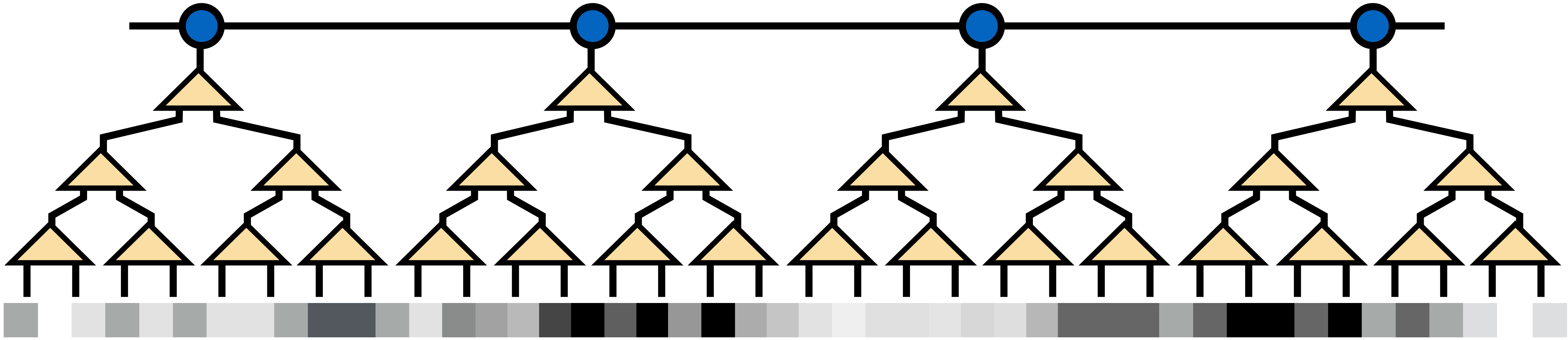
Tensor Networks for Machine Learning



E.M. Stoudenmire

May 2024 - Padua ETN School

FLATIRON INSTITUTE

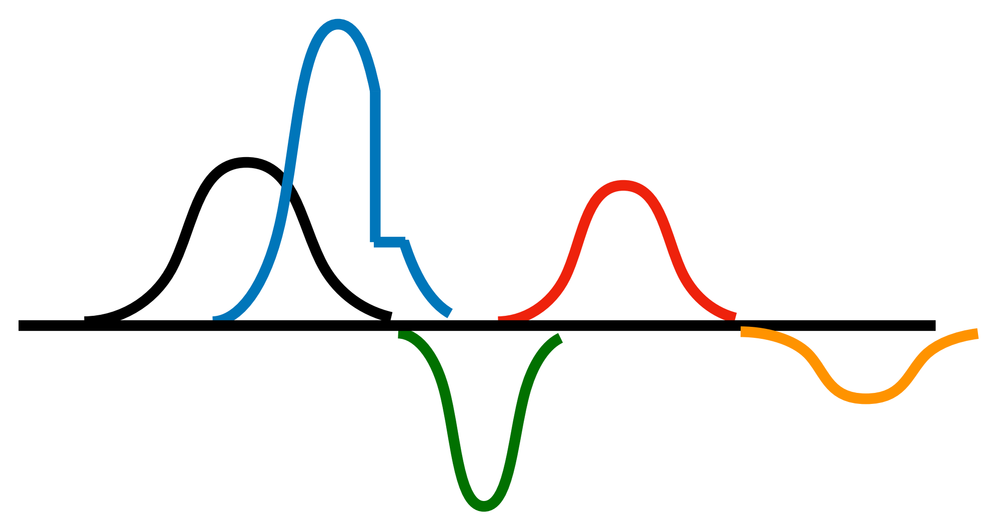
Sample Codes

Quick demo – tensor ML is powerful!

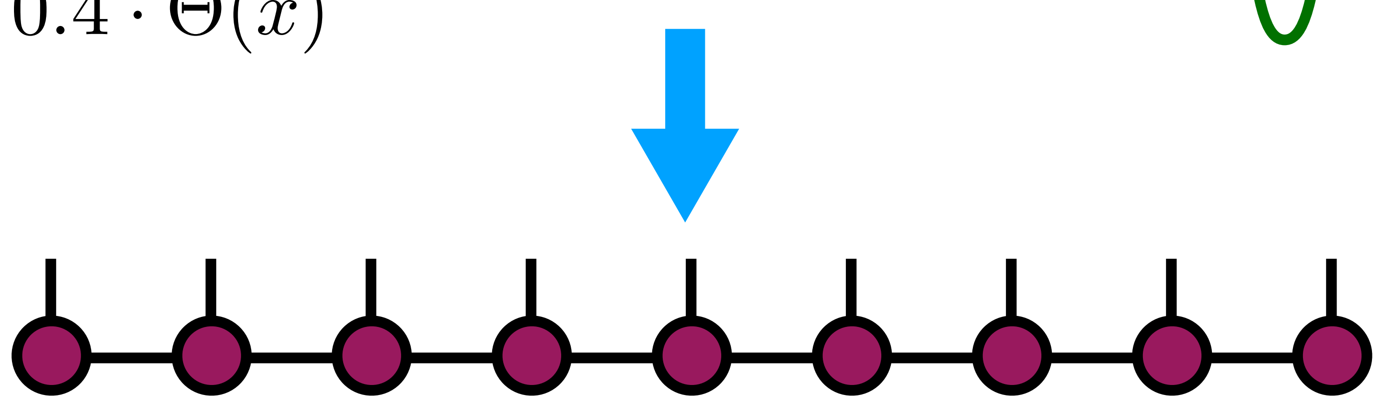
Let's machine learn the following function
into a tensor network:

40 Gaussians, random location, width, & height
+ a sharp step at 0.4

$$f(x) = \sum_{g=1}^{N_g} a_g e^{-w_g (x - x_g)^2}$$
$$+ 0.4 \cdot \Theta(x)$$



$f(x) \approx$



Today's Talk

Brief review of tensor networks

Why tensor networks for machine learning?
Inspiration from DMRG.

Basis and amplitude encodings of data

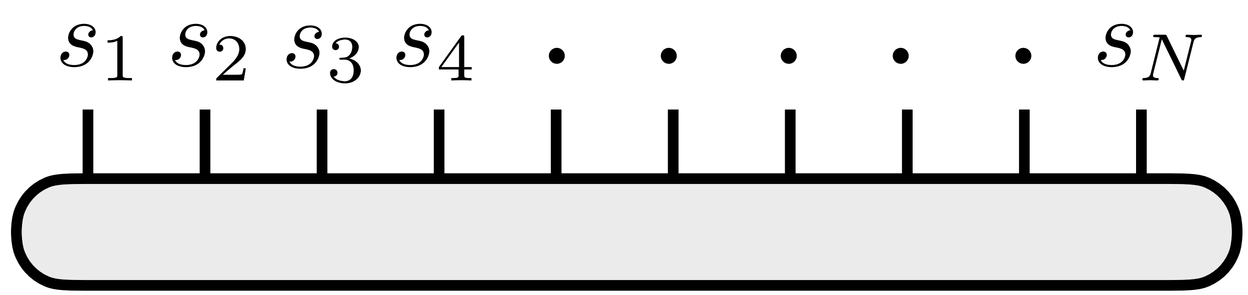
Example applications

Future of tensor network machine learning

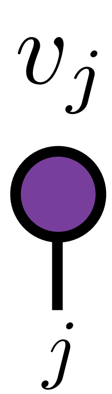
Tensors – Penrose Diagram Notation

N-index tensor = shape with N lines

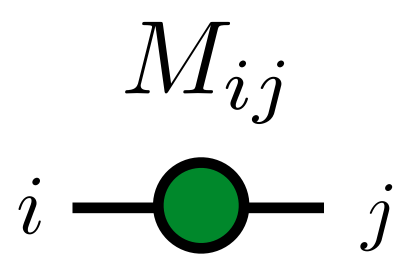
$$T^{s_1 s_2 s_3 \cdots s_N} =$$



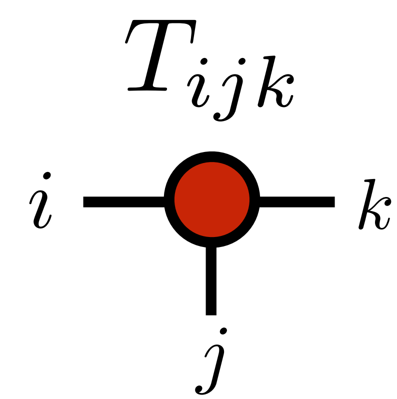
Low-order examples:



vector



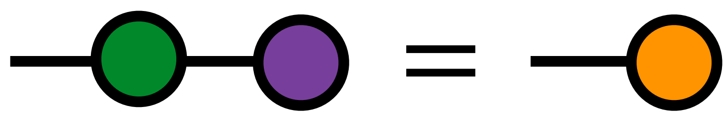
matrix



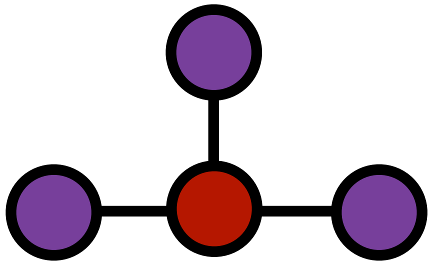
order-3 tensor

Tensors – Penrose Diagram Notation

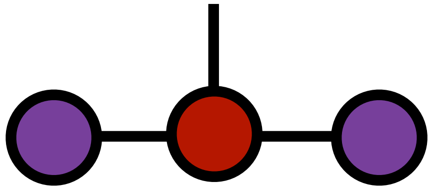
Joining wires means contraction:



$$\sum_j M_{ij} v_j = w_i$$



$$\sum_{i,j,k} T^{ijk} v_i v_j v_k$$



$$\sum_{i,k} T^{ijk} v_i v_k = z^j$$

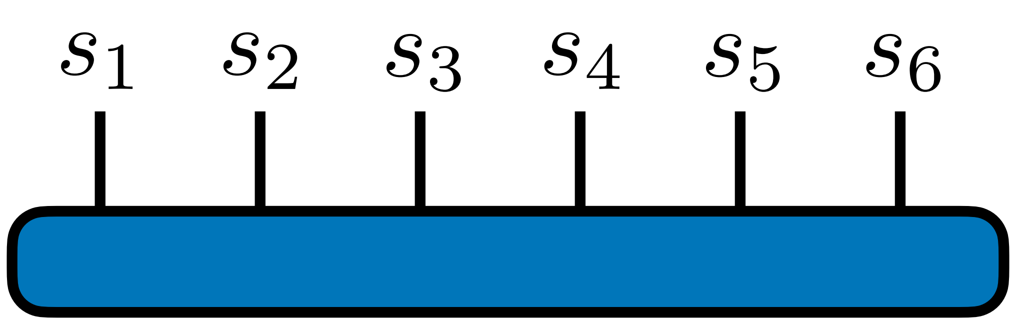
Tensors

We are familiar with tensors from quantum many-body

$$|\Psi\rangle = \sum_{s_1 s_2 s_3 \cdots s_n} \Psi^{s_1 s_2 s_3 \cdots s_n} |s_1 s_2 s_3 \cdots s_n\rangle \qquad s_j \in 0, 1$$

Amplitudes form a big tensor!

