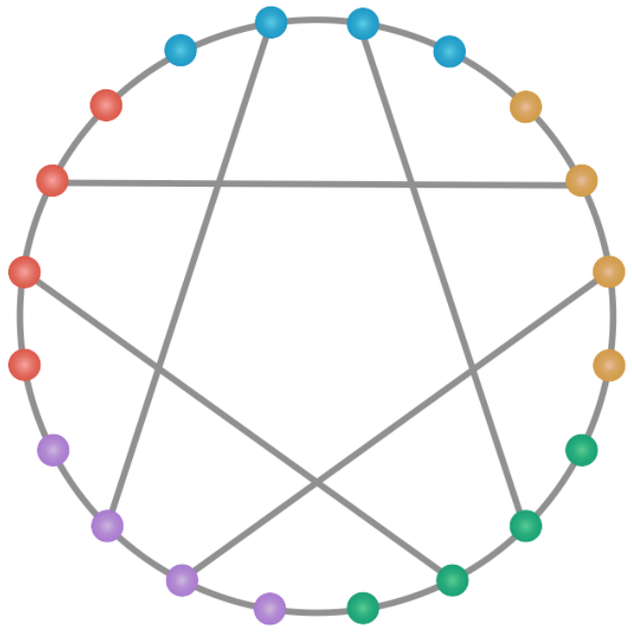
CATN.jl

Contrating Arbitrary Tensor Networks using Julia



Pan Zhang
ITP,CAS

JuliaCN 2020

Outline

Introductions to tensor networks

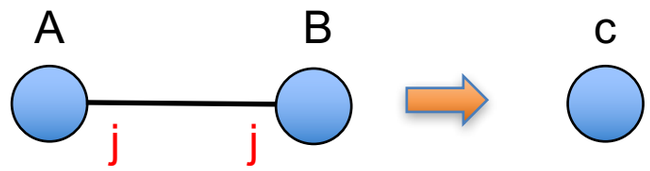
- Tensor diagrams
- Canonical Polyadic, Tucker, Matrix Product States

Tensor network contractions

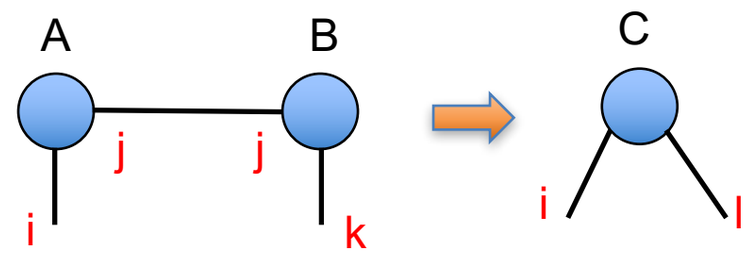
- Contraction order
- Low-dimensional approximation

Einsum notations

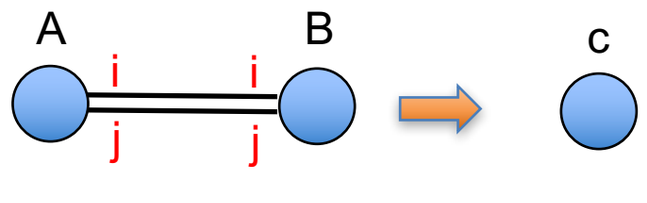
c = einsum(A,B,"j,j") $c = \sum_j A_j B_j$



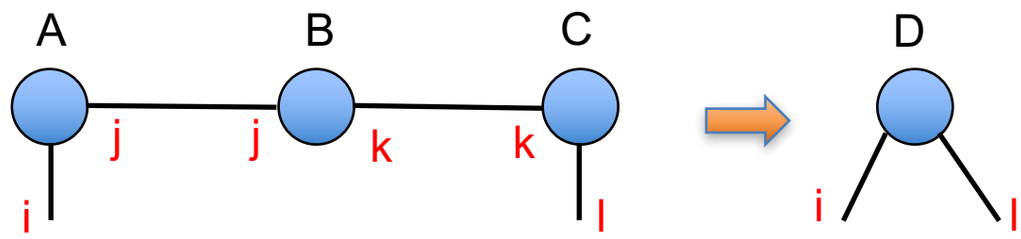
C = einsum(A,B,"ij,jk->ik") $C_{ij} = \sum_j A_{ij} B_{jk}$



c = einsum(A,B,"ij,ij") $c = \mathrm{Tr}(AB) = \sum_i \sum_j A_{ij} B_{ij}$

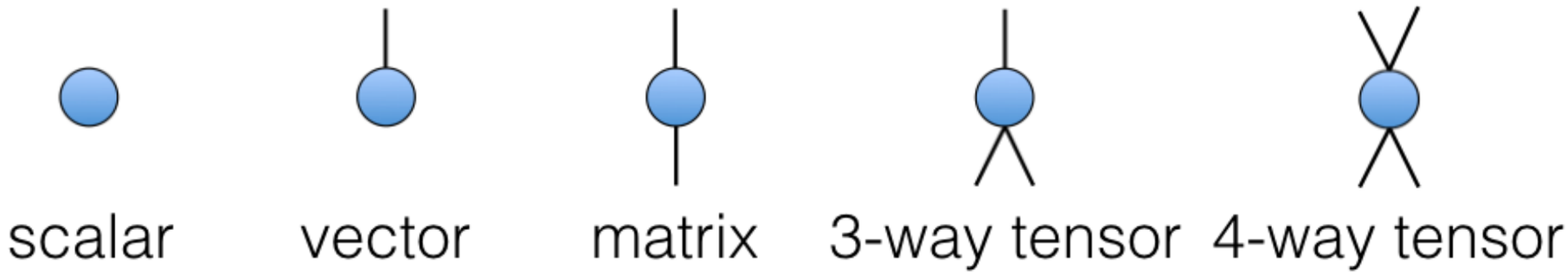


D = einsum(A,B,C,"ij,jk,kl")



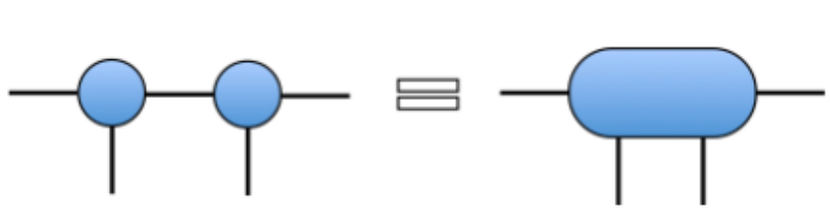
Computational complexity: product of dimensions of all unique indices

Diagram notations

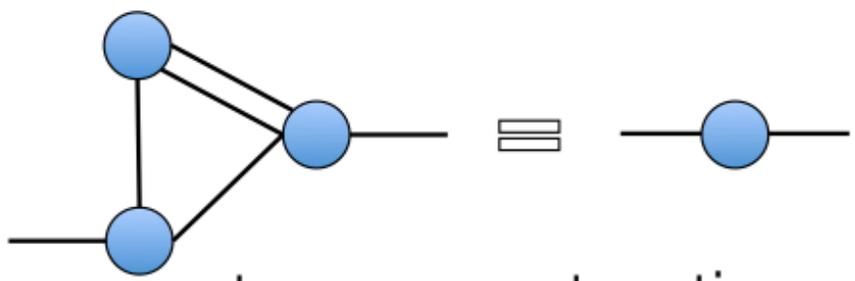


tensor product

Trace

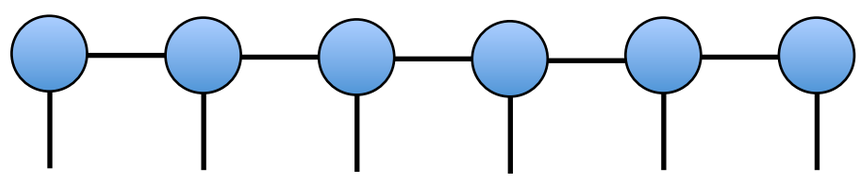


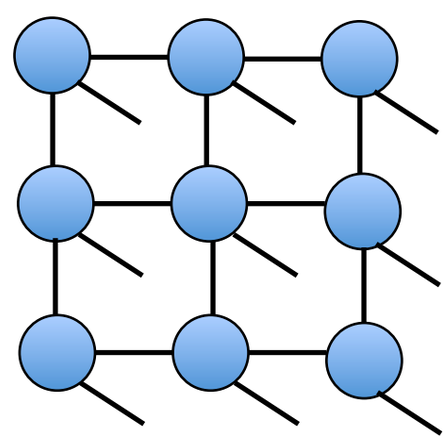
tensor contraction

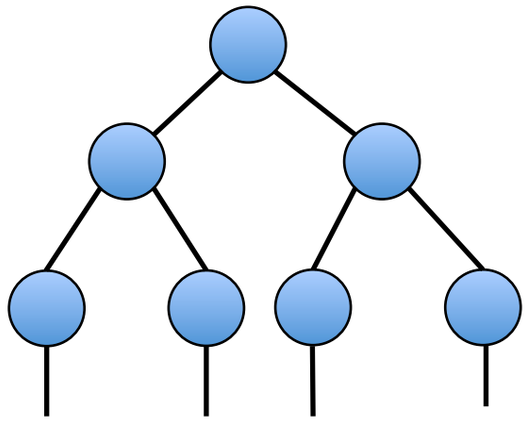


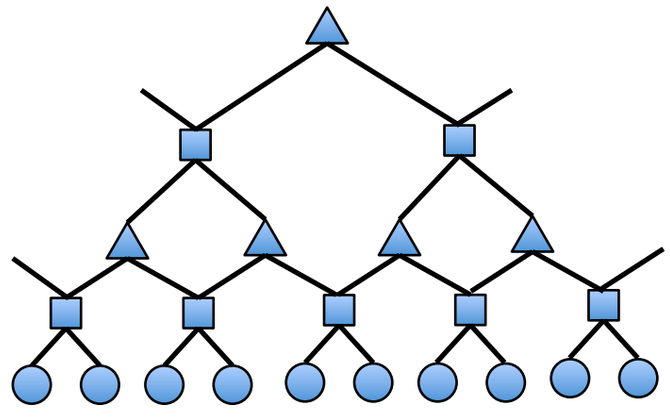
tensor contraction

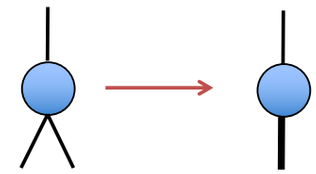
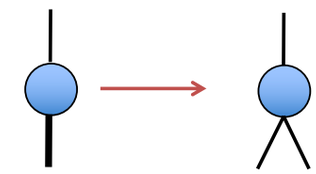
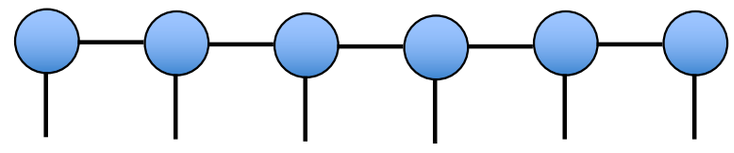
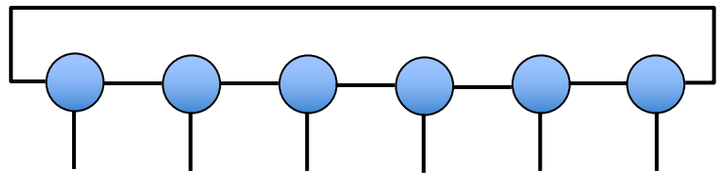
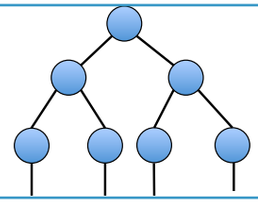
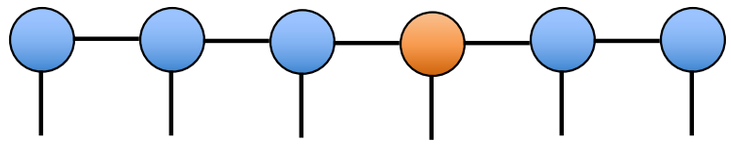
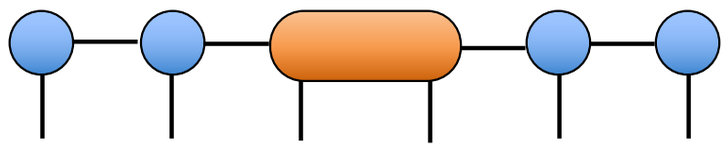
Tensor networks in physics:
imposing prior of physical wave function



