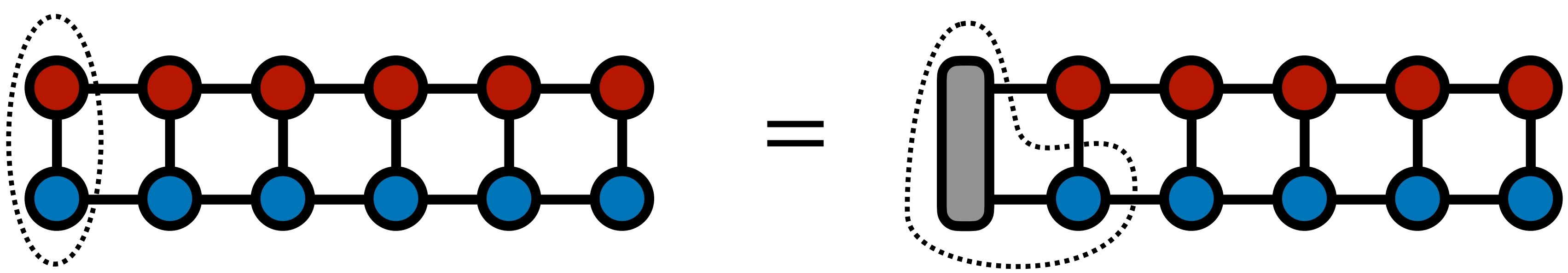
Tensor Networks

and ITensor



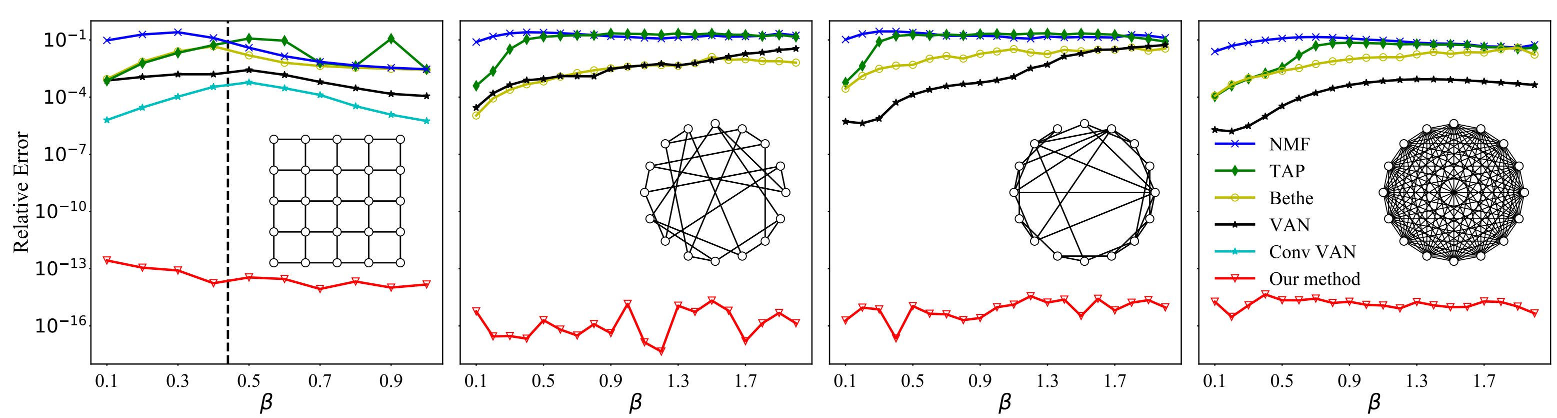


Fig. Credit: Pan, Zhou, Li, Zhang, Phys. Rev. Lett. **125**, 060503 (2020)

E.M. Stoudenmire

Oct 2020 - Flatiron FWAM

FLATIRON INSTITUTE
SIMONS FOUNDATION

Today's Talk:

- Introduction to Tensors
- Tensor Networks
- Applications of Tensor Networks
- Brief Intro to the ITensor Software

For ITensor activity in Discussion Later

Installing Julia:

https://julialang.org/downloads/

https://itensor.org/

Once you have installed Julia on your machine,

1. enter the command `julia` to launch an interactive Julia session (a.k.a. the Julia "REPL")
2. type `]` to enter the package manager (`pkg>` prompt should now show)
3. enter the command `add ITensors`
4. after installation completes, press backspace to return to the normal `julia>` prompt

Sample screenshot:

```
:: julia
julia> ]
(@v1.4) pkg> add ITensors
   Updating registry at `~/.julia/registries/General`
   Updating git-repo `https://github.com/JuliaRegistries/General.git`
```

Introduction to Tensors

What is a tensor?
Where do tensors occur?

What is a Tensor?

At a practical level – and for the purpose of this talk – a tensor is a *multi-dimensional array*

A generalization of a vector or a matrix

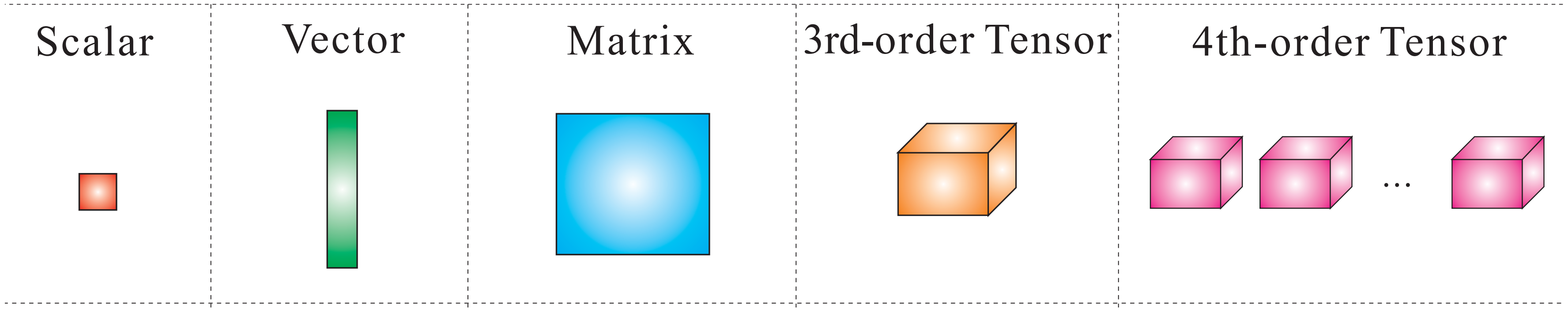


Figure credit: Cichocki et al., "Low-Rank Tensor Networks...", arxiv:1609.00893

What is a Tensor?

The modern definition of a tensor is:

a multi-linear function of vectors

$$T(\mathbf{x}, \mathbf{y}, \mathbf{v}, \mathbf{w}) \rightarrow \mathbb{R}$$

$$T(a\,\mathbf{x}_1 + b\,\mathbf{x}_2, \mathbf{y}, \mathbf{v}, \mathbf{w}) = a\,T(\mathbf{x}_1, \mathbf{y}, \mathbf{v}, \mathbf{w}) + b\,T(\mathbf{x}_2, \mathbf{y}, \mathbf{v}, \mathbf{w})$$

and similar for each argument

Tensor taking N vectors are "*order-N*" *tensors*

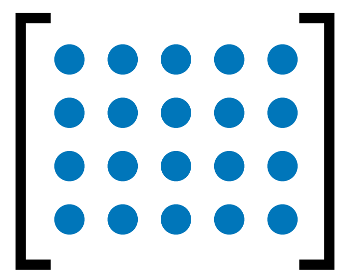
Jeevanjee. "An introduction to tensors and group theory for physicists". Birkhäuser, 2011

What is a Tensor?

Connection to multi-dim. array through plugging in standard basis vectors:

$M(\mathbf{v}, \mathbf{w})$ order-2 tensor M

$$M(\mathbf{e}_i, \mathbf{e}_j) = M_{ij}$$



$$\mathbf{e}_j = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} j$$

Can view an order-2 tensor as a ***matrix*** as long as basis is understood

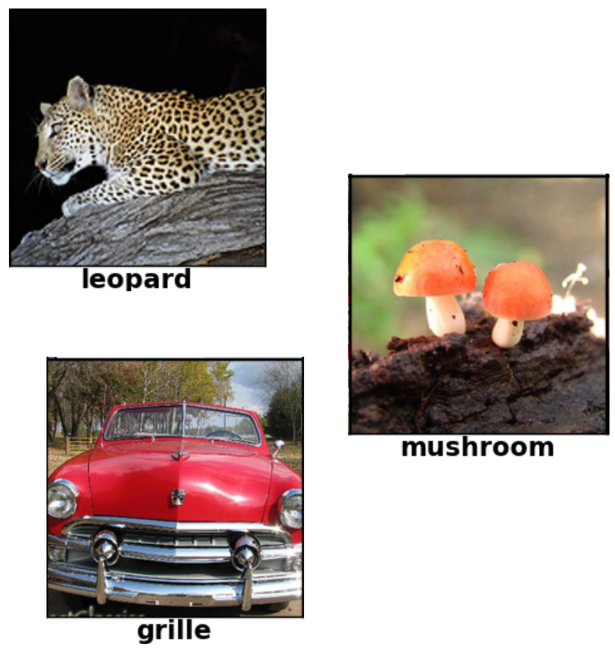
Matrix sufficient to specify M through linearity

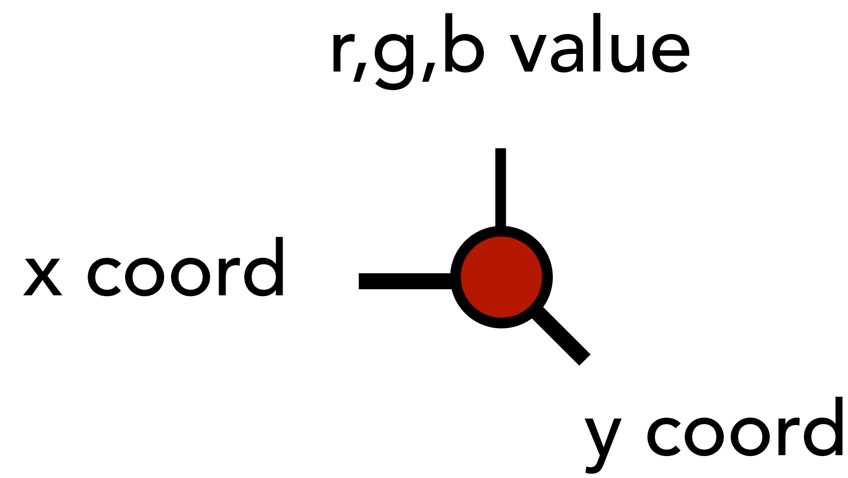
Jeevanjee. "An introduction to tensors and group theory for physicists". Birkhäuser, 2011

Where do Tensors Occur?

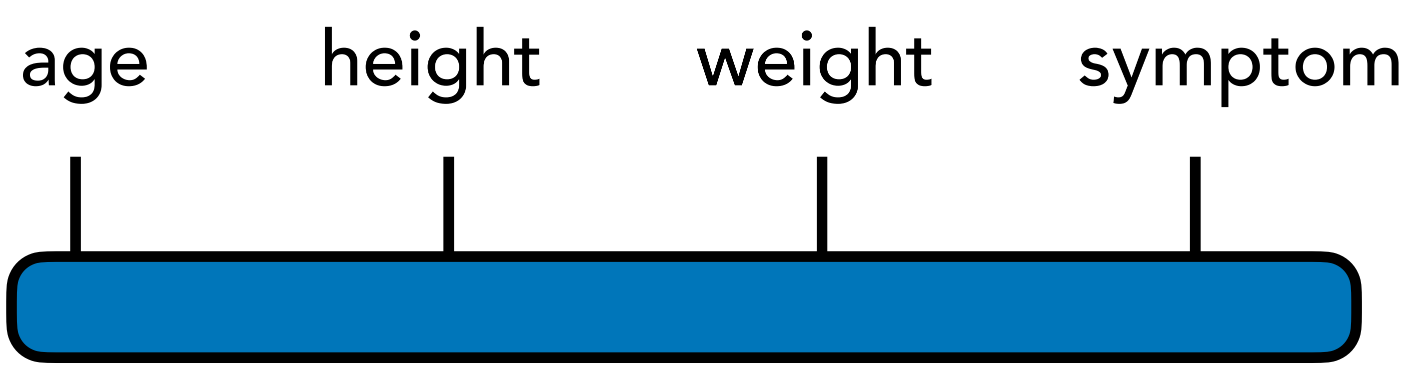
Multi-Dimensional Data

Image Data



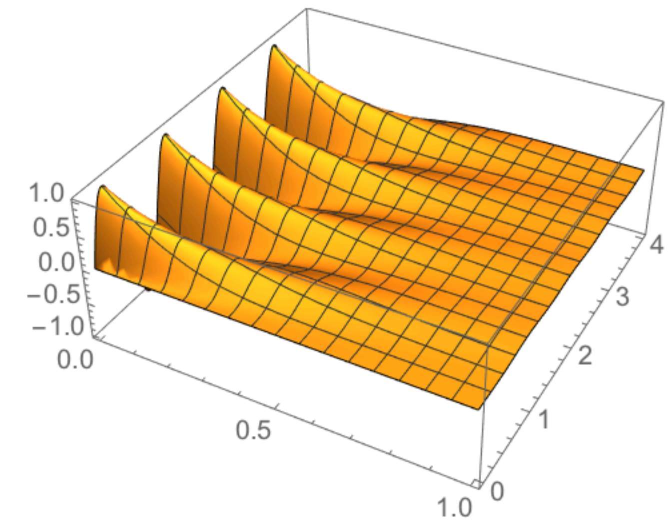


Medical Data



Where do Tensors Occur?

Discretization of functions



$$T_{nmpq} = f(x_n, x_m, x_p, x_q)$$

$$x_n = n \cdot a$$

Note: this is how tensors come up in quantum physics, as discretizations of probability distribution functions or "wavefunctions"*

Image credit: Wolfram Language System Documentation: The Numerical Method of Lines

**technically amplitudes which square to probabilities*

The Curse of Dimensionality

Tensors beyond a few indices become *exponentially* costly to store and manipulate

$$T_{n_1 n_2 n_3 n_4 n_5 n_6} \qquad n_j = 1, 2, ..., 10$$

$$\underbrace{}_{10^6 \text{ entries}}$$

$T_{n_1 n_2 n_3 \cdots n_N}$ has 10^N entries, exponential in N

The Curse of Dimensionality

Tensors beyond a few indices become hard to visualize

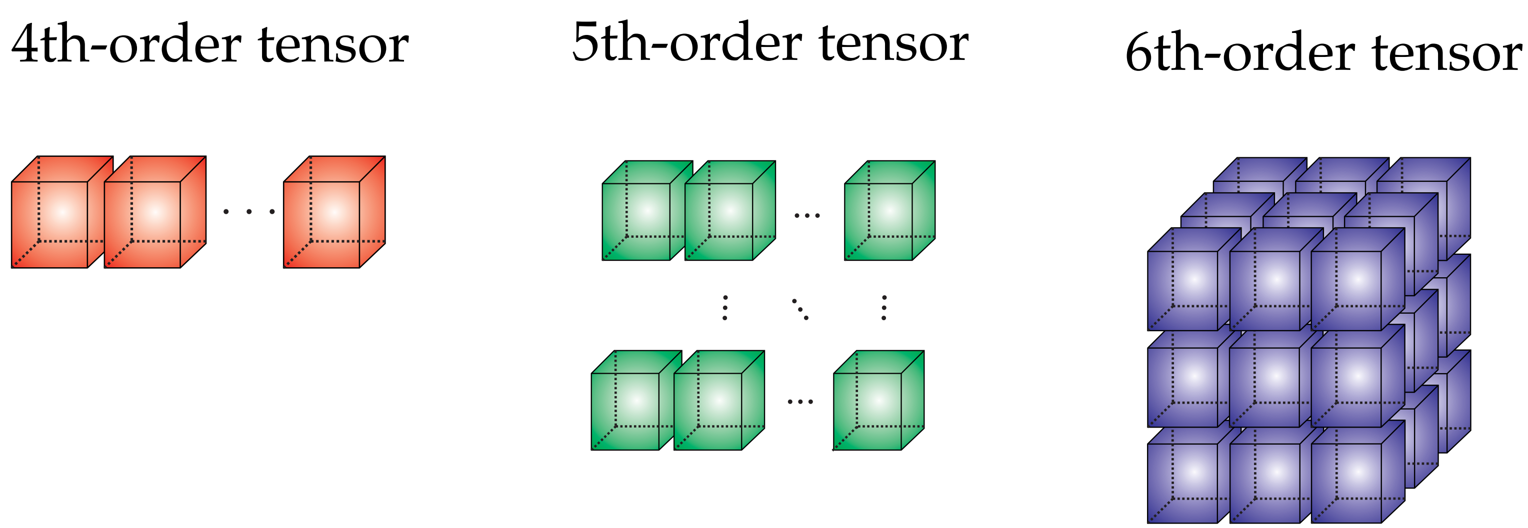


Figure credit: Cichocki et al., "Low-Rank Tensor Networks...", arxiv:1609.00893

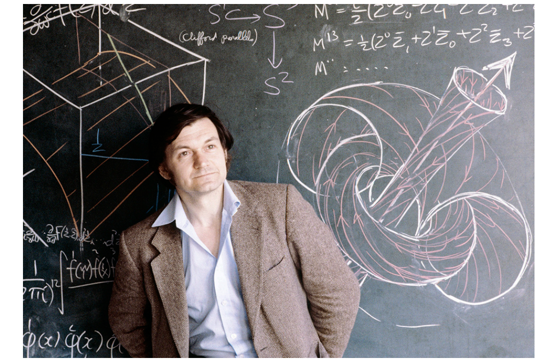
The Curse of Dimensionality

Complicated expressions like

$$T^{n_1 n_2 n_3 n_4 n_5 n_6} = \sum_{\mathbf{a}} A^{n_1}_{a_1} A^{n_2}_{a_1 a_2} A^{n_3}_{a_2 a_3} A^{n_4}_{a_3 a_4} A^{n_5}_{a_4 a_5} A^{n_6}_{a_5 a_6} A^{n_7}_{a_6}$$

difficult for traditional index notation

Tensor Diagram Notation

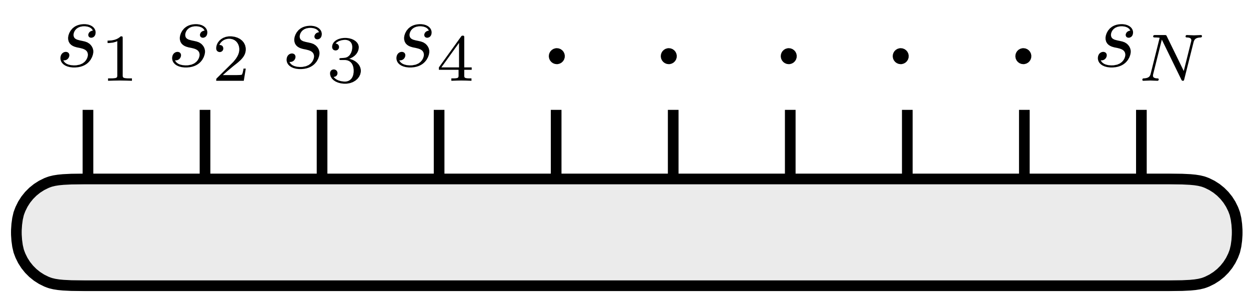


Roger Penrose

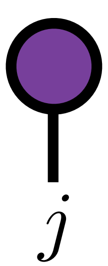
Fortunately there is a way out!