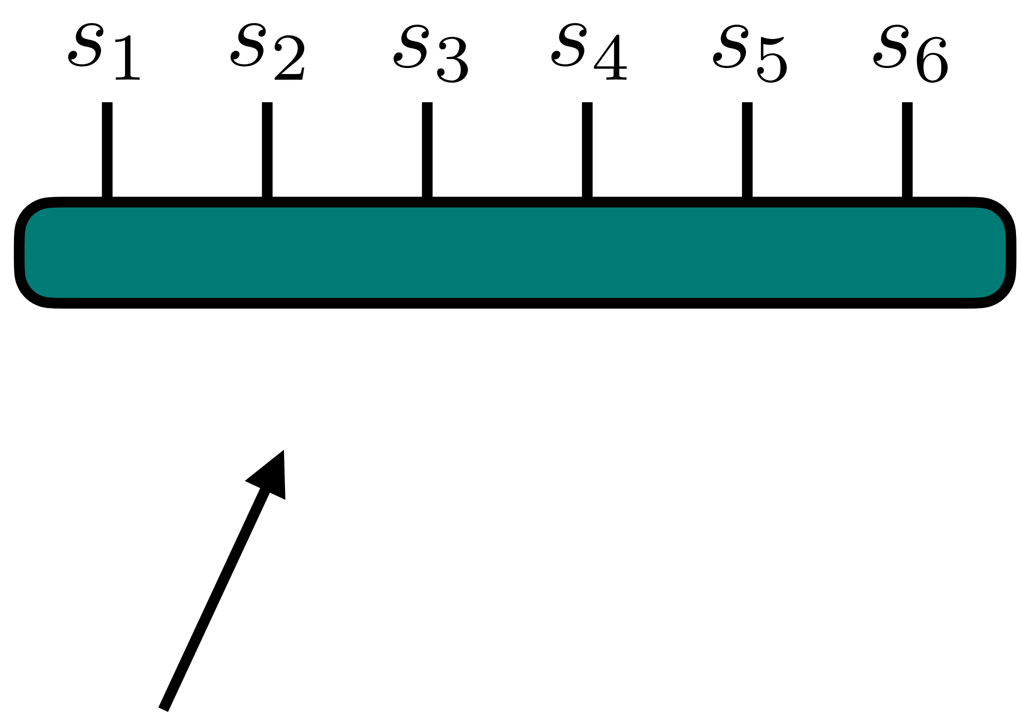
$$\Psi^{s_1 s_2 s_3 s_4 s_5 s_6} =$$



Tensors

Any function of discrete variables can be represented as a tensor

$$f(s_1, s_2, s_3, s_4, s_5, s_6) =$$

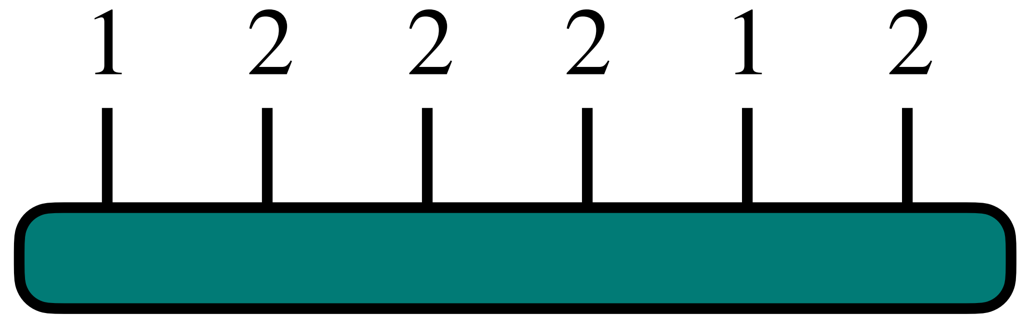


All values of f inside

Tensors

Any function of discrete variables can be represented as a tensor

$f(1, 2, 2, 2, 1, 2) =$

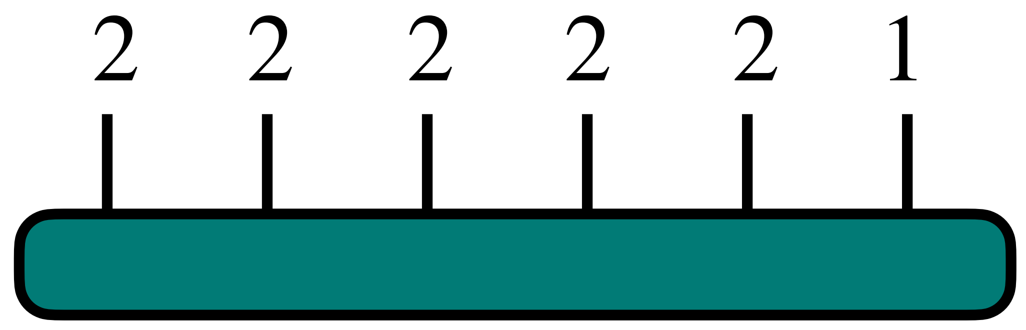


$= 1.2$

Tensors

Any function of discrete variables can be represented as a tensor

$$f(2, 2, 2, 2, 2, 1) \ =$$

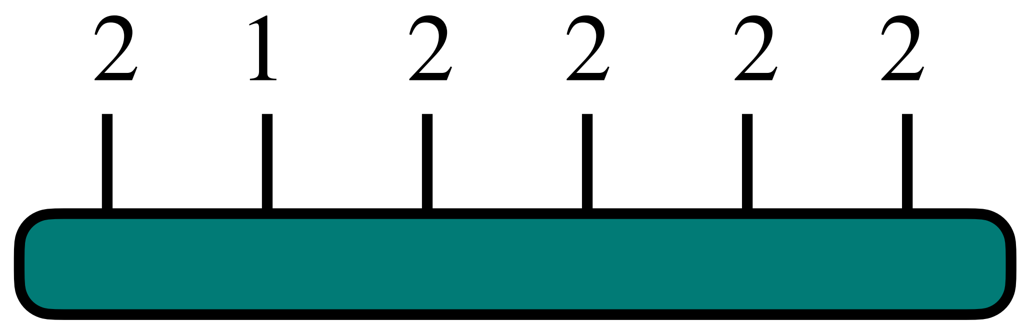


$$= \ 0.3$$

Tensors

Any function of discrete variables can be represented as a tensor

$f(2, 1, 2, 2, 2, 2) =$

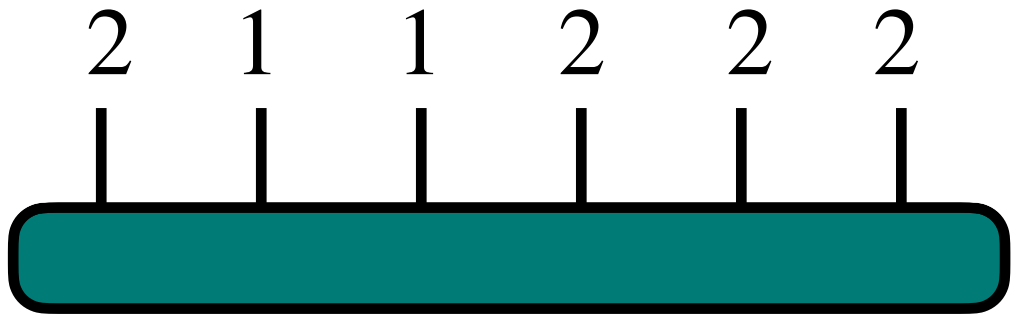


$= -0.2$

Tensors

Any function of discrete variables can be represented as a tensor

$$f(2,1,1,2,2,2) \;=$$

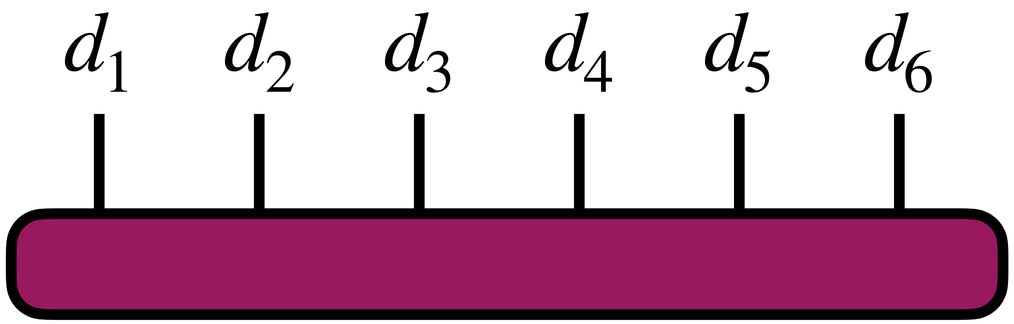


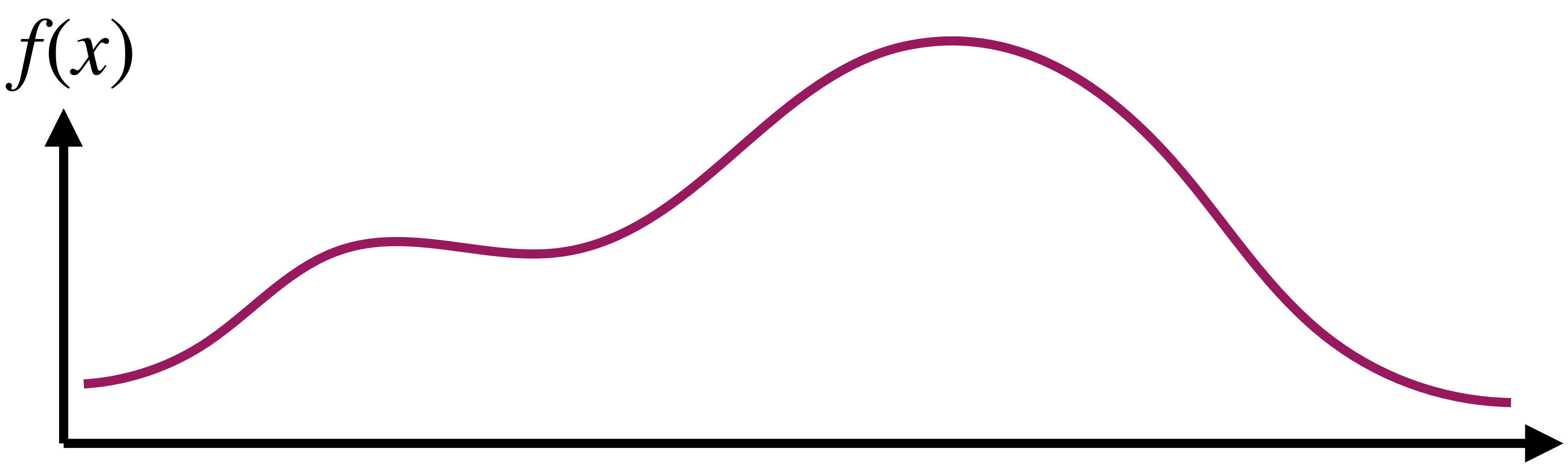
$$= \;\; 2.7$$

Tensors

Later we will see technique to encode **continuous variable** functions too

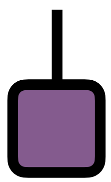
$$f(x) \approx f(0.d_1d_2d_3d_4d_5d_6) =$$



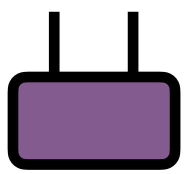


Tensor Networks

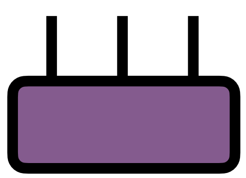
Why are tensors challenging?



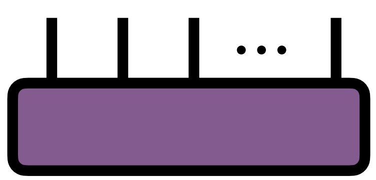
2 parameters (vector)



4 parameters (matrix)



8 parameters (3-index tensor)



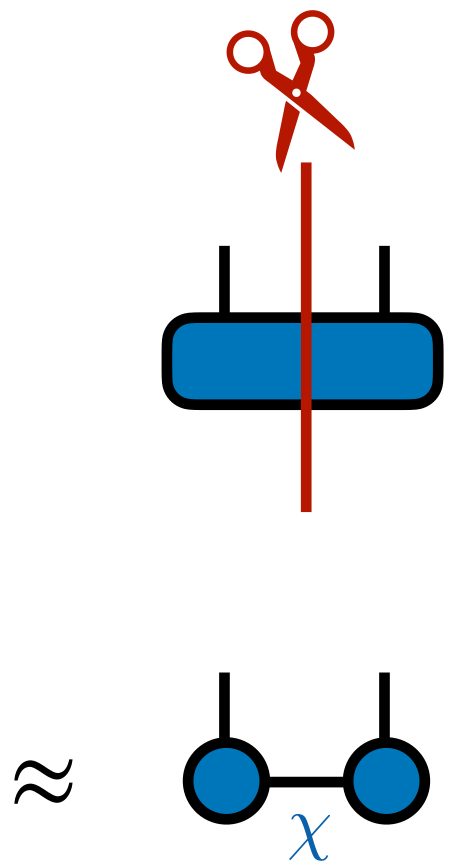
2^n parameters for n-index tensor

Tensor with 50 indices would have

1,125,899,906,842,624 ~ $\mathbf{10}^{15}$ parameters

Tensor Networks

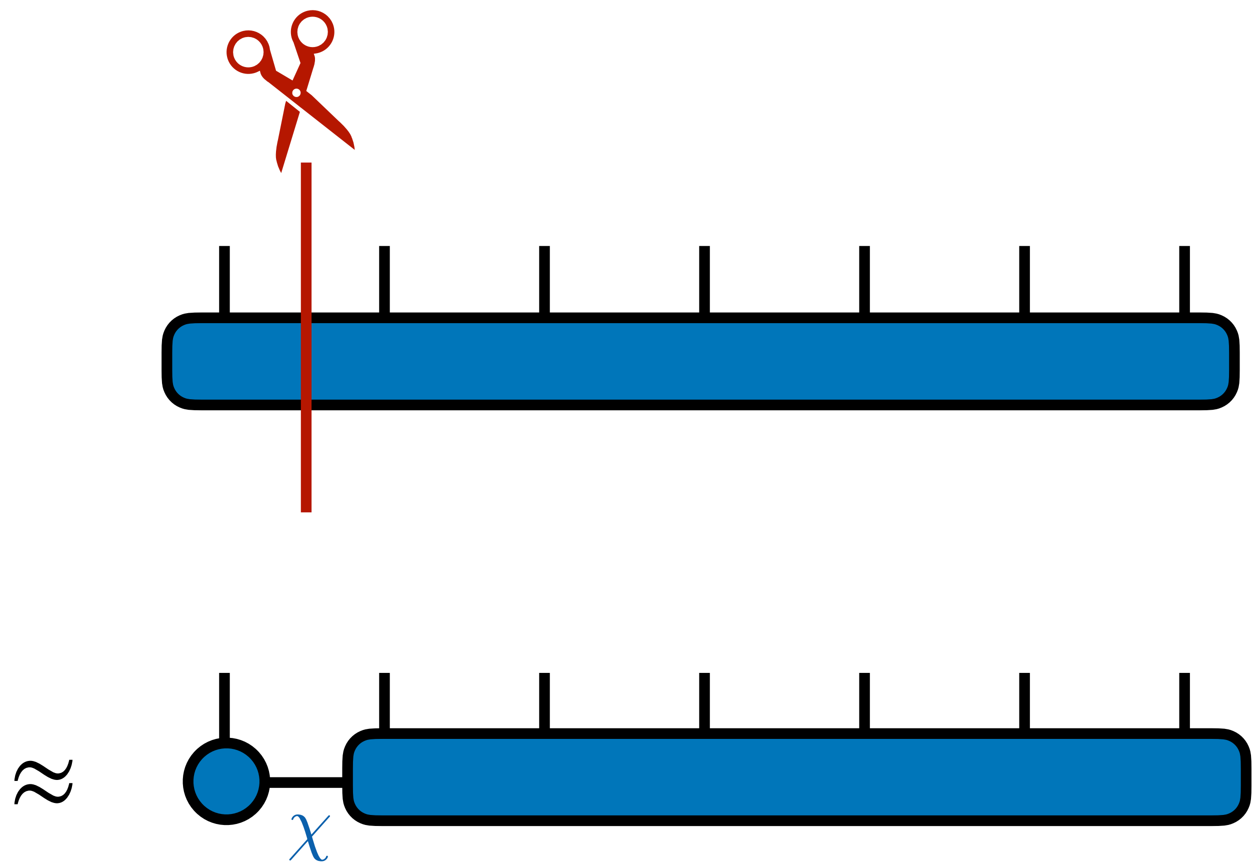
Just as factorizing (SVD) a matrix reduces cost of memory and compute



