : $C_{ik} = \sum_j A_{ij} B_{jk}$



Tensor times matrix: $C_{ilk} = \sum_j A_{ilj} B_{jk}$



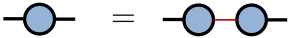
Kronecker/outer product: $T_{ijkl} = A_{ik} B_{jl}$



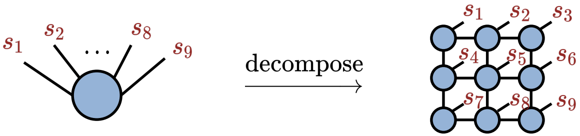
Khatri-Rao product: $T_{ijl} = A_{il} B_{jl}$

Tensor decomposition: break the curse of dimensionality

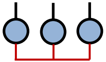
Matrix factorization:



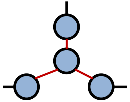
Tensor decomposition: represents a tensor with a (low-rank) tensor network



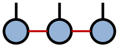
Canonical polyadic (CP) decomposition



Tucker decomposition



Tensor train decomposition



Applications of tensor decompositions and tensor networks

Tensor decompositions:

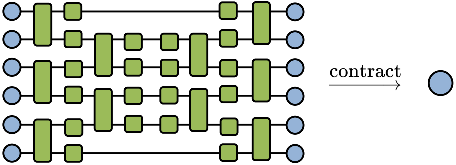
Data science: detect latent structure[1,2]

Quantum chemistry: accelerate high-accuracy methods[3]

Quantum physics: represent wavefunctions and Hamiltonians[4]

Tensor network contractions:

Quantum computing: simulate quantum algorithm[5]



[1] Kolda and Bader, Tensor decompositions and applications, SIAM review 2009

[2] Sidiropoulos et al, Tensor decomposition for signal processing and machine learning, IEEE Signal Processing 2017

[3] Hohenstein et al, Communication: Tensor hypercontraction. III. Least-squares tensor hypercontraction for the determination of correlated wavefunctions, JCS 2012

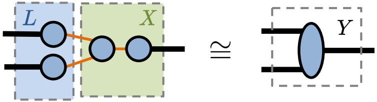
[4] Verstraete et at, Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems, Advances in physics 2008

[5] Markov and Shi, Simulating quantum computation by contracting tensor networks, SIAM Journal on Computing 2008

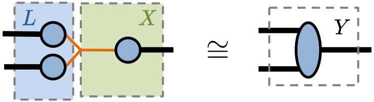
(Rank-constrained) linear least squares with tensor networks

$$\min_{X,\mathrm{rank}(X)\leq R} \|\, L \; X \; - \; Y \,\|_F$$

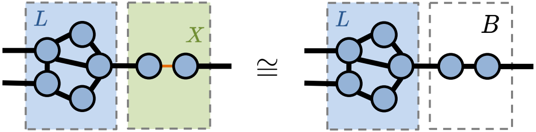
Tucker decomposition



CP decomposition



Tensor network contraction



An overview of thesis contributions

Accelerating alternating minimization of tensor decompositions[1,2,3]

- Pairwise perturbation for CP and Tucker decompositions
- AutoHOOT: an automatic differentiation system for tensors

Sketching for tensor decompositions and tensor networks[4,5]

Approximate tensor network contraction algorithms[6,7]

Use low-rank CP decomposition to simulate and analyze quantum algorithms[8,9]

- We simulate Grover's search, quantum Fourier transform, quantum phase estimation
- A new upper bound on CP rank of specific quantum states

---

[1] [Ma and Solomonik, NLA 2022] [2] [Ma and Solomonik, IPDPS 2021] [3] [Ma, Ye and Solomonik, PACT 2020]
[4] [Ma and Solomonik, NeurIPS 2021] [5] [Ma and Solomonik, NeurIPS 2022]
[6] [Ma, Ibrahim, Safro, and Solomonik, in preparation] [7] [Ma, Fishman, Stoudenmire, and Solomonik, in preparation]
[8] [Ma and Yang, JCS 2022] [9] [Schatzki, Ma, Solomonik, and Chitambar, 2022]

Presentation overview

Outline of the presentation:

- Introduction to tensors
- An overview of thesis contributions
- Sketching for tensor decompositions and tensor networks
- Algorithms for approximate tensor network contractions