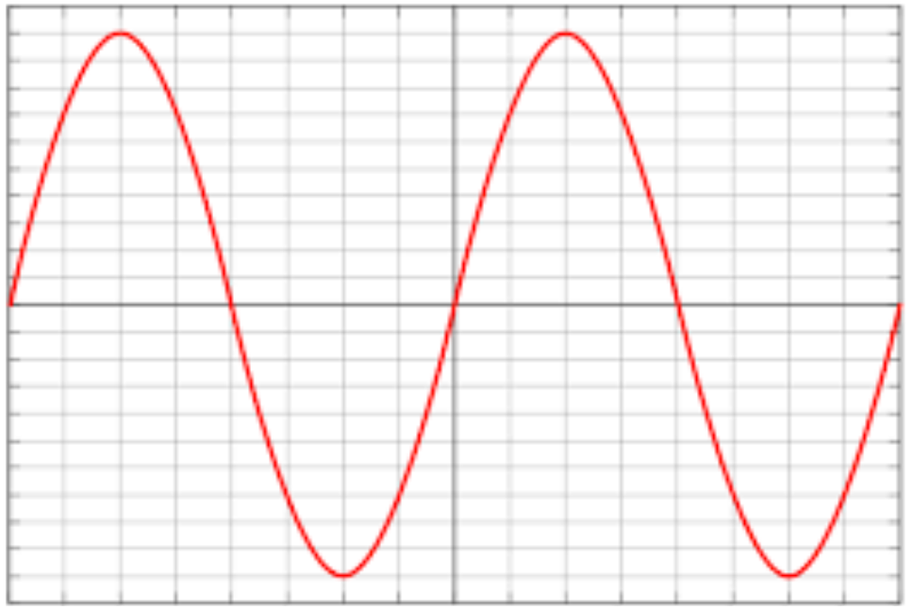






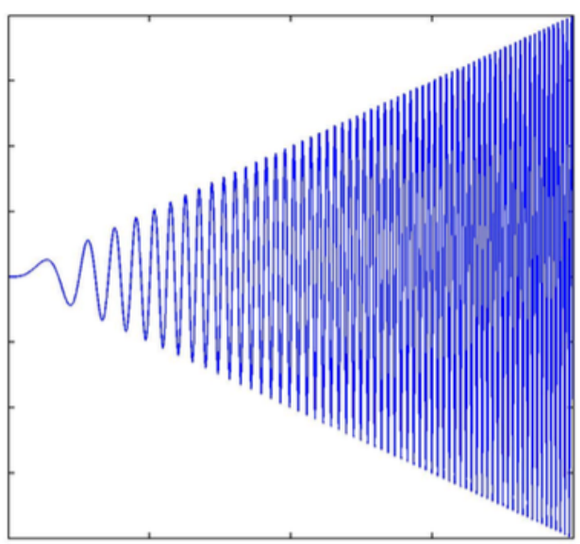
In Physics	Out of Physics	Diagram
grouping of indices	unfolding, matricization	
splitting of indices	tensorizing	
matrix product states	tensor train decomposition	
periodic boundary MPS	tensor chain decomposition	
tree tensor networks	hierarchical Tucker decompostion	
single-site DMRG	alternating least square	
two-site DMRG	modified alternating least square	

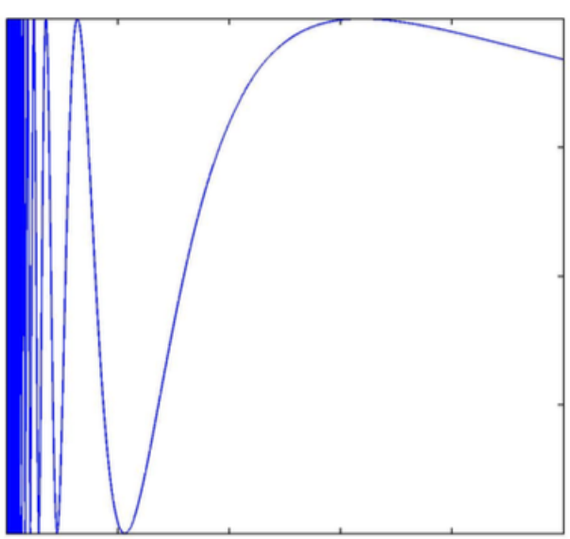
Exploring internal structures in the data

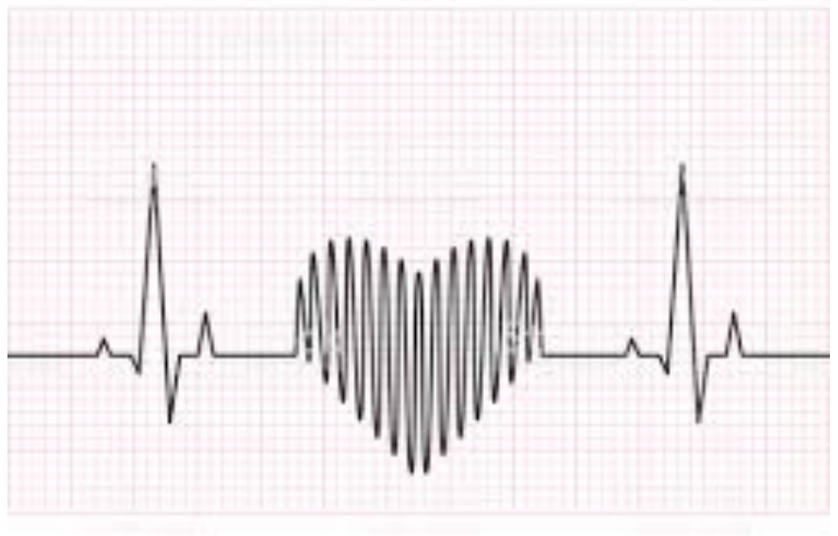


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
.....
.....
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