χ is matrix rank

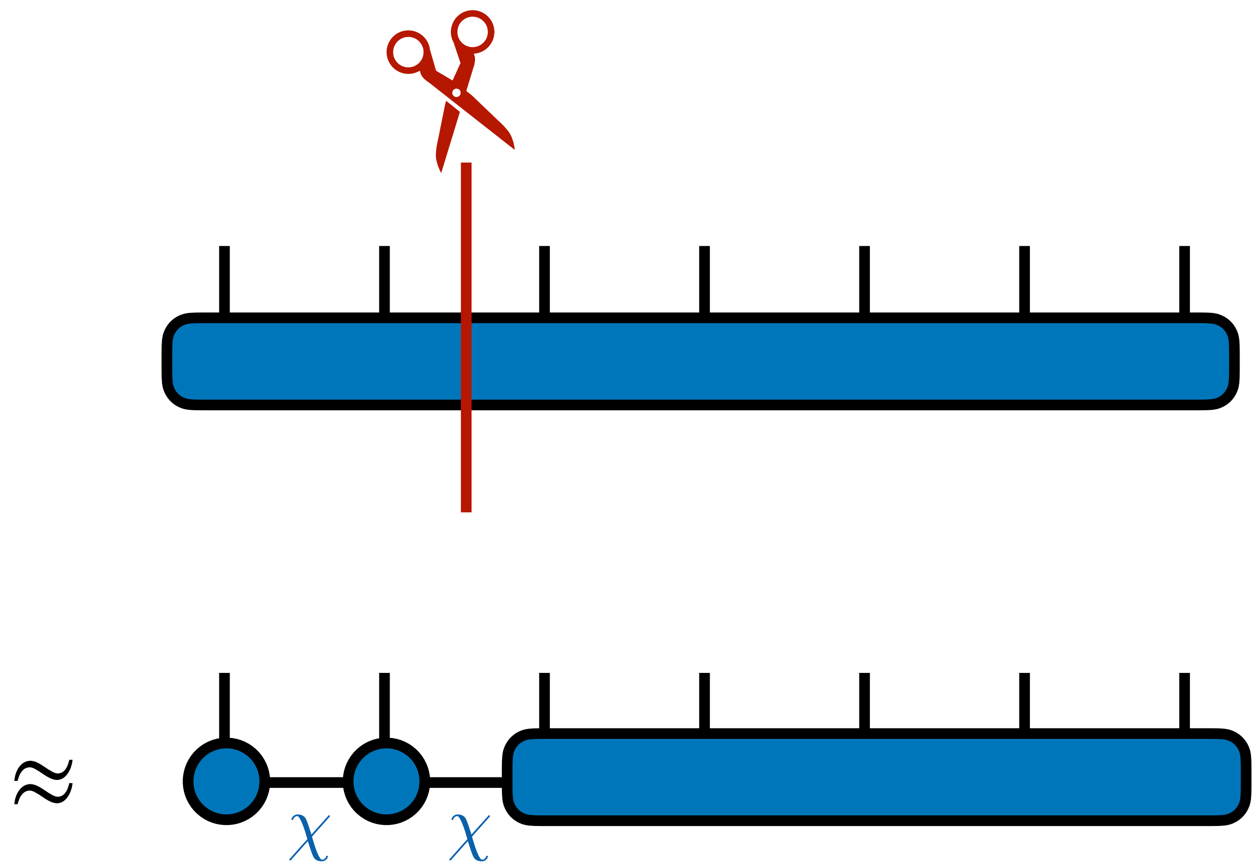
Tensor Networks

Can recursively factor (compress) a tensor as well



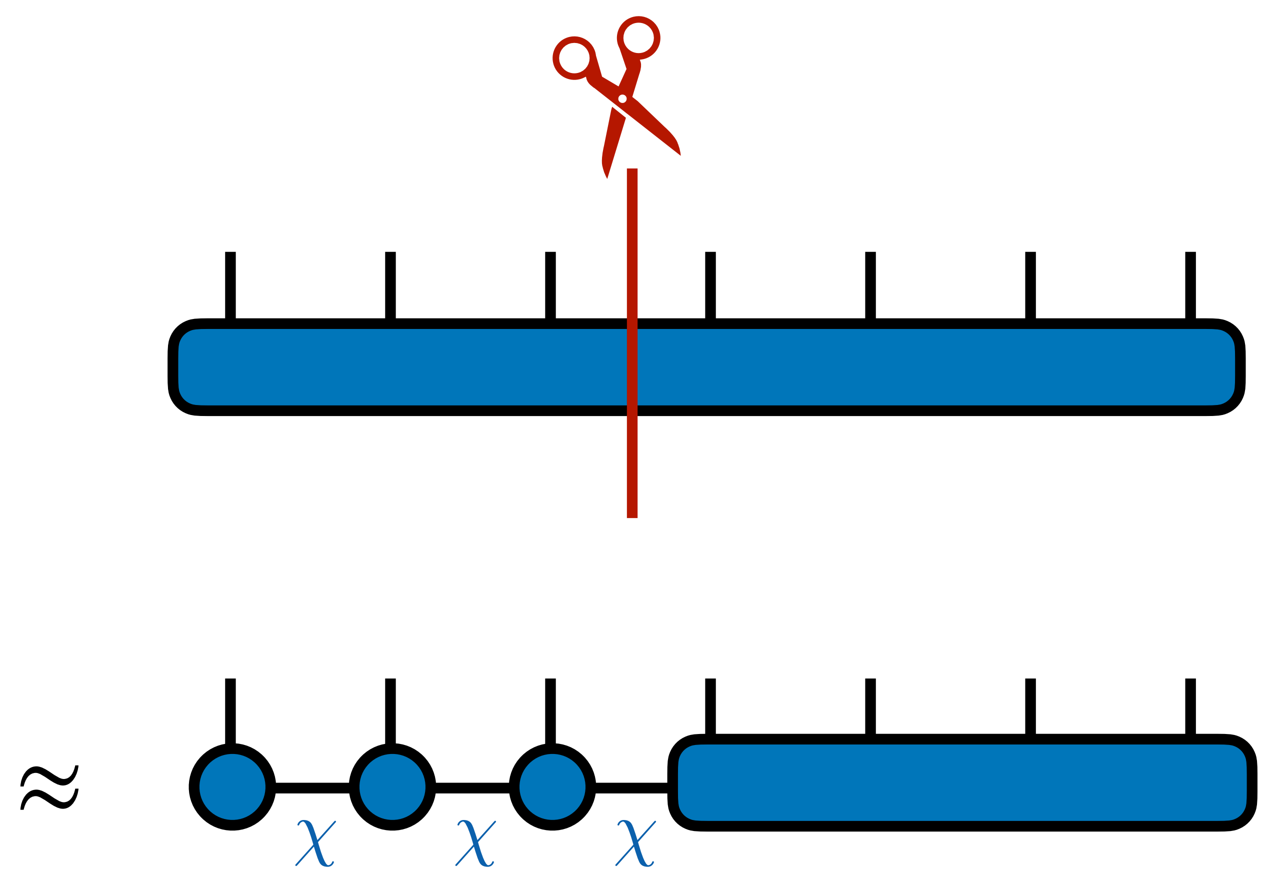
Tensor Networks

Can recursively factor (compress) a tensor as well



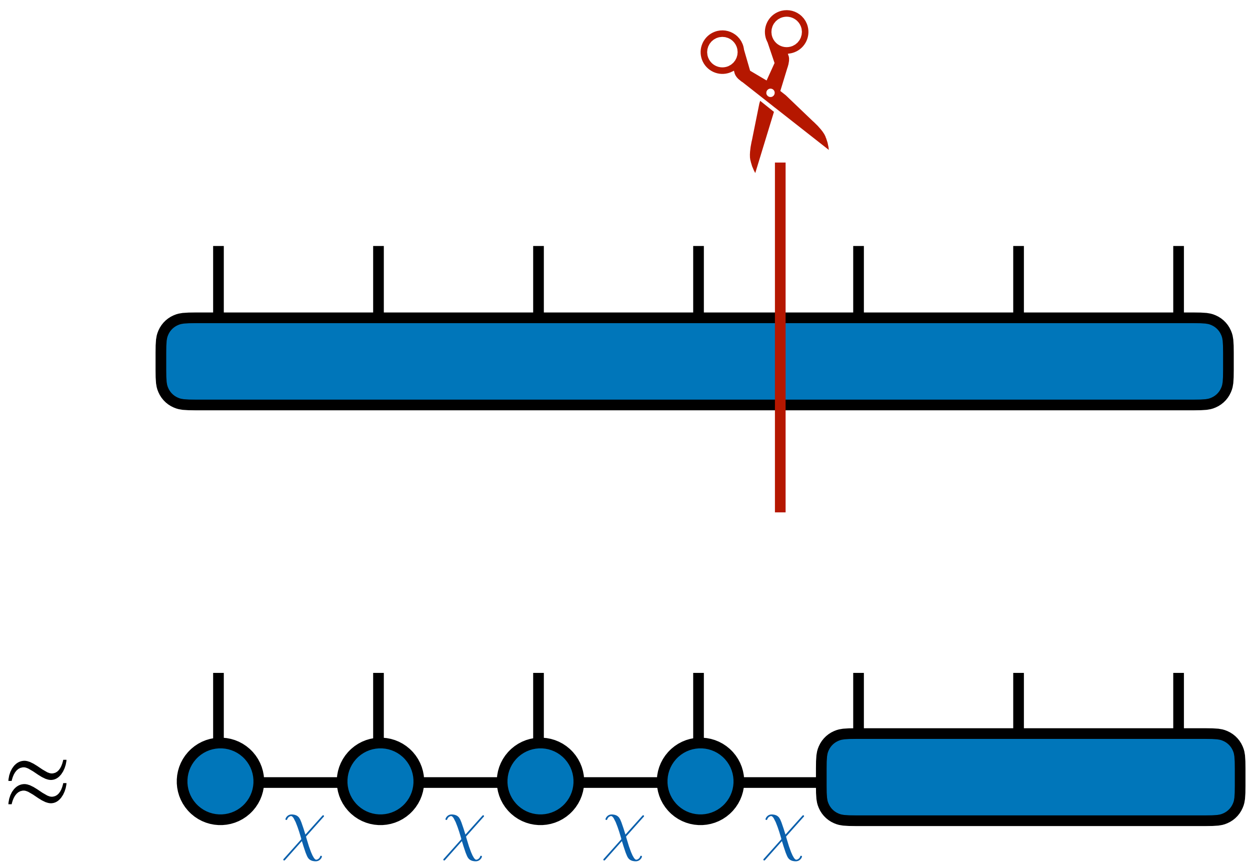
Tensor Networks

Can recursively factor (compress) a tensor as well



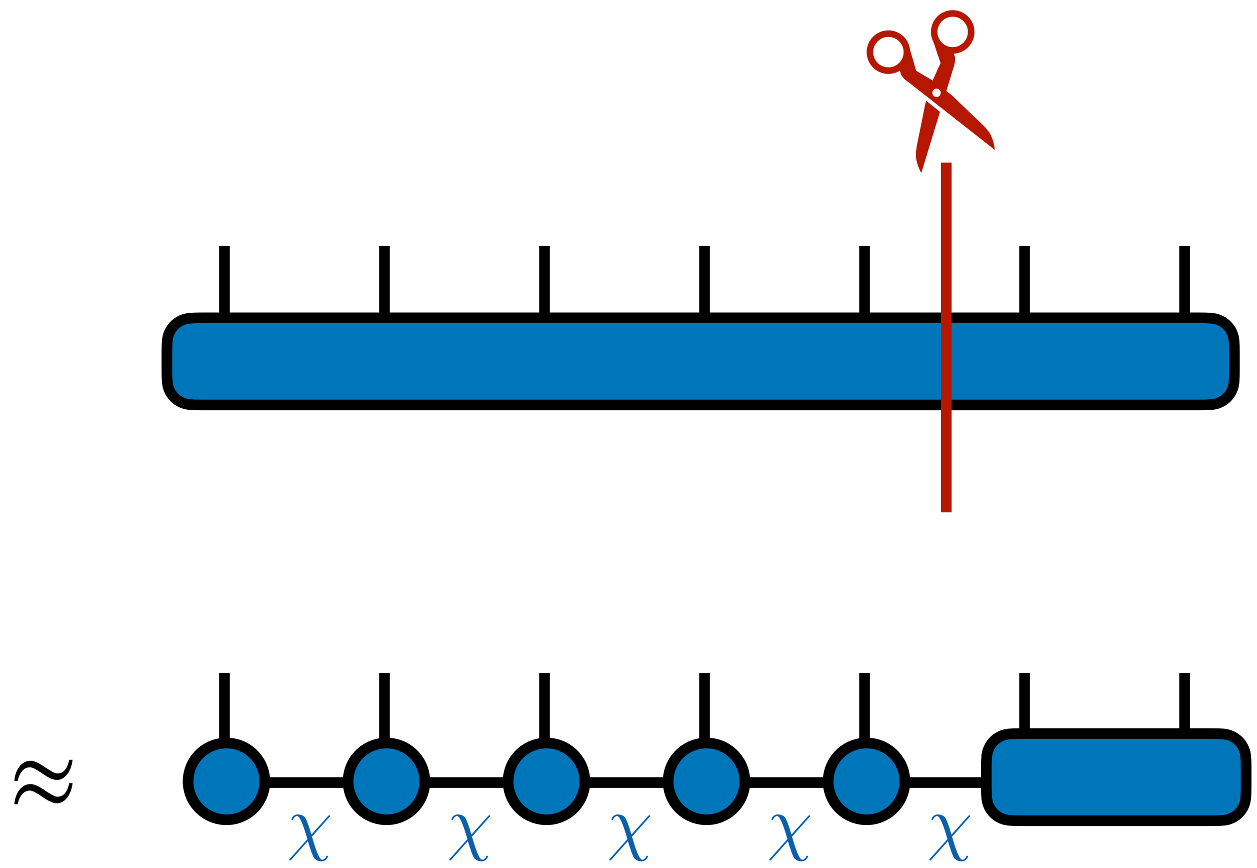
Tensor Networks

Can recursively factor (compress) a tensor as well



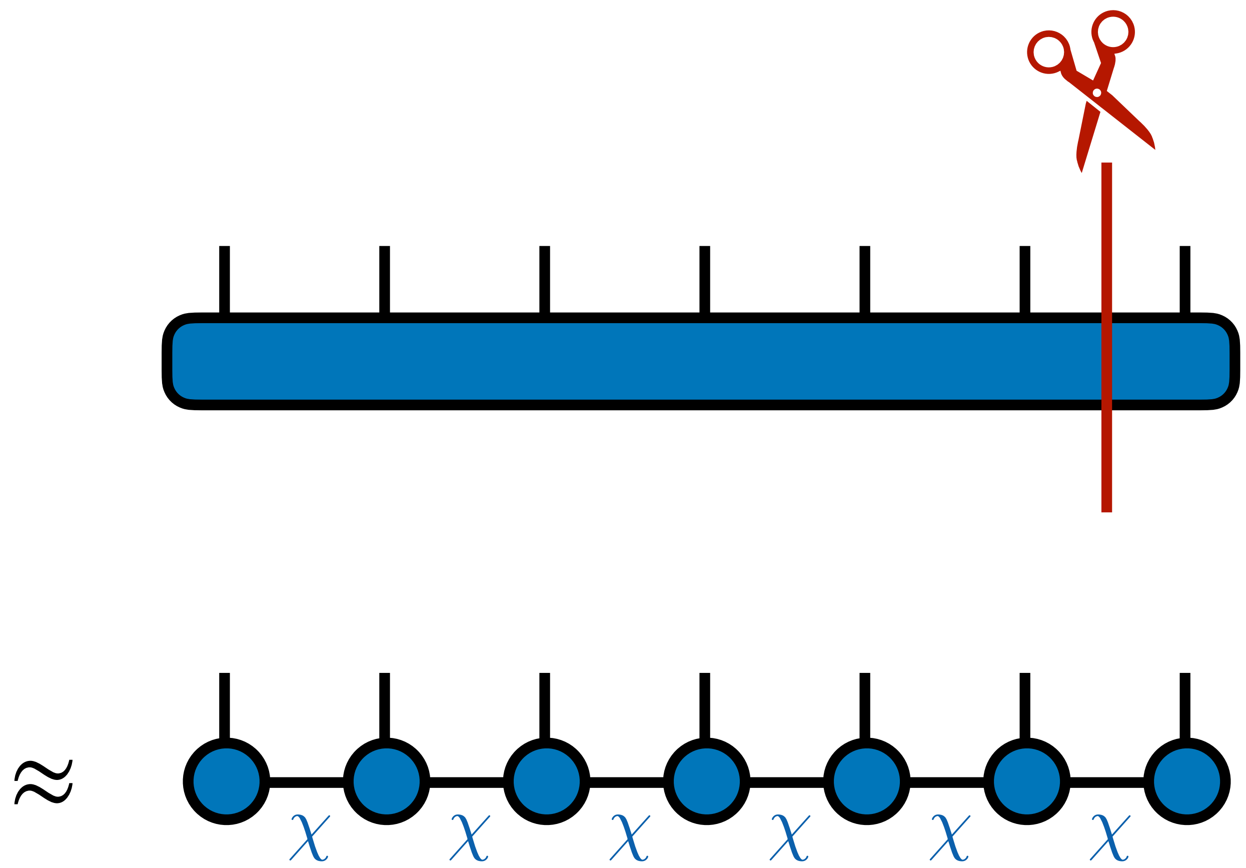
Tensor Networks

Can recursively factor (compress) a tensor as well



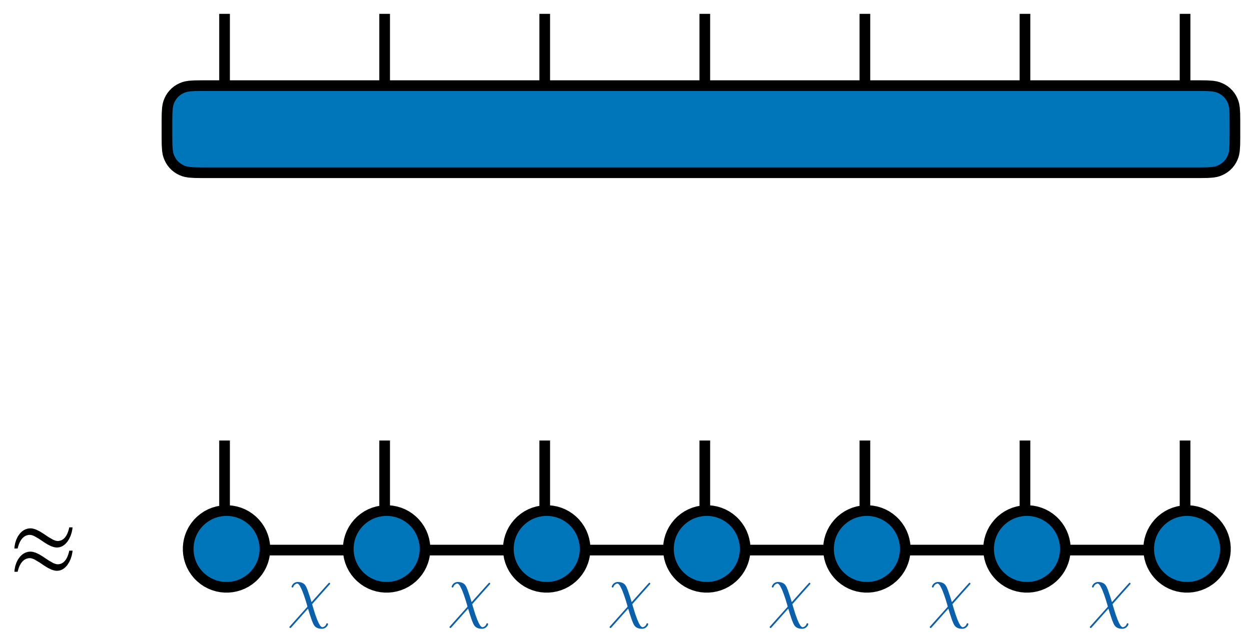
Tensor Networks

Can recursively factor (compress) a tensor as well



Tensor Networks

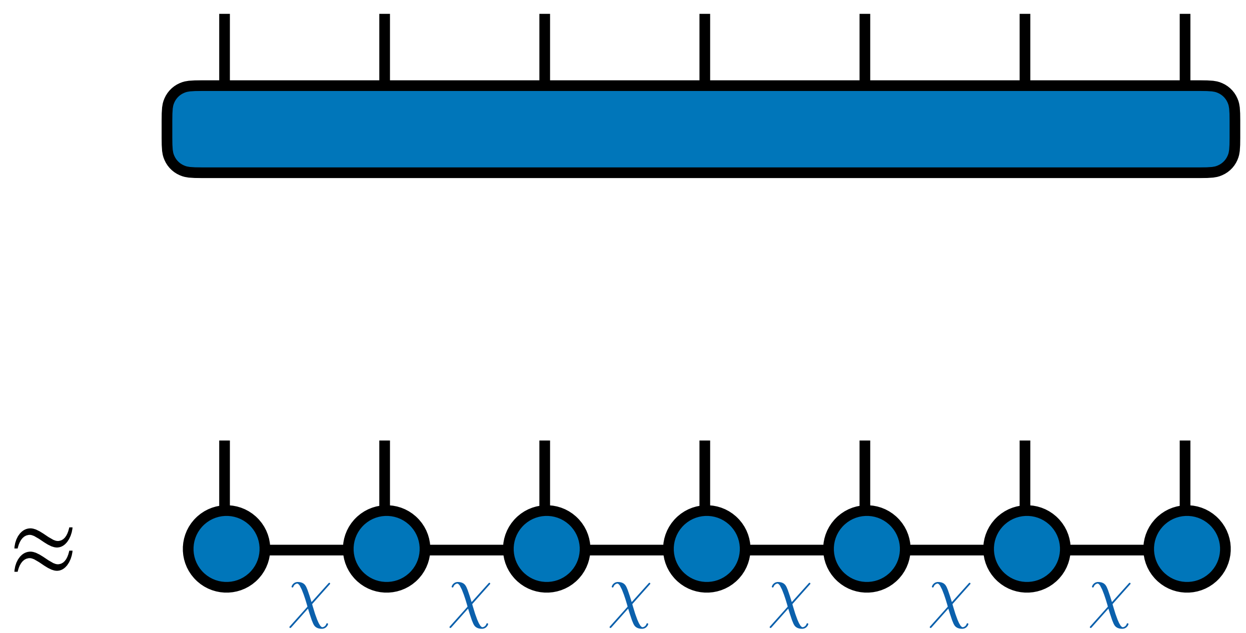
Result is a **matrix product state** or **tensor train**



Advantage if internal indices small, yet accuracy is good
(small "bond dimension" or "rank" χ)

Tensor Networks

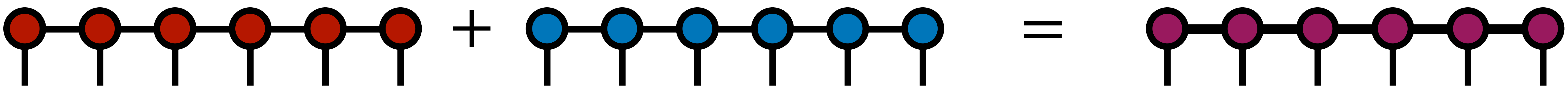
For large enough χ $(=2^{N/2})$, MPS can represent any tensor



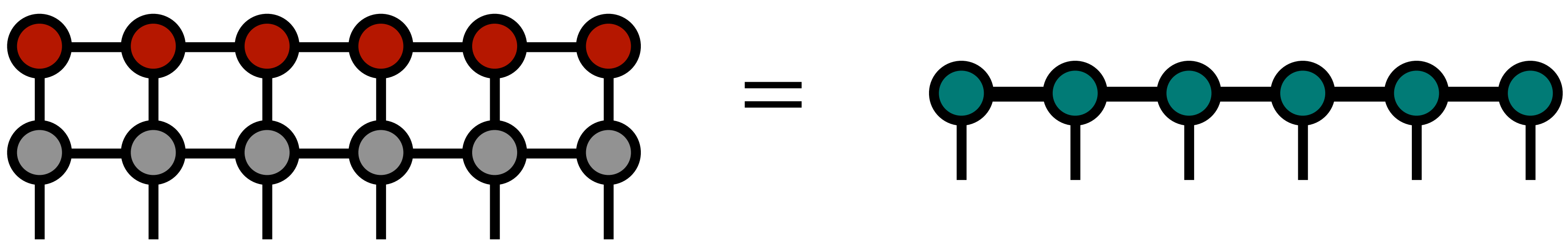
Most algorithms require χ^3 computation,

χ^2 memory

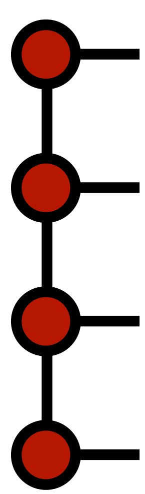
Can efficiently sum MPS in compressed form:



multiply by other networks:



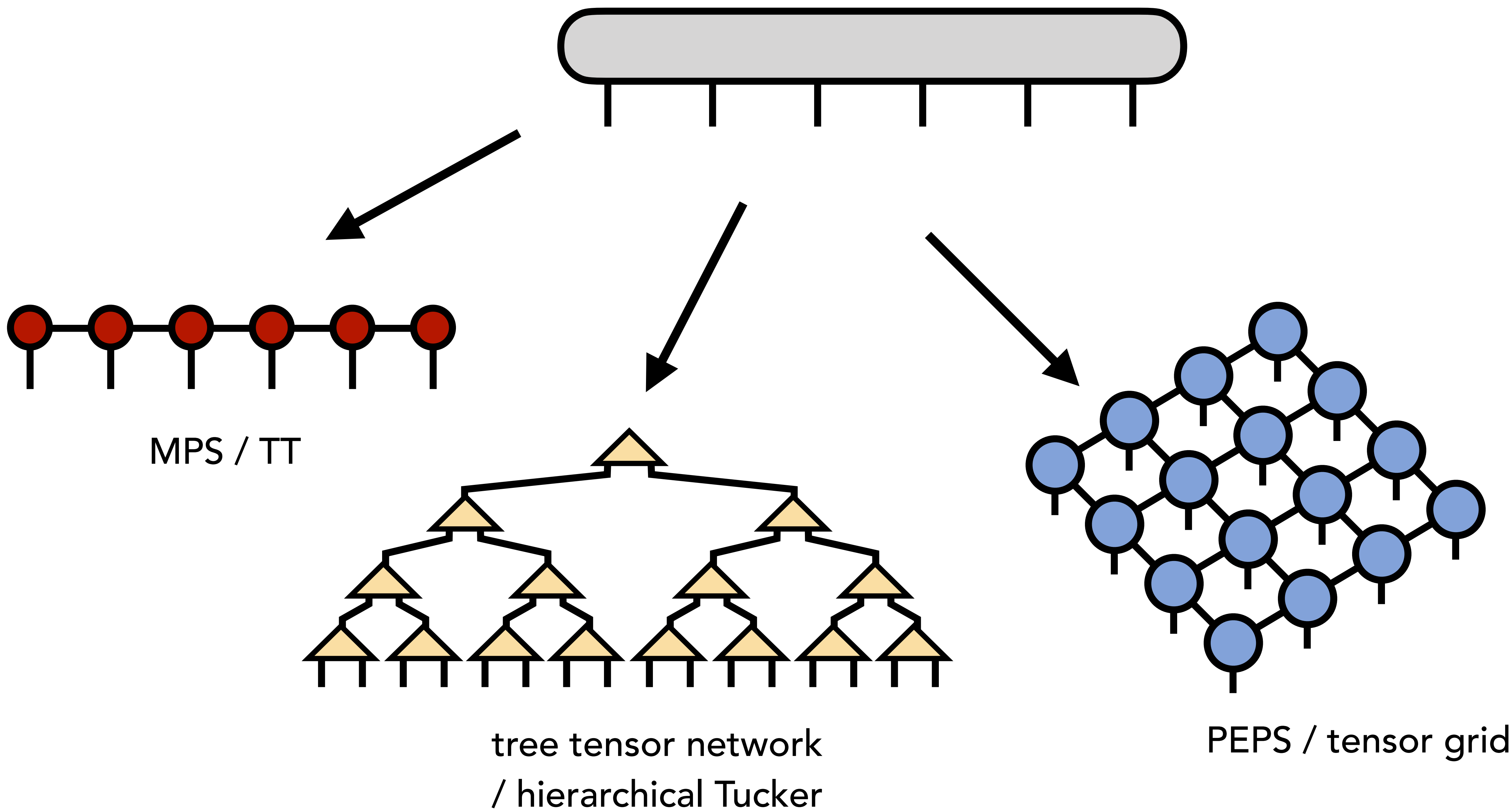
and perfectly sample:



0	0	0	0	1	0
0	1	1	0	1	0
0	1	0	1	0	0
0	0	0	0	1	0

Tensor Networks

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



Tensor Network Algorithms

Power of tensor networks is
algorithms

Seminal tensor network algorithm is DMRG
(density matrix renormalization group)

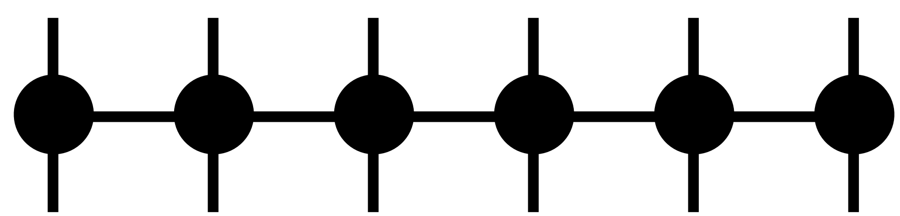
$$\min_{\psi} \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle} = E_0$$

Finds **ground state** and its **energy**

Tensor Network Algorithms

DMRG algorithm

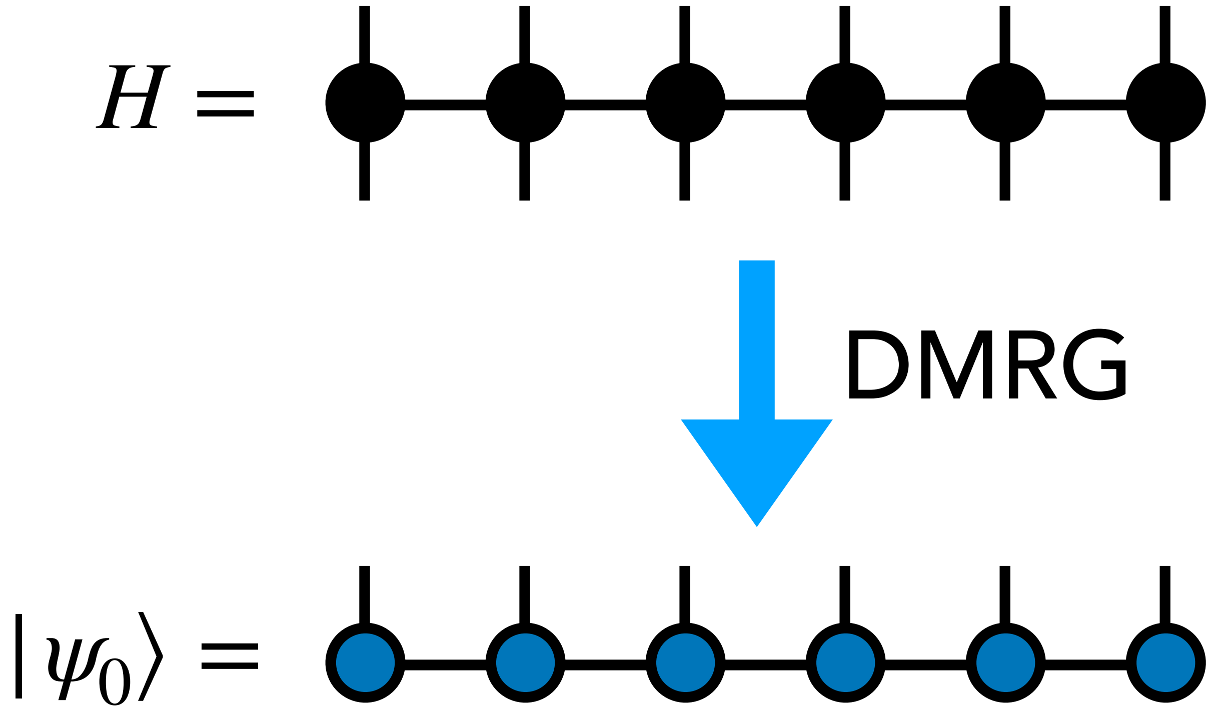
Assume we can write H as a tensor network

$H =$ 

Tensor Network Algorithms

DMRG algorithm

DMRG finds its ground state as an MPS tensor network

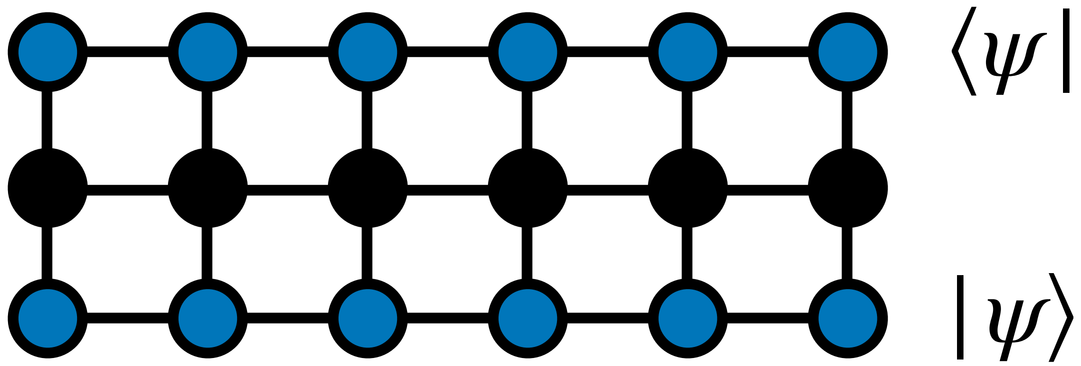


Tensor Network Algorithms

DMRG algorithm

Energy is

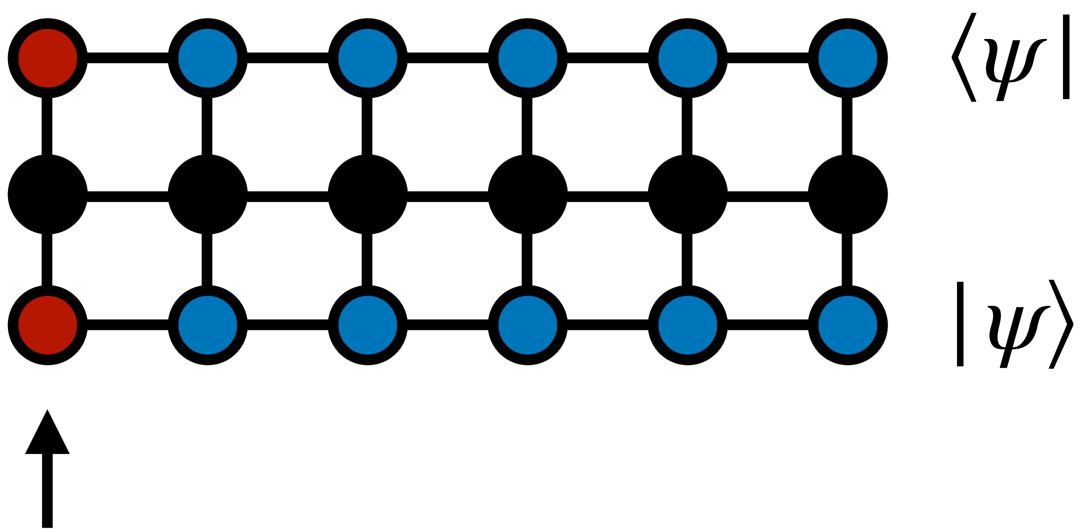
$E =$



Tensor Network Algorithms

DMRG algorithm

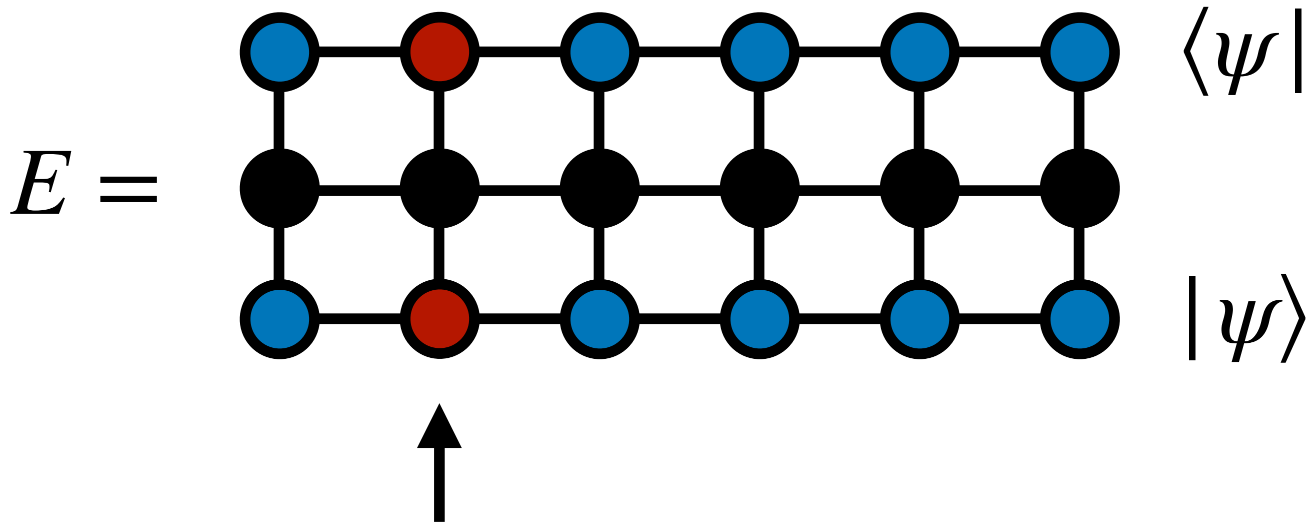
DMRG uses an "alternating" strategy to optimize

$E =$ 

Tensor Network Algorithms

DMRG algorithm

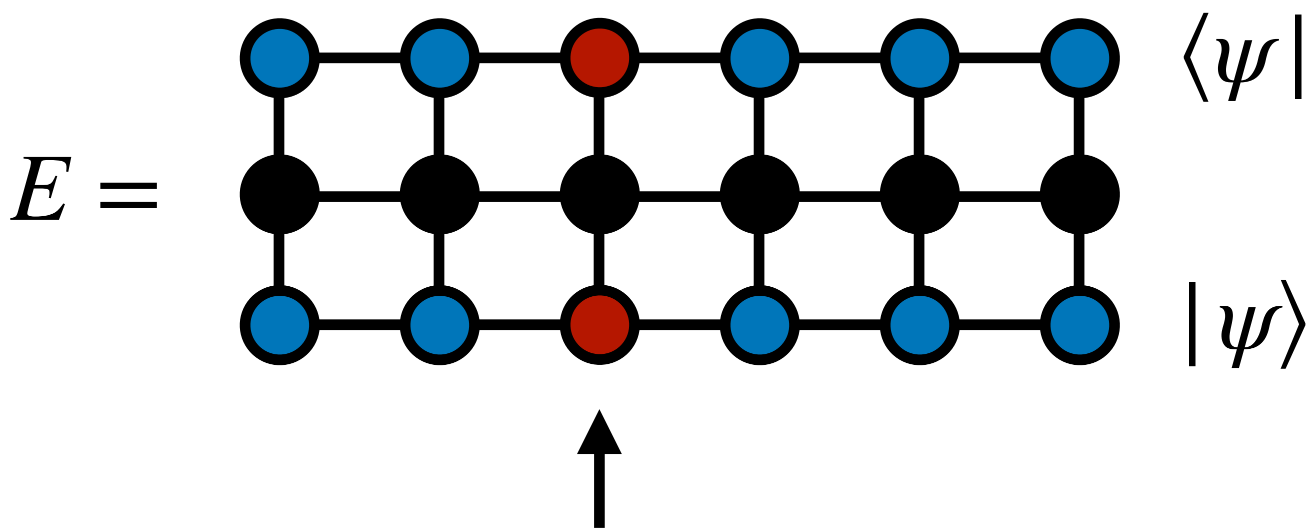
DMRG uses an "alternating" strategy to optimize



Tensor Network Algorithms

DMRG algorithm

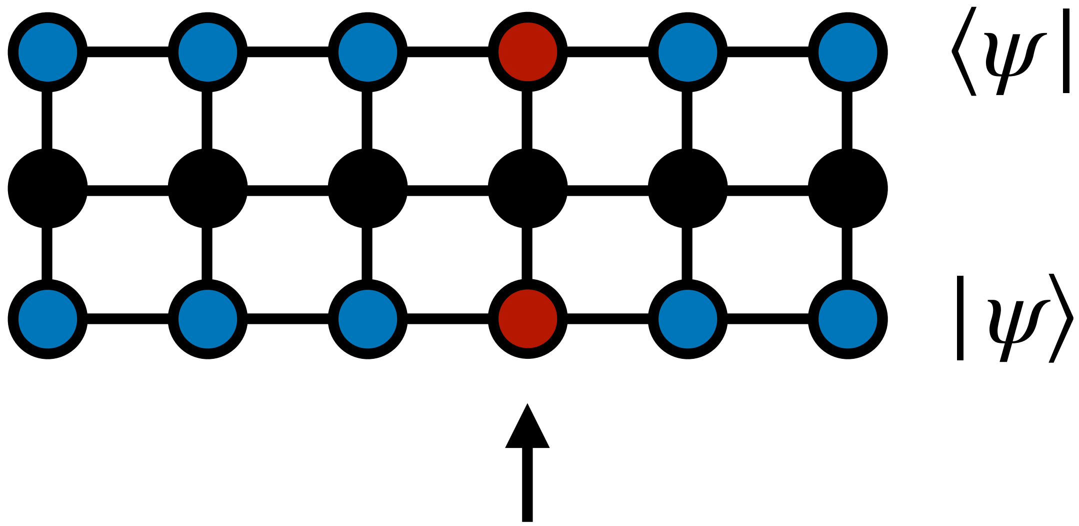
DMRG uses an "alternating" strategy to optimize