N-index tensor = shape with N lines

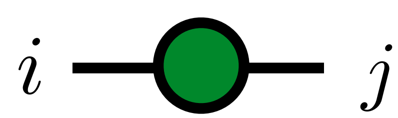
$$T^{s_1 s_2 s_3 \cdots s_N} =$$



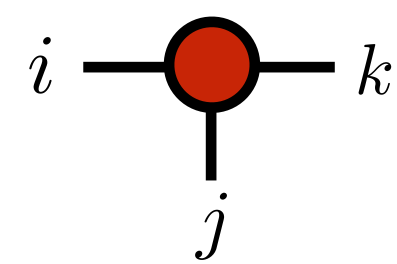
Low-order tensor examples:



v_j



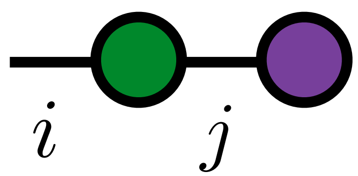
M_{ij}



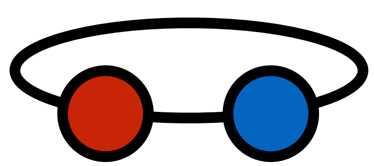
T_{ijk}

Tensor Diagram Notation

Joining lines implies contraction, can omit names



$$\longleftrightarrow \quad \sum_j M_{ij} v_j$$



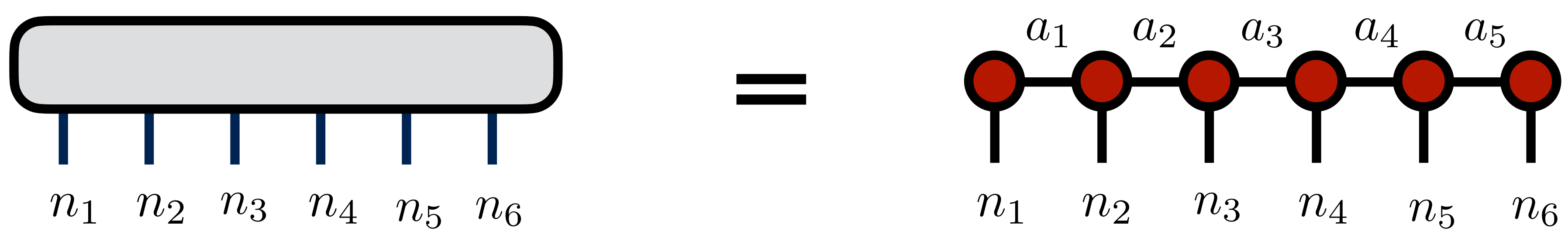
$$\longleftrightarrow \quad A_{ij} B_{ji} \quad = \mathrm{Tr}[AB]$$

Tensor Diagram Notation

Complicated expressions like

$$T^{n_1 n_2 n_3 n_4 n_5 n_6} = \sum_{\mathbf{a}} A^{n_1}_{a_1} A^{n_2}_{a_1 a_2} A^{n_3}_{a_2 a_3} A^{n_4}_{a_3 a_4} A^{n_5}_{a_4 a_5} A^{n_6}_{a_5 a_6} A^{n_7}_{a_6}$$

much clearer in diagram notation



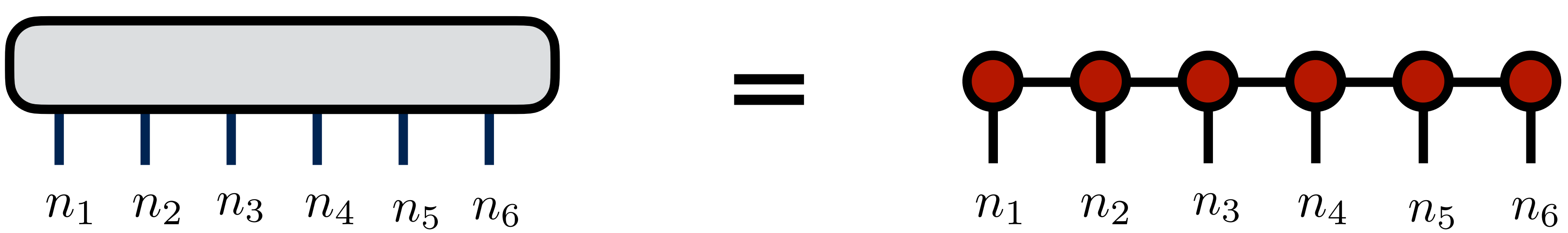
and equally rigorous

Tensor Diagram Notation

Complicated expressions like

$$T^{n_1 n_2 n_3 n_4 n_5 n_6} = \sum_{\mathbf{a}} A^{n_1}_{a_1} A^{n_2}_{a_1 a_2} A^{n_3}_{a_2 a_3} A^{n_4}_{a_3 a_4} A^{n_5}_{a_4 a_5} A^{n_6}_{a_5 a_6} A^{n_7}_{a_6}$$

much clearer in diagram notation



and equally rigorous

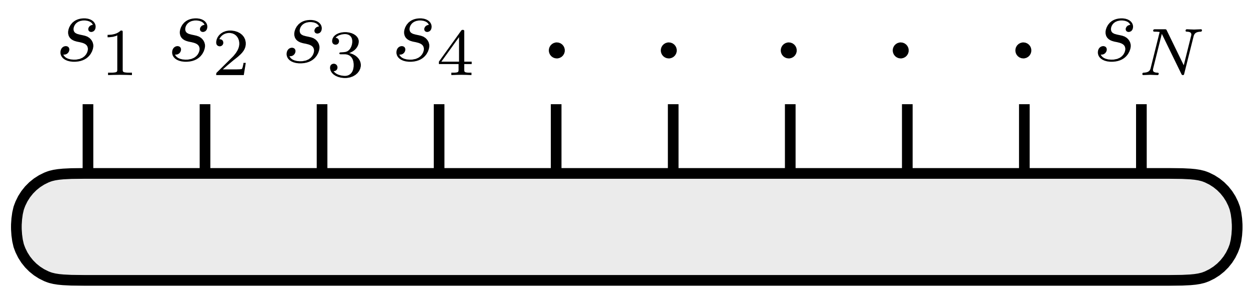
Tensor Networks

Breaking the curse of dimensionality

Key problem:

cannot store or manipulate tensor with N indices

$$T^{s_1 s_2 s_3 \cdots s_N} =$$

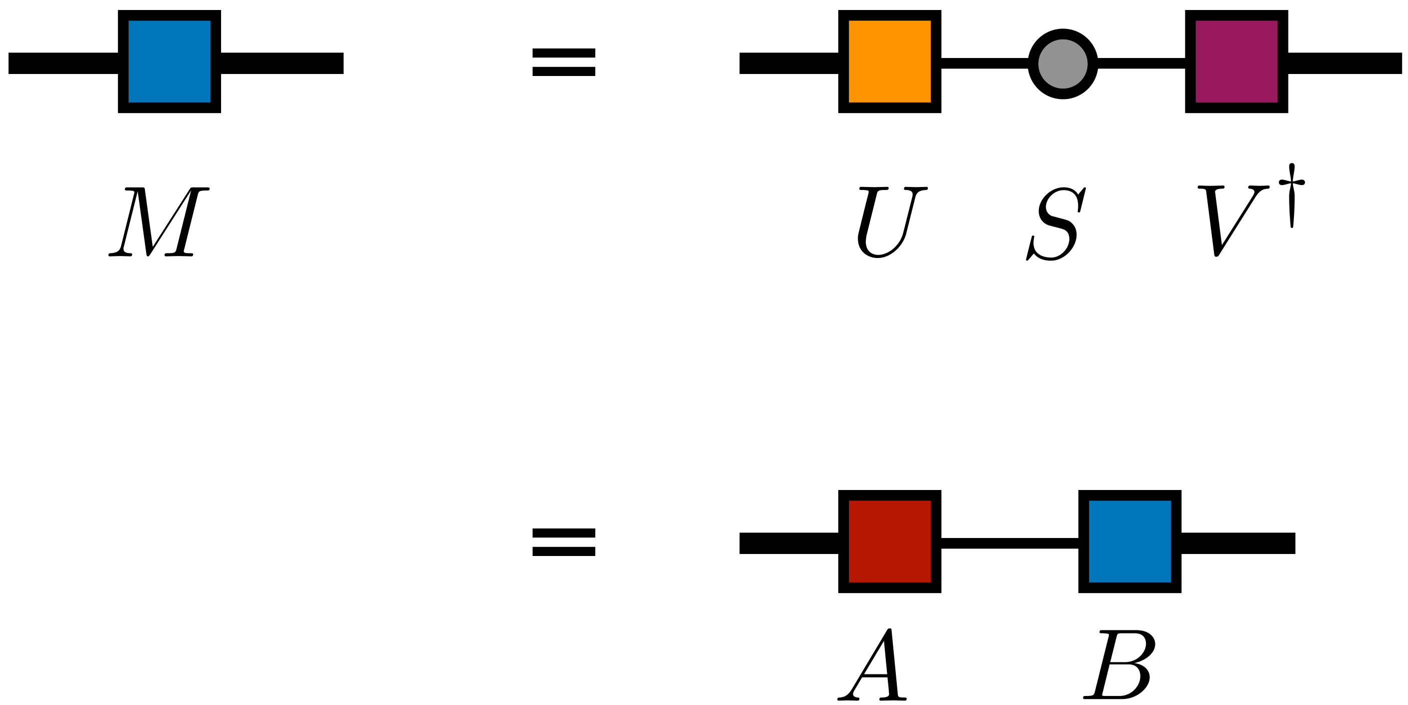


Some ways out:

- sparsity, if applicable
- sampling
- *low-rank structure*

Low-rank Structure

Uncovering low-rank structure
straightforward for matrices



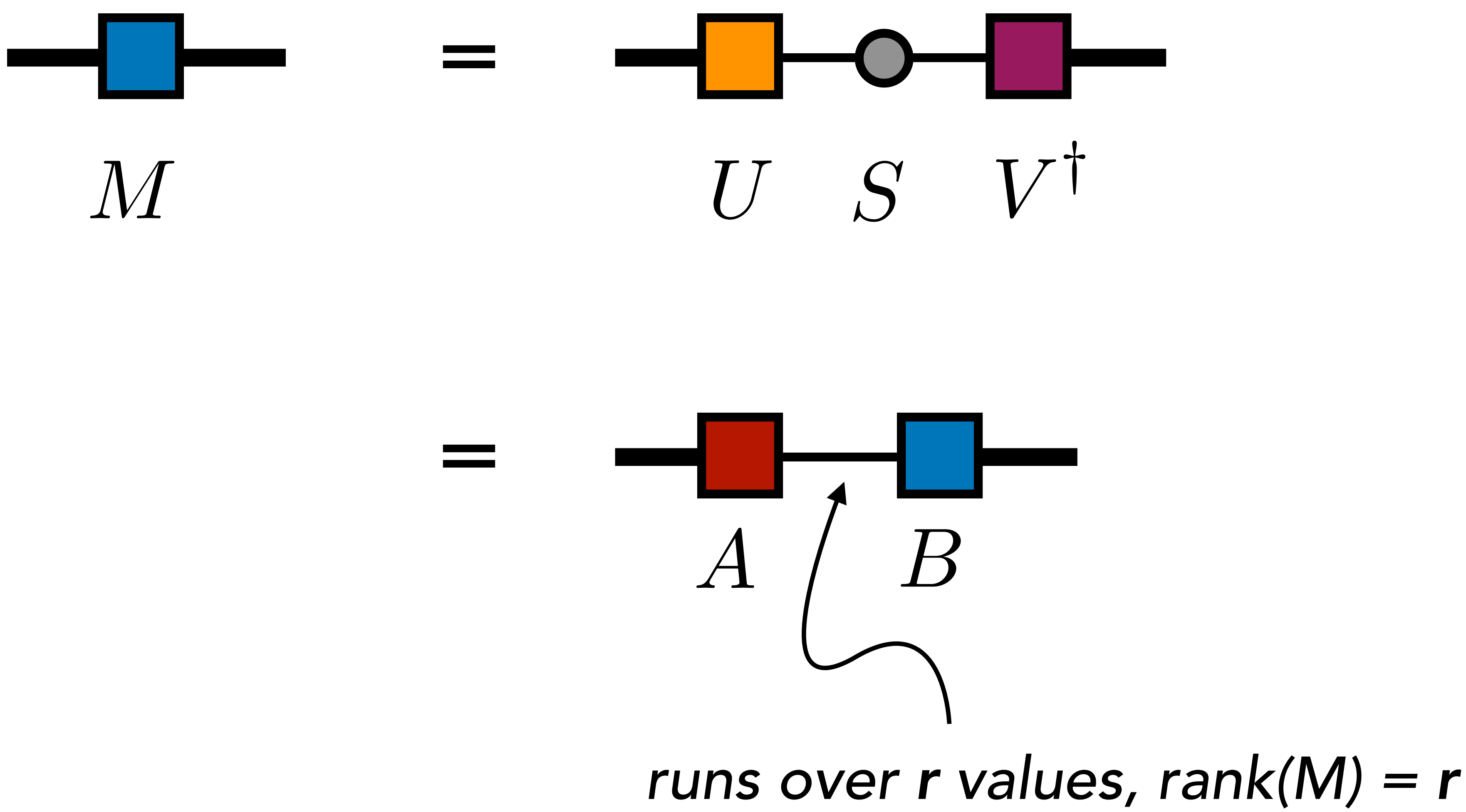
$$A = U\sqrt{S}$$

$$B = \sqrt{S}V^{\dagger}$$

Solved by singular value decomposition (SVD)

Low-rank Structure

Uncovering low-rank structure straightforward for matrices



$A = U\sqrt{S}$

$B = \sqrt{S}V^\dagger$

Solved by singular value decomposition (SVD)

Review: *Singular Value Decomposition (SVD)*

Given rectangular (4x3) matrix M

$$M = \begin{bmatrix} 0.435839 & 0.223707 & 0.10 \\ 0.435839 & 0.223707 & -0.10 \\ 0.223707 & 0.435839 & 0.10 \\ 0.223707 & 0.435839 & -0.10 \end{bmatrix}$$

Can factorize as

$$\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix} \begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0.300 & 0 \\ 0 & 0 & 0.200 \end{bmatrix} \begin{bmatrix} 0.707107 & 0.707107 & 0 \\ -0.707107 & 0.707107 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$
\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix}
\begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0.300 & 0 \\ 0 & 0 & 0.200 \end{bmatrix}
\begin{bmatrix} 0.707107 & 0.707107 & 0 \\ -0.707107 & 0.707107 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$U \qquad S \qquad V^T$$

Matrices U and V have orthonormal columns:

$$U^T U = 1$$

$$V^T V = 1$$

$$
\underbrace{\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix}}_{U}
\underbrace{\begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0.300 & 0 \\ 0 & 0 & 0.200 \end{bmatrix}}_{S}
\underbrace{\begin{bmatrix} 0.707107 & 0.707107 & 0 \\ -0.707107 & 0.707107 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{V^T}
$$

Matrices U and V have orthonormal columns:

$$U^T U = 1$$

$$V^T V = 1$$

S diagonal = "singular values"

Elements of S always:

1) Real
2) Non-negative
3) Decreasing

Keep fewer and fewer elements of S:

$$\underset{U}{\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix}} \underset{S}{\begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0.300 & 0 \\ 0 & 0 & 0.200 \end{bmatrix}} \underset{V^T}{\begin{bmatrix} 0.707107 & 0.707107 & 0 & \\ -0.707107 & 0.707107 & 0 & \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$= M = \begin{bmatrix} 0.435839 & 0.223707 & 0.10 \\ 0.435839 & 0.223707 & -0.10 \\ 0.223707 & 0.435839 & 0.10 \\ 0.223707 & 0.435839 & -0.10 \end{bmatrix}$$

$$||M - M||^2 = 0$$

Keep fewer and fewer elements of S:

$$\overset{U}{\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix}} \overset{S}{\begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0.300 & 0 \\ 0 & 0 & 0.200 \end{bmatrix}} \overset{V^T}{\begin{bmatrix} 0.707107 & 0.707107 & 0 \\ -0.707107 & 0.707107 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$= M_2 = \begin{bmatrix} 0.435839 & 0.223707 & 0 \\ 0.435839 & 0.223707 & 0 \\ 0.223707 & 0.435839 & 0 \\ 0.223707 & 0.435839 & 0 \end{bmatrix}$$

$$||M - M||^2 = 0$$

Keep fewer and fewer elements of S:

$$\underset{U}{\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix}} \underset{S}{\begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0.300 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \underset{V^T}{\begin{bmatrix} 0.707107 & 0.707107 & 0 \\ -0.707107 & 0.707107 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$= M_2 = \begin{bmatrix} 0.435839 & 0.223707 & 0 \\ 0.435839 & 0.223707 & 0 \\ 0.223707 & 0.435839 & 0 \\ 0.223707 & 0.435839 & 0 \end{bmatrix}$$

$$||M_2 - M||^2 = 0.04 = (0.2)^2$$

Keep fewer and fewer elements of S:

$$\underbrace{\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0.300 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{S} \underbrace{\begin{bmatrix} 0.707107 & 0.707107 & 0 & \\ -0.707107 & 0.707107 & 0 & \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{V^T}$$

$$= M_3 = \begin{bmatrix} 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \end{bmatrix}$$

$$||M_2 - M||^2 = 0.04 = (0.2)^2$$

Keep fewer and fewer elements of S:

$$\underset{U}{\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix}} \underset{S}{\begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \underset{V^T}{\begin{bmatrix} 0.707107 & 0.707107 & 0 \\ -0.707107 & 0.707107 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$= M_3 = \begin{bmatrix} 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \end{bmatrix}$$

$$||M_3 - M||^2 = 0.13 = (0.3)^2 + (0.2)^2$$

Keep fewer and fewer elements of S:

$$\underset{U}{\begin{bmatrix} 1/2 & -1/2 & 1/2 \\ 1/2 & -1/2 & -1/2 \\ 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 \end{bmatrix}} \underset{S}{\begin{bmatrix} 0.933 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \underset{V^T}{\begin{bmatrix} 0.707107 & 0.707107 & 0 \\ -0.707107 & 0.707107 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$= M_3 = \begin{bmatrix} 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \\ 0.329773 & 0.329773 & 0 \end{bmatrix}$$

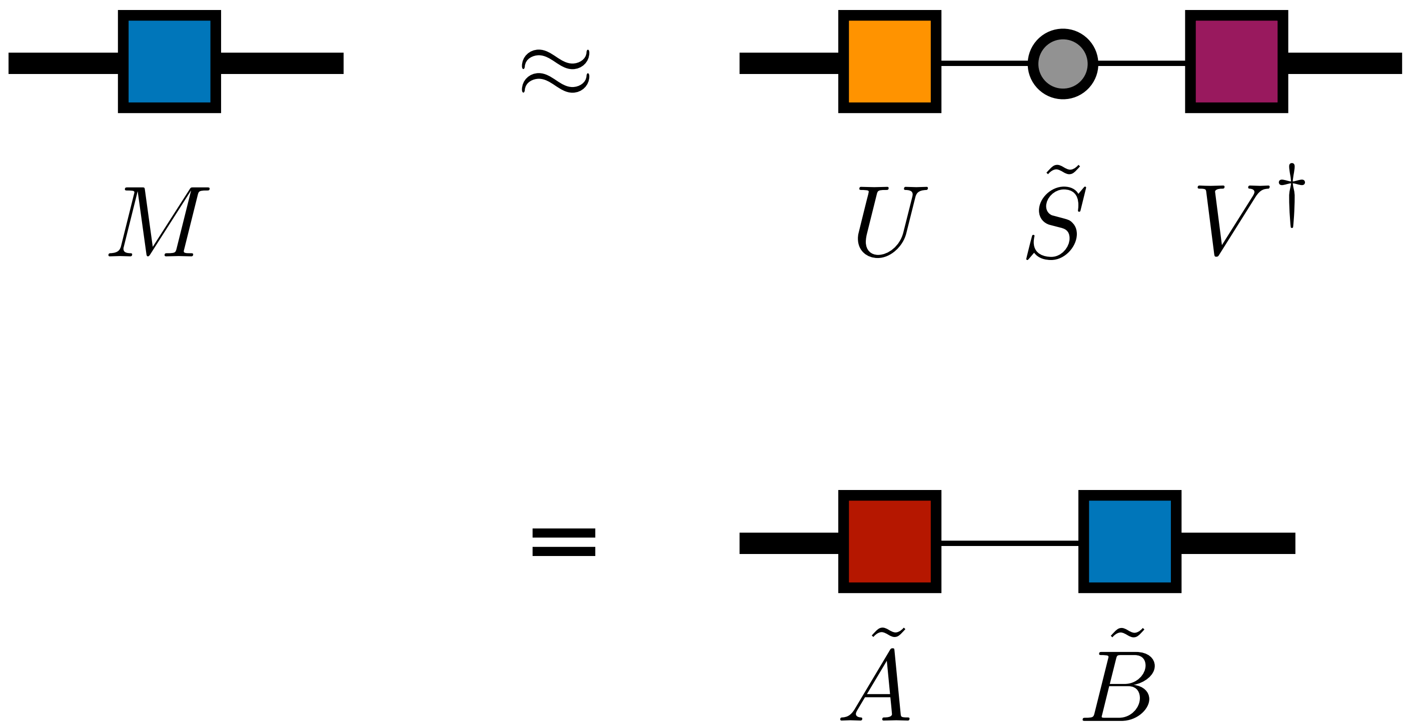
Truncating SVD =

Controlled approximation for M

$$||M_3 - M||^2 = 0.13 = (0.3)^2 + (0.2)^2$$

Low-rank Structure

If matrix M approximately low-rank,
truncating singular values of SVD gives optimal approximation

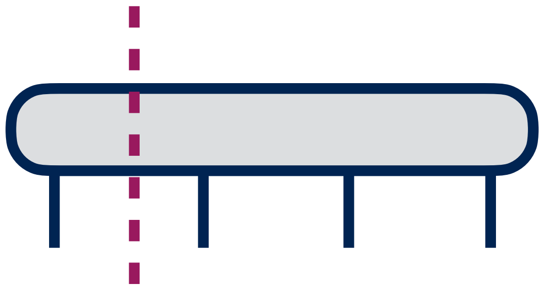


Let's apply SVD to a tensor - how?