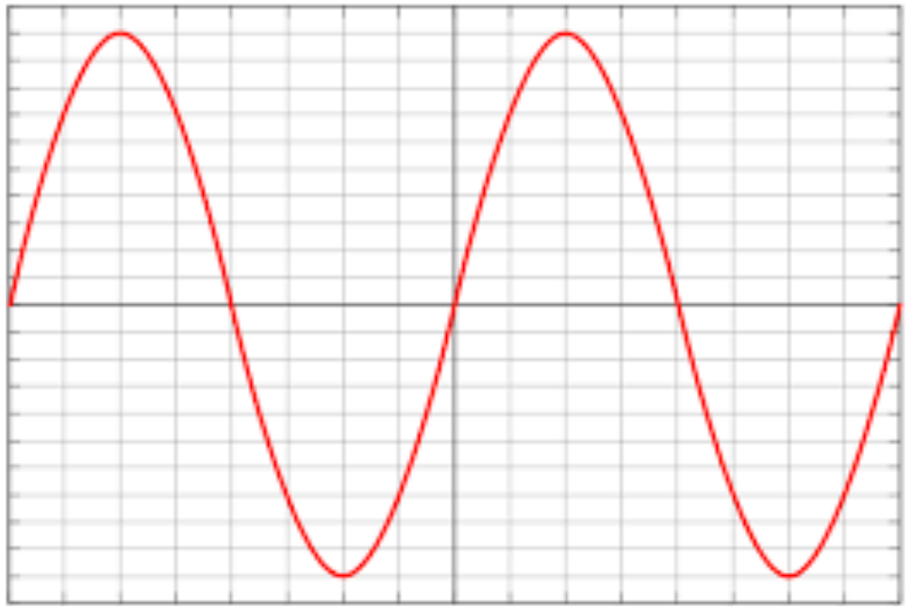
10^6 data points in a vector





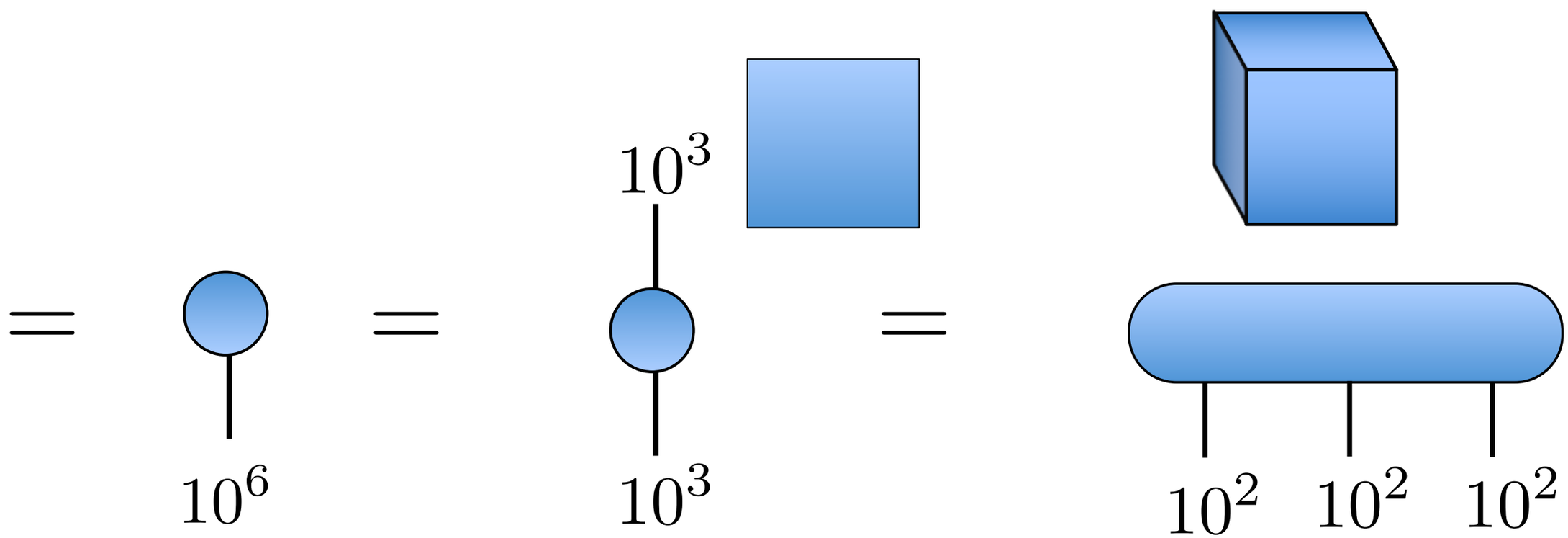


Exploring internal structures in the data

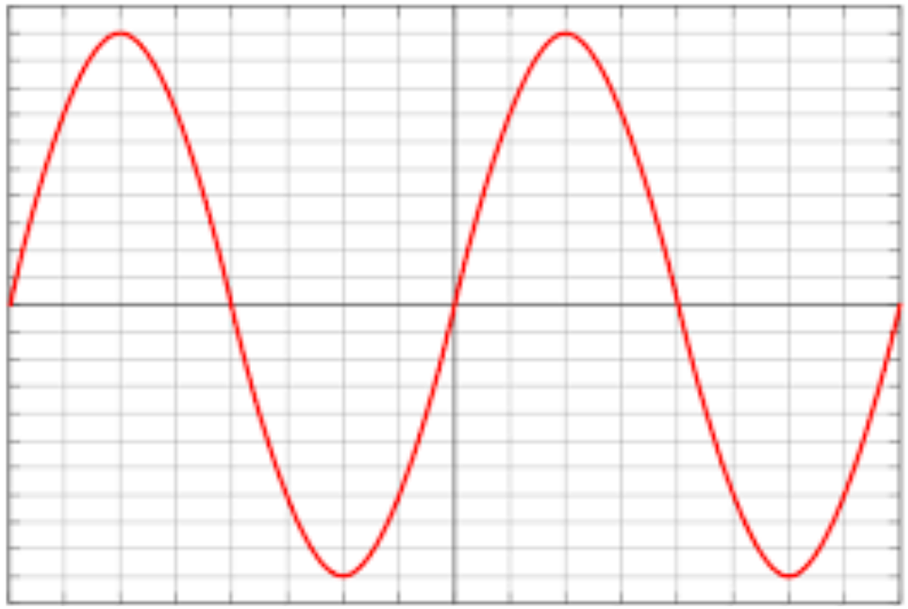


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
.....
.....
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector

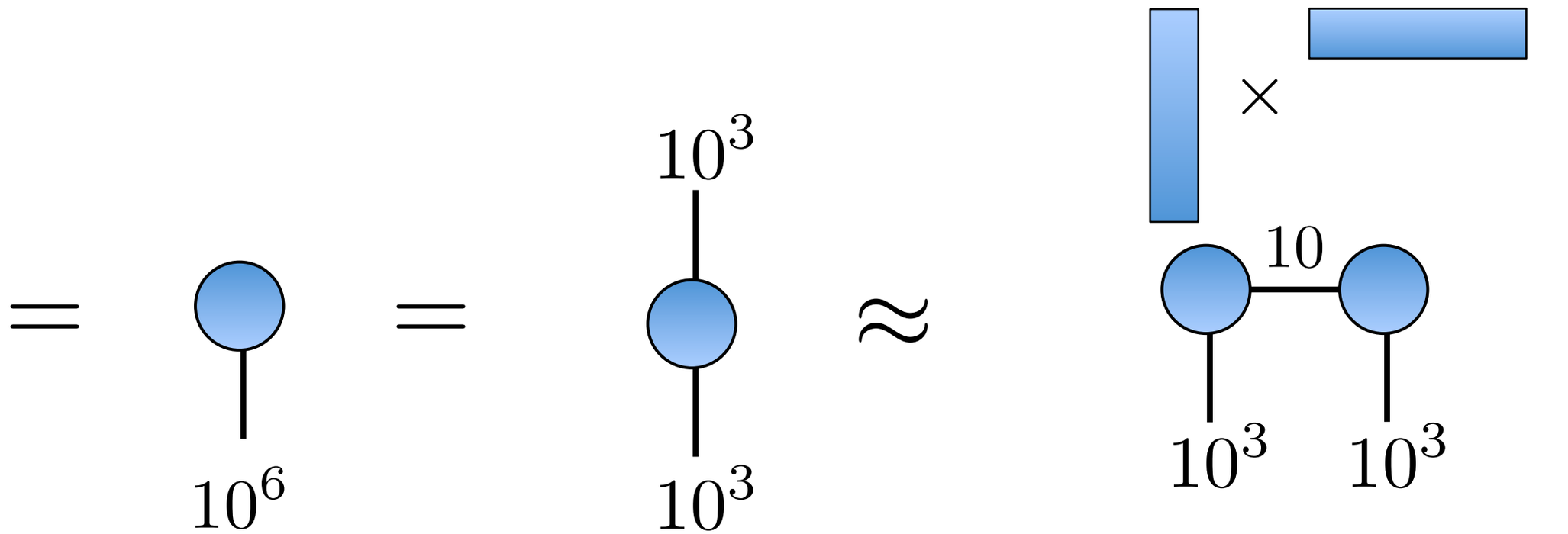


Exploring internal structures in the data



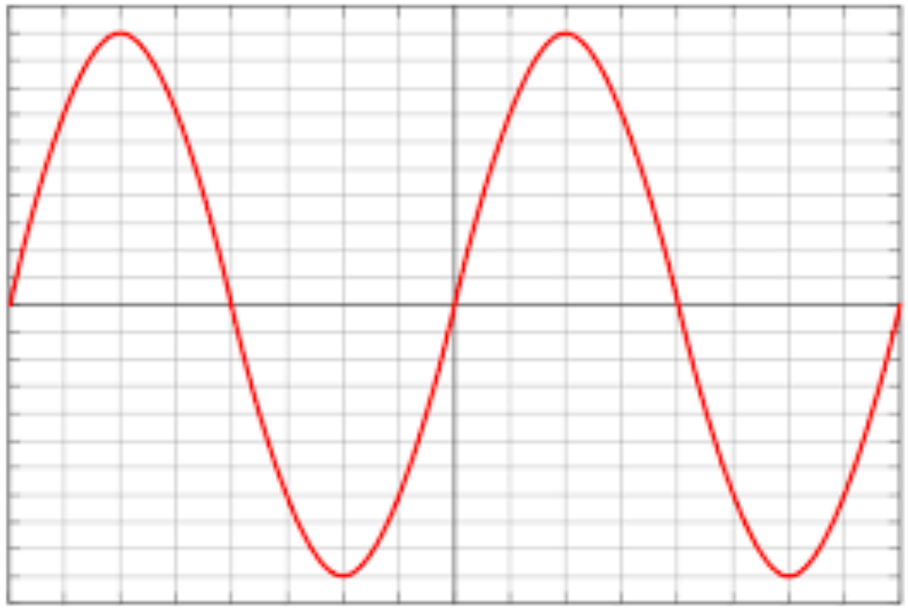
[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
.....
.....
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector



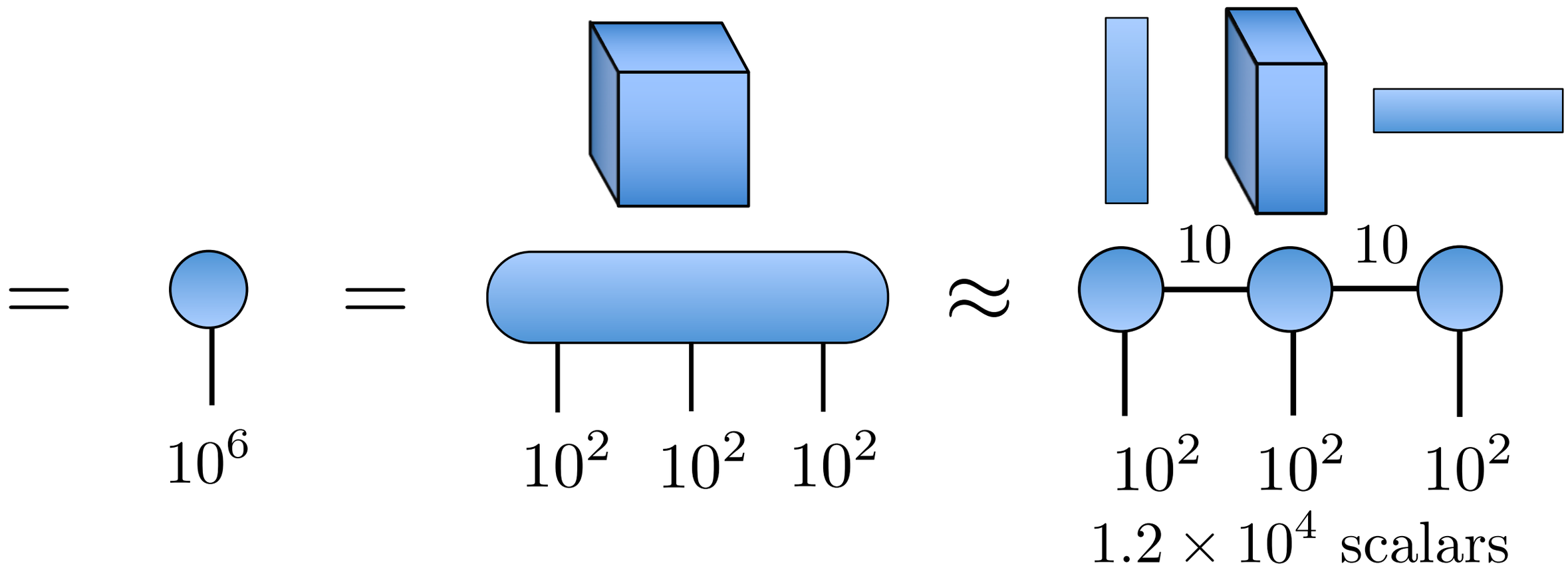
2×10^4 scalars

Exploring internal structures in the data



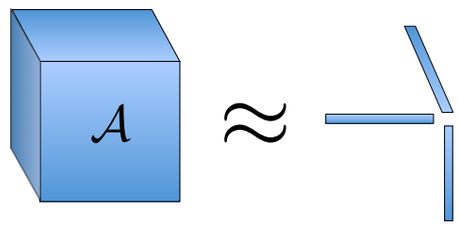
[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
.....
.....
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector



Representation of a large tensor: rank-one

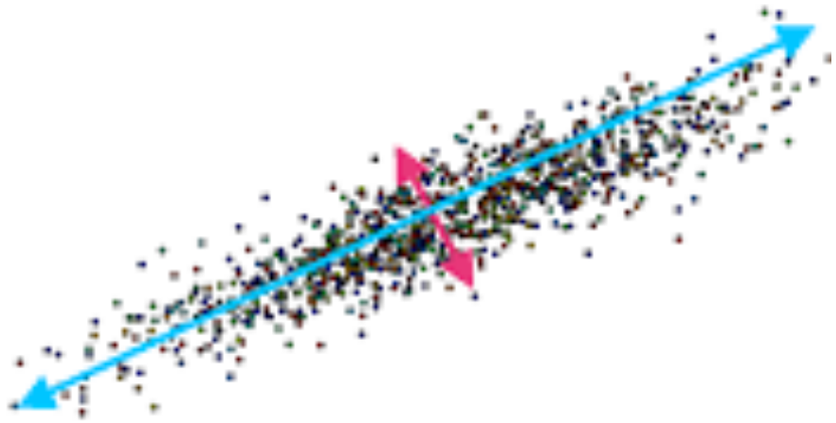
$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \mathbf{a}^{(n)}$$



independent modes

factorized pure state

principled component

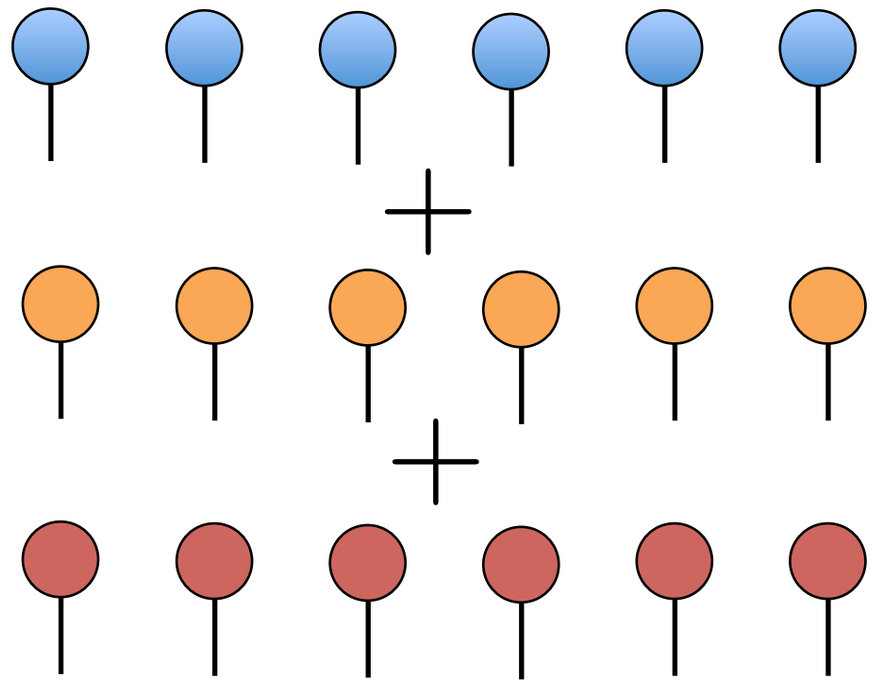


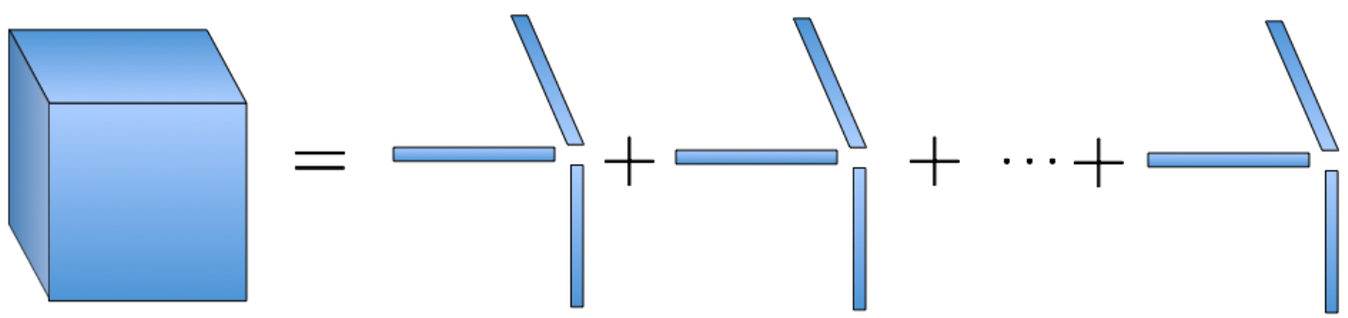
reminiscent of the *mean-field* approximation

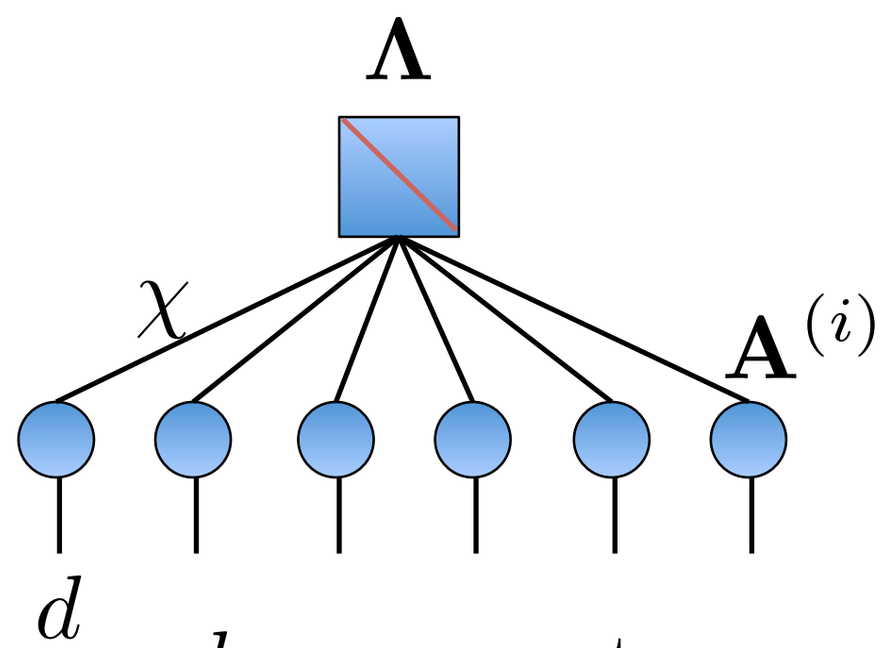
$$p(\{x_1, x_2, ..., x_n\}) = p(x_1)p(x_2)...p(x_n)$$

Canonical Polyadic (CP) decomposition

$$\mathcal{A} = \sum_{r=1}^{\chi} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(n)}$$
$$= \mathbf{\Lambda} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_n \mathbf{A}^{(\mathbf{n})}$$
$$= [\mathbf{\Lambda}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(\mathbf{n})}]$$

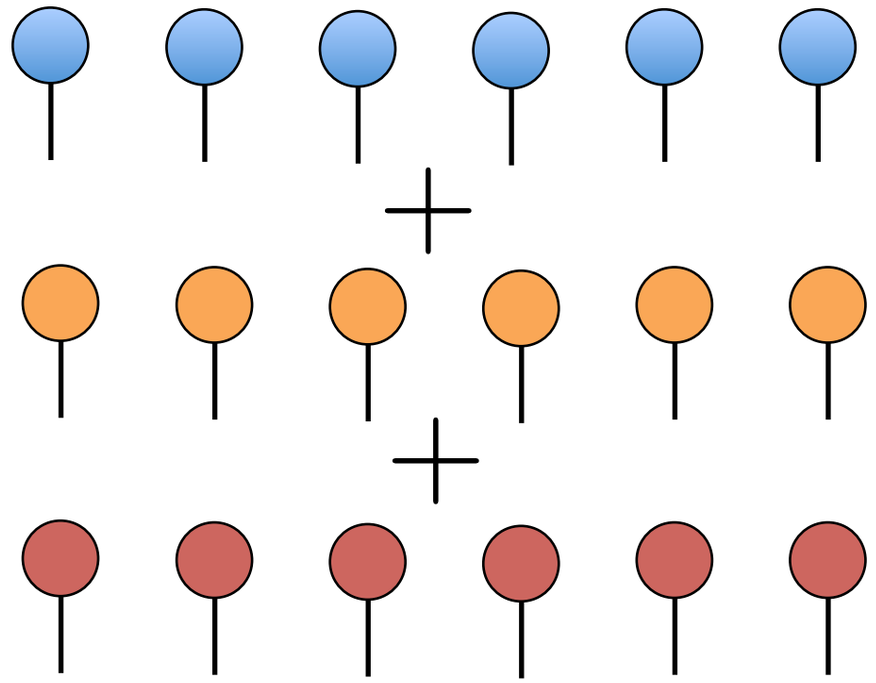






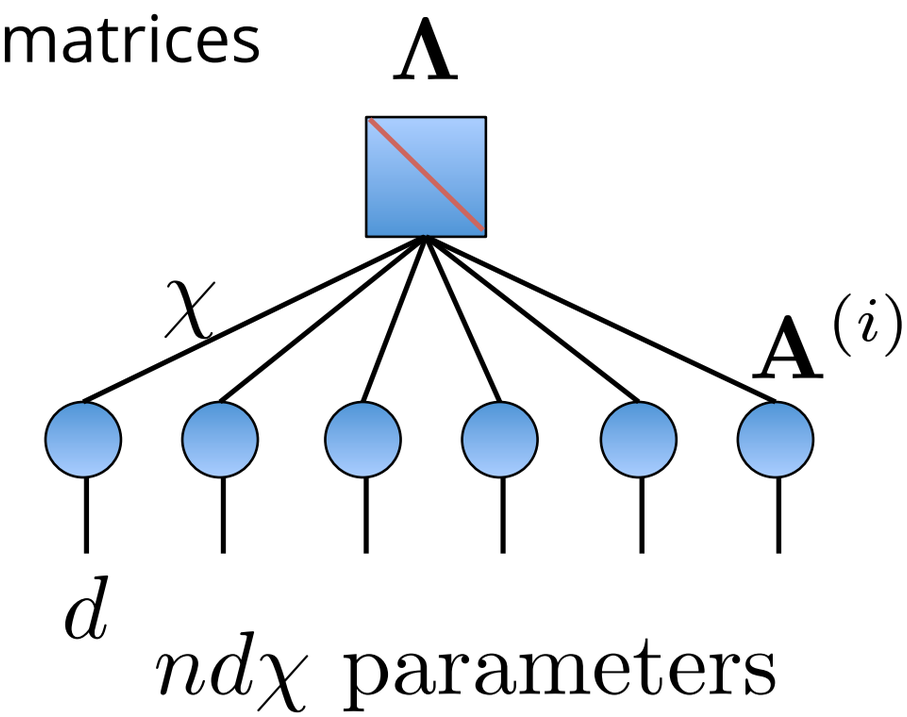
Canonical Polyadic (CP) decomposition

$$\begin{aligned}\mathcal{A} &= \sum_{r=1}^{\chi} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(n)} \\ &= \mathbf{\Lambda} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_n \mathbf{A}^{(\mathbf{n})} \\ &= [\mathbf{\Lambda}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(\mathbf{n})}]\end{aligned}$$