Tensor Network Algorithms

DMRG algorithm

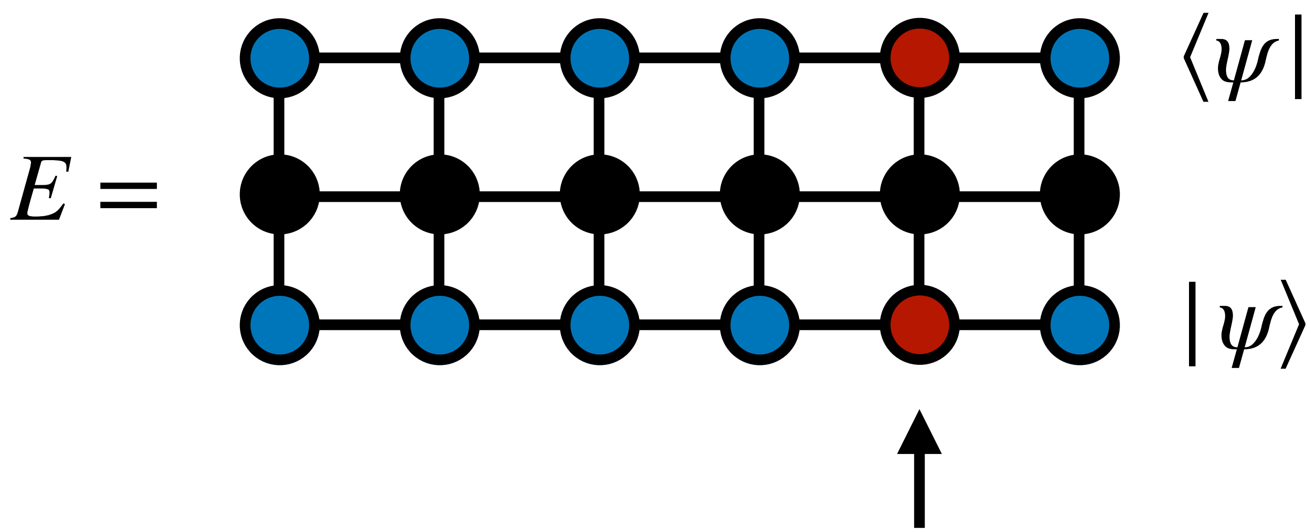
DMRG uses an "alternating" strategy to optimize

$E =$ 

Tensor Network Algorithms

DMRG algorithm

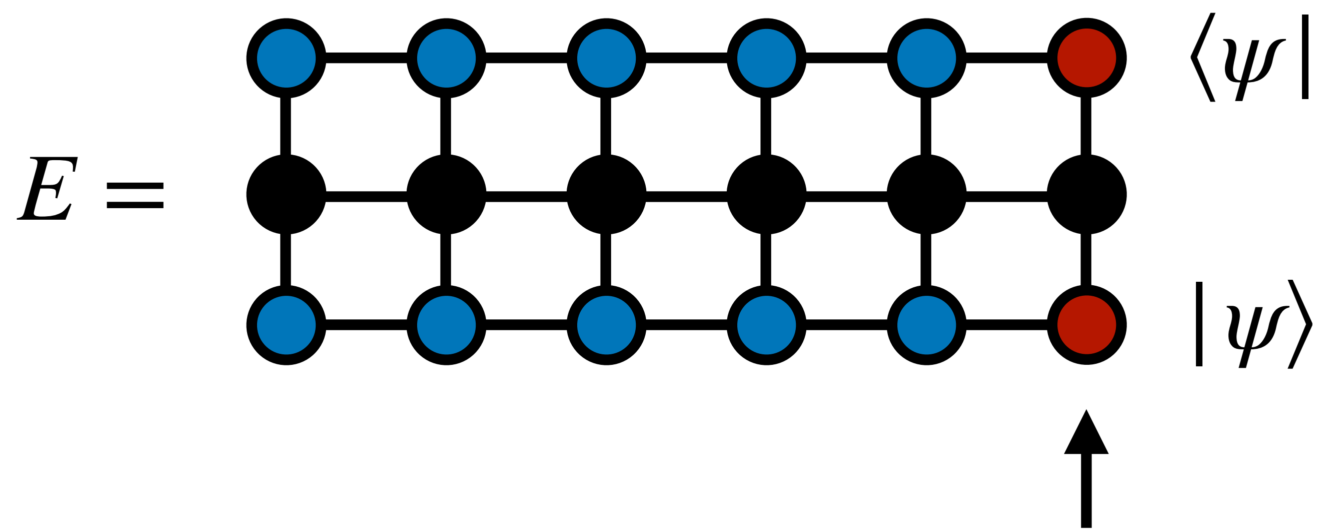
DMRG uses an "alternating" strategy to optimize



Tensor Network Algorithms

DMRG algorithm

DMRG uses an "alternating" strategy to optimize

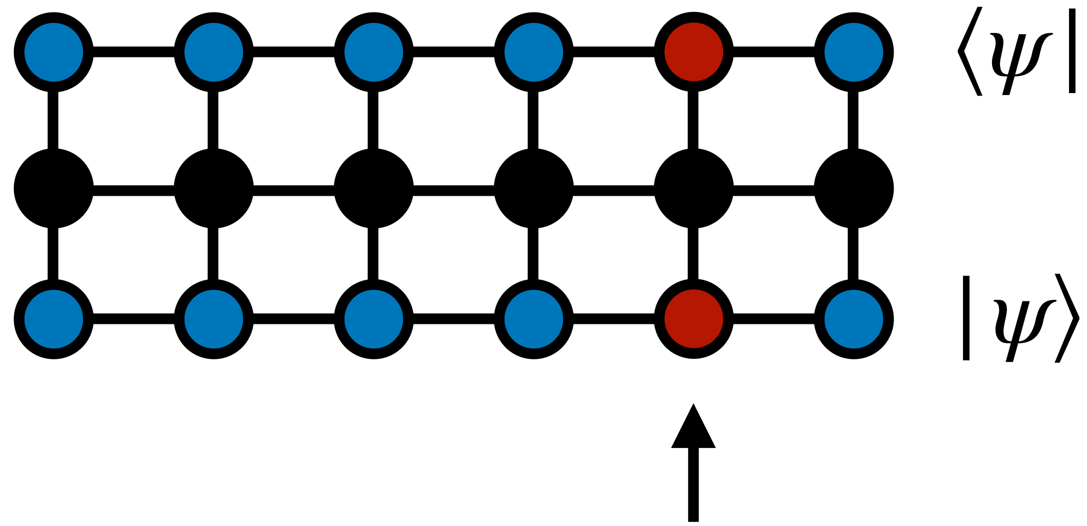


Tensor Network Algorithms

DMRG algorithm

DMRG uses an "alternating" strategy to optimize

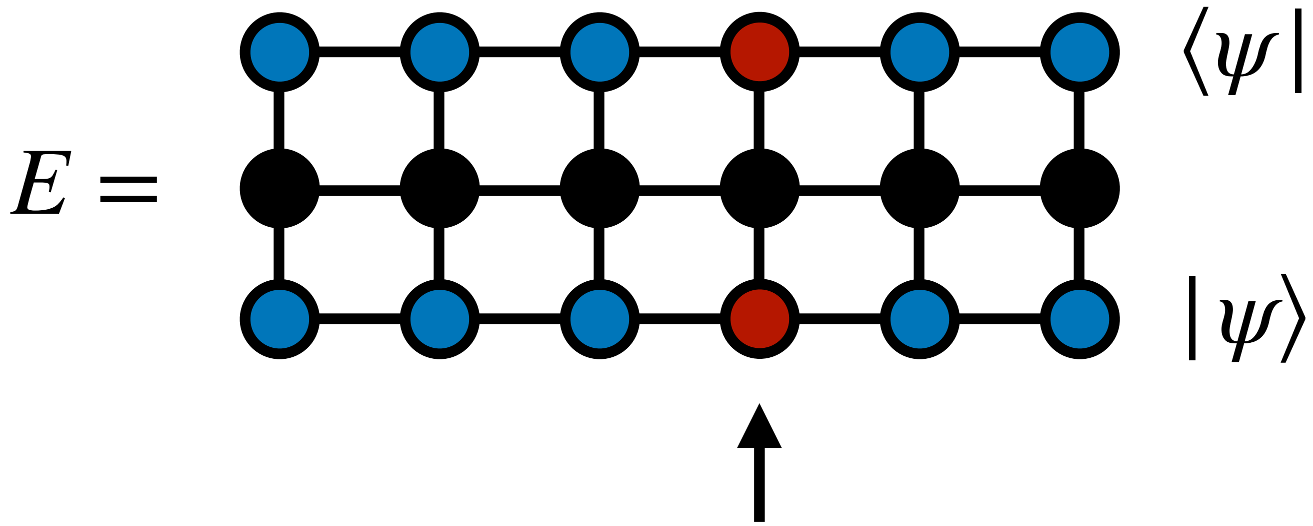
$E =$



Tensor Network Algorithms

DMRG algorithm

DMRG uses an "alternating" strategy to optimize

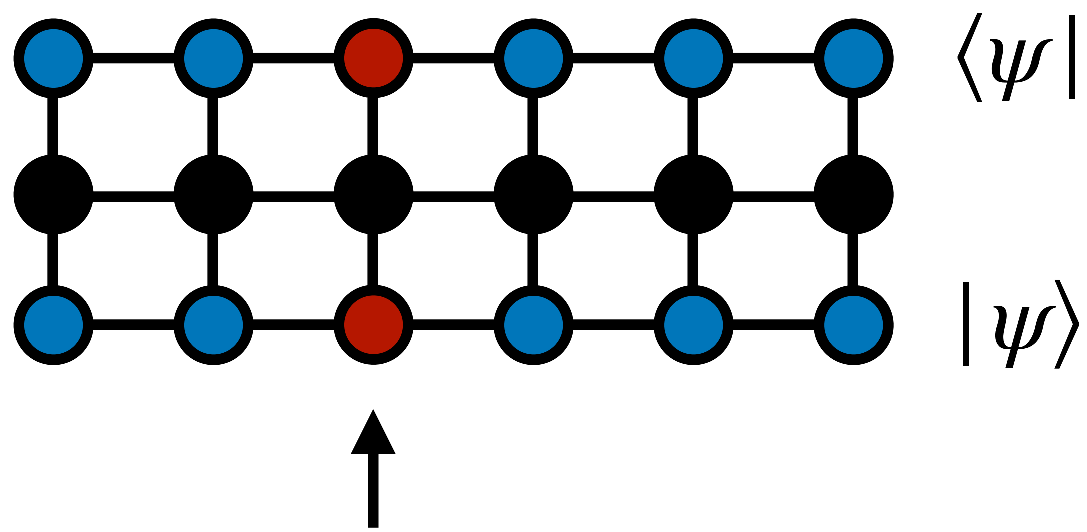


Tensor Network Algorithms

DMRG algorithm

DMRG uses an "alternating" strategy to optimize

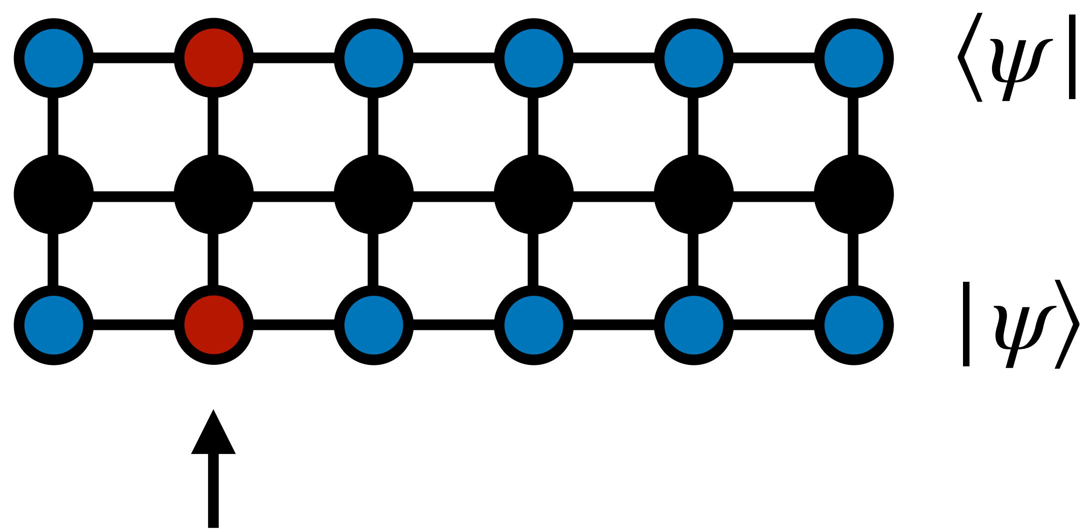
$E =$



Tensor Network Algorithms

DMRG algorithm

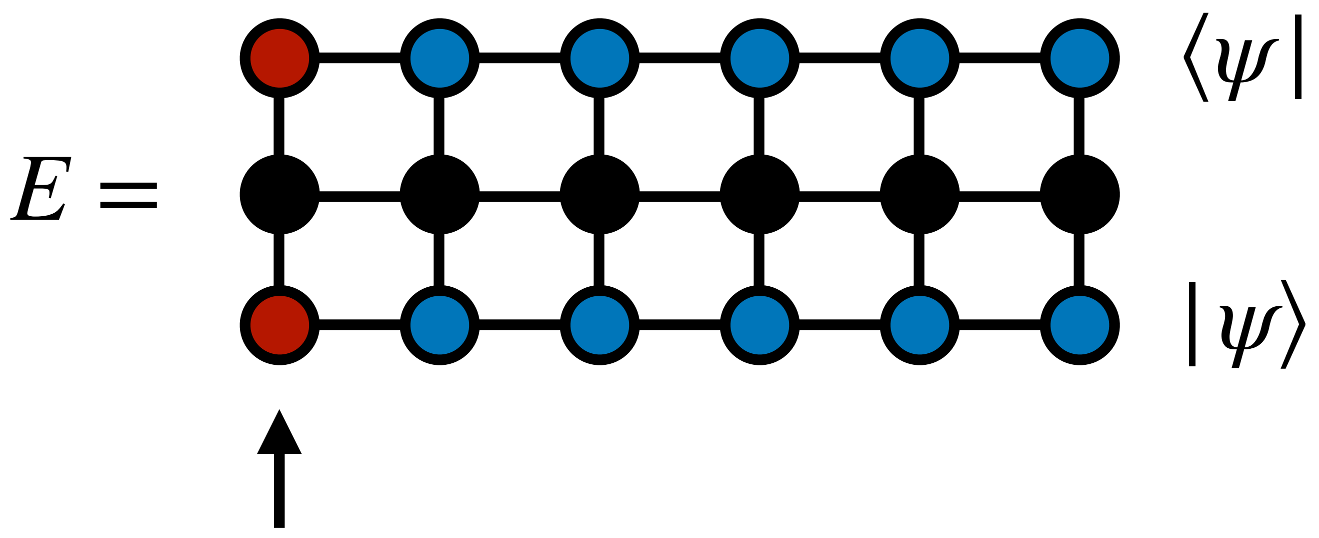
DMRG uses an "alternating" strategy to optimize

$E =$ 

Tensor Network Algorithms

DMRG algorithm

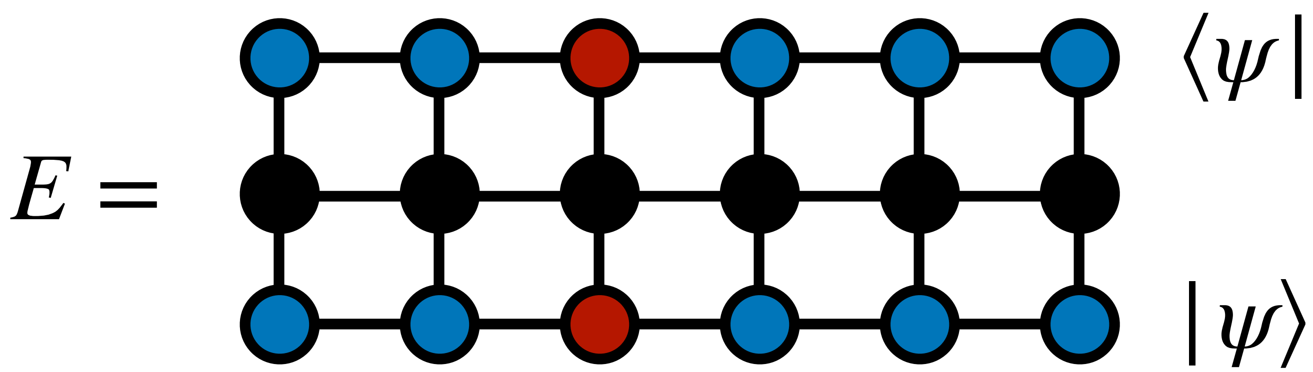
DMRG uses an "alternating" strategy to optimize



Tensor Network Algorithms

DMRG algorithm

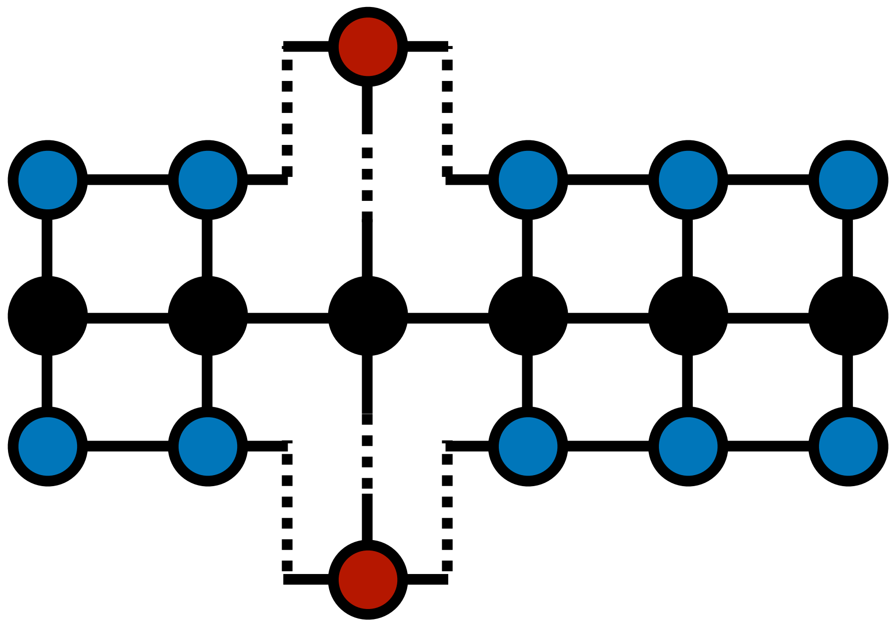
At each step, solve a "mini" diagonalization problem