Sketching for linear least squares

Sketching: randomly project a data L to low dimensional spaces

$$L \longrightarrow SL$$

- $L \in \mathbb{R}^{s\times n}$, $S \in \mathbb{R}^{m\times s}$ with the sketch size $m \ll s$
- S is a random matrix (called embedding)

Sketching for linear least squares

Sketching: randomly project a data L to low dimensional spaces

$$L \longrightarrow SL$$

- $L \in \mathbb{R}^{s\times n}$, $S \in \mathbb{R}^{m\times s}$ with the sketch size $m \ll s$
- S is a random matrix (called embedding)

Standard LLS:

$$X^* = \underset{X}{\operatorname{argmin}} \|LX - Y\|_F$$

Sketched LLS:

$$\hat{X} = \underset{X}{\operatorname{argmin}} \|SLX - SY\|_F$$

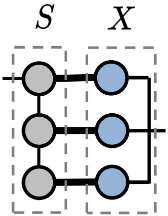
- Gaussian random matrix is standard for embedding
- Sparse embedding[1] can be used when L, Y are sparse (computing SL only costs $\mathrm{nnz}(L)$)

[1] Charikar et al, Finding frequent items in data streams, 2002

Sketching general tensor networks

Problem: Find a tensor network embedding S for the tensor network X, so that

- The embedding is (ϵ, δ)-accurate
- The sketch size (number of rows of S) 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]

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

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

Outline: sketching for tensor networks

$$\min_X \|LX - Y\|_F \quad \rightarrow \quad \min_X \|SLX - SY\|_F$$

Sketching for low-rank Tucker decomposition of large and sparse tensors[1]

- L is a Kronecker product of matrices and has orthonormal columns
- A new sketch size upper bound on the problem
- Reach at least 98% of the standard algorithm's accuracy with better cost

[1] **Ma** and Solomonik, Fast and accurate randomized algorithms for low-rank tensor decompositions, NeurIPS 2021

Outline: sketching for tensor networks

$$\min_X \|LX - Y\|_F \quad \rightarrow \quad \min_X \|SLX - SY\|_F$$

Sketching for low-rank Tucker decomposition of large and sparse tensors[1]

- L is a Kronecker product of matrices and has orthonormal columns
- A new sketch size upper bound on the problem
- Reach at least 98% of the standard algorithm's accuracy with better cost

A cost-efficient algorithm to sketch arbitrary tensor network[2]

- L has arbitrary tensor network structure
- Find accurate and cost-optimal embeddings S
- Asymptotically faster than previous works for CP decomposition

[1] **Ma** and Solomonik, Fast and accurate randomized algorithms for low-rank tensor decompositions, NeurIPS 2021
[2] **Ma** and Solomonik, Cost-efficient Gaussian tensor network embeddings for tensor-structured inputs, NeurIPS 2022

Sketching for Tucker decomposition

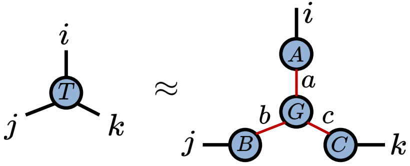
Goal: efficiently sketch the rand-constrained linear least squares problem arising in alternating least squares for Tucker decomposition

Alternating least squares for Tucker decomposition

Tucker decomposition

$$\min_{G,A,B,C} \sum_{i,j,k} \left(T_{ijk} - \sum_{a,b,c} G_{abc} A_{ia} B_{jb} C_{kc} \right)^2$$

- $T \in \mathbb{R}^{s\times s\times s}$, $X \in \mathbb{R}^{R\times R\times R}$
- $A, B, C \in \mathbb{R}^{s\times R}$ with orthonormal columns, $R < s$

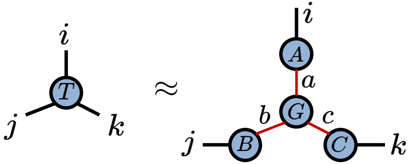


Alternating least squares for Tucker decomposition

Tucker decomposition

$$\min_{G,A,B,C} \sum_{i,j,k} \left(T_{ijk} - \sum_{a,b,c} G_{abc} A_{ia} B_{jb} C_{kc} \right)^2$$