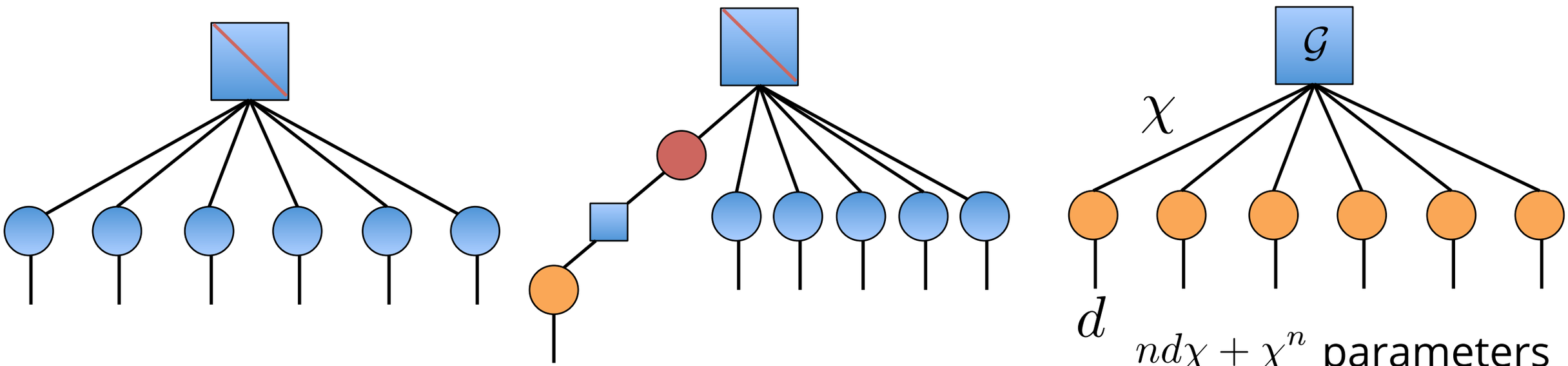
Convenient representations of CP using unfolded matrices

$$\mathbf{A}_{(i)} = \mathbf{A}^{(i)} \mathbf{\Lambda} \left(\mathbf{A}^{(1)} \odot \mathbf{A}^{(2)} \odot \cdots \odot \mathbf{A}^{(n)} \right)$$



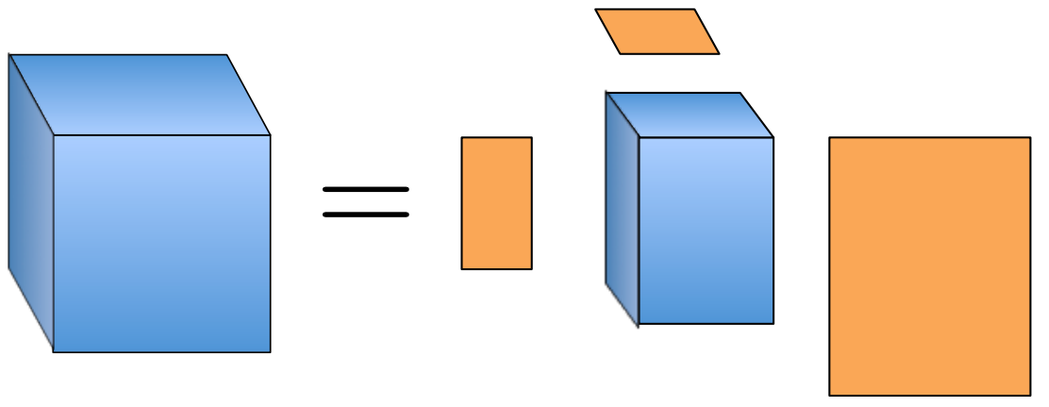
From CP to Tucker

The CP formats is lack of canonical forms and orthogonalization
by imposing orthogonalization, the core tensor is no more diagonal



this leads to the *Tucker Decomposition*

$$\mathcal{A} = [\mathcal{G}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(\mathbf{n})}]$$
$$= \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_n \mathbf{A}^{(n)}$$



First introduced by Tucker in *1963,* known as *N-mode PCA/SVD/Factor analysis.*
Usually treated as a multilinear extension of PCA

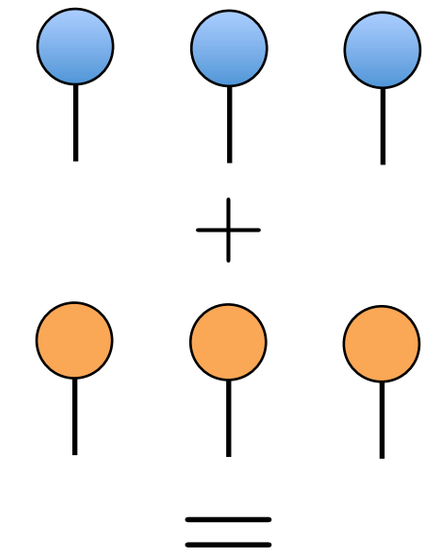
From CP to MPS

Summing two rank-1 tensors

$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \mathbf{a}^{(3)}$$

$$\mathcal{B} = \mathbf{b}^{(1)} \circ \mathbf{b}^{(2)} \circ \mathbf{b}^{(3)}$$

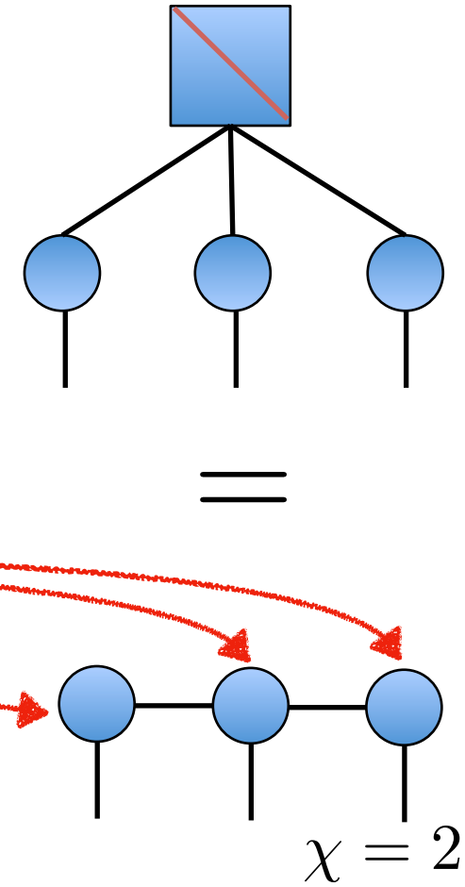
$$\mathcal{C} = \mathcal{A} + \mathcal{B}$$



$$c_{s_1,s_2,s_3} = a^{(1)}_{s_1} a^{(2)}_{s_2} a^{(3)}_{s_3} + b^{(1)}_{s_1} b^{(2)}_{s_2} b^{(3)}_{s_3}$$

$$= \begin{pmatrix} a^{(1)}_{s_1} & b^{(1)}_{s_1} \end{pmatrix} \begin{pmatrix} a^{(2)}_{s_2} & 0 \\ 0 & b^{(2)}_{s_2} \end{pmatrix} \begin{pmatrix} a^{(3)}_{s_3} \\ b^{(3)}_{s_3} \end{pmatrix}$$

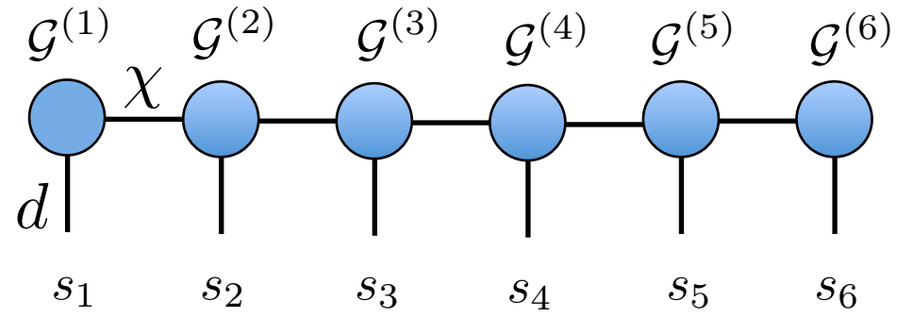
$$\mathbf{C}^{(1)}(s_1,:) \quad \mathcal{C}^{(2)}(:,s_2,:) \quad \mathbf{C}^{(3)}(:,s_3)$$



Matrix Product States and Matrix Product Operator

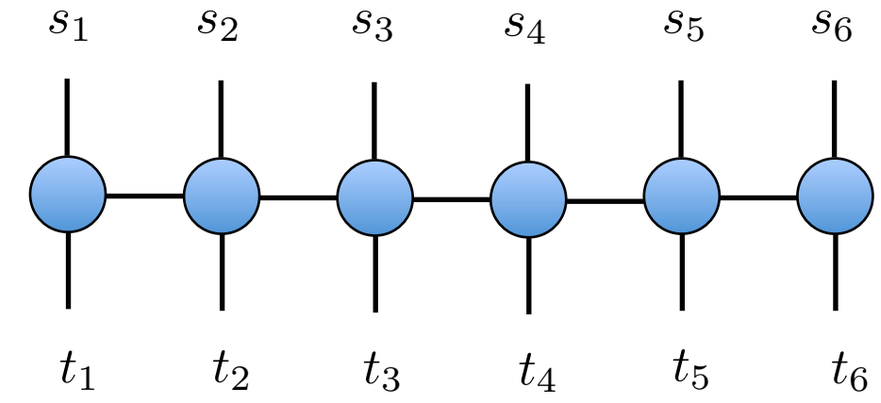
$$\mathcal{A} = \langle \mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \cdots, \mathcal{G}^{(n)} \rangle$$
$$= \mathcal{G}^{(1)} \times_1 \mathcal{G}^{(2)} \times_1 \cdots \times_1 \mathcal{G}^{(n)}$$

$$a_{s_1,s_2,\cdots,s_n} = \mathbf{G}^{(1)}_{s_1} \mathbf{G}^{(2)}_{s_2} \cdots \mathbf{G}^{(n)}_{s_n}$$



$$\approx nd\chi^2 \text{ parameters}$$

$$a_{s_1,s_2,\cdots,s_n} = \mathbf{G}^{(1)}_{s_1,t_1} \mathbf{G}^{(2)}_{s_2,t_2} \cdots \mathbf{G}^{(n)}_{s_n,t_n}$$

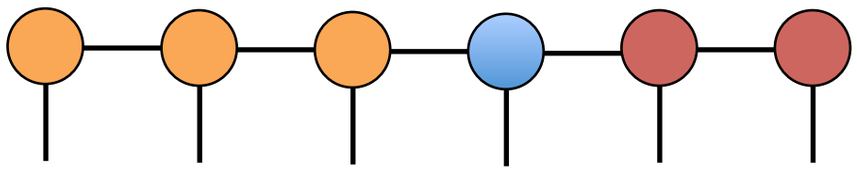


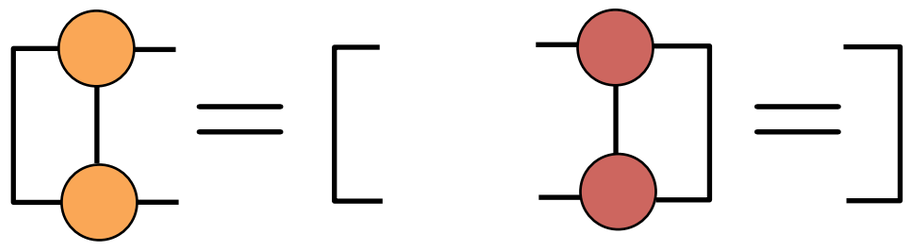
with χ=1, MPS is a rank-one tensor

besides CP, MPS is another generalization of rank-one tensors

Canonical forms of MPS

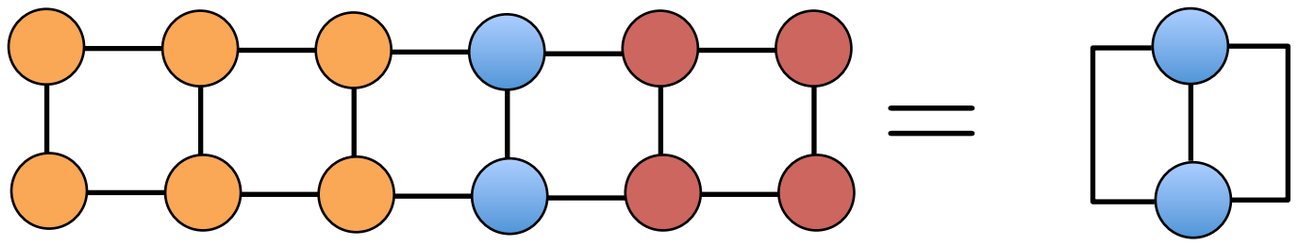
Analogous to the Tucker decomposition and HOSVDs, MPS has the benefits of orthogonality.