Reshape as a matrix:

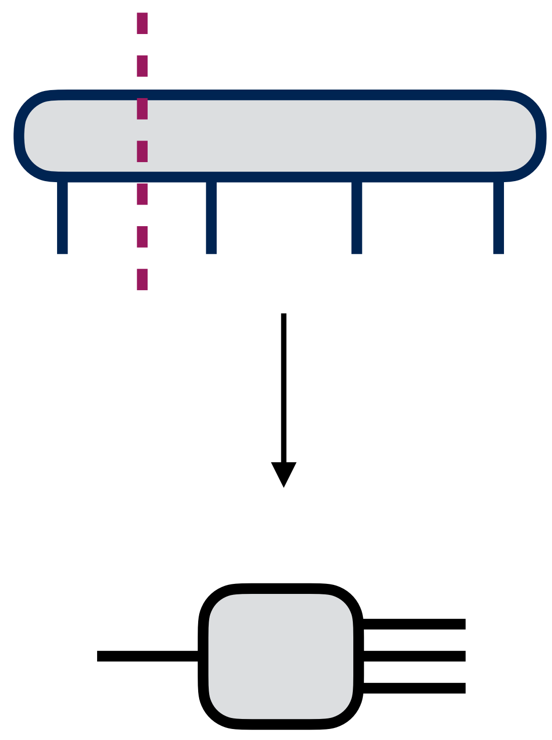
Let's apply SVD to a tensor - how?

Reshape as a matrix:



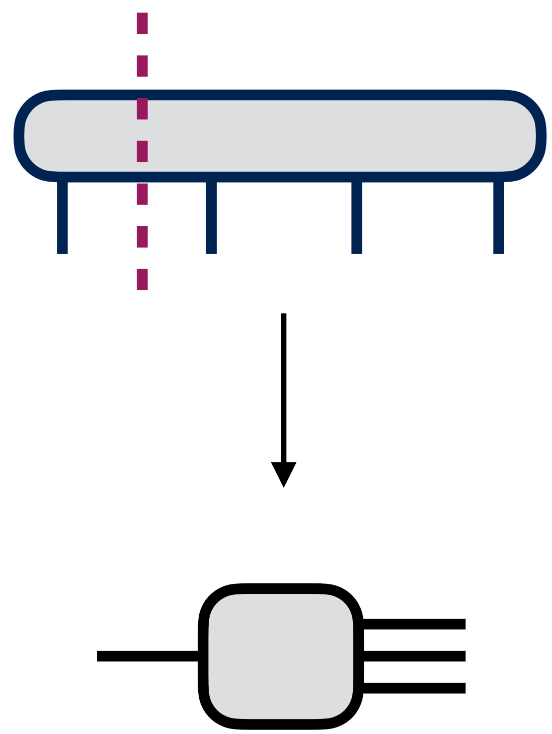
Let's apply SVD to a tensor - how?

Reshape as a matrix:

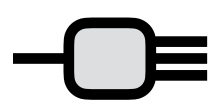


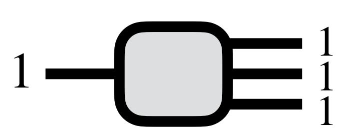
Let's apply SVD to a tensor - how?

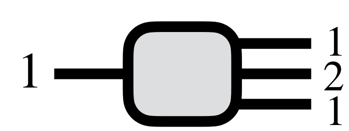
Reshape as a matrix:

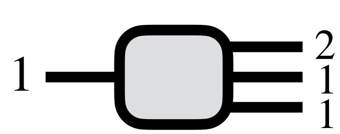


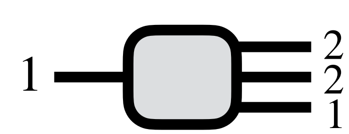
[

Reshaping  as a matrix means treating as a 2x8 matrix M, where:

 $= M_{11}$

 $= M_{13}$

 $= M_{12}$

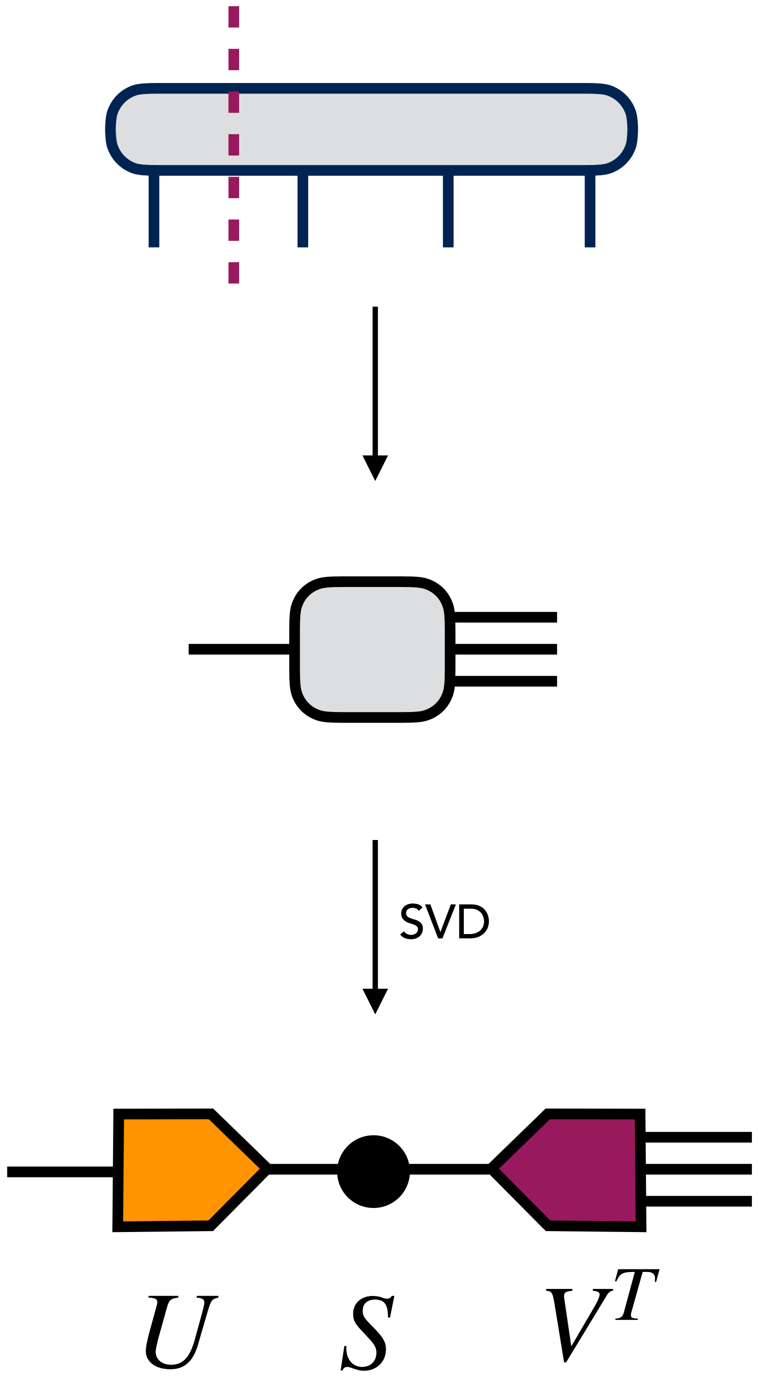
 $= M_{14}$

etc.

]

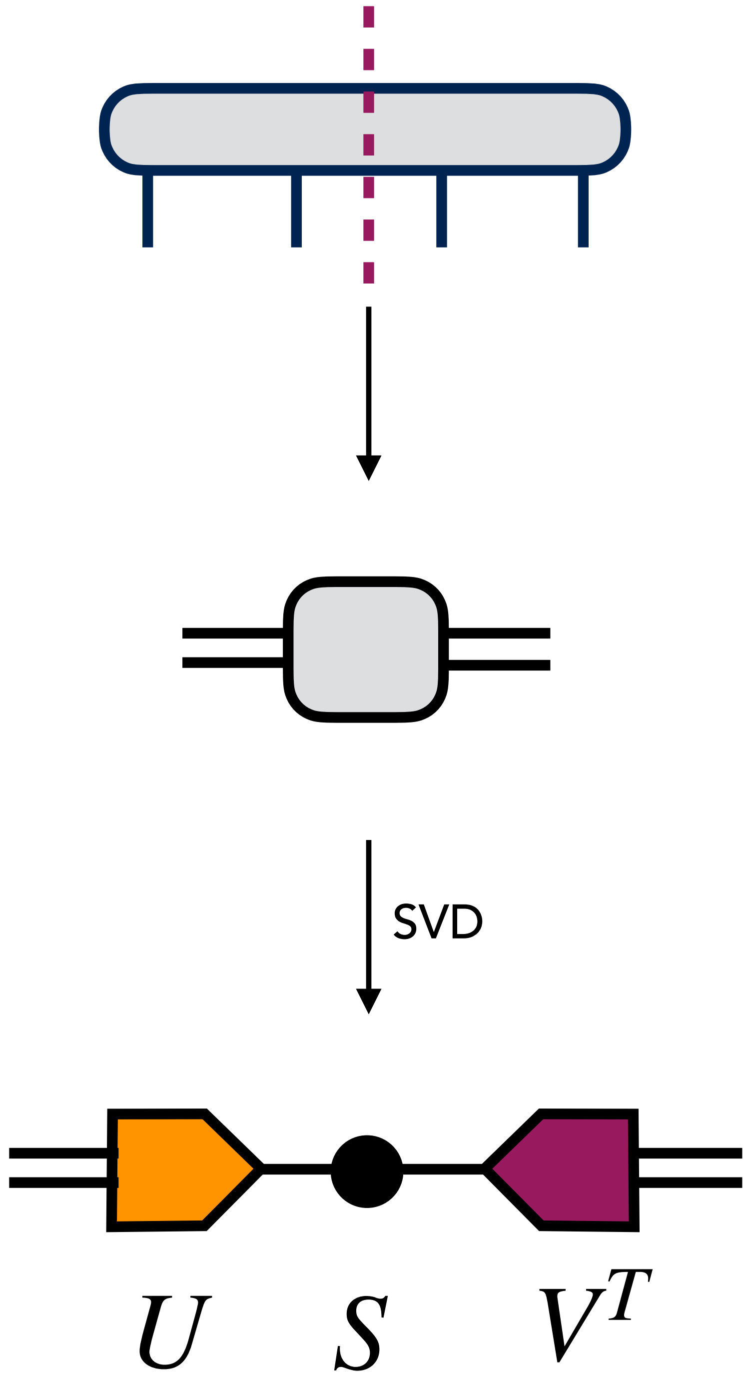
How to generalize SVD to tensors?

Reshape as a matrix:



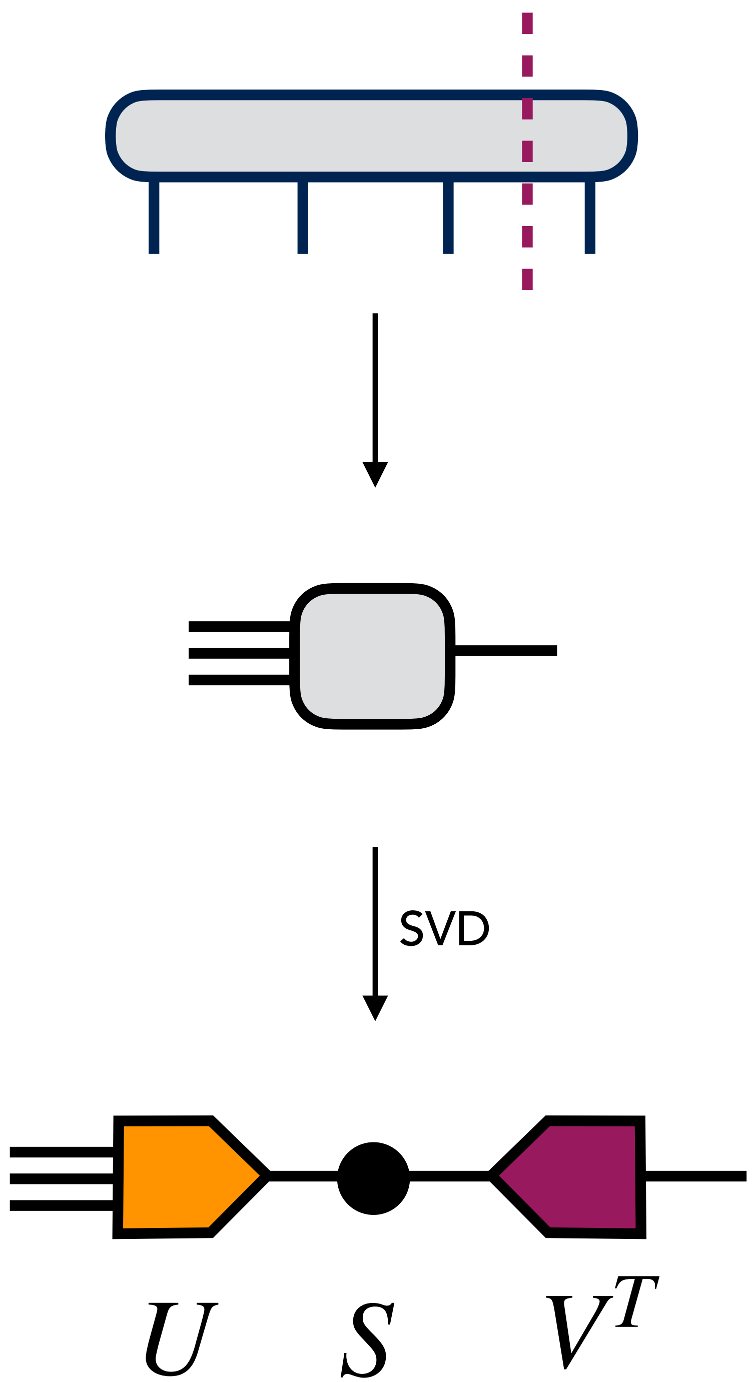
How to generalize SVD to tensors?

Other partitions:

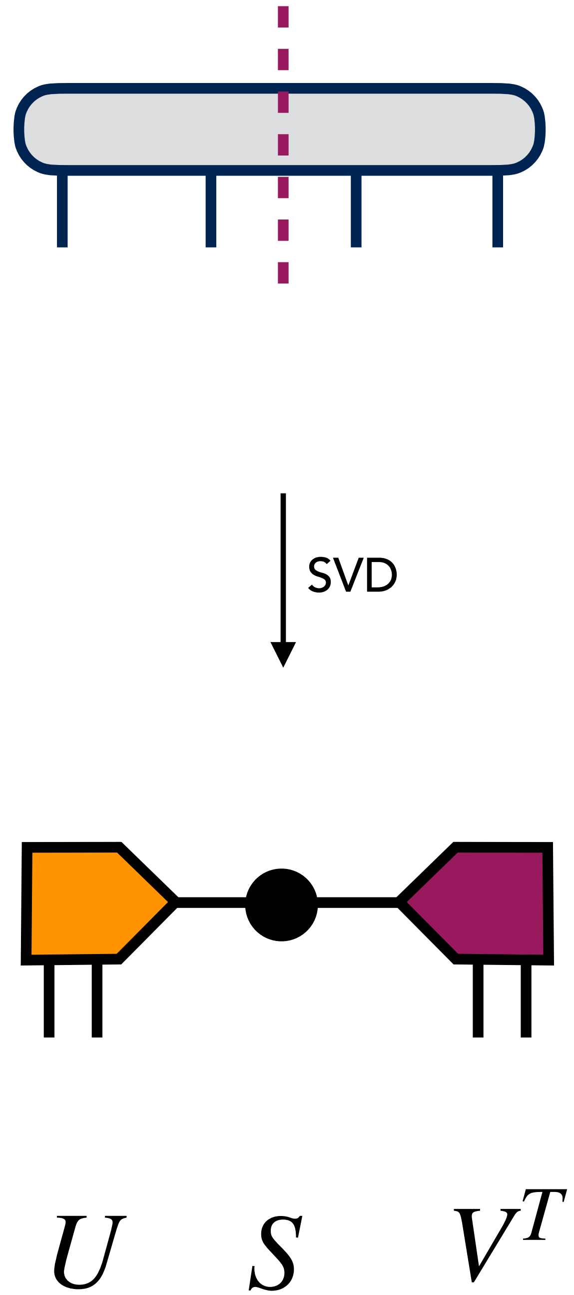


How to generalize SVD to tensors?