Tensor Network Algorithms

DMRG algorithm

At each step, solve a "mini" diagonalization problem*

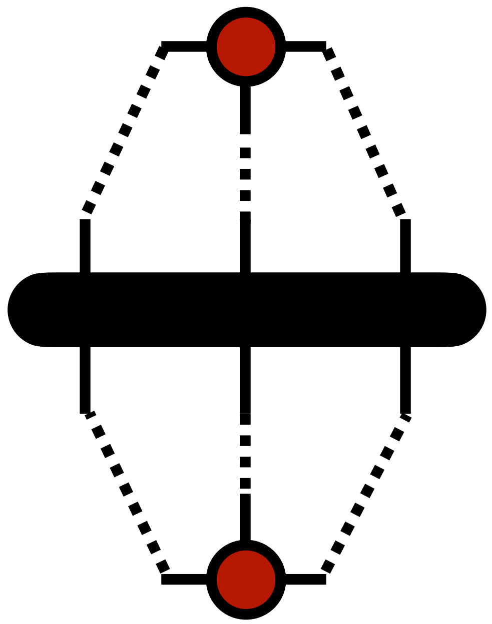


*technical note: for efficiency, frozen tensors are contracted in three groups, not exactly as shown above

Tensor Network Algorithms

DMRG algorithm

At each step, solve a "mini" diagonalization problem*



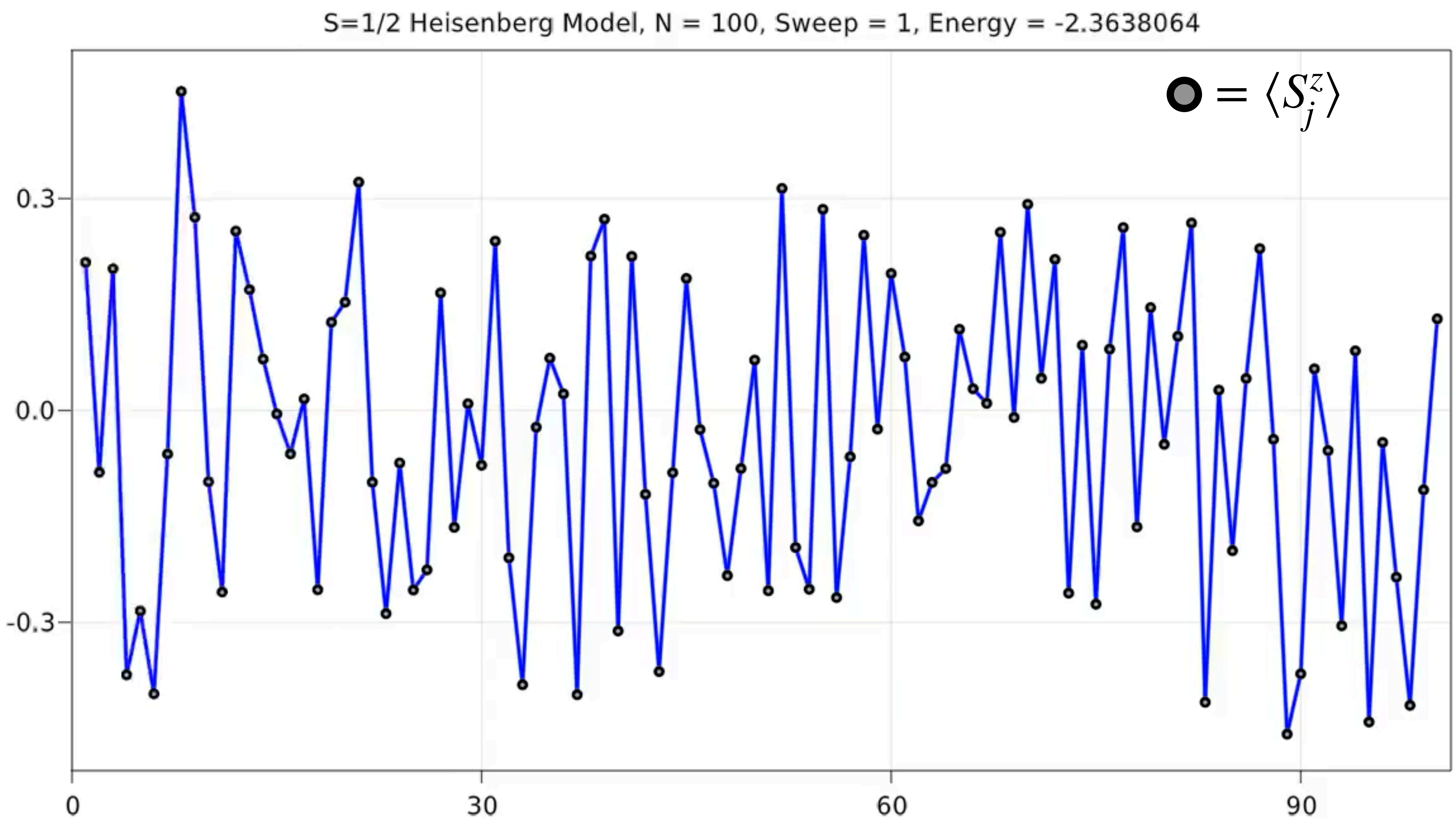
$$= E$$

*technical note: for efficiency, frozen tensors are contracted in three groups, not exactly as shown above

Tensor Network Algorithms

DMRG algorithm

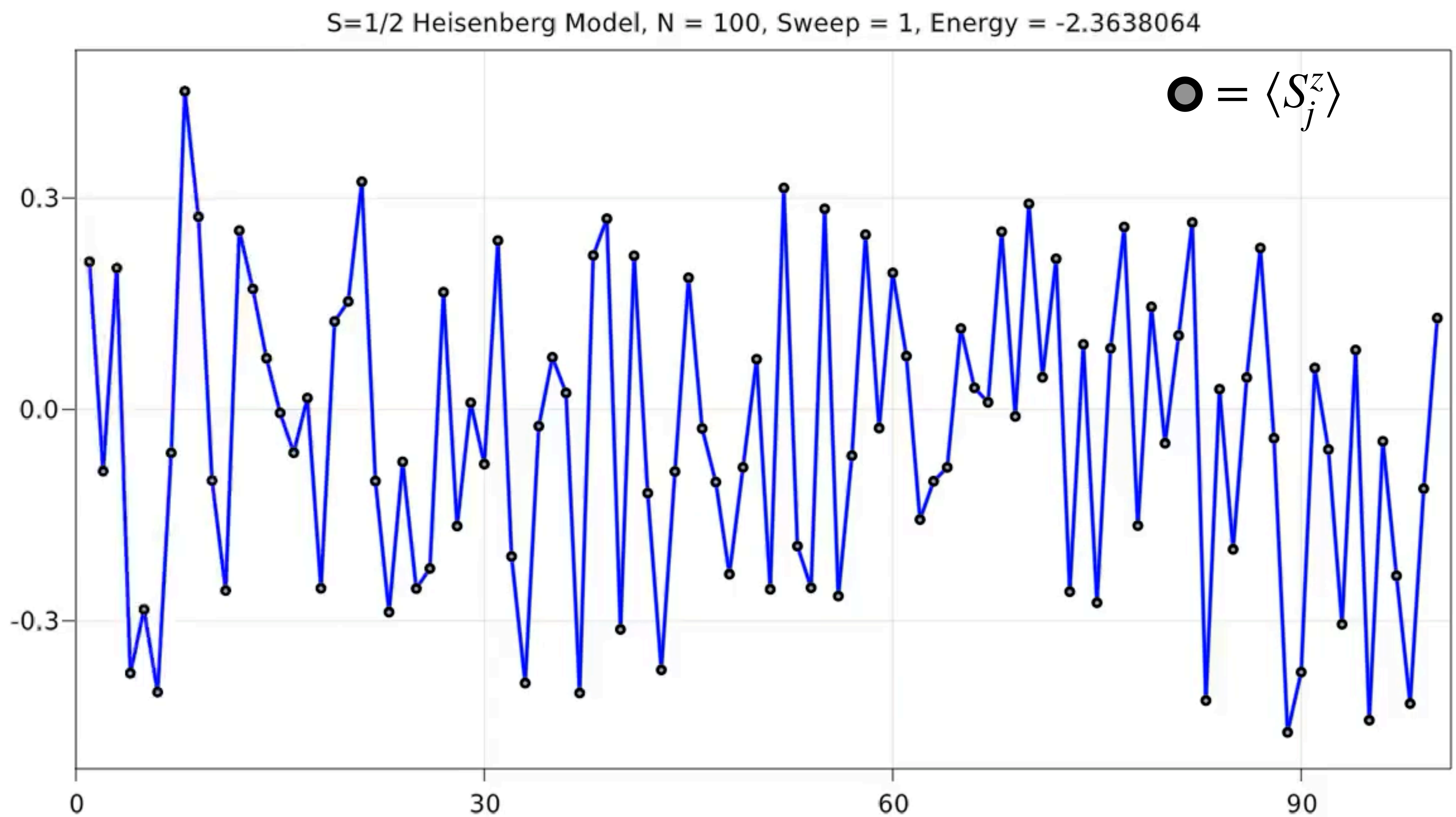
DMRG in action – solving Heisenberg chain



Tensor Network Algorithms

DMRG algorithm

DMRG in action – solving Heisenberg chain



Tensor Network Algorithms

DMRG algorithm is extremely powerful

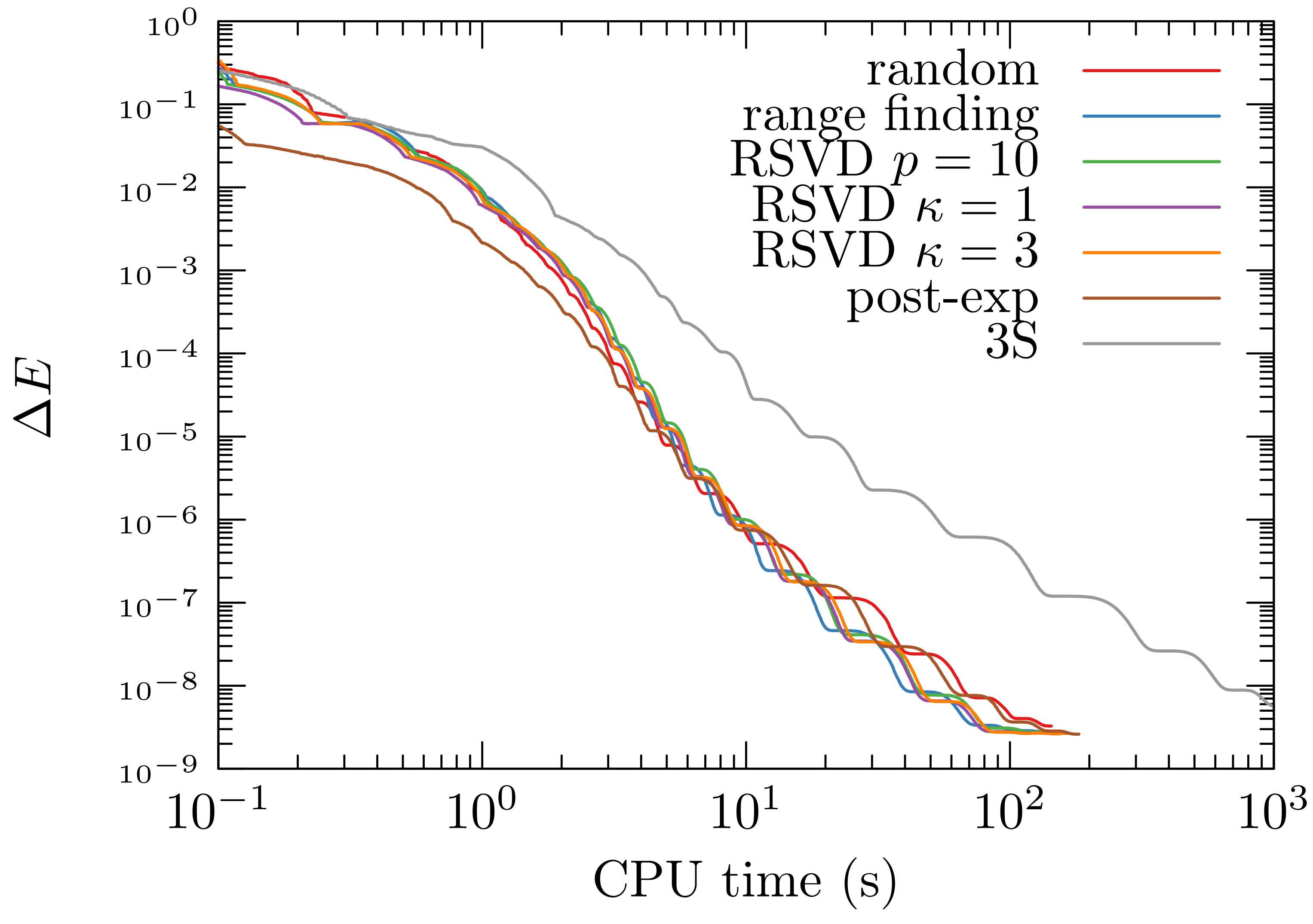
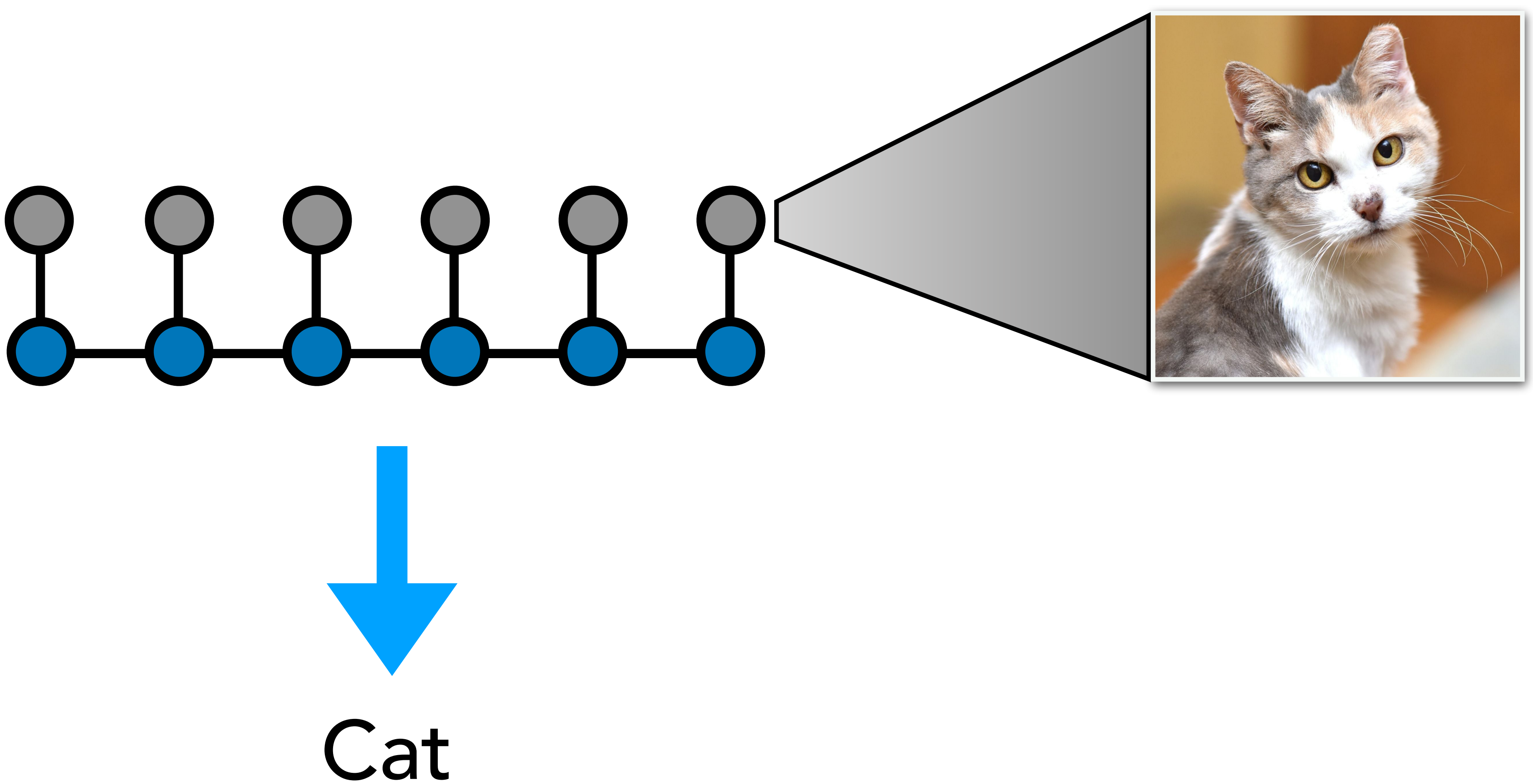


FIG. 5. CPU time (seconds) for the Hubbard-Holstein model.

McCulloch, Osborne, arxiv:2403.00562 (2024)

Tensor Network Machine Learning

Can we harness the power of **tensor networks** for **machine learning**?