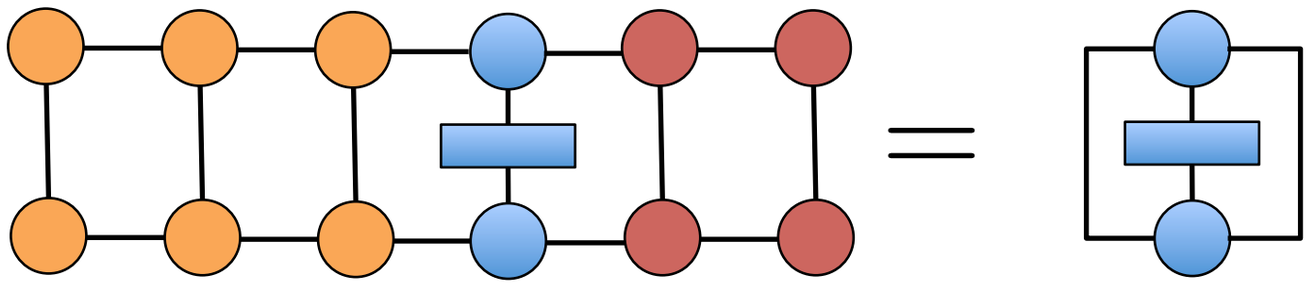


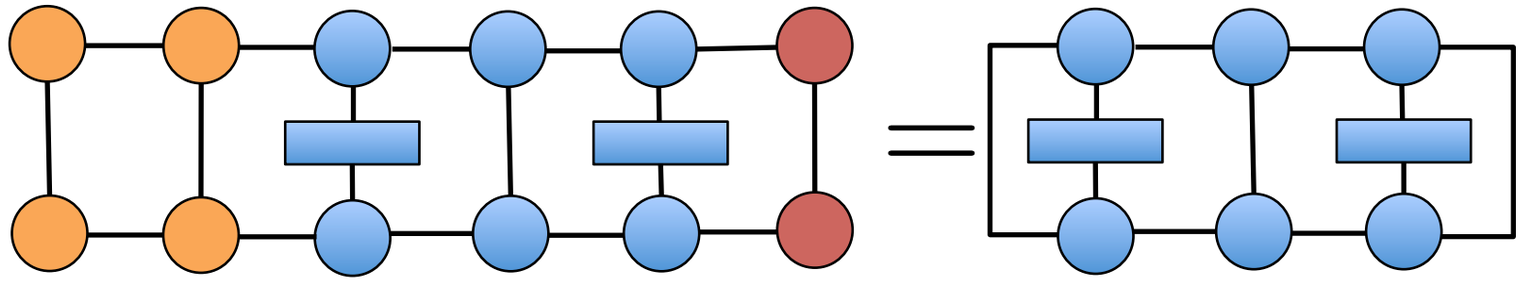


Benefits

- Fixed gauge, no ambiguity
- Easy norm computation
- Easy expectation/ correlation computation
- Always good conditioned







Outline

Introductions to tensor networks

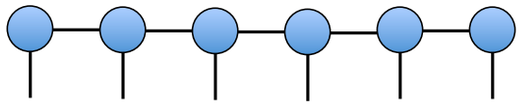
- Tensor diagrams
- Canonical Polyadic, Tucker, Matrix Product States

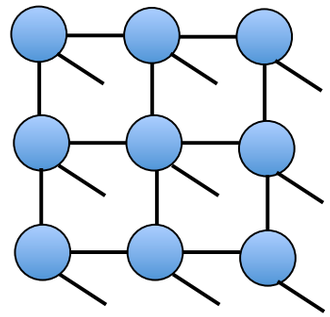
Tensor network contractions

- Contraction order
- Low-dimensional approximation

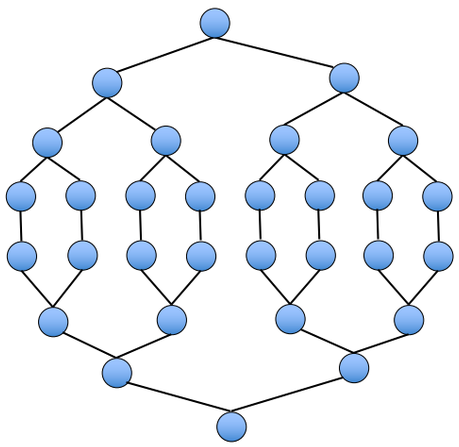
Contracting tensor networks corresponds to

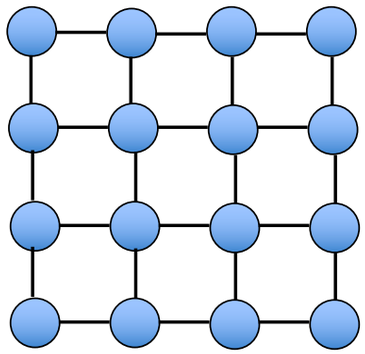
- Evaluating a wave function



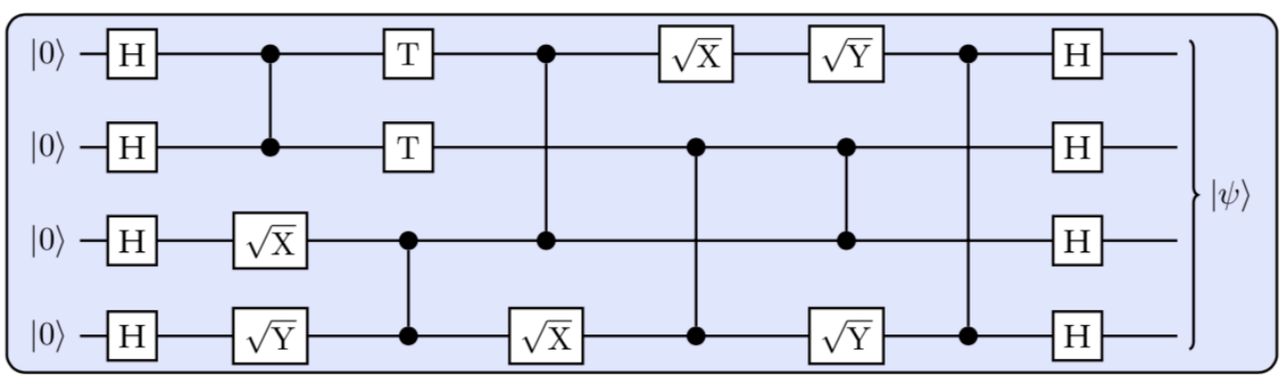


- Marginalizing a graphical model



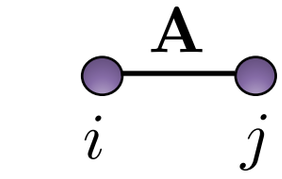


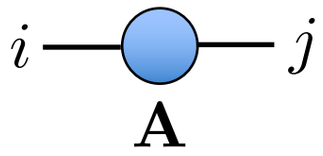
- Simulating a quantum circuit



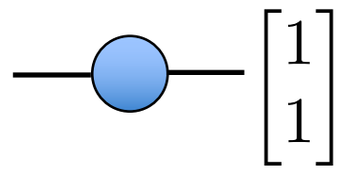
Robeva et al, arXiv:1710.01437
Boixo et al, arXiv:1712.05384
Chen et al, arXiv:1805.01450

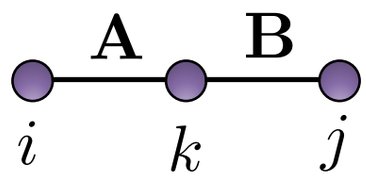
Probability distribution is a tensor, hence can be represented by a tensor network in general.

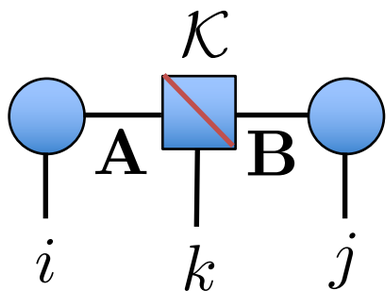




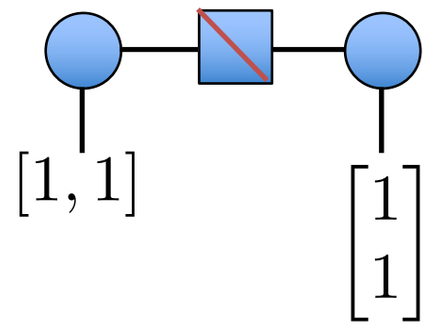
$$Z = \sum_{i,j} A_{ij} = \mathbf{1}^T A \mathbf{1} =$$





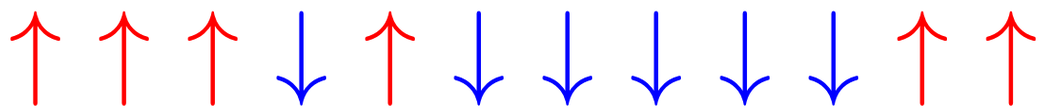


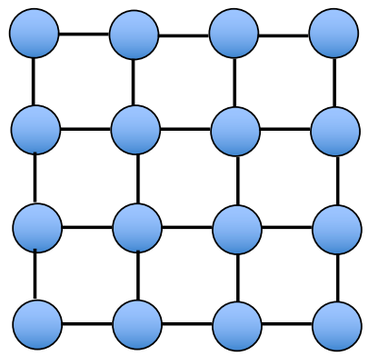
$$Z = \mathbf{1}^{\mathrm{T}} \mathbf{A} \mathbf{B} \mathbf{1} =$$



Example: Ising model

$$\mathbf{S} = \{+1, -1\}^n$$



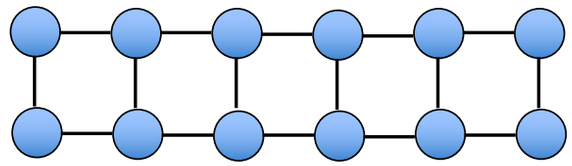


computing partition function == contracting a tensor network

$$P(\mathbf{S}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{S})} \qquad Z = \sum_{\mathbf{S}} \mathrm{e}^{-\beta E(\mathbf{S})}$$