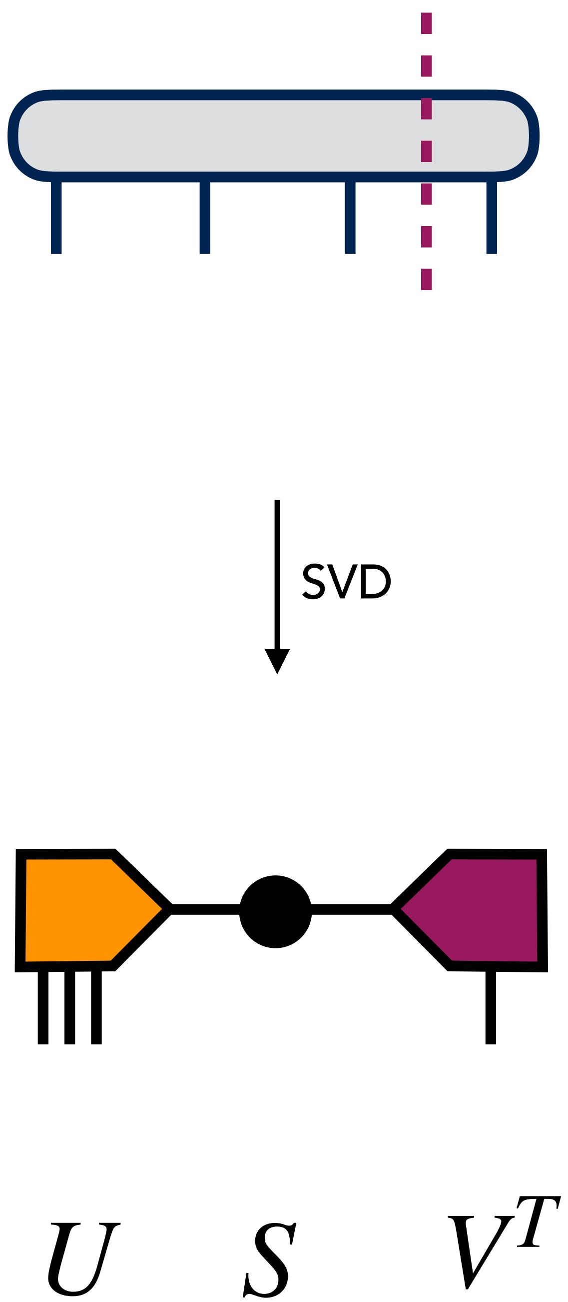
Other partitions:



From now on, reshaping steps are *implicit*:

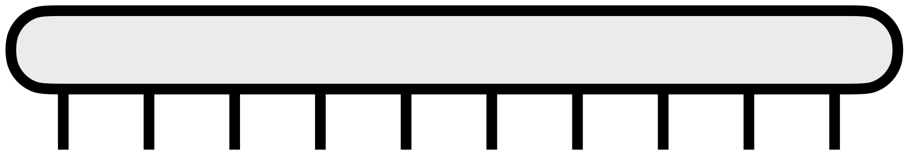


From now on, reshaping steps are *implicit*:

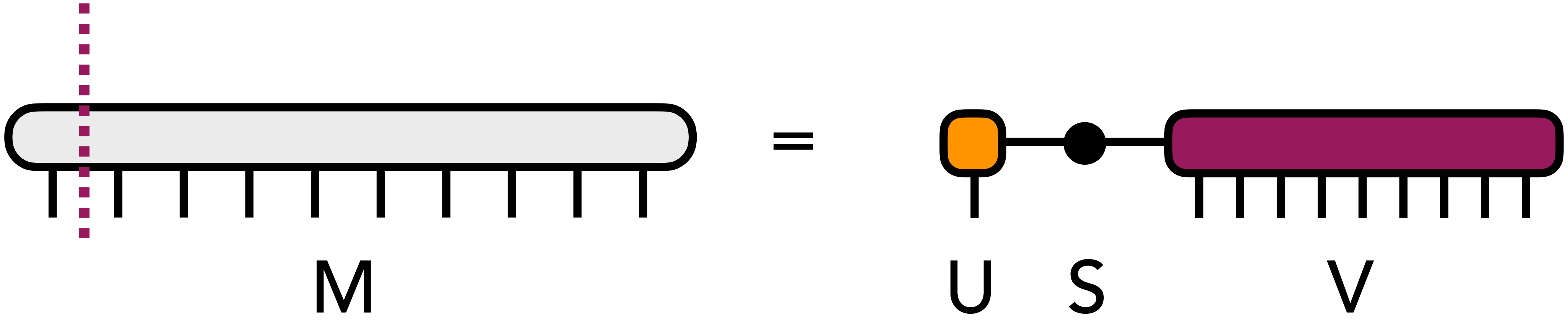


For N-index tensor, which partition to choose?

N-index tensor:

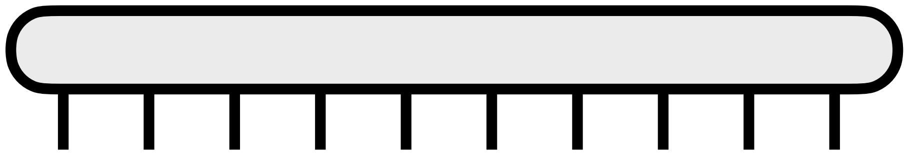


Could reshape as 2 x 2^{N-1} matrix and SVD

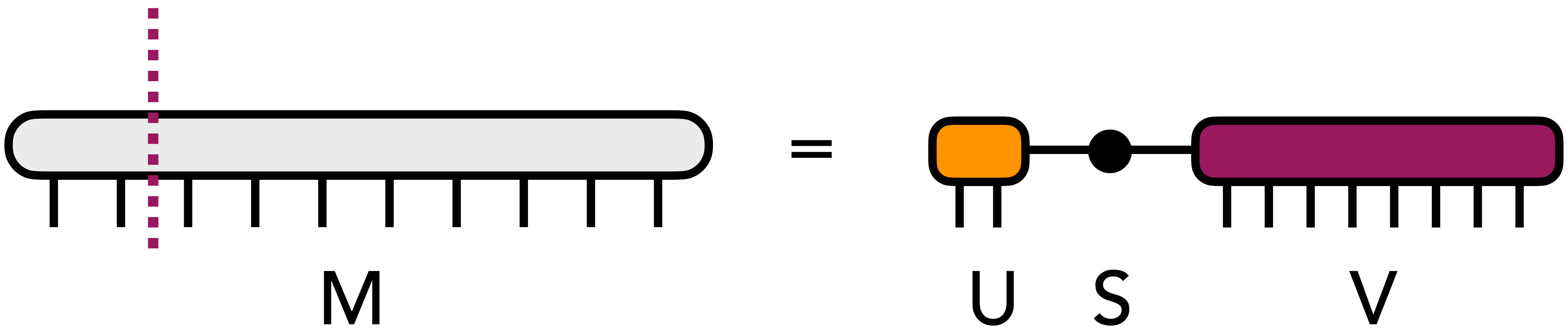


For N-index tensor, which partition to choose?

N-index tensor:

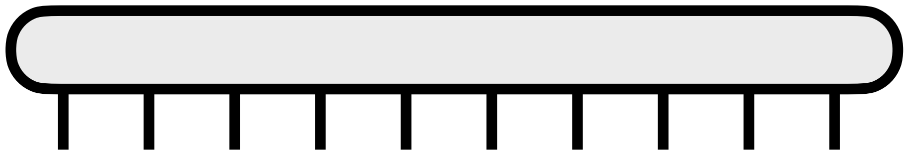


Or reshape to 2^2 x 2^{N-2} matrix and SVD

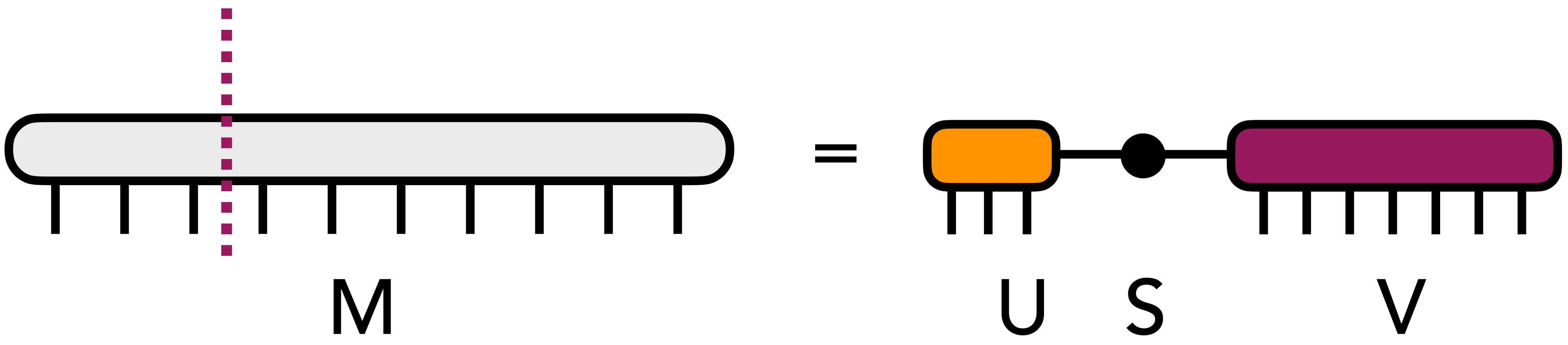


For N-index tensor, which partition to choose?

N-index tensor:

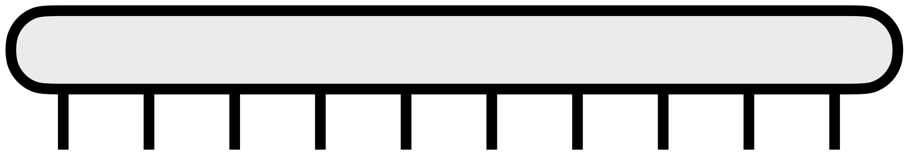


Or reshape to $2^3 \times 2^{N-3}$ matrix and SVD

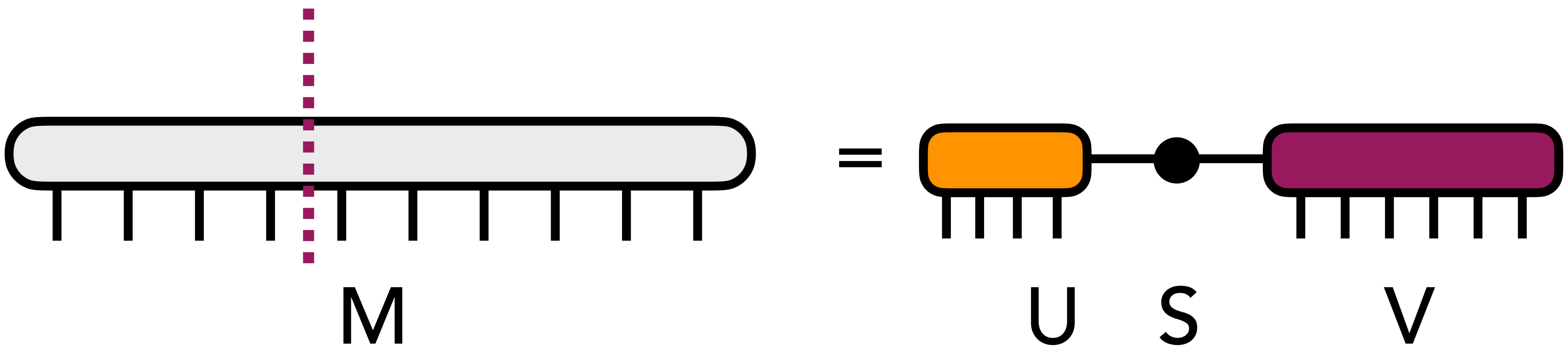


For N-index tensor, which partition to choose?

N-index tensor:

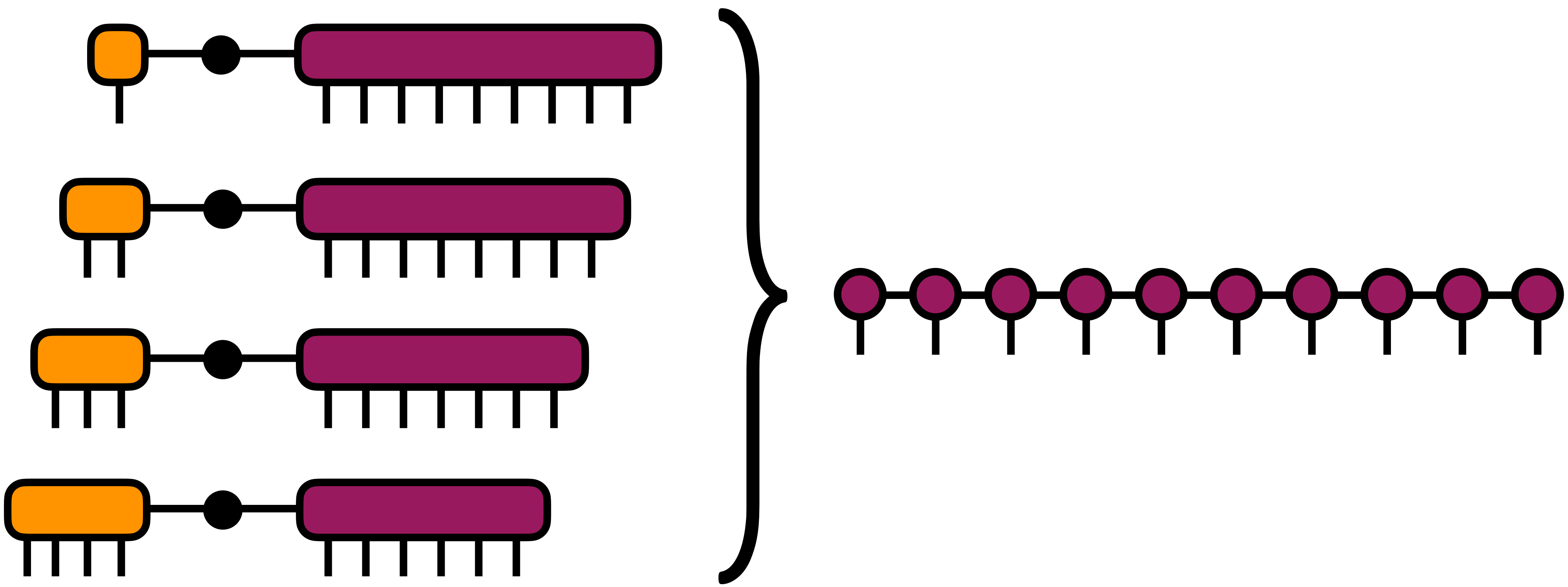


Or reshape to 2^4 x 2^{N-4} matrix and SVD



Can combine all SVD's simultaneously

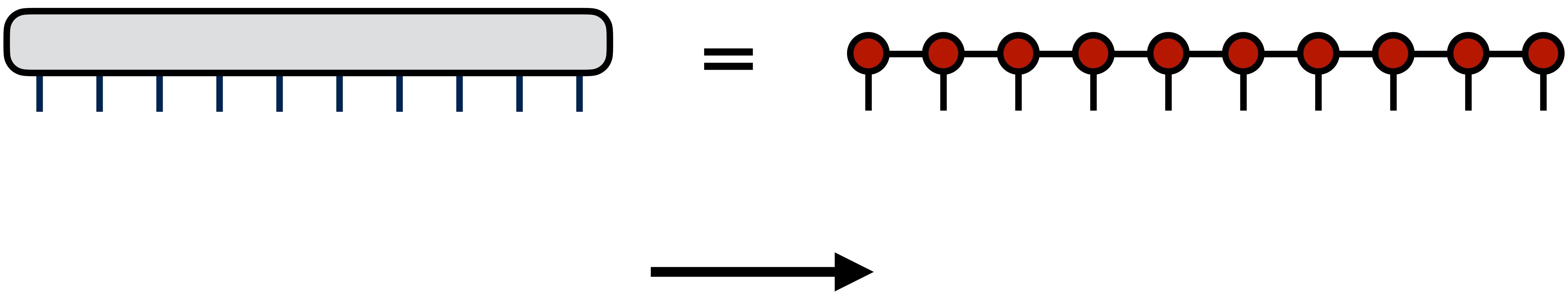
Result known as *matrix product state (MPS)*



MPS = vast generalization of SVD for tensors

also known as *tensor train* (TT) in math literature

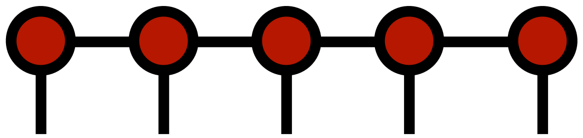
Matrix product state (MPS) tensor network



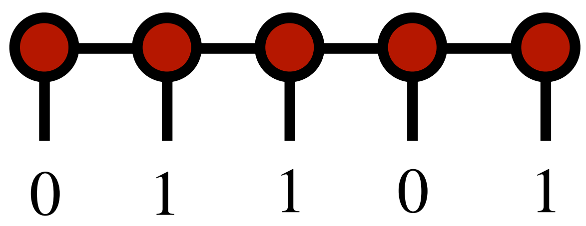
Can view as multi-SVD of a tensor

Or special class or subspace of tensors
(low-rank subspace)

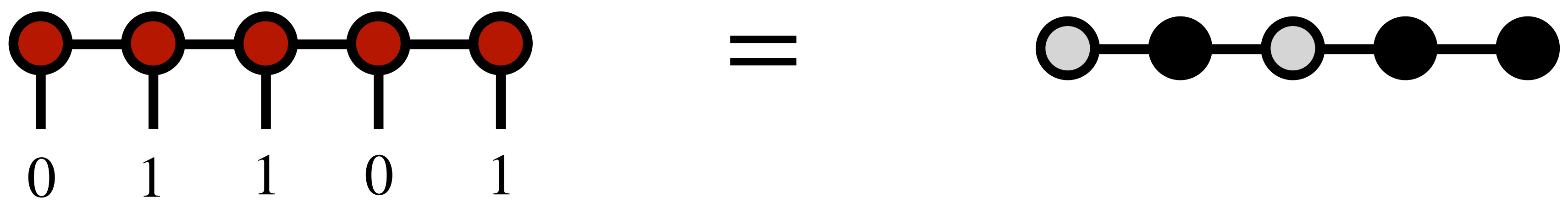
Name matrix product state refers to retrieving elements:



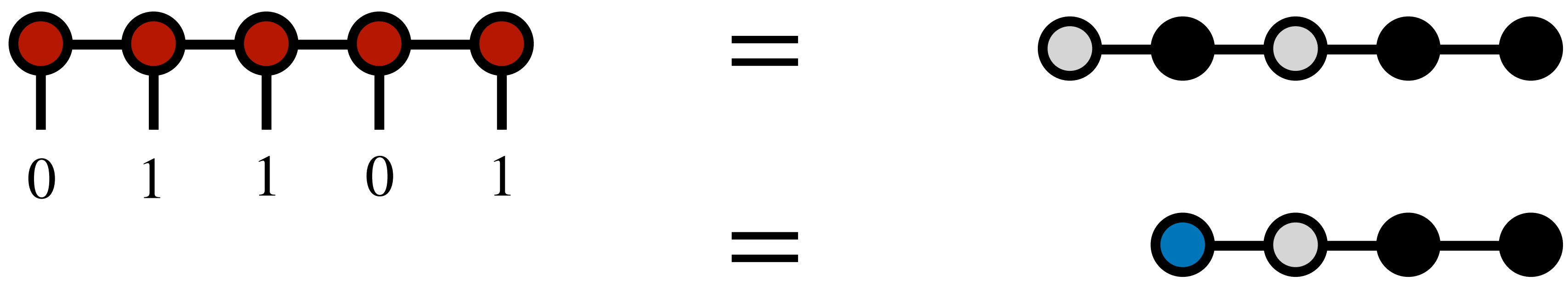
Name matrix product state refers to retrieving elements:



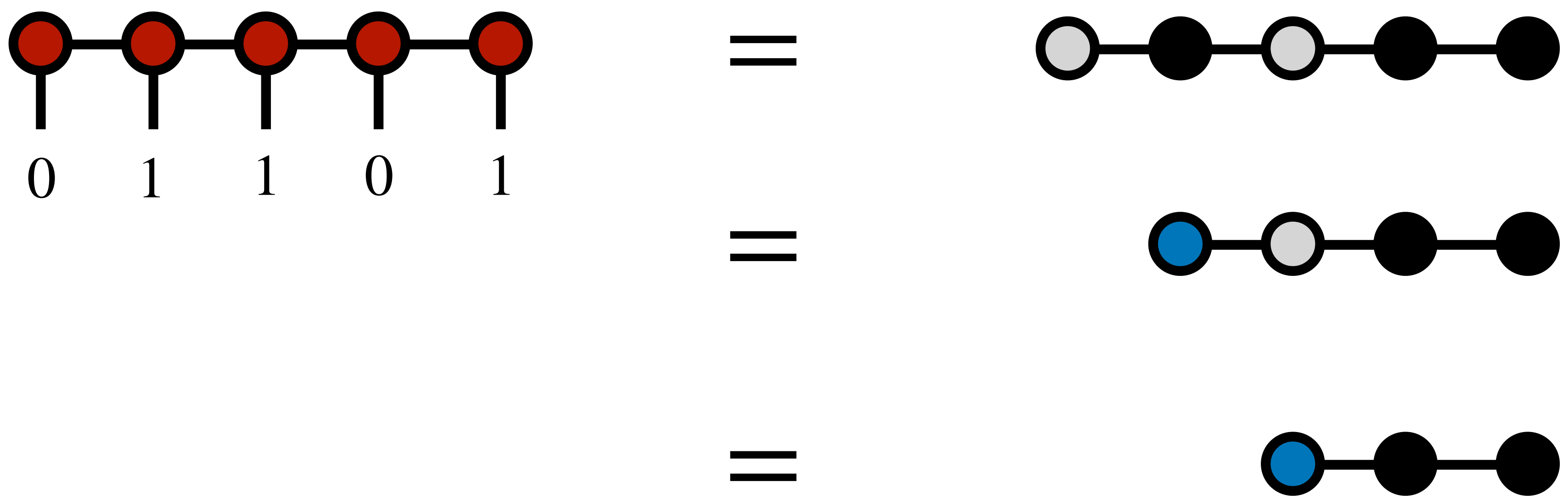
Name matrix product state refers to retrieving elements:



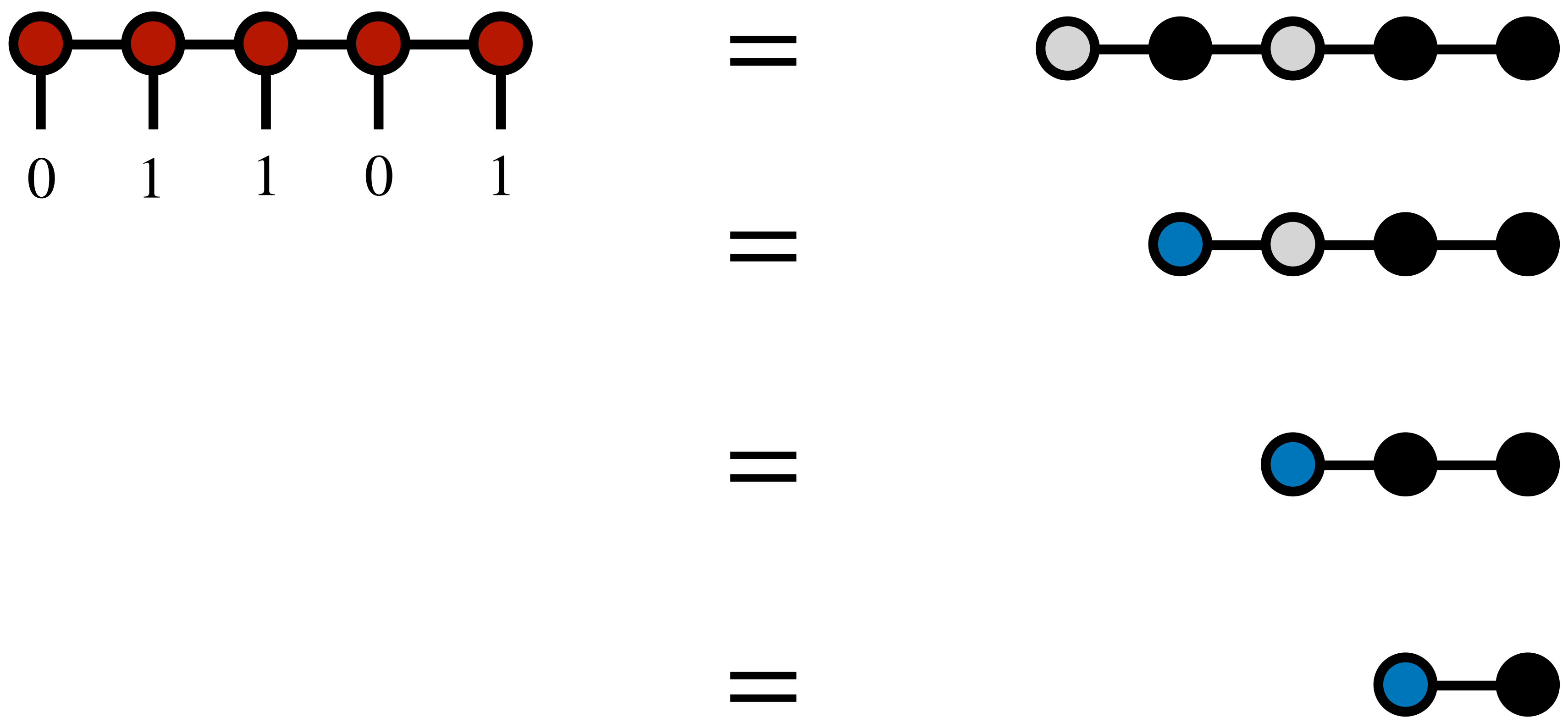
Name matrix product state refers to retrieving elements:



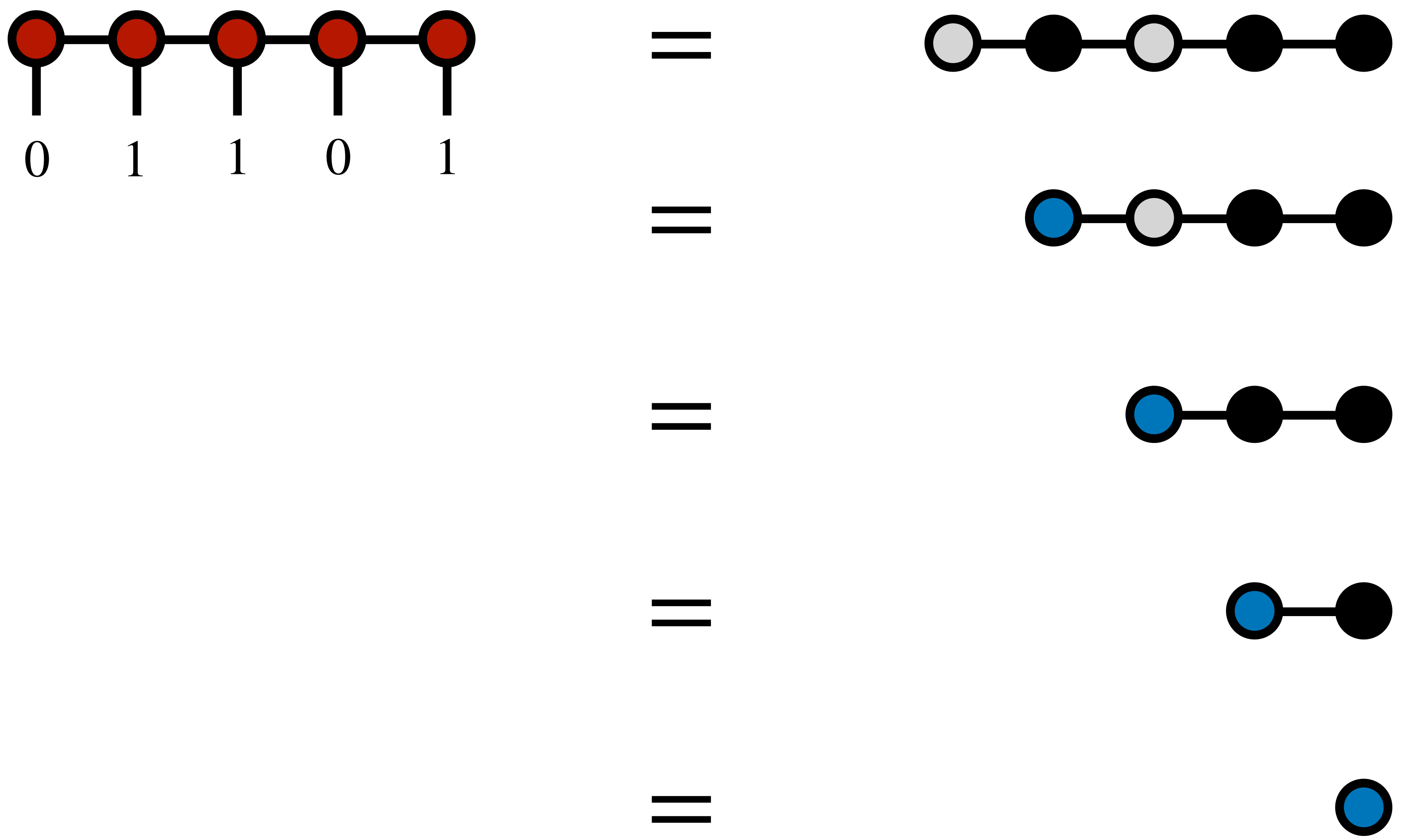
Name matrix product state refers to retrieving elements:



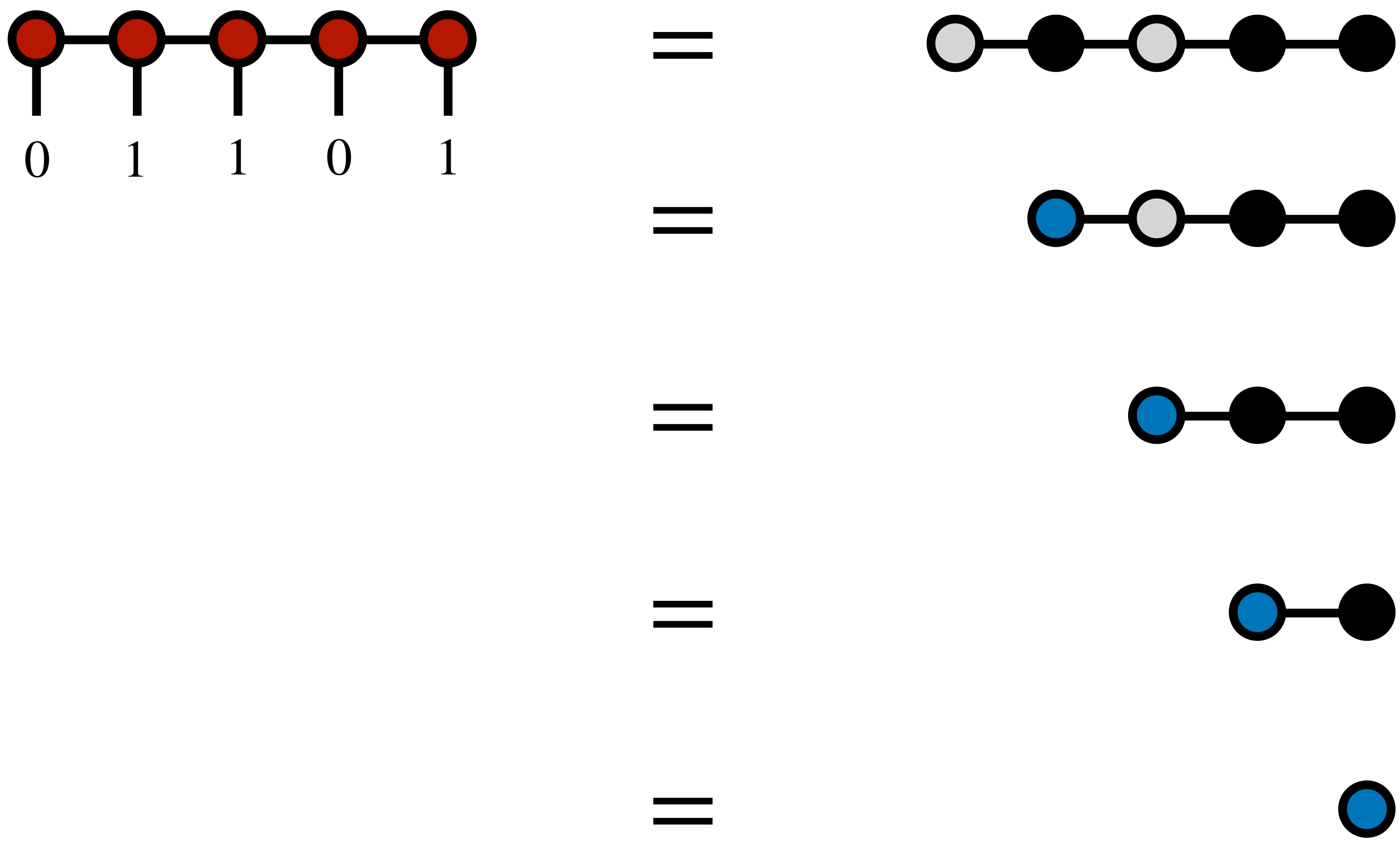
Name matrix product state refers to retrieving elements:



Name matrix product state refers to retrieving elements:

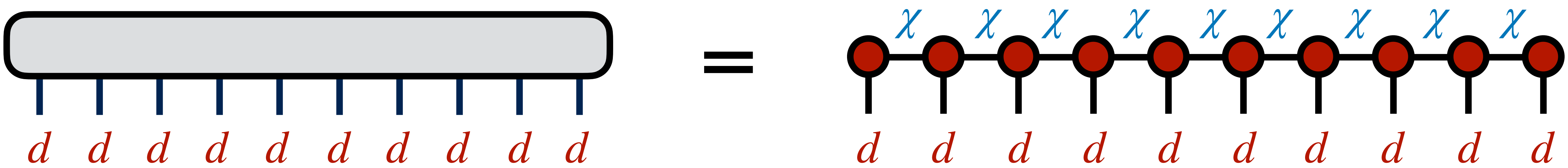


Name matrix product state refers to retrieving elements:



$$\approx \quad T^{01101}$$

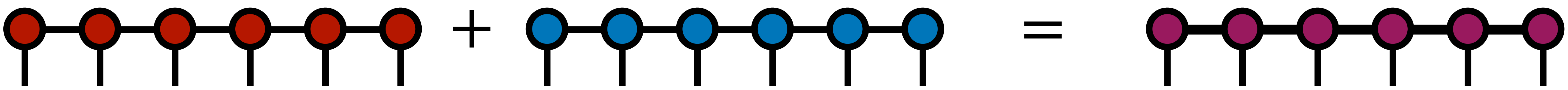
Hyper-parameter of matrix product state (MPS) is bond dimension χ



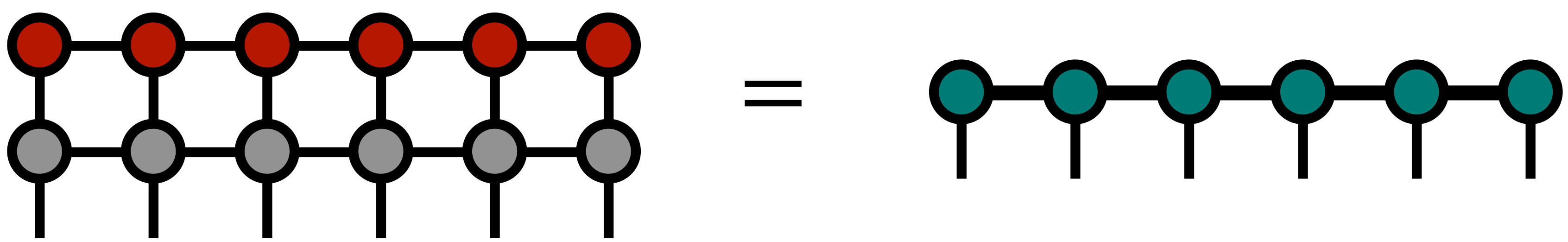
If modest χ yields good approximation, obtain massive compression:

$$d^N \longrightarrow N d \chi^2$$

Can efficiently sum MPS in compressed form:



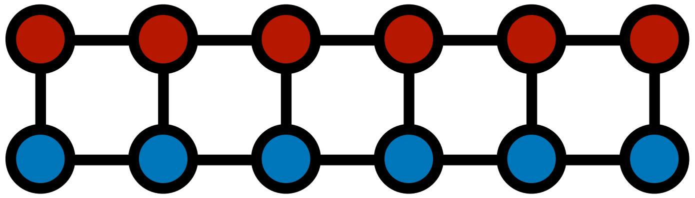
Or multiply by other networks:



Typical cost χ^3, memory usage χ^2

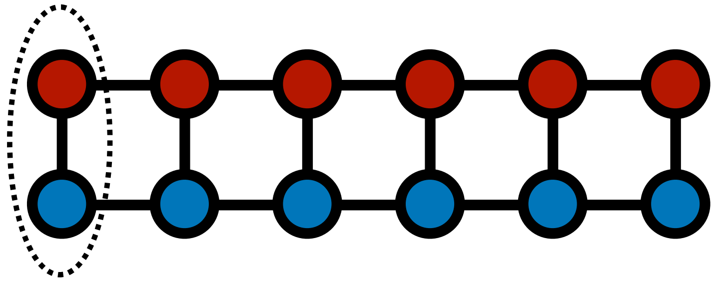
More detailed tensor network algorithm

Inner product of two MPS tensors



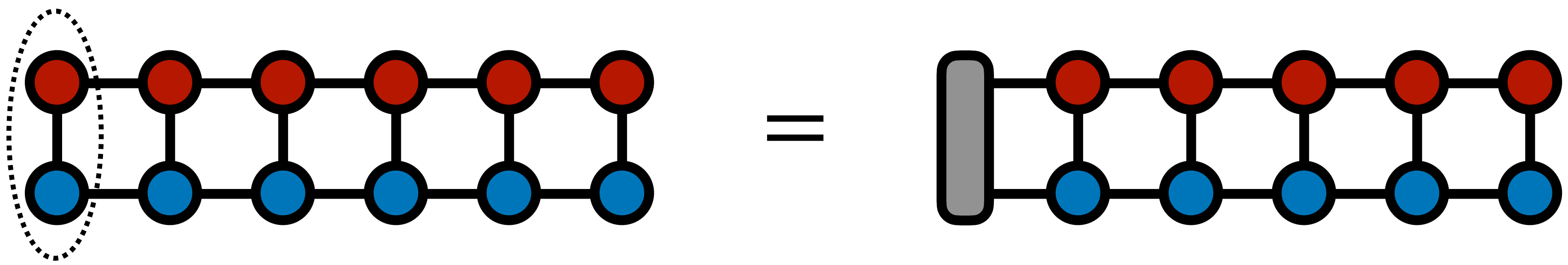
More detailed tensor network algorithm

Inner product of two MPS tensors



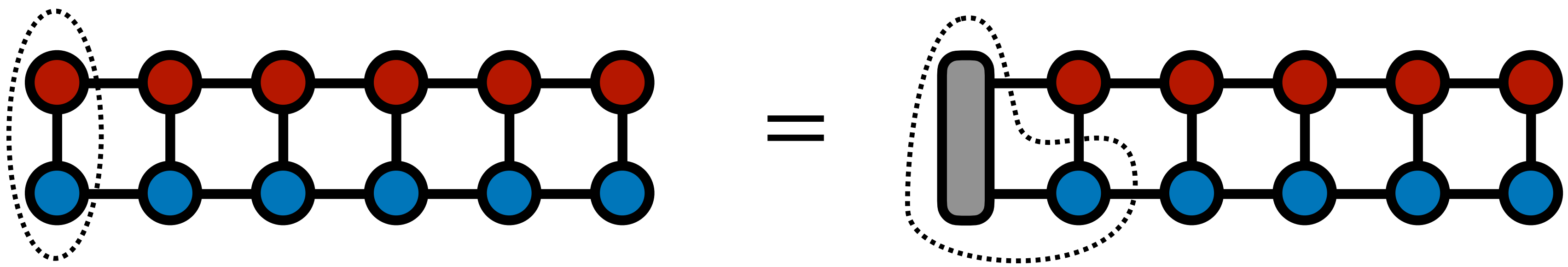
More detailed tensor network algorithm

Inner product of two MPS tensors



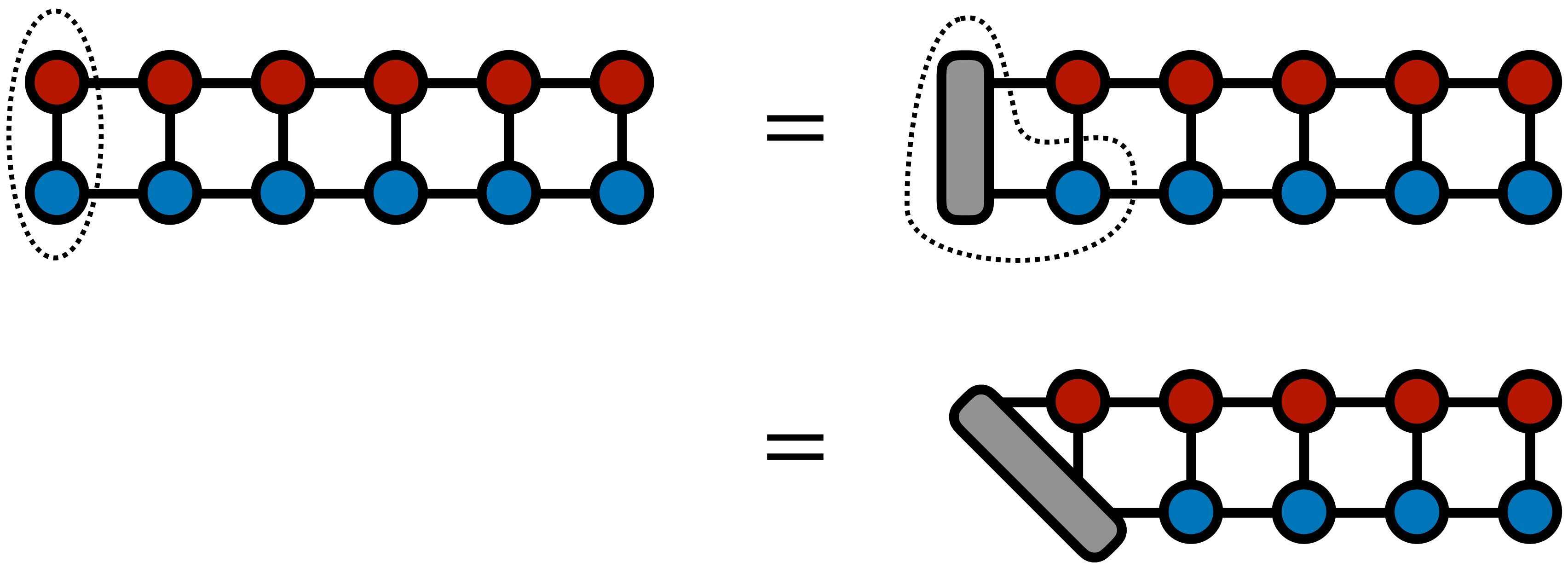
More detailed tensor network algorithm

Inner product of two MPS tensors



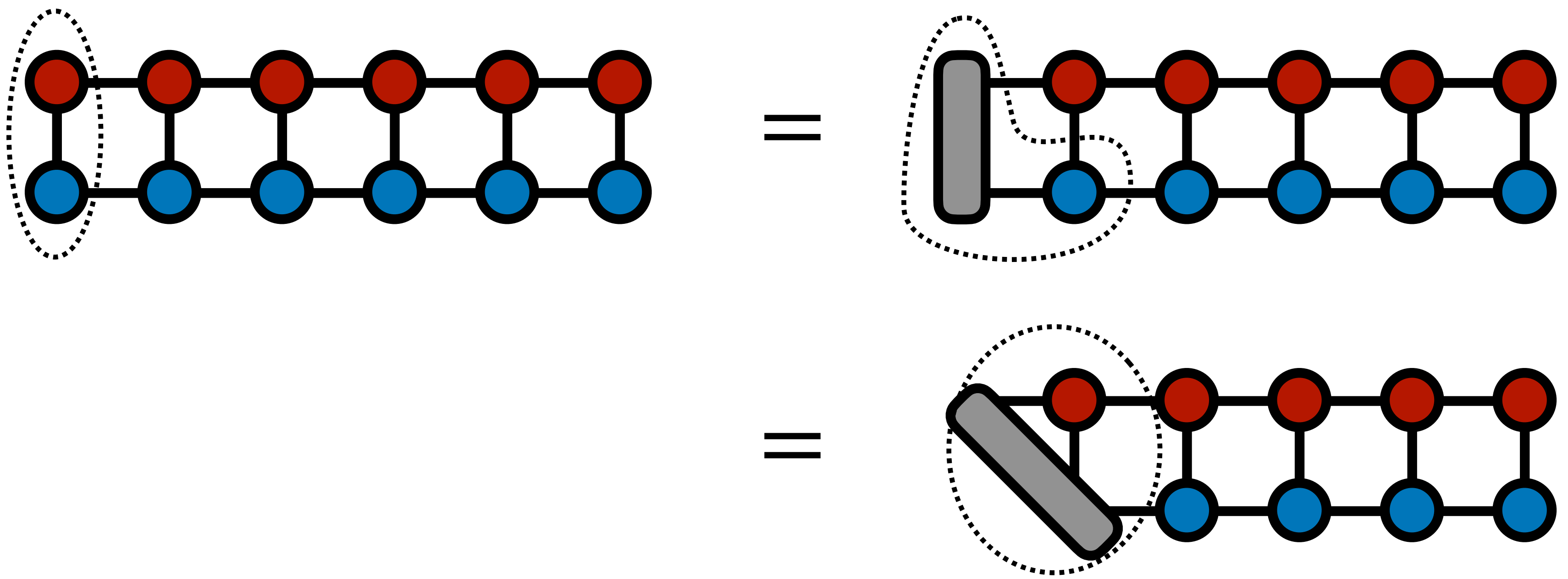
More detailed tensor network algorithm

Inner product of two MPS tensors



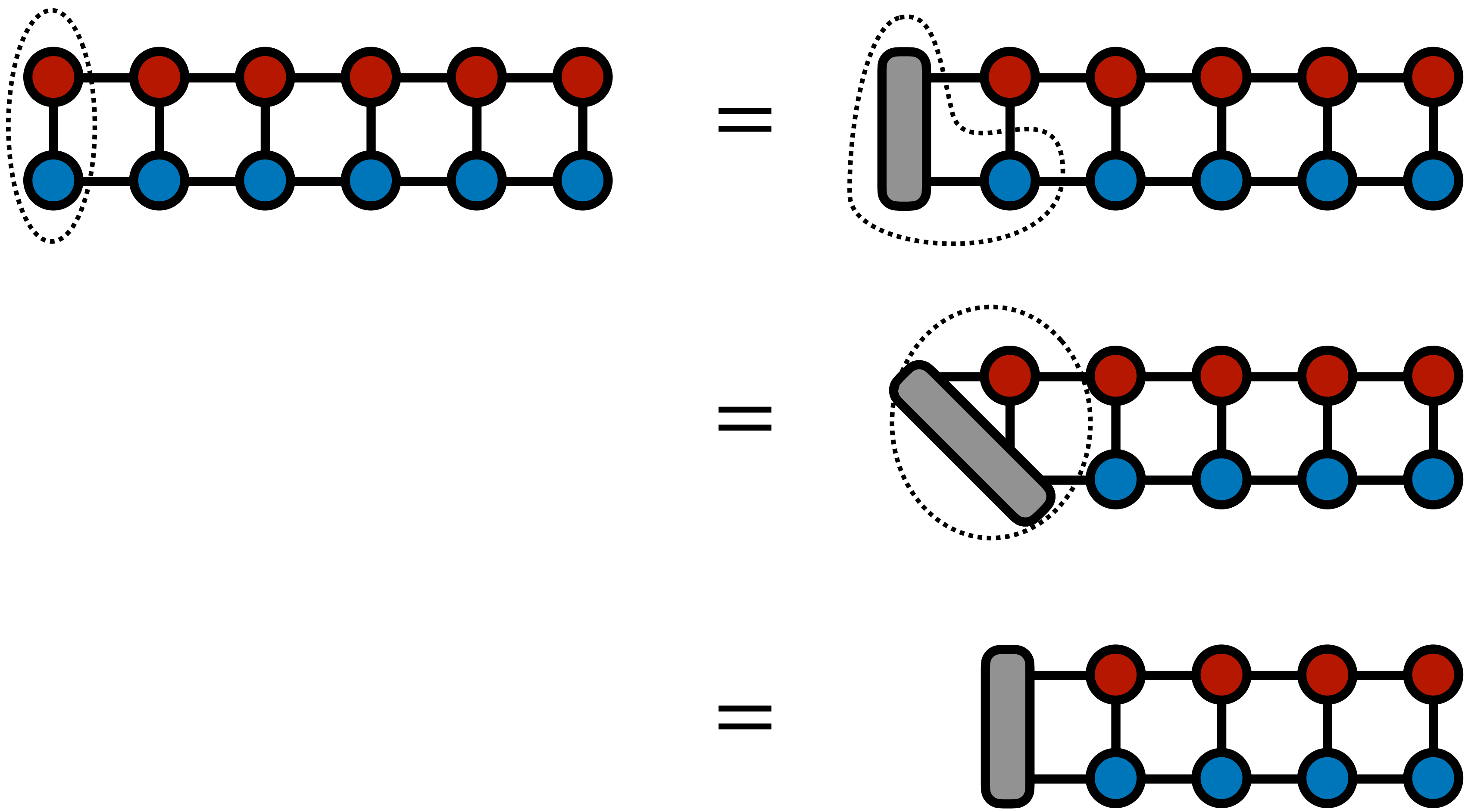
More detailed tensor network algorithm

Inner product of two MPS tensors



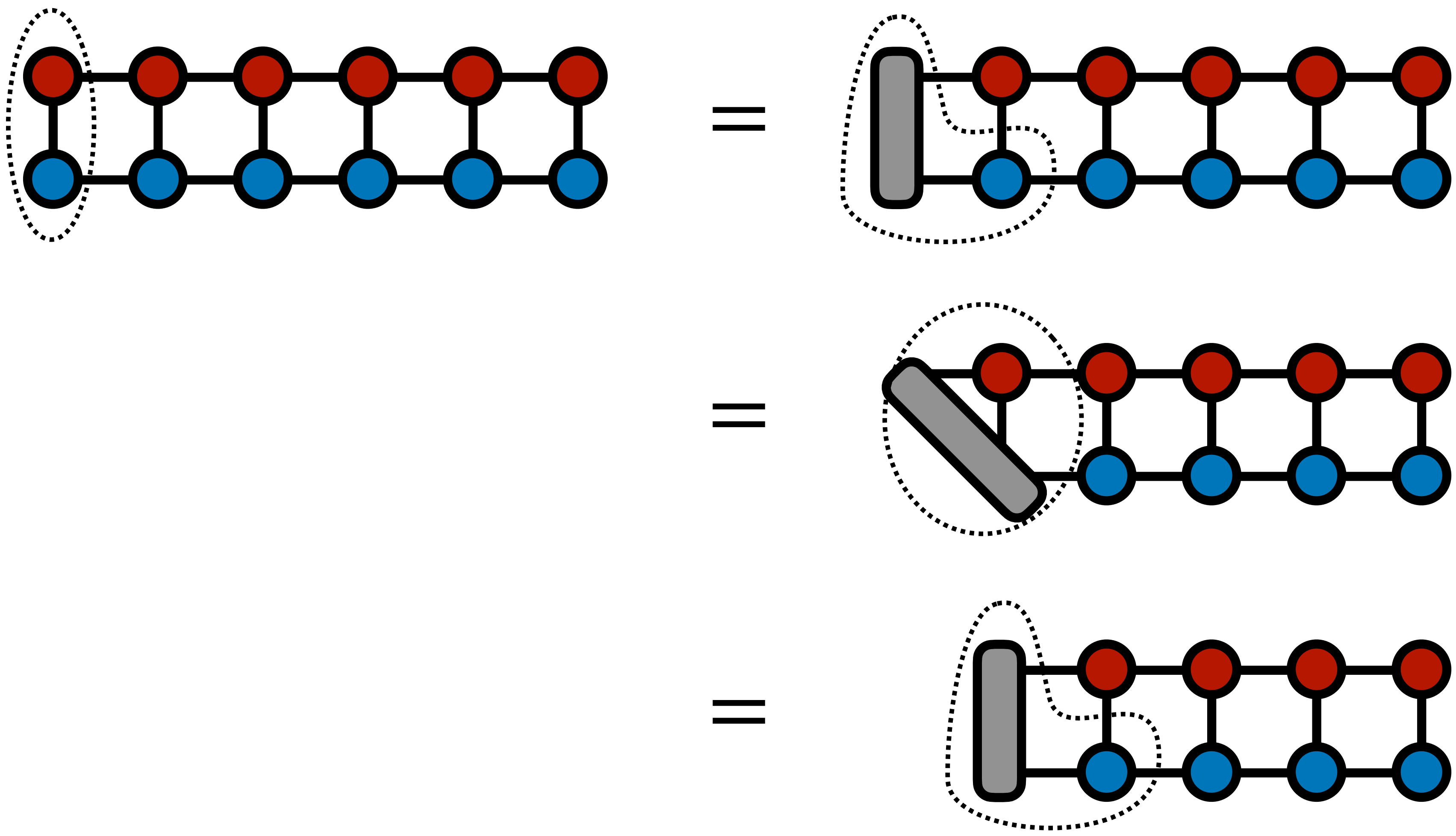
More detailed tensor network algorithm

Inner product of two MPS tensors



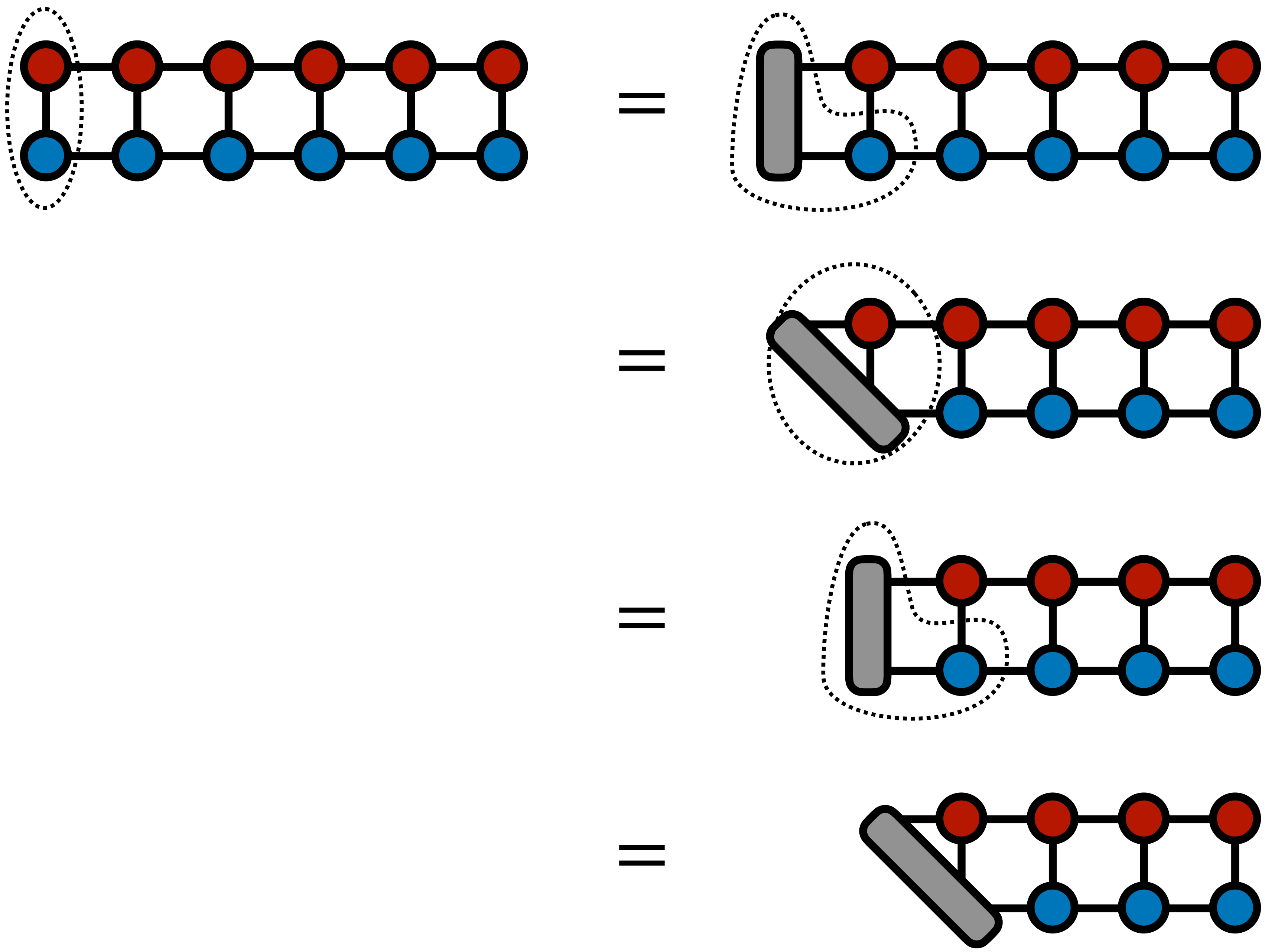
More detailed tensor network algorithm

Inner product of two MPS tensors



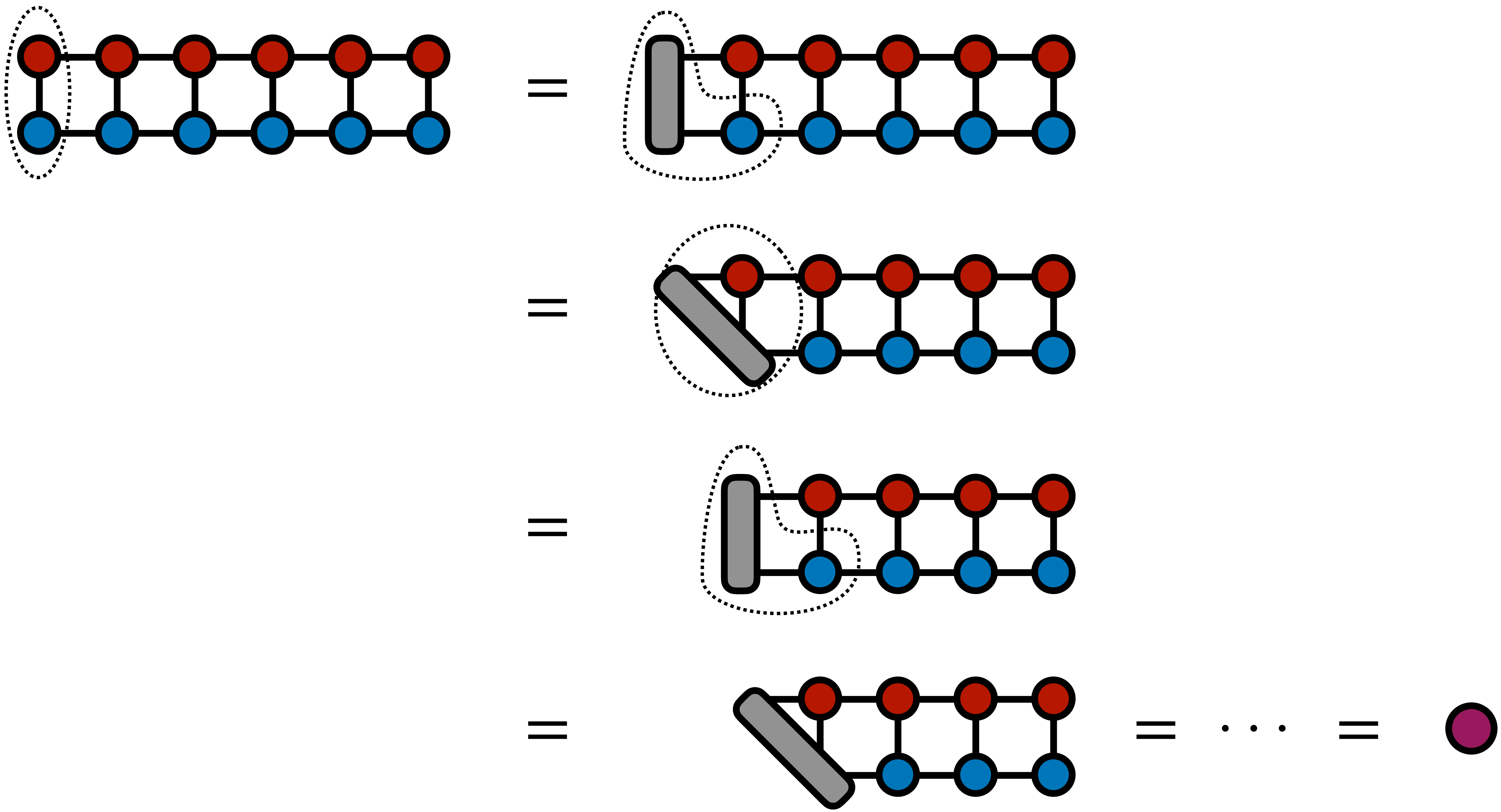
More detailed tensor network algorithm

Inner product of two MPS tensors



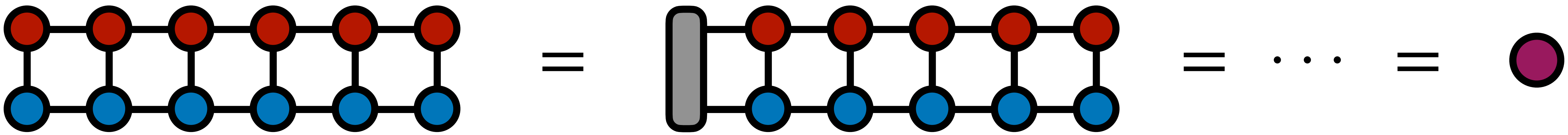
More detailed tensor network algorithm

Inner product of two MPS tensors



Cost $\sim \chi^3$, memory usage $\sim \chi^2$

Inner product of two MPS tensors

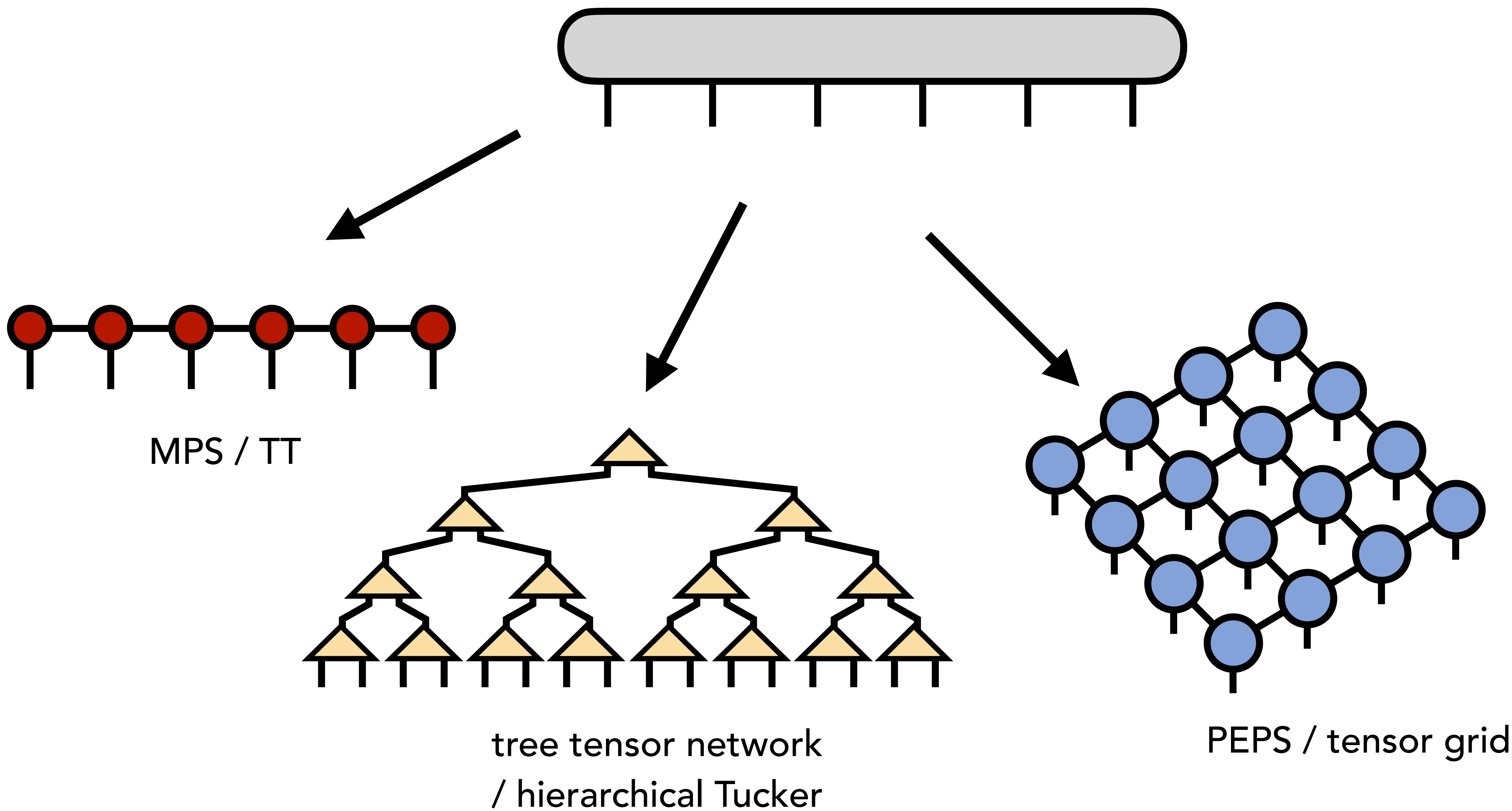


If each MPS represents tensor with 40 indices of dimension 2

Then above algorithm computes dot product of two 'vectors' of a *trillion* entries each

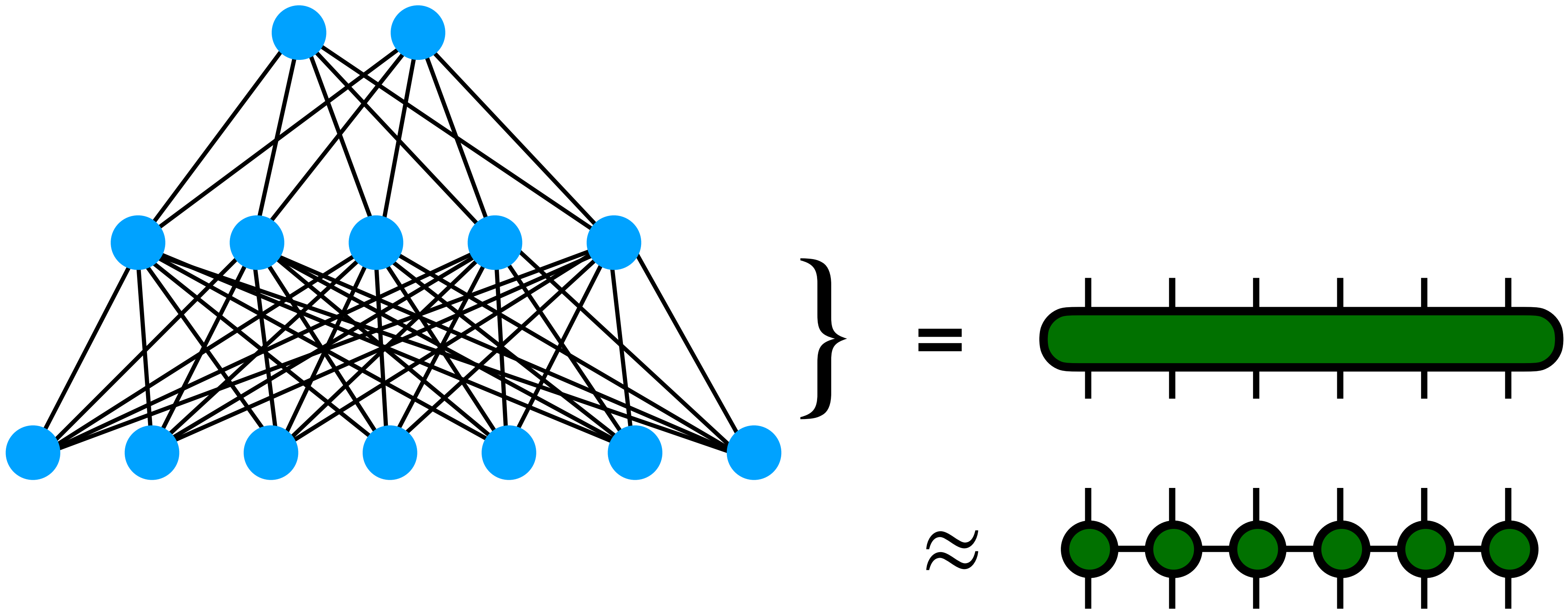
Takes ~ 10ms for bond dimension $\chi = 100$

There are other tensor networks too,
with their own algorithms and degrees of expressive power