Tensor Network Machine Learning

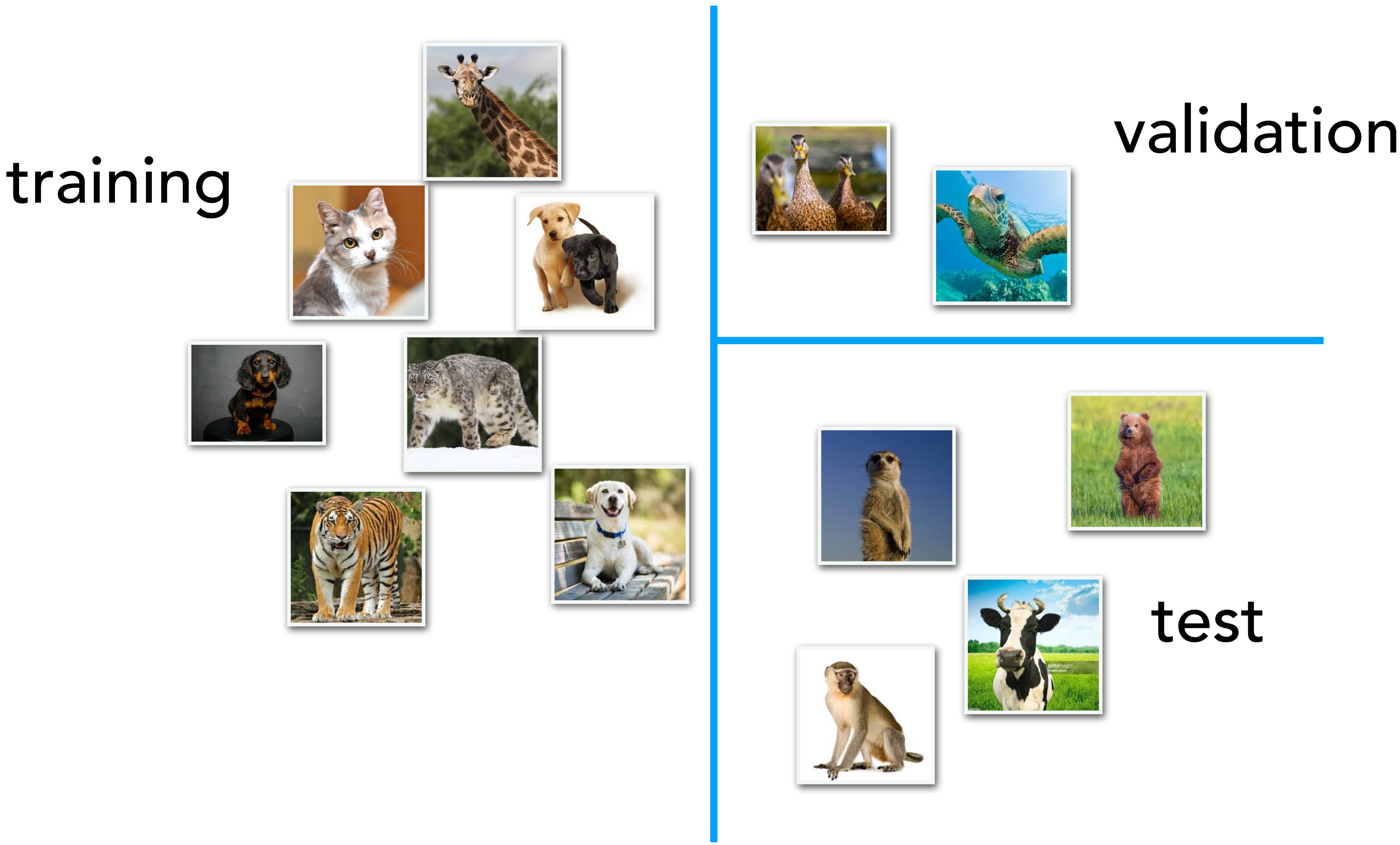
Lightning review of machine learning concepts ⚡

Machine Learning Concepts

Sometimes have large "data set" up front:

Machine Learning Concepts

Can divide into training / validation / test



Machine Learning Concepts

Sometimes given no data, but can call a function:

```
distance_from_goal(position) → number
```

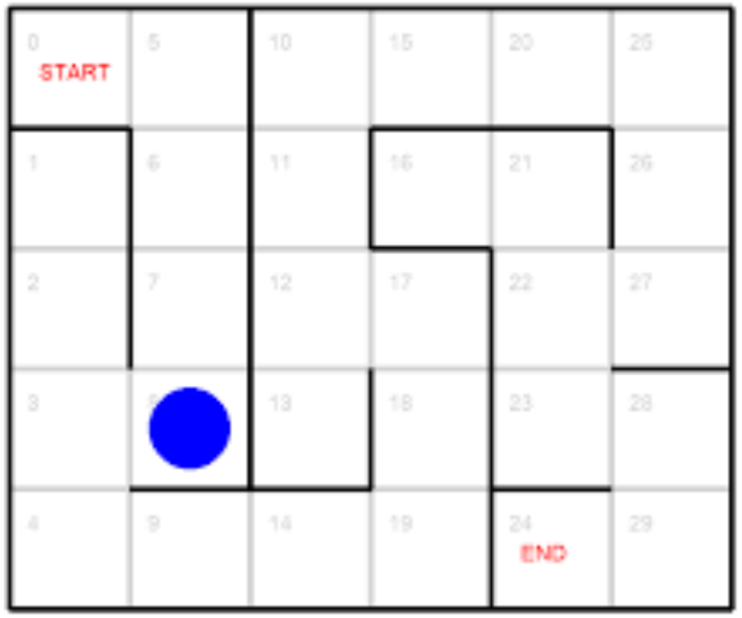


Image credit: Sofia Kolchanova, JetBrains

Machine Learning Concepts

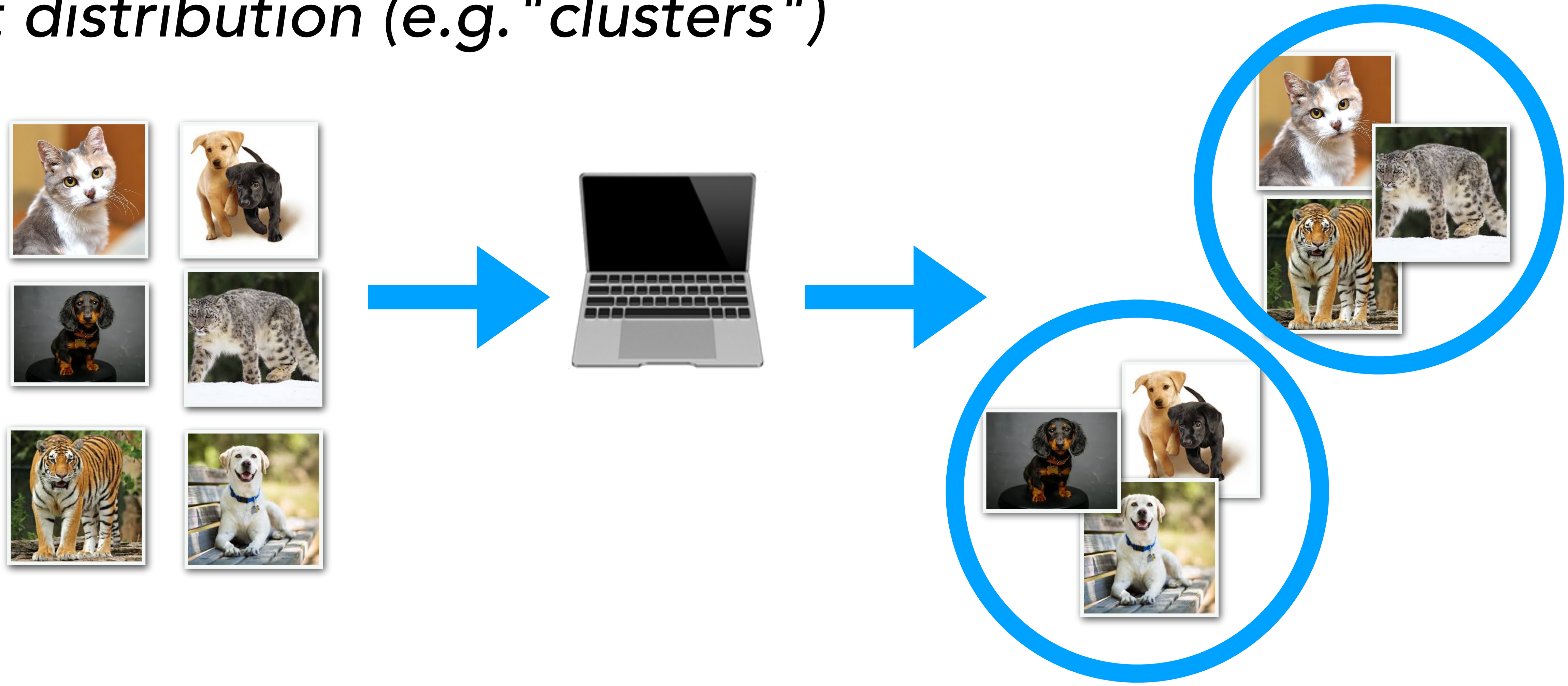
Various "tasks" in machine learning:

- **Supervised learning**
  predict labels for data, classify data

Machine Learning Concepts

Various "tasks" in machine learning:

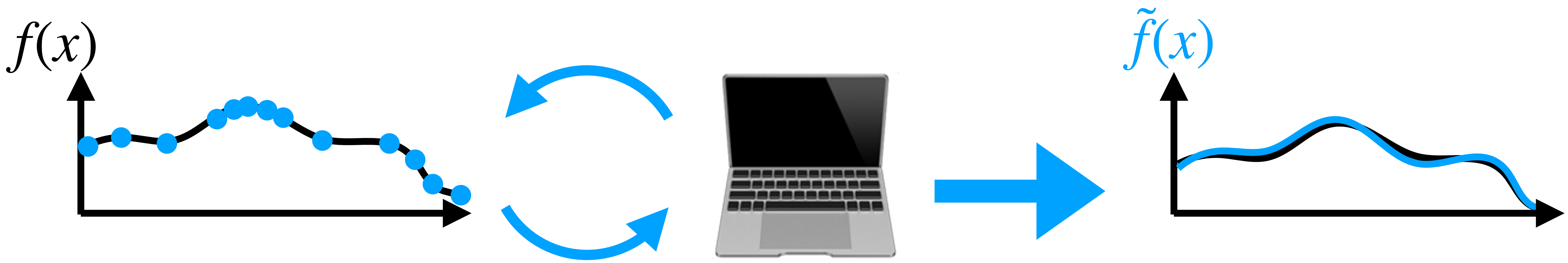
- **Unsupervised learning / generative modeling**
  recover <u>distribution</u> of data, or properties of that distribution (e.g. "clusters")



Machine Learning Concepts

Various "tasks" in machine learning:

- **Active learning**
  recover a function by querying at points



Machine Learning Concepts

In all cases, seek a model function with parameters

data input $\vec{x}$

$$f_{\vec{\theta}}(\vec{x})$$

model parameters $\vec{\theta}$

Optimize parameters $\vec{\theta}$ until function accomplishes task

Machine Learning Concepts

For example, in supervised learning, model is

$$f^{\ell}_{\vec{\theta}}(\vec{x})$$

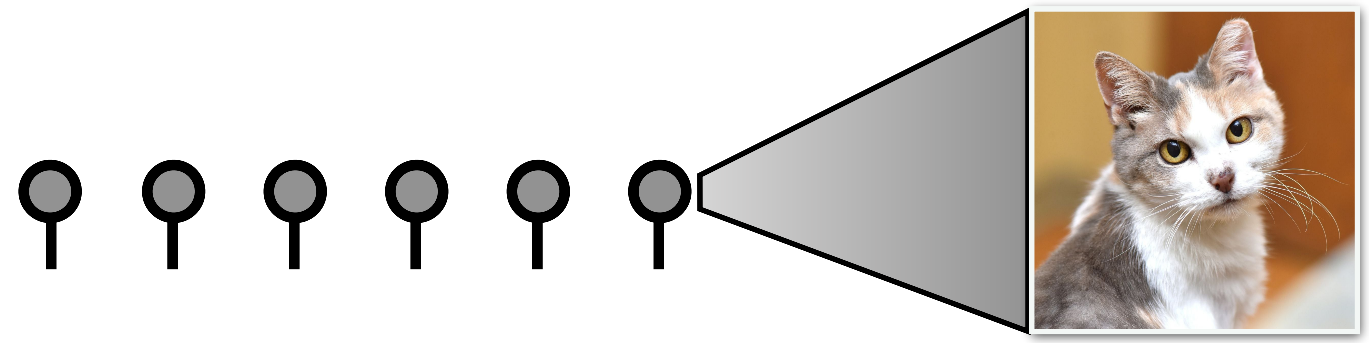
ℓ = label

Optimize parameters $\vec{\theta}$ until f^{ℓ} outputs maximum value when ℓ is the correct label of input $\vec{x}$

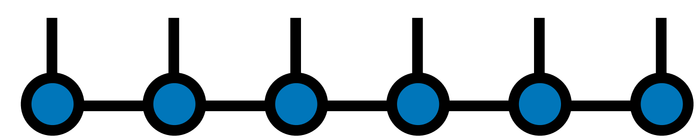
Tensor Network Machine Learning

Three challenges for tensor networks:

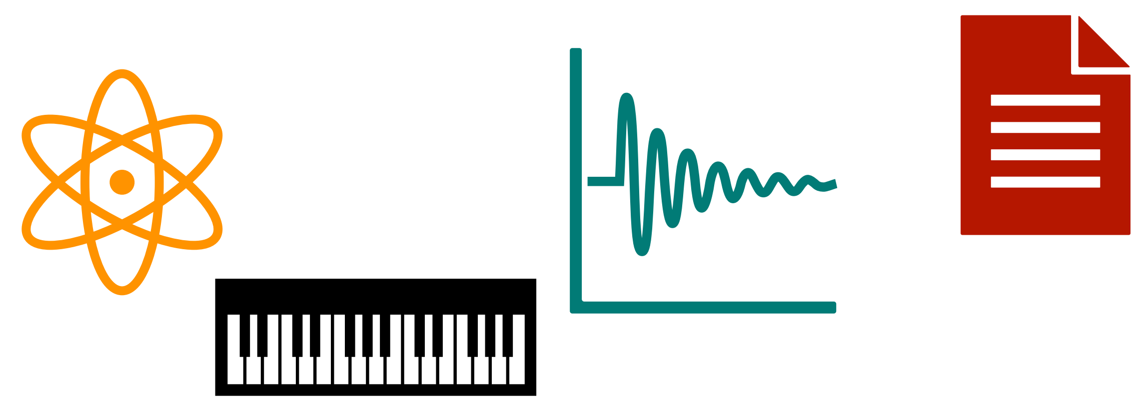
- representation of **data**



- training **algorithms**



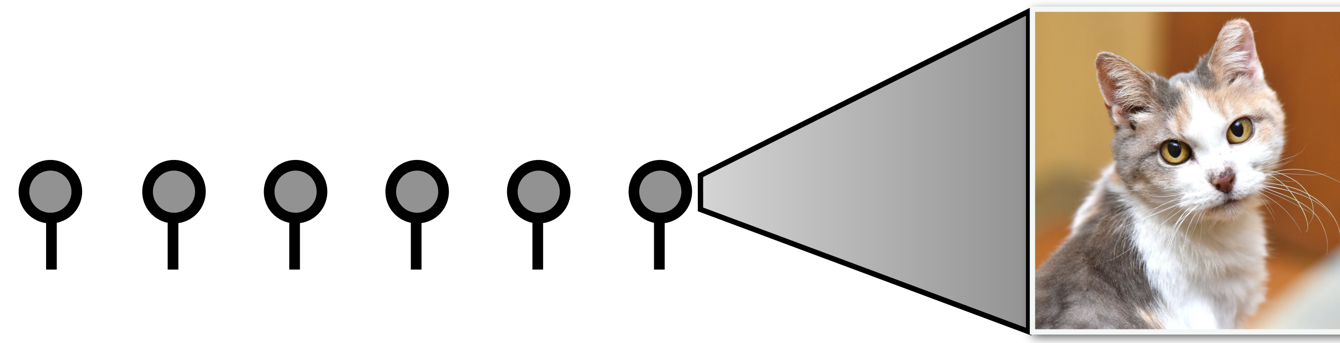
- good **problem** selection



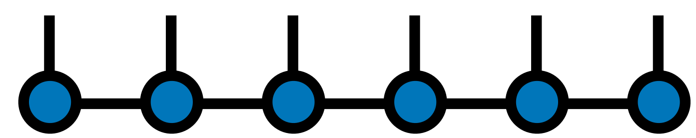
Tensor Network Machine Learning

Three challenges for tensor networks:

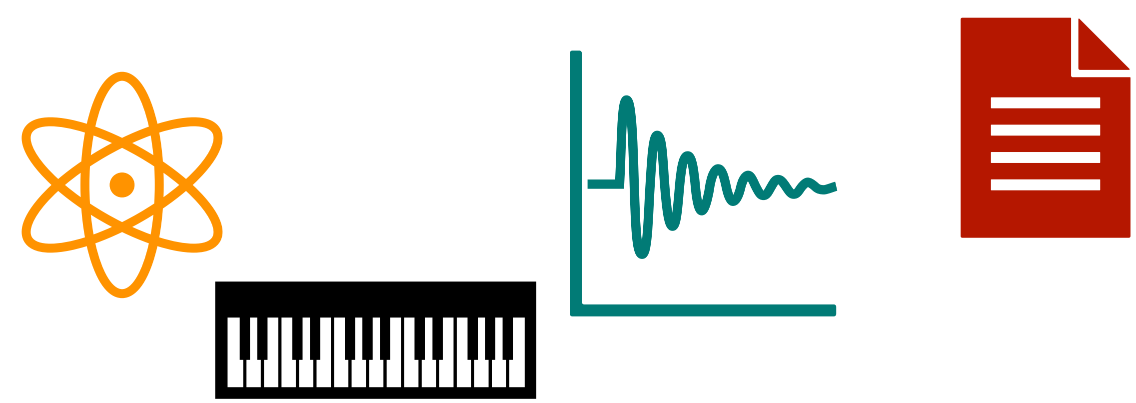
- representation of **data**



- training **algorithms**



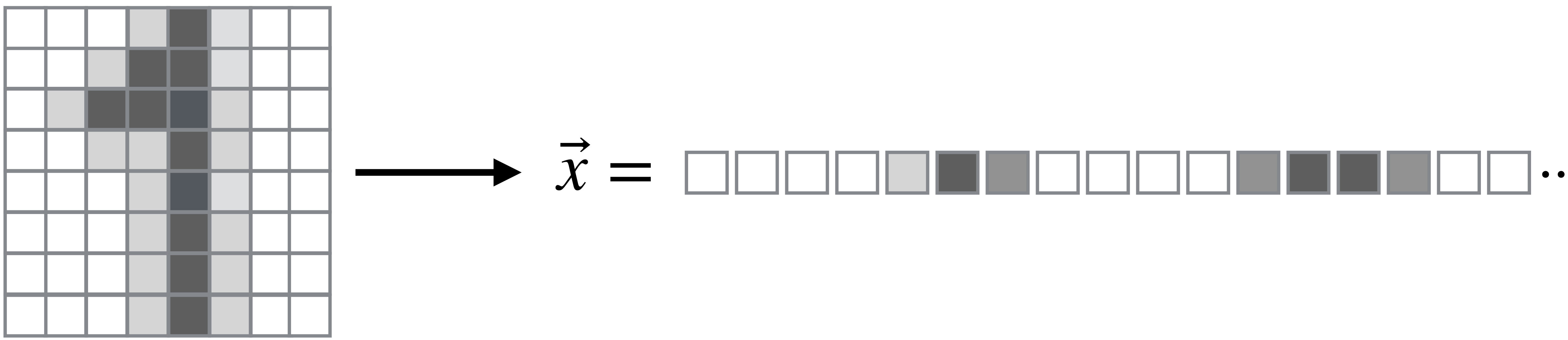
- good **problem** selection



Tensor Network Machine Learning

Representations of data

Say we are given a piece of data with N components



View as vector of length N

$$\vec{x} = [x_1, x_2, x_3, \ldots, x_N]$$

Tensor Network Machine Learning

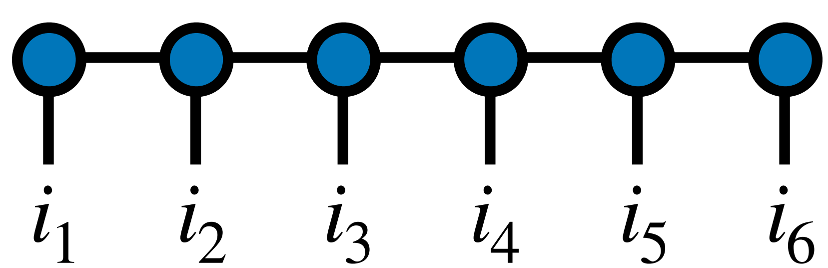
Representations of data

If data entries are integers, nothing else to do

$$\vec{x} = \left[i_1, i_2, i_3, \ldots, i_N\right] \qquad i_j \in \mathbb{Z}$$

Just use tensor network as model:

$f(\vec{x}) =$



Test your knowledge: what are parameters $\vec{\theta}$?

Tensor Network Machine Learning

Representations of data

Say we are given a piece of data with N components

What about continuous entries? $x_j \in \mathbb{R}$

$$\vec{x} = \left[x_1, x_2, x_3, \ldots, x_N\right]$$

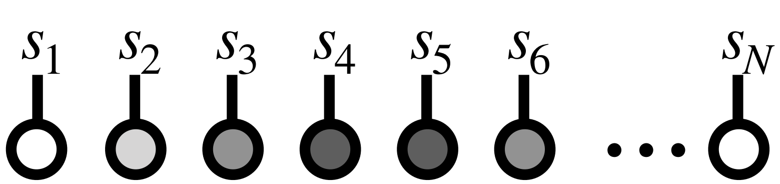
Tensor Network Machine Learning

Representations of data

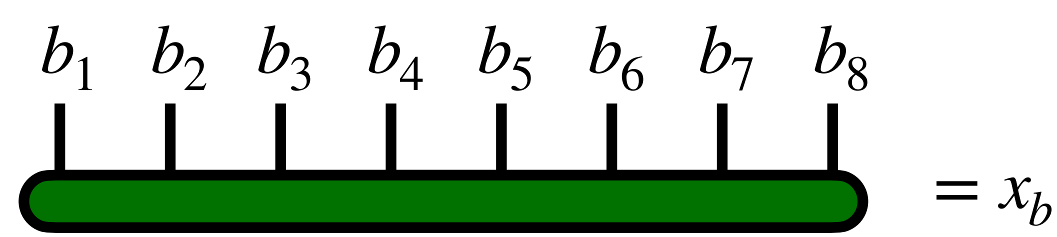
Two main encodings of continuous data into tensors

$$\vec{x} = \left[x_1, x_2, x_3, \ldots, x_N\right]$$

Basis encoding



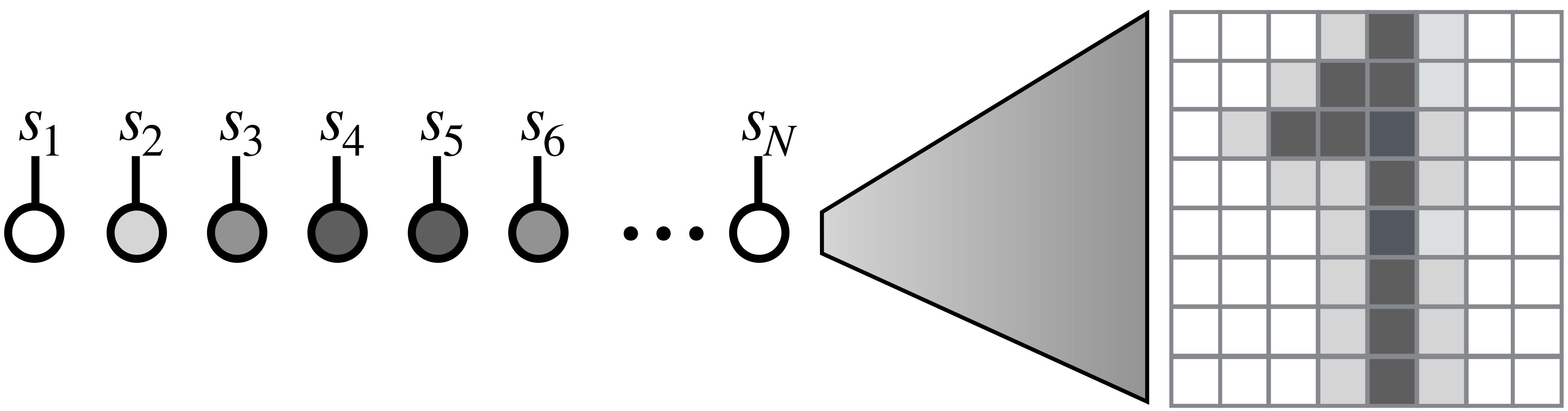
Amplitude encoding



Tensor Network Machine Learning

Representations of data

Basis encoding



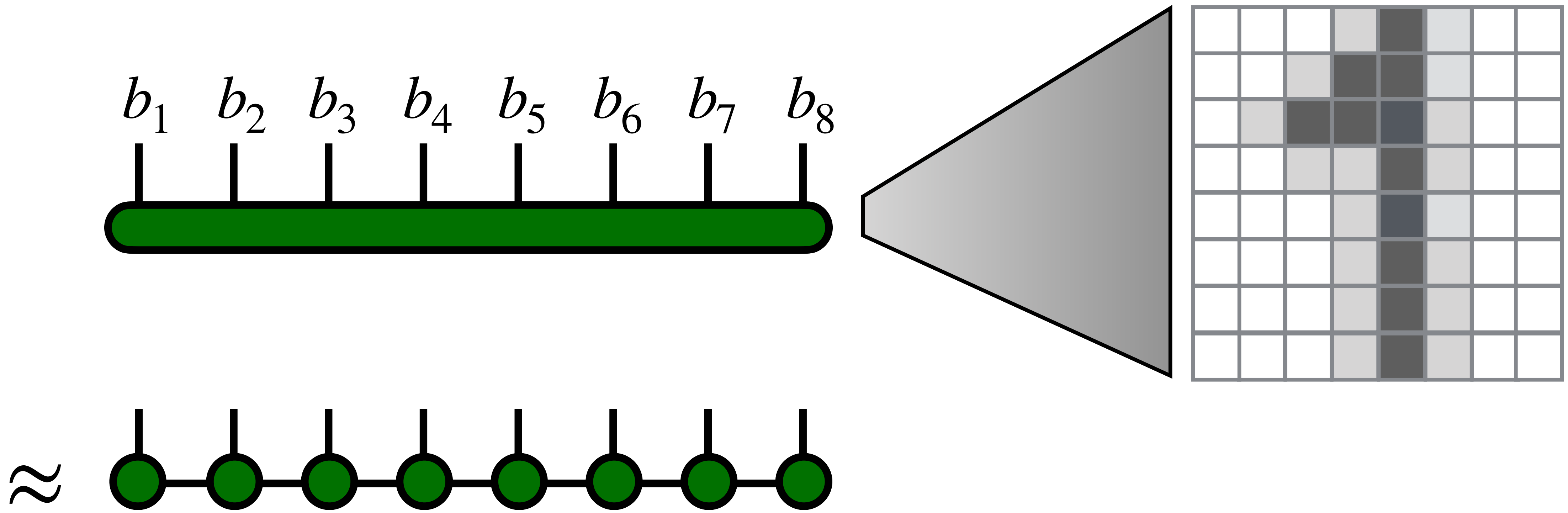
For input input size N, use N indices (high dimensional)

also known as "state encoding" or "product encoding"

Tensor Network Machine Learning

Representations of data

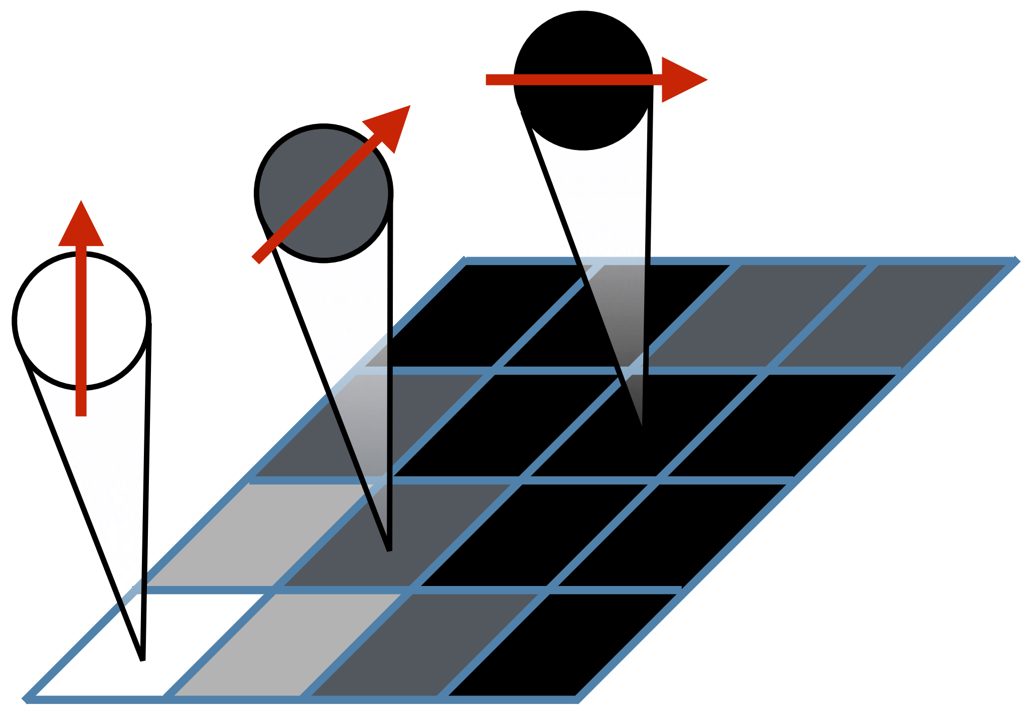
Amplitude encoding



For input size N, $\log(N)$ indices (low dimensional)

Tensor Network Machine Learning

Basis encoding



Map each pixel to a vector

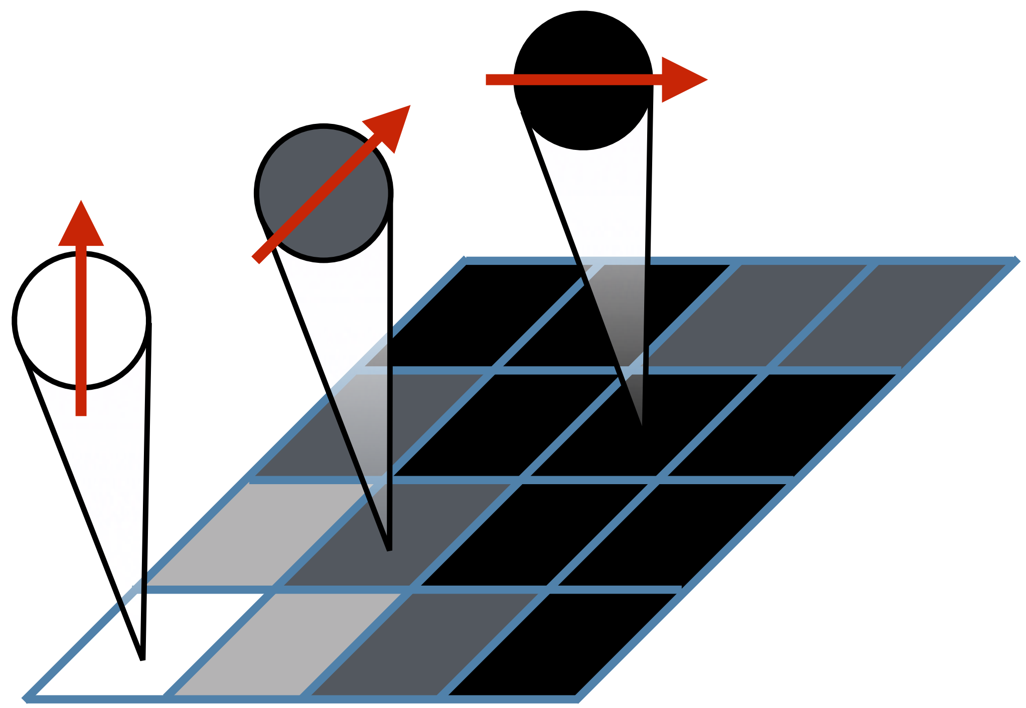
$$x_j \rightarrow \begin{bmatrix} \cos(\frac{\pi}{2} x_j) \\ \sin(\frac{\pi}{2} x_j) \end{bmatrix}$$

Take (formal) outer product

$$\vec{x} \rightarrow \begin{bmatrix} \cos(\frac{\pi}{2} x_1) \\ \sin(\frac{\pi}{2} x_1) \end{bmatrix} \begin{bmatrix} \cos(\frac{\pi}{2} x_2) \\ \sin(\frac{\pi}{2} x_2) \end{bmatrix} \begin{bmatrix} \cos(\frac{\pi}{2} x_3) \\ \sin(\frac{\pi}{2} x_3) \end{bmatrix} \cdots$$

Tensor Network Machine Learning

Basis encoding



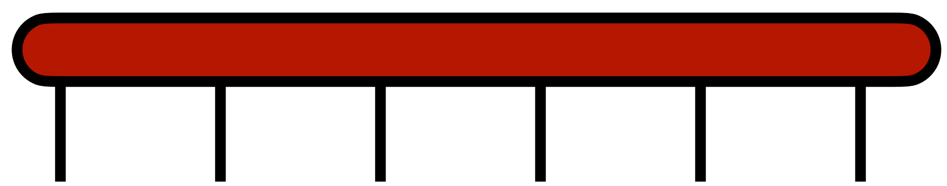
Another choice of "local feature map" is

$$x_j \rightarrow \begin{bmatrix} 1 \\ x_j \end{bmatrix} \qquad \vec{x} \rightarrow \begin{bmatrix} 1 \\ x_1 \end{bmatrix} \begin{bmatrix} 1 \\ x_2 \end{bmatrix} \begin{bmatrix} 1 \\ x_3 \end{bmatrix} \cdots$$

Tensor Network Machine Learning

Can make into a function, or machine learning model, by contracting with a tensor

$$f(x_1, x_2, \ldots, x_N) =$$



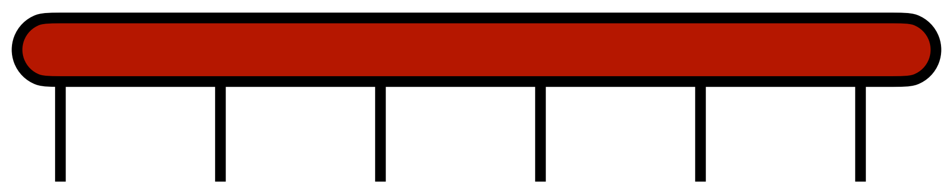
W weight tensor

$$\begin{bmatrix}1\\x_1\end{bmatrix}\begin{bmatrix}1\\x_2\end{bmatrix}\begin{bmatrix}1\\x_3\end{bmatrix}\begin{bmatrix}1\\x_4\end{bmatrix}\begin{bmatrix}1\\x_5\end{bmatrix}\begin{bmatrix}1\\x_6\end{bmatrix} \leftarrow \vec{x}$$

Tensor Network Machine