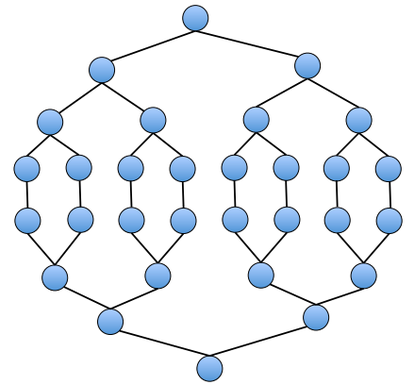
For regular tensor networks

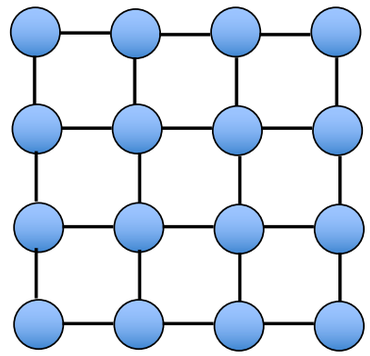
Exact for 1D and Tree TN



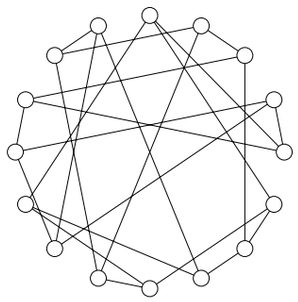
Challenging for 2D TN

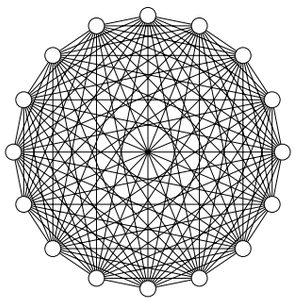
- Boundary MPS
- Tensor Renormalization Group
- High-Order TRG
- Corner Transfer Matrix RG
- Loop TNR

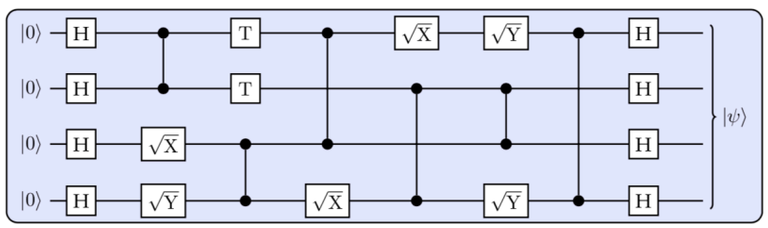




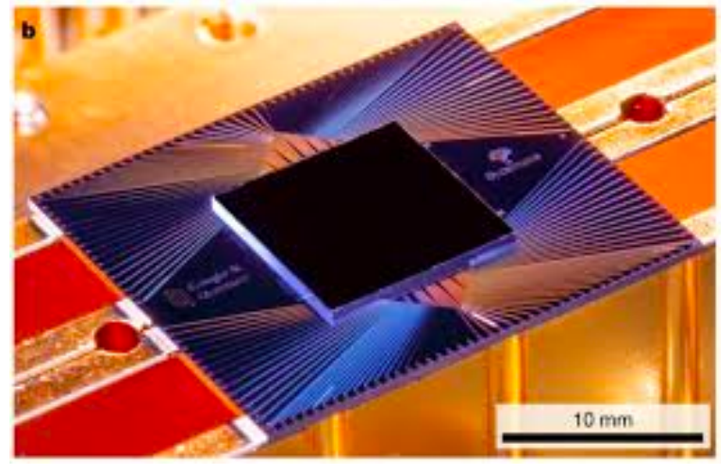
Very challenging for
Contracting Arbitrary TN







Specifical Task: Quantum Supremacy

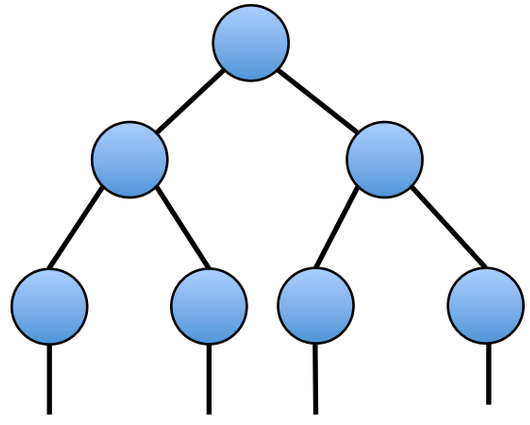


- Google's original estimate [**Arute et. al. 2019**]
  - 10,000 years for 20 cycles / 1.1 year for 14 cycles
- IBM's estimate [**Pednault et al 2019**]:
  - 250PB memory using all hard disks of the Summit
- Cotengra [**Gray/Kourtis 2020**]
  - Balanced Partitioning (Kahypar) + Greedy/optimal + Slicing
  - 1.4 hour for single amplitude of 14 cycles / 3000 years for 20 cycles (Single GPU)
- Ali's simulator [**Huang et. al. 2020**]
  - Hierarchical partitioning (Kahypar) + Greedy/optimal + Slicing + rejection sampling
  - 243 Second for single/batch amplitude of 14 cycles (Single GPU)
  - 833 Second for single/batch amplitude of 20 cycles (using more than 20,000 GPUs)

Challenges in TN contractions

Choosing a contraction order

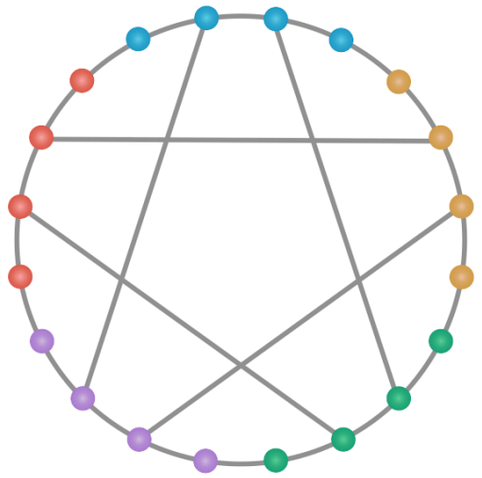
- NP problem
- Heuristics to find a good order



Markov/Shi SIAM JC, 38 963 (2008)
Boixo et al, arXiv:1712.05384
Gray/Koutis, arXiv:2002.01935
Huang et al, arXiv:2005.06787

Large intermediate tensors

- Compress the intermediate tensors
- Efficient approximations



Phys. Rev. Lett. 125 060503 (2020)

Contraction Order

Markov/Shi, SIAM Journal on Computing, 38 963 (2008)

- Tensor network contraction -> quantum circuit simulation

Boixo et al, arXiv:1712.05384
Chen et al, arXiv:1805.01450

- TN as a graphical model
- Employing QuickBB for finding a order

Koutis et al, SciPostPhys.7.5.060 (2019)
Gray/Koutis, arXiv:2002.01935
Huang et al, arXiv:2005.06787

- Using contraction tree
- Partitioning + optimal/greedy order

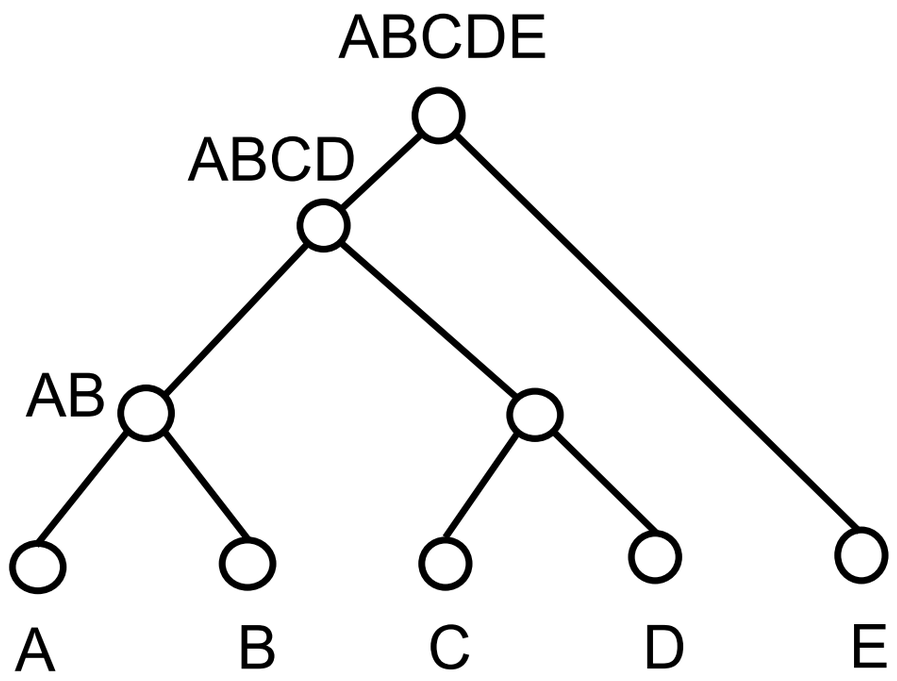
Contraction Tree

Contacion order: {{ {AB} {CD} } E}

#orders >= #contraction trees

Space complexity = Maximum size of tree node

Time complexity = summation of product of edge dimensions for each node

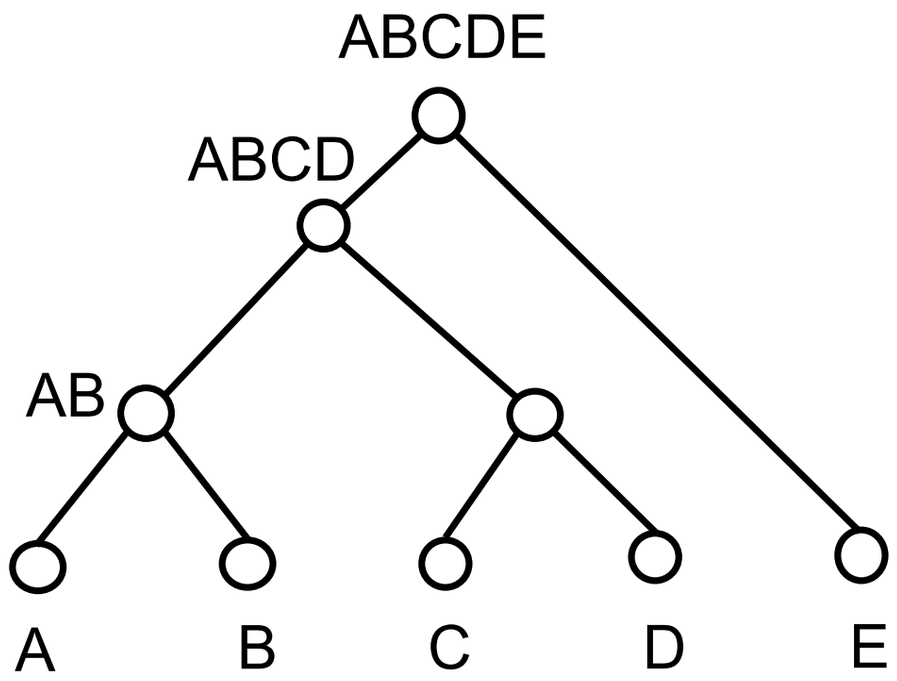


Exact algorithms for finding optimal solutions

Depth first search of orders {{ {AB} {CD} } E} + bounding.