Applications of Tensor Networks

Compressing Neural Network Weight Layers



View weight layer (size 2^N x 2^N)
as <u>*tensor*</u> with 2N indices (each of dimension 2)

Training through tensor-network approx. of weight layer yields state-of-art performance while giving 80x compression (only 1% decrease on CIFAR-10)

Novikov et al., "Tensorizing Neural Networks", NeurIPS 28 (2015)

Garipov et al., "Ultimate Tensorization...", NeurIPS (2016) arxiv:1611.03214

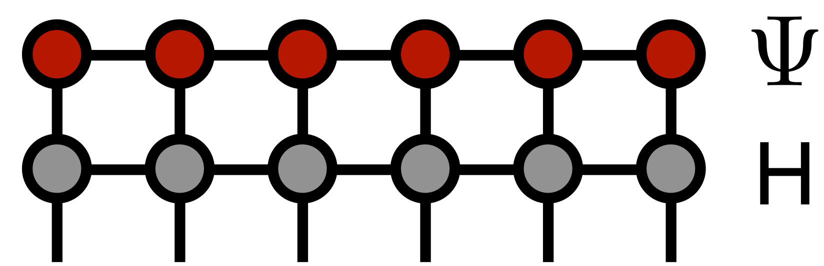
Solving PDE's with Tensor Networks

$$i\frac{\partial}{\partial t}\Psi(\{\mathbf{r}\},t) = H\Psi(\{\mathbf{r}\},t)$$

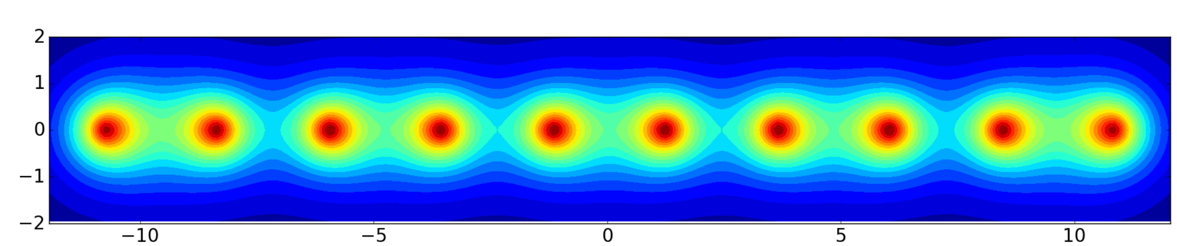
Schrödinger equation, quantum mechanics

Discretize H and Ψ by introducing a basis set

Represent by tensor networks:



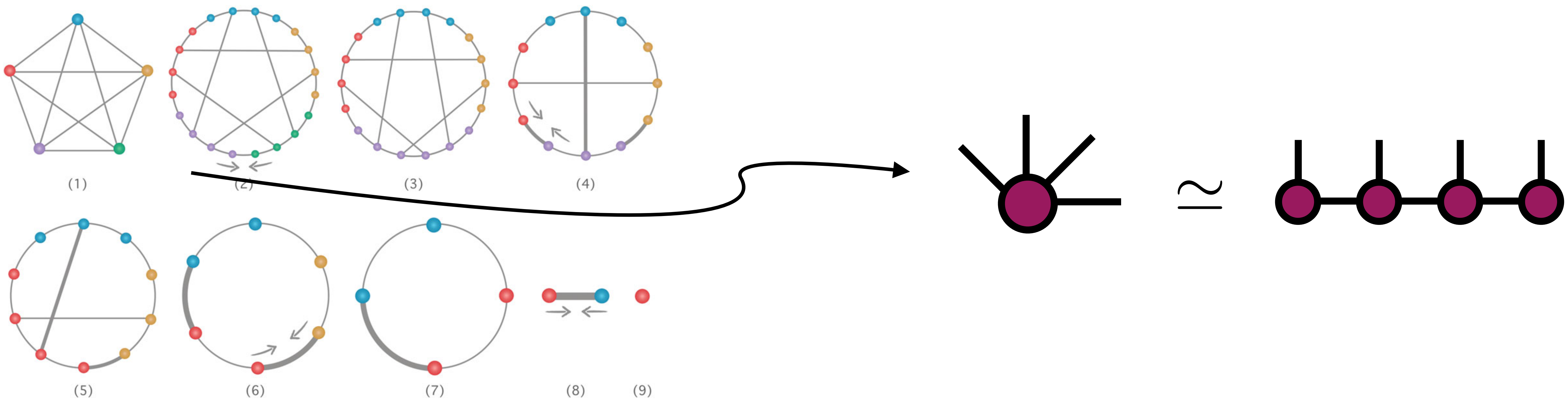
Stationary solution for 1000's of Hydrogen atoms:



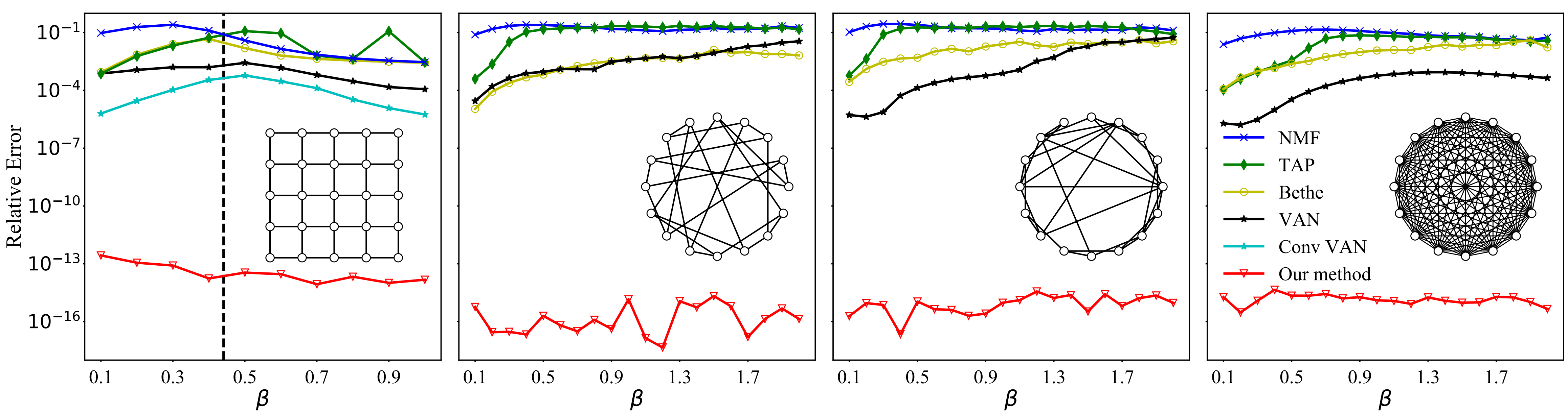
Stoudenmire, White, PRL 119, 046401 (2017)

Contracting Arbitrary Tensor Networks

Pan, Zhou, Li, Zhang, Phys. Rev. Lett. **125**, 060503 (2020)

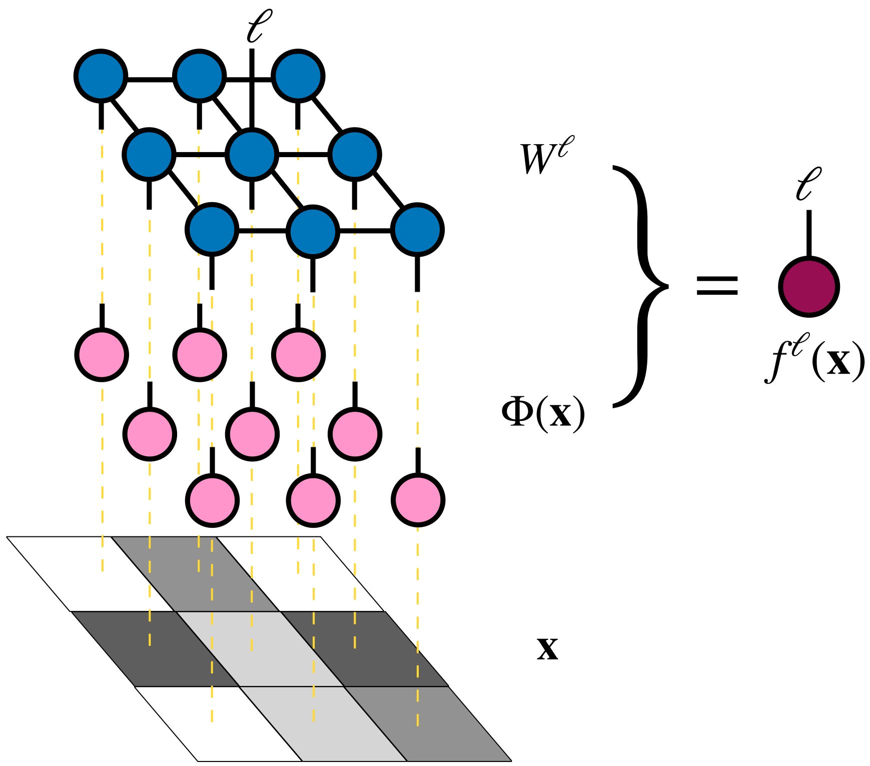


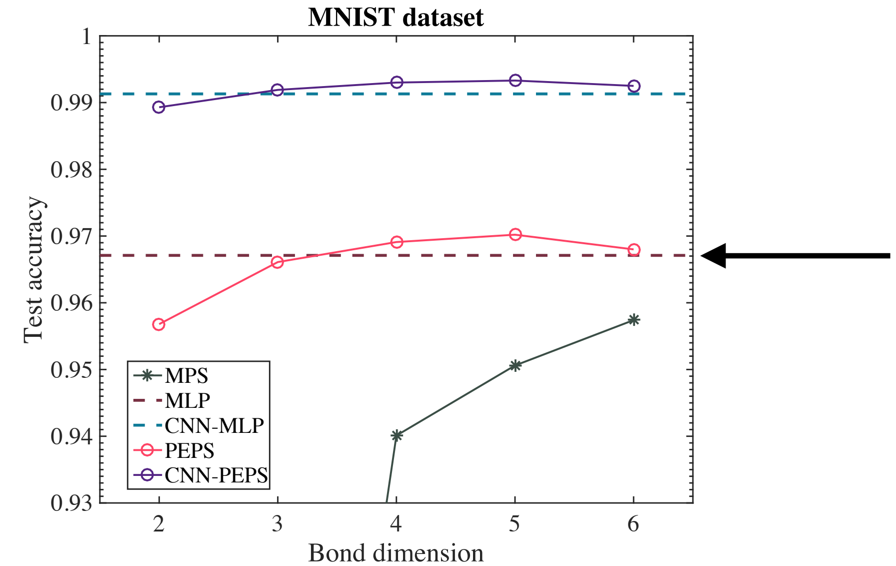
Application to graphical models (e.g. Ising spin glass):



See also: Jermyn, "Automatic Contraction of Unstructured Tensor Networks", SciPost Phys. 8, 005 (2020)

Tensor Network Machine Learning Models





neural net

Only beaten by one other model for FashionMNIST dataset

Cheng, Wang, Zhang, "Supervised Learning with PEPS" arxiv:2009.09932

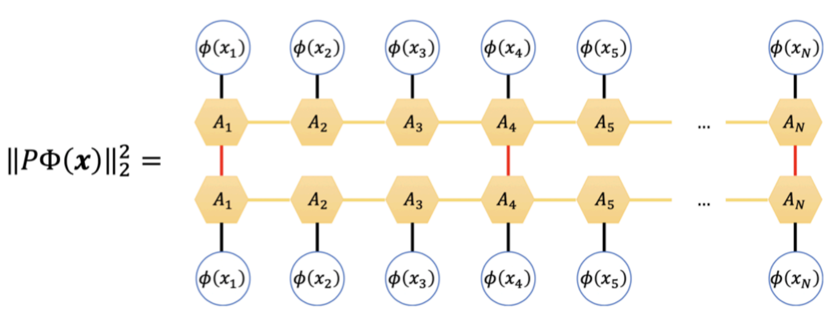


Table 3: Mean AUROC scores (in %) and standard errors on ODDS datasets.