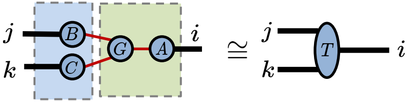
- $T \in \mathbb{R}^{s\times s\times s}$, $X \in \mathbb{R}^{R\times R\times R}$
- $A, B, C \in \mathbb{R}^{s\times R}$ with orthonormal columns, $R < s$



Higher order orthogonal iteration (HOOI)[1]

- Costs $\Omega(\mathrm{nnz}(T)R)$ for arbitrary tensor order
- Fast convergence (usually in around 10 iterations)

$$\min_{X,\mathrm{rank}(X)\le R} || \; L \; X \; - \; Y \; ||_F$$

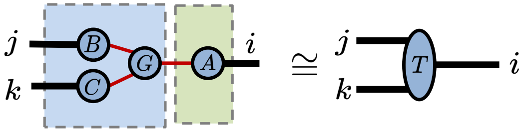


[1] Lathauwer et al, On the best rank-1 and rank-$(R_1, R_2, \ldots, R_n)$ approximation of higher-order tensors, SIMAX 2000

Sketching for Tucker decomposition: previous work

Sketch alternating unconstrained least squares (AULS)[1]

- Advantage: cost with t iterations is $O\left(\text{nnz}(T) + t\left(sR^5 + R^7\right)\right)$
- Disadvantage: not an orthogonal iteration and has slow convergence



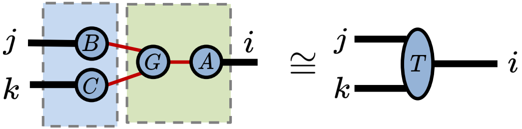
Apply sketching on high-order SVD[2]

- Apply randomized SVD on matricizations of T
- Disadvantages: accuracy lower than HOOI and costs $\Omega(\text{nnz}(T)R)$

[1] Malik and Becker, Low-rank tucker decomposition of large tensors using Tensorsketch, NeurIPS 2018
[2] Ahmadi-Asl et al, Randomized algorithms for computation of Tucker decomposition and HOSVD, IEEE Access 2021

Sketched HOOI for Tucker decomposition

$$\min_{X,\mathrm{rank}(X)\leq R} || \; L \;\; X \; - \; Y \; ||_F$$



HOOI: solve and truncate

$$X^* \leftarrow \underset{X}{\mathrm{argmin}} \|LX - Y\|_F^2$$

$$X_R^* \leftarrow \text{rank-}R \text{ approximation of } X^*$$

$$GA \leftarrow X_R^*$$

Sketched HOOI: sketch, solve and truncate

$$\hat{X} \leftarrow \underset{X}{\mathrm{argmin}} \|SLX - SY\|_F^2$$

$$\hat{X}_R \leftarrow \text{rank-}R \text{ approximation of } \hat{X}$$

$$\hat{G}\hat{A} \leftarrow \hat{X}_R$$

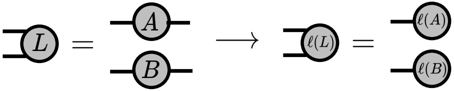
Sketched HOOI for Tucker decomposition

We use efficient embeddings S for solving $\min_X \|SLX - SY\|_F^2$

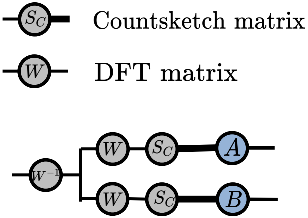
- L is a Kronecker product of factor matrices and changes over iterations
- Y is a matricization of the input tensor and can be sparse

Leverage score sampling

- Sample each row of L based on the leverage score vector $\ell(L)$



Tensorsketch: tensorized Countsketch[1]



[1] Pham and Pagh, Fast and scalable polynomial kernels via explicit feature maps, KDD 2013

Sketched HOOI for Tucker decomposition

We derive sketch size bounds so that

$$\left\| L\hat{X}_R - Y \right\|_F^2 \leq (1 + O(\epsilon)) \left\| LX_R^* - Y \right\|_F^2$$

- $X_R^*, \hat{X}_R$: optimal and the sketched solution
- We apply Mirsky's inequality[1] to bound change in singular values of the sketched L
- Sketch size upper bound is at most $O(1/\epsilon)$ times that for unconstrained LS

[1] Mirsky, The Quarterly journal of mathematics, 1960

Sketched HOOI for Tucker decomposition

We derive sketch size bounds so that

$$\left\| L\hat{X}_R - Y \right\|_F^2 \leq (1 + O(\epsilon)) \left\| LX_R^* - Y \right\|_F^2$$

- $X_R^*, \hat{X}_R$: optimal and the sketched solution
- We apply Mirsky's inequality[1] to bound change in singular values of the sketched L
- Sketch size upper bound is at most $O(1/\epsilon)$ times that for unconstrained LS

Algorithm performs well in experiments

- Sketched HOOI converges to at least 98% of the accuracy of standard HOOI
- With leverage score sampling, cost with t iterations is $O\left(\text{nnz}(T) + t\left(sR^3 + R^6\right)\right)$

[1] Mirsky, The Quarterly journal of mathematics, 1960

Sketching general tensor networks

Goal: accurately and efficiently sketch arbitrary tensor network structure

Sketching general tensor networks

Previous work:

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

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

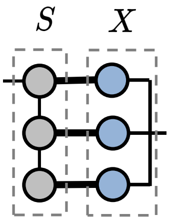
Sketching general tensor networks

Previous work:

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

Assumptions throughout our analysis:

- Multiply $A, B \in \mathbb{R}^{n\times n}$ has a cost of $O(n^3)$
- S is a Gaussian tensor network defined on graphs
- Each dimension to be sketched has large size

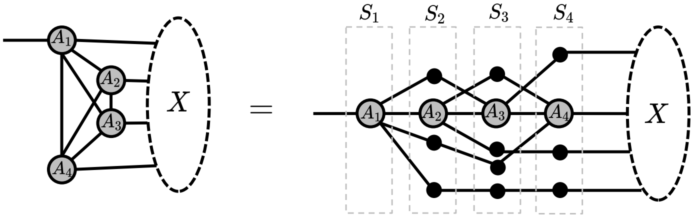


[1] Ahle et al, Oblivious sketching of high-degree polynomial kernels, SODA 2020
[2] Rakhshan and Rabusseau, Tensorized random projections, AISTATS 2020

Sufficient condition for (ϵ, δ)-accurate embedding

The embedding is accurate if we can rewrite $S = S_1 \cdots S_N$ and

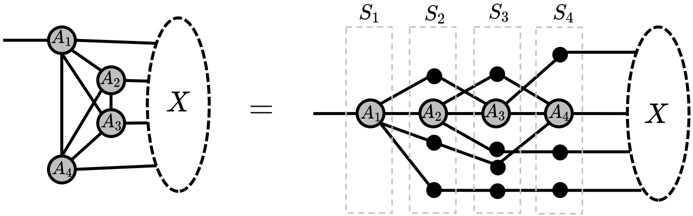
- S_i is the Kronecker product of A_i (a Gaussian random matrix) and identity matrices
- A_i has row size $\Omega(N \log(1/\delta)/\epsilon^2)$



Sufficient condition for (ϵ, δ)-accurate embedding

The embedding is accurate if we can rewrite $S = S_1 \cdots S_N$ and

- S_i is the Kronecker product of A_i (a Gaussian random matrix) and identity matrices
- A_i has row size $\Omega(N \log(1/\delta)/\epsilon^2)$

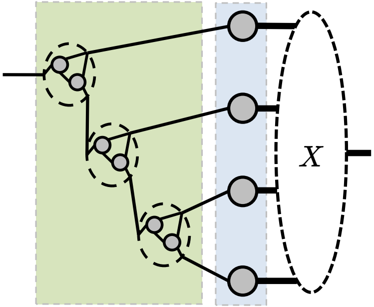


Two key prior results used in the proof[1]

- If A_i is (ϵ, δ)-accurate, so is the Kronecker product between A_i and identity matrices
- If $S_1, \ldots, S_N$ are $(\epsilon/\sqrt{N}, \delta)$-accurate, $S_1 \cdots S_N$ is $(O(\epsilon), \delta)$-accurate

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

A sketching algorithm with efficient computational cost and sketch size



Embedding containing a Kronecker product embedding + binary tree of gadgets

Each small gadget sketches the product of two tensors

- Each gadget contains a pair of tensors
- Dimension sizes in each gadget are chosen based on data tensors to minimize cost
- Can reduce cost by $O(\sqrt{m})$ compared to containing one tensor

Analysis of the algorithm

c: asymptotic sketching cost for our algorithm

c_{opt}: optimal asymptotic sketching cost under the embedding sufficient condition

m: sketch size

Input data tensor network structure	Optimality of the algorithm
General hypergraph	$c = O(\sqrt{m} \cdot c_{\text{opt}})$
General graph	$c = O(m^{0.375} \cdot c_{\text{opt}})$
Each data tensor has a dimension to be sketched (e.g. Kronecker product, tensor train)	$c = c_{\text{opt}}$

Applications

Low-rank CP decomposition with alternating least squares

- R: CP rank, N: tensor order
- Our algorithm is $\Omega(NR)$ times better than prior work[1]
- Larger preparation cost is needed (can be reduced by using sparse embeddings)

Truncation of high-rank tensor train

- Our algorithm is more efficient the standard algorithm
- We show the recently proposed truncation algorithm is also optimal[2]

[1] Malik, More Efficient Sampling for Tensor Decomposition With Worst-Case Guarantees, ICML 2022
[2] Daas et al, Randomized algorithms for rounding in the Tensor-Train format, SISC 2023

Presentation overview

Outline of the presentation:

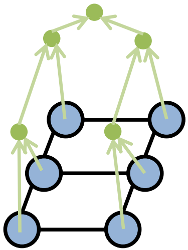
- Introduction to tensors
- An overview of thesis contributions
- Sketching for tensor decompositions and tensor networks
- Algorithms for approximate tensor network contractions

Tensor network contraction

Tensor network: denoted by undirected hypergraph $G = (V, E)$

Contraction tree: rooted binary tree T

- A leaf of T represents a tensor in G
- A non-leaf vertex represents its children's contraction output



Find contraction cost-optimal contraction tree: NP-hard[1], many heuristics are used[2,3]

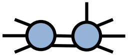
Cost under optimal contraction tree: exponential to the treewidth of G's line graph[4]

---

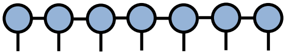
[1] O'Gorman, Parameterization of Tensor Network Contraction, TQC 2019
[2] Gray and Kourtis, Hyper-optimized tensor network contraction, Quantum 2021
[3] Liu et al, Computing solution space properties of combinatorial optimization problems via generic tensor networks, SISC 2023
[4] Markov and Shi, Simulating quantum computation by contracting tensor networks, SIAM Journal on Computing 2008

Approximate tensor network contractions: previous work

Idea: approximate each contraction output as a bounded-rank tensor network

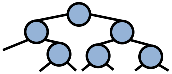


Tensor train/matrix product state (MPS)[1,2]



We propose an algorithm for cost-efficient contraction tree

Binary tree tensor network[3]



We propose to contract with flexible and cost-efficient low-rank approximation

[1] Pan et al, Contracting arbitrary tensor networks: general approximate algorithm and applications in graphical models and quantum circuit simulations, PRL 2020

[2] Chubb, General tensor network decoding of 2D Pauli codes, 2021

[3] Jermyn, Automatic contraction of unstructured tensor networks, SciPost Physics 2020

Outline: approximate tensor network contraction algorithms

Cost-efficient contraction tree for the tensor train-based algorithm[1]

- Solves a linear ordering problem to minimize edge crossings
- Achieves 5.9X speed-up when compared to previous works

Contraction with a flexible and cost-efficient low-rank approximation[2]

- Uses normal equations to improve efficiency and can flexibly select the environment
- Achieves 9.2X speed-up when compared to previous works

[1] **Ma**, Ibrahim, Safro, and Solomonik, An efficient swap-based algorithm for approximate tensor network contractions, in preparation

[2] **Ma**, Fishman, Stoudenmire, and Solomonik, Tensor network contraction with flexible environment incorporation and a cost-efficient density matrix algorithm for tree approximation, in preparation

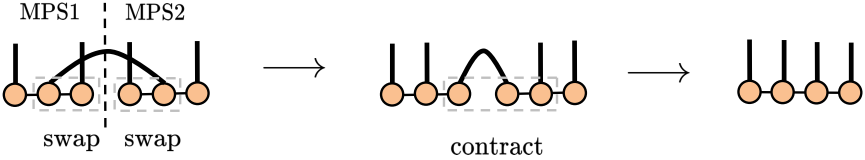
Accelerate tensor train-based algorithm

Goal: find efficient contraction trees for tensor train-based approximate tensor network contraction

Contraction of two tensor trains into a tensor train

Algorithm: move contracted edges to the center through adjacent swaps, then eliminate them[1]

- Each swap uses low-rank approximation to maintain a bounded rank



Observation: The total number of swaps is lower bounded by the convex crossing number[2]

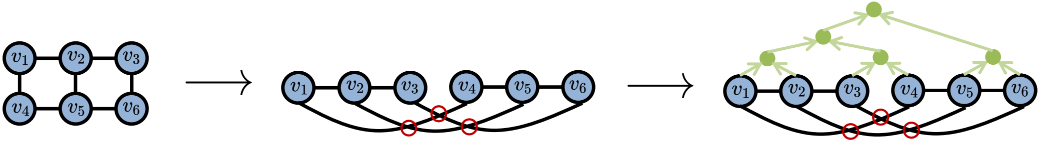
[1]Pan et al, Contracting arbitrary tensor networks: general approximate algorithm and applications in graphical models and quantum circuit simulations, PRL 2020

[2]Shahrokhi et al, Book embeddings and crossing numbers, WG'94

CATN-GO: build contraction tree constrained by a vertex ordering

Our approach: find a vertex ordering that minimizes edge crossings, then find a contraction tree constrained by the ordering

- Inspired by prior work on building exact tensor network contraction trees[1]

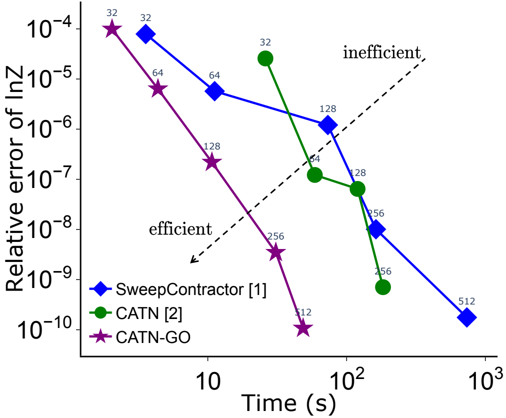


Find the optimal vertex ordering: NP-hard problem, heuristics are used[2]

Contraction tree optimization: minimize the cost using dynamic programming

[1] Ibrahim et al, Constructing Optimal Contraction Trees for Tensor Network Quantum Circuit Simulation, HPEC 2022
[2] Shahrokhi et al, Book embeddings and crossing numbers, WG'94

Experimental results



Results for contracting an Ising model tensor network defined on a $5 \times 5 \times 5$ lattice

- Number on each point: maximum tensor train rank
- Achieve 5.9X speed-up relative to previous works to reach a relative error of 10^{-8}

[1] Chubb, General tensor network decoding of 2D Pauli codes, 2021

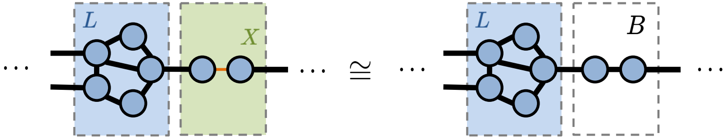
[2] Pan et al, Contracting arbitrary tensor networks: general approximate algorithm and applications in graphical models and quantum circuit simulations, PRL 2020

Efficient low-rank approximation for tensor network contraction

Goal: efficiently and accurately perform low-rank approximation in approximate tensor network contraction

Motivation for a new low-rank approximation subroutine

$$\min_{X,\mathrm{rank}(X)\le R} \left|\left| \; L \; X \; - \; L \; B \; \right|\right|_F$$



Accuracy: environment (L) typically comprises a small part of the whole tensor network[1,2]

- Small $L \to$ minimizes local rather than global error

Efficiency: Orthogonalization (via implicit QR factorization) on L is performed

- QR factorization can be expensive when L is not a tree

[1]Pan et al, Contracting arbitrary tensor networks: general approximate algorithm and applications in graphical models and quantum circuit simulations, PRL 2020

[2]Chubb, General tensor network decoding of 2D Pauli codes, 2021

Normal equations for low-rank approximation

$$X^* = \underset{X, \mathrm{rank}(X) \leq r}{\mathrm{argmin}} \|LX - LB\|_F$$

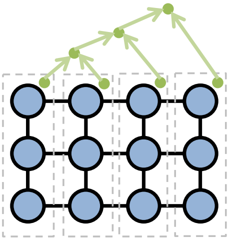
Orthogonalization-based: $Q_L, R_L \leftarrow \mathtt{QR}(L)$, then use the rank-$r$ approximation of $R_L B$ to update solution

Normal equations-based: compute the leading r eigenvectors of $B^T L^T LB$, and $X^* = BVV^T$

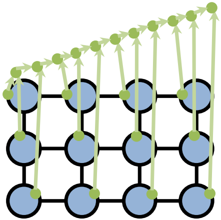
The asymptotic cost to form normal equations ($B^T L^T LB$) is upper-bounded by doing QR

Partitioned Contract: use partial contraction tree for flexible environment

Contraction tree over partitions



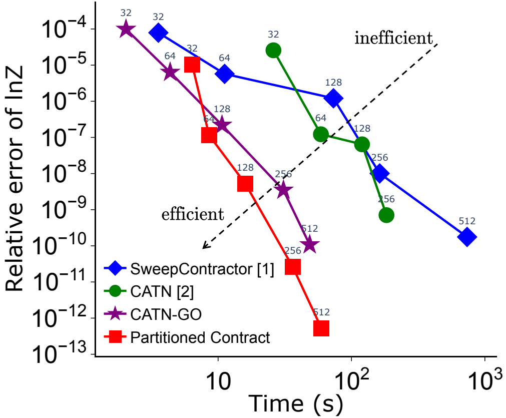
Complete contraction tree



Each contraction outputs a binary tree tensor network

- The input pair of partitions are considered the environment
- Larger partition implies larger environment $\rightarrow$ minimizes the global error

Experimental results



Results for contracting an Ising model tensor network defined on a $5 \times 5 \times 5$ lattice

- Number on each point: maximum tensor train rank
- Achieve 9.2X speed-up relative to previous works to reach a relative error of 10^{-9}

[1] Pan et al, Contracting arbitrary tensor networks: general approximate algorithm and applications in graphical models and quantum circuit simulations, PRL 2020

[2] Chubb, General tensor network decoding of 2D Pauli codes, 2021

Conclusion

Introduce efficient numerical algorithms for tensor decompositions and tensor networks

Applications include machine learning with large-scale datasets and simulation of large quantum circuits

Our contributions to tensor network libraries automate the development of fast algorithms

Future work

Tensor network sketching

- Generalize the analysis to other embeddings, such as Countsketch[1] and Tensorsketch[2]

Approximate tensor network contraction

- For CATN-GO: devise heuristics for finding vertex orderings with fewer edge crossings
- For Partitioned Contract: find efficient partial contraction trees

[1] Charikar et al, Finding frequent items in data streams, 2002
[2] Pham and Pagh, Fast and scalable polynomial kernels via explicit feature maps, KDD 2013

Backup slides

Experimental results

Vertex ordering	$8 \times 8 \times 8$ lattice			$(6, 300)$-rand regular graph		
	# crossings	Time (s)	GFlops	# crossings	Time (s)	GFlops
Baseline	34.6k	2.2k	9.4k	133k	10.8k	52k
Recursive bisection	16.8k	1.0k	4.6k	37.5k	2.8k	13.8k
Relative improvements	2.1X	2.2X	2.1X	3.5X	3.8X	3.8X

Vertex orderings with fewer edge crossings yield less contraction time

- Baseline: sequential traversal for lattice, and random ordering for a random graph
- Random regular graph has 300 vertices and degree 6

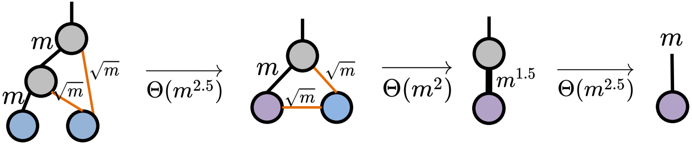
Analysis of the sketching algorithm

Lower bound analysis

- When the data contains 2 tensors, sketching lower bound can be derived
- Kronecker product case: when the data has two vectors with size m (sketch size), the sketching computational cost is $\Omega(m^{2.5})$
- When the data has more tensors, for a given contraction path the lower bound is the sum of two-tensor-contraction lower bounds

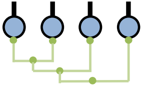
Algorithm design

- For the 2-tensor data, can design embedding attaining the lower bound



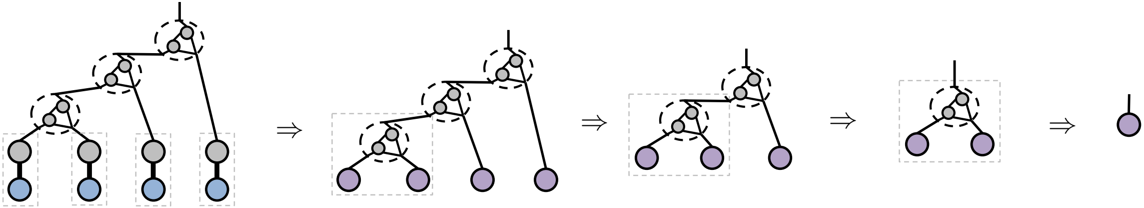
- For the data with more tensors, we can derive the optimal way to sketch using the two-tensor scheme for a given contraction path
- We can try all data contraction paths to get the optimal sketching path

Example: sketching Kronecker product data



Consider contracting an input Kronecker product from left to the right

Sketching contraction path as follows



Our algorithm reduces cost by up to $O(\sqrt{m})$ for the same accuracy compared to using tree embeddings[1]

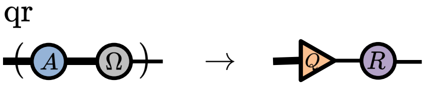
[1] Ahle et al, Oblivious sketching of high-degree polynomial kernels, SODA 2020

Randomized SVD using sketching

Given a matrix $A \in \mathbb{R}^{m \times n}$, find a rank-$r$ approximation with $r \ll m, n$ in the SVD form

Randomized range finder[1]

- Generate a random embedding matrix $\Omega \in \mathbb{R}^{n \times \Theta(r)}$
- $Q, R \leftarrow \mathrm{qr}(A\Omega)$, so $Q \in \mathbb{R}^{m \times \Theta(r)}$

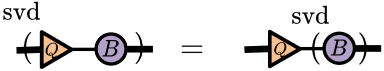


Dimensionality reduction

- $B \leftarrow Q^T A$

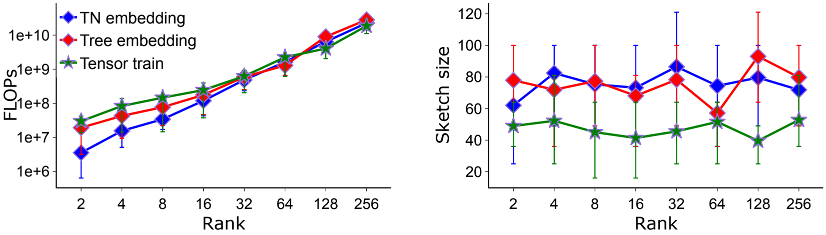
SVD on the low-rank matrix QB

- $Q_B, \Sigma, V_B^T \leftarrow \mathrm{svd}(B)$
- Return QQ_B, Σ, V_B^T



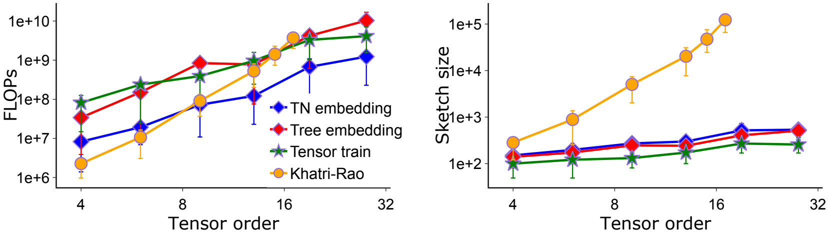
[1] Nathan, Martinsson, and Tropp, Finding structure with randomness, SIAM review 2011

Experiments: sketching a MPS data



- Input MPS: order 6, each dimension size $s = 500$ with varying MPS rank
- TN embedding: Kronecker product + a binary tree of small networks
- Tree embedding: Kronecker product + a binary tree tensor network
- Sketching error is within 0.1
- Our TN embedding achieves the best asymptotic cost for all MPS ranks

Experiments: sketching a Kronecker product data



- Input data: each dimension size $s = 1000$ with varying number of orders
- Sketching error is within 0.1
- Our TN embedding achieves the best asymptotic cost
- TN, tree, and MPS embeddings have efficient sketch size