Top-bottom depth first search of all contraction trees + bounding.

Breathe first search of orders (easier to apply bounding).



Heuristic algorithms

Any-time Branch-and-Bound

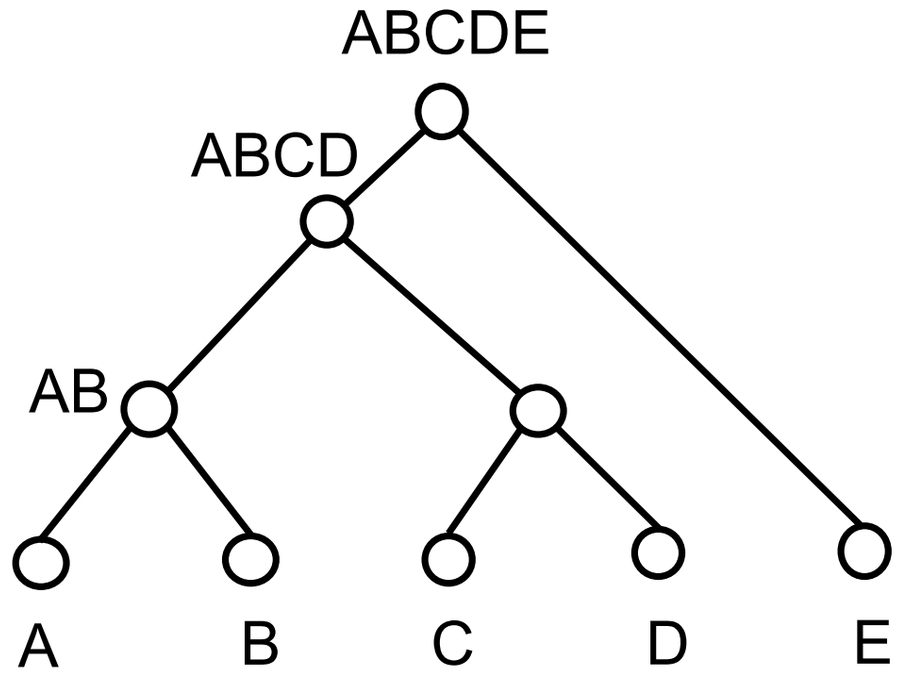
- QuickBB

Greedy algorithms

- Minimizing size of intermediate tensors
- Minimizing amount of reductions

Partitioning based algorithms

- Koutis et al, SciPostPhys.7.5.060: community detections for top nodes in the contraction tree, greedy for bottom nodes.
- Cotengra: Kahypar for top nodes in the contraction tree, greedy for bottom nodes.
- Huang et al, arXiv:2005.06787: Kahypar for hierarchically determining top nodes in the contraction tree



Basic data structures for contraction scheme of CATN.jl

```
struct OpenGraph
    n::Any # number of variables
    neis::Array{Set{Any},1} # neighbors, list of sets
    A::SparseMatrixCSC{Int,Int} # Weighted adjacency matrix,
    log2dim::SparseMatrixCSC{Float64,Int} # Weighted adjacenc
    node_log2dim::Array{Float64,1} # log dimension of each
    node_outlog2dim::Array{Float64,1} # log dimension of eac
end
```

```
mutable struct CTree_Node{}
    """

    Node struct of contraction tree

    """
    gid::Int # id in the graph
    father::Set{Any} # Set([1,2,3,4,5])
    left::Set{Any} # Set([1,2])
    right::Set{Any} # Set([3,5])
    sc::Float64 # size of the node
    tc::Float64 # time complexity
end
```

```
mutable struct Contraction_Tree{}
    """

    Contraction tree struct

    It stores time and space complexity,
    """
    nodes::Dict{Any,Any} # map node_id (S
    root::Set{Any}
    complex::Array{Any} # [space_complexi
end
```

Approximating intermediate tensors in CATN.jl

How to deal with large intermediate tensors during the contraction process?

- Stored and compressed using MPS

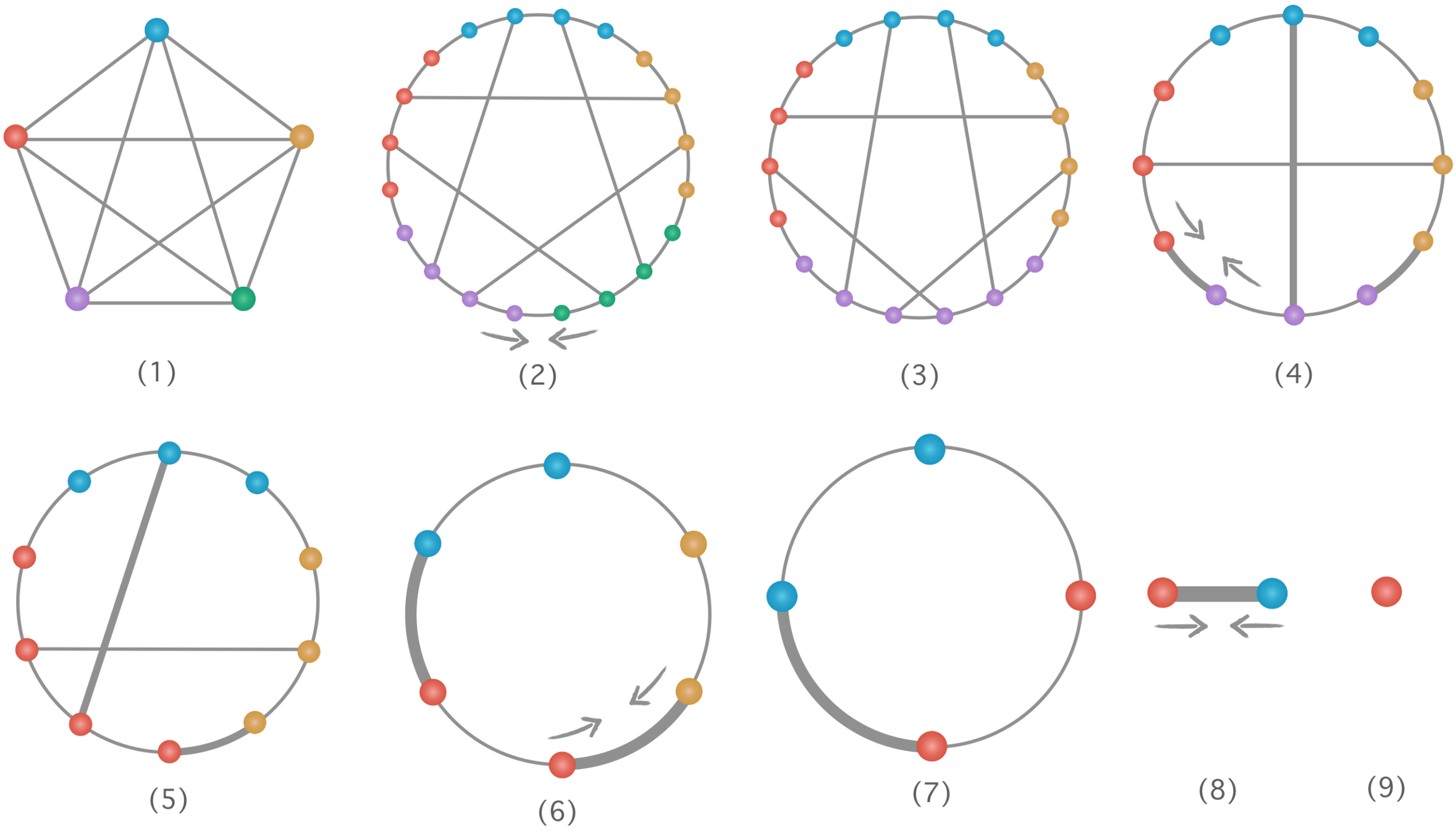
How to do accurate approximations?

- Canonical forms and SVD

Scalability of approximations?

- Trade off between accuracy and bond dimension

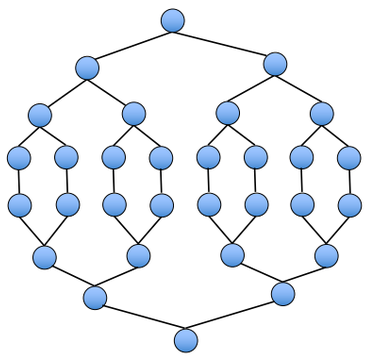
CATN: Contracting Arbitrary Tensor Networks

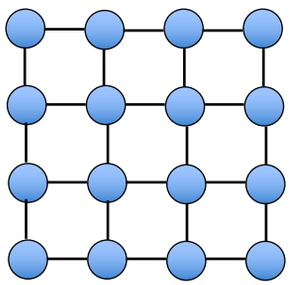


Phys. Rev. Lett. 125 060503 (2020)

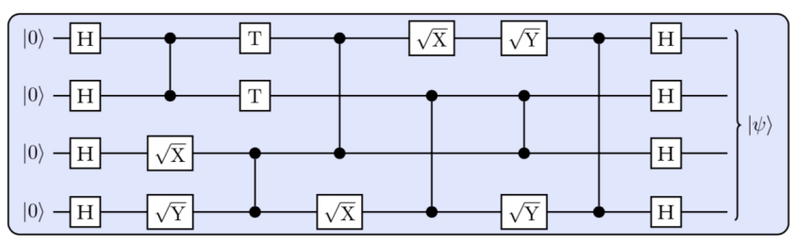
Take-home message

Tensor network contractions are useful





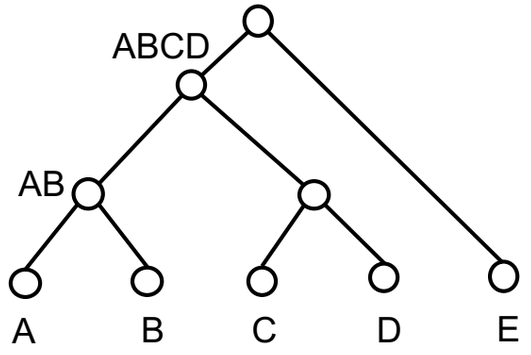
- Evaluating a wave function
- Marginalizing a graphical model
- Simulating a quantum circuit
- ...

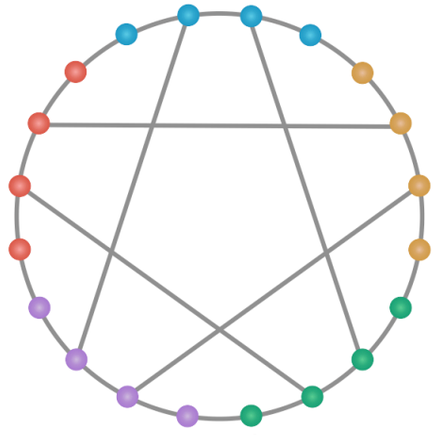


ABCDE

Two key parts of TN contraction

- Contraction order
- Low-dimensional approximations





CATN.jl will be public soon!

Phys. Rev. Lett. 125 060503 (2020)