Dataset	OC-SVM	IF	GOAD	DAGMM	TNAD
Wine	**60.0**	46.0 ± 8.4	48.2 ± 24.7	51.7 ± 19.3	$\mathbf{97.3 \pm 4.5}$
Glass	**62.0**	57.2 ± 1.6	53.5 ± 13.6	52.5 ± 12.9	$\mathbf{81.8 \pm 7.3}$
Thyroid	98.8	$\mathbf{99.0 \pm 0.1}$	95.8 ± 1.3	88.8 ± 6.8	$\mathbf{99.0 \pm 0.1}$
Satellite	**79.9**	77.2 ± 0.9	60.6 ± 5.3	72.1 ± 4.7	$\mathbf{81.3 \pm 0.5}$
Forest	**97.7**	71.7 ± 2.6	64.6 ± 4.7	60.9 ± 8.9	$\mathbf{98.8 \pm 0.6}$

New state-of-the-art result for anomaly detection task for tabular (heterogeneous) data

Geometrical framework for anomaly detection

Wang, Roberts, Vidal, Leichenauer, "Anomaly detection with tensor networks" arxiv:2006.02516

Quantum-inspired machine learning

Friday, October 23, 2020
10:00 – 18:00

Google Calendar · ICS

FRIDAY, OCTOBER 23, 2020 ON ZOOM
10:00 AM - 6:00 PM EDT

REGISTER TO JOIN

Anomaly detection with tensor networks
Stefan Leichenauer, X (Google)

Language modeling with reduced densities
Tai-Danae Bradley, X (Google)

Probabilistic modeling with tensor networks
Jacob Miller, Université de Montreal

Progress in tensor network algorithms for machine learning:
Miles Stoudenmire, Flatiron Institute

Differentiable programming with tensor networks
Lei Wang, Institute of Physics, Chinese Academy of Sciences

Organizers:
David Schwab, The Graduate Center, CUNY
John Terilla, Queens College and The Graduate Center, CUNY

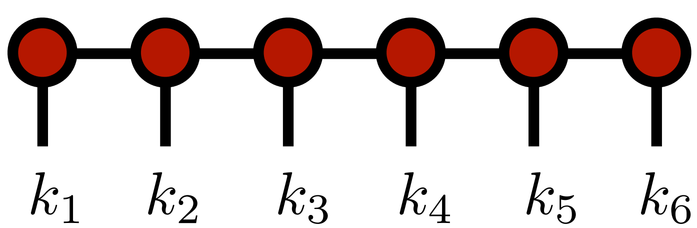
High-Dimensional Integration with Tensor Networks

Goal to compute $$I = \int_{[0,1]^d} f(x_1, x_2, .., x_d)$$

1. approximate as sum (quadrature):

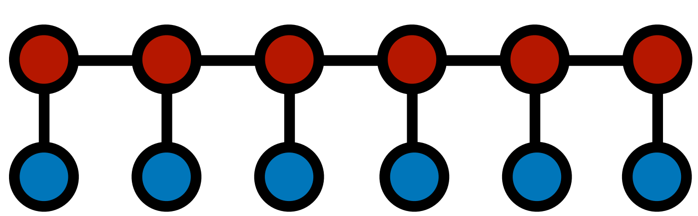
$$I \simeq \sum_{\mathbf{k}} f(x_{k_1}, x_{k_2}, \ldots, x_{k_d}) w_{k_1} w_{k_2} \cdots w_{k_d}$$

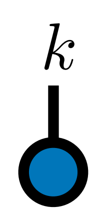
2. optimize MPS to represent f (most expensive step)



$$\simeq f(x_{k_1}, x_{k_2}, \ldots, x_{k_d})$$

3. sum with weights – very efficient

$I \simeq$ 



$= w_k$

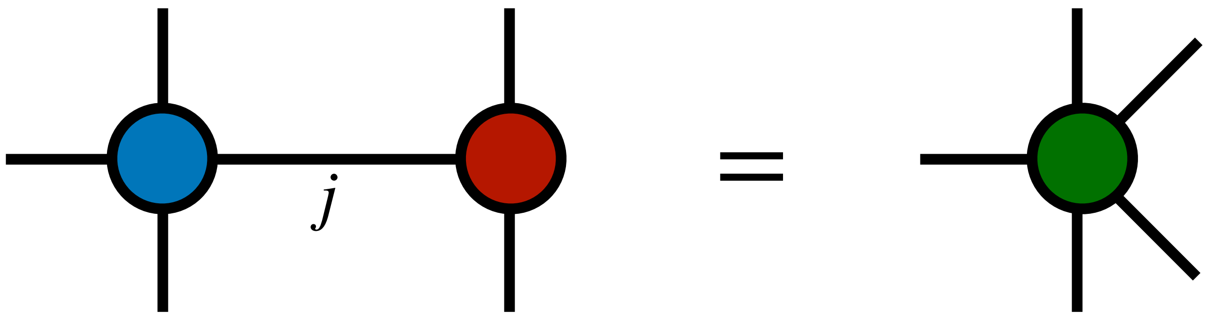
The ITensor Software

Inspired by tensor diagrams

For tensor network algorithms,

contractions take up the majority of:

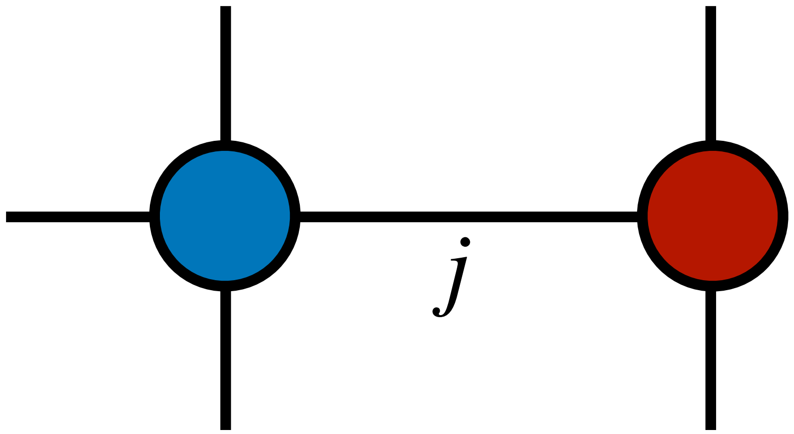
- conceptual steps (= correctness of algorithm)
- computational time



$$\sum_j A_{ijk} B_{mjp} = C_{ikmp}$$

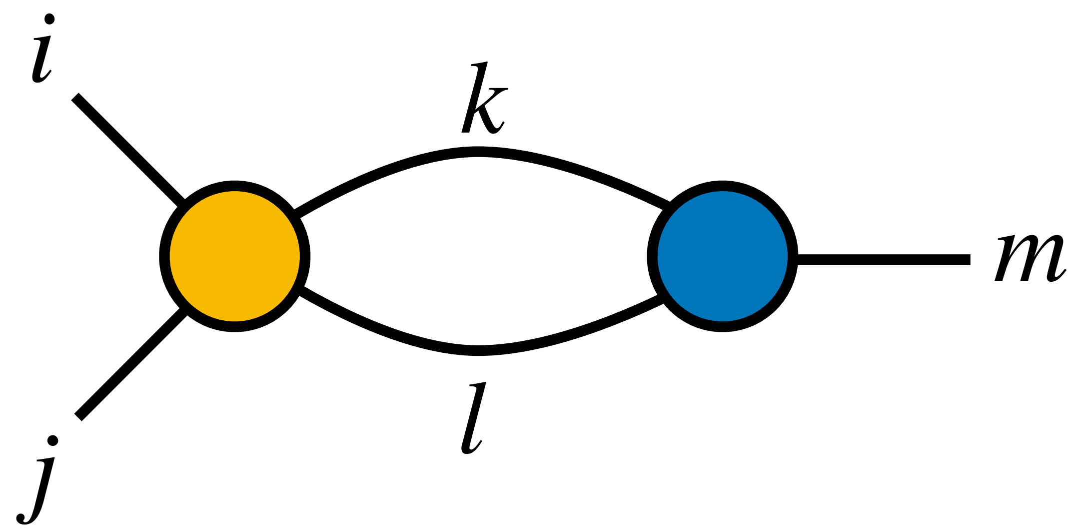
What can go wrong?

- contract the wrong indices
- too much human time inputting contractions
- take too long to compute



$$\sum_{j} A_{ijk} B_{mjp}$$

Conventional tensor library (*not ITensor*)

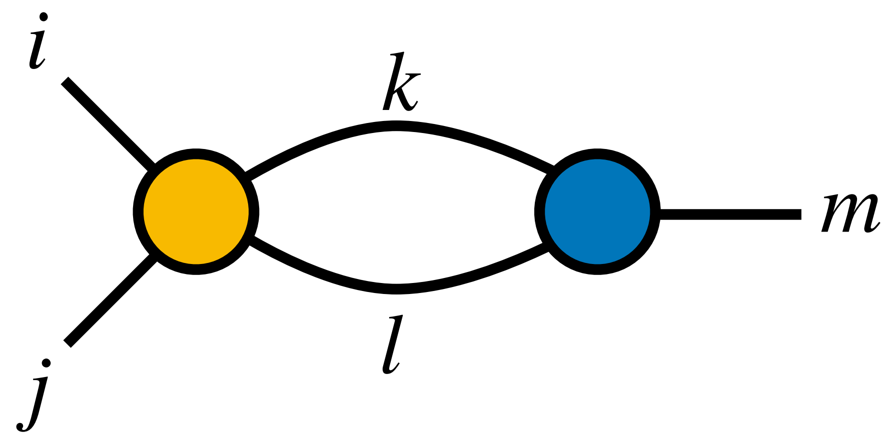



```
C = A["i,j,k,l"] * B["k,l,m"]
```

- Index labels are temporary
- Must think about index ordering
- Possible to mistake same-size indices

ITensor introduces "intelligent" indices

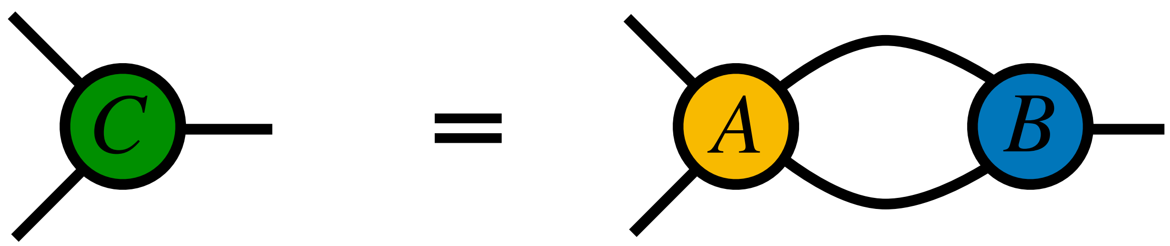
which recognize each other




```
k = Index(5,"k")

l = Index(7,"l")

...

A = ITensor(i,j,k,l)

B = ITensor(l,m,k)

...

C = A * B
```

ITensor introduces "intelligent" indices

which recognize each other:




```
C = A * B
```

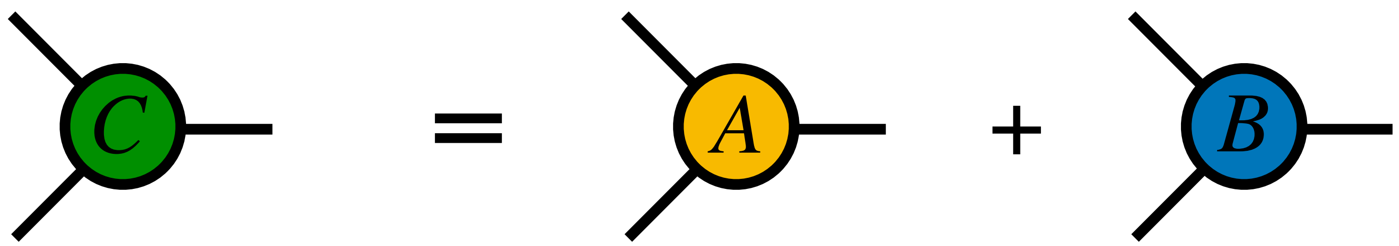
Immediately rules out:

- mental burden of index ordering
- contraction of wrong indices

Only think about *topology* of network –

like tensor diagrams

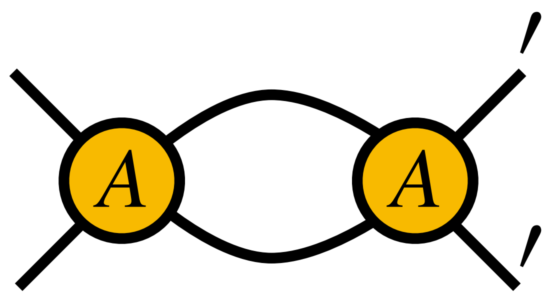
Adding ITensors "just works"




```
C = A + B
```

No thinking about index ordering

To prevent indices from contracting



Can put "primes" on indices and remove them after

$$\sum_{kl} A_{ijkl} A_{i'j'kl}$$

```
R = A * prime(A,i,j)
```

ITensor – Summary

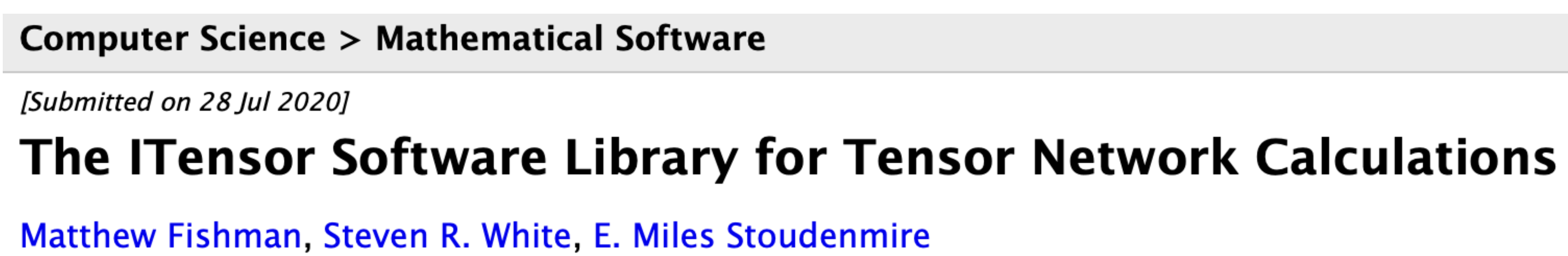
Tensor library with unique interface to accelerate development, reduce bugs

Ported in 2020 to the Julia programming language

Delivering on speed, rapid development times


Matt Fishman (CCQ ADS)

New paper to appear in SciPost Phys. Codebases:


Computer Science > Mathematical Software

[Submitted on 28 Jul 2020]

The ITensor Software Library for Tensor Network Calculations

Matthew Fishman, Steven R. White, E. Miles Stoudenmire

Further Topics

Tensor network optimization algorithms (putting numbers into a T.N. for some task):

- DMRG / alternating least-squares
- density-matrix algorithm
- TT-cross / skeleton algorithm
- ...

Other applications:

- simulating quantum computers
- large-scale PCA and other iterative methods
- branch-and-bound spin glass algorithm
- ...

Computational strategies

- tensor renormalization group
- block-sparse tensor networks
- ...

Concluding Thoughts

With hindsight, tensor networks may be "right" way to do linear algebra in exponentially large spaces

Big developments in tensor-network algorithms still to come (e.g. analogues of matrix factorizations)

Intimate connection to hierarchical matrices only beginning to be understood

Probably still under-used – many application domains to be explored. Yours may be next – let's discuss!

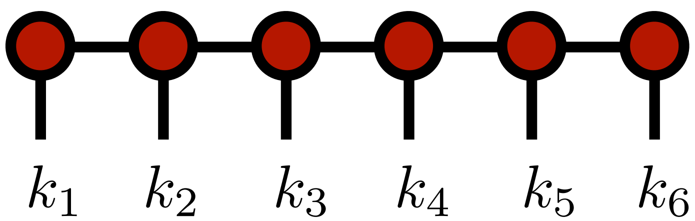
High-Dimensional Integration with ITensor

http://itensor.org/sine_int.jl

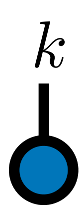
http://itensor.org/sine_util.jl

$$I = \int_{[0,1]^d} f(x_1, x_2, .., x_d)$$

$$\simeq \sum_{\mathbf{k}} f(x_{k_1}, x_{k_2}, \ldots, x_{k_d}) w_{k_1} w_{k_2} \cdots w_{k_d}$$

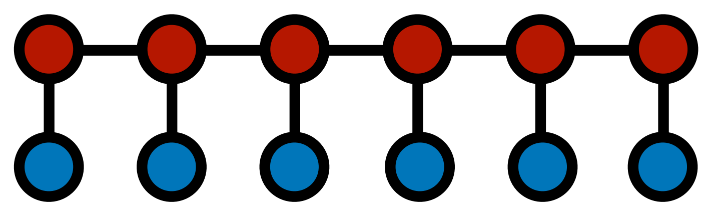


$$\simeq f(x_{k_1}, x_{k_2}, \ldots, x_{k_d})$$

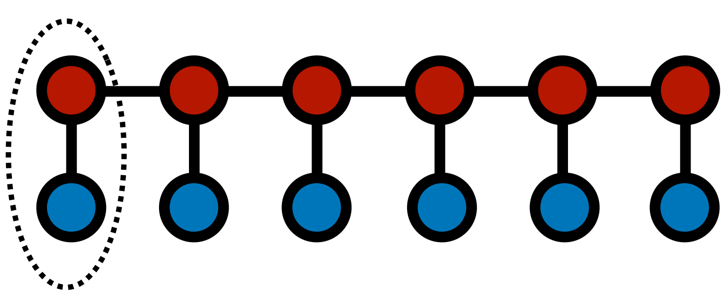


$$= w_k$$

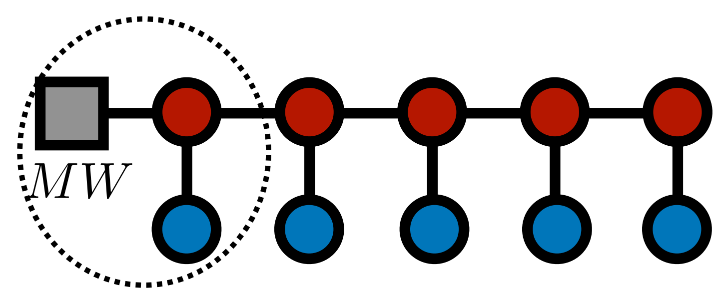
$$I \simeq$$



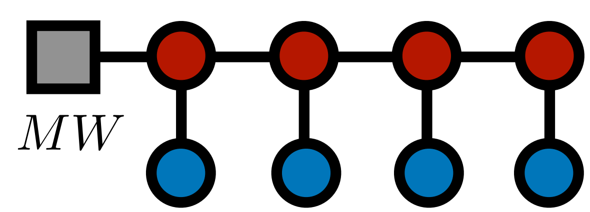
$$I \simeq$$



$$=$$



$$=$$



$$= \cdots =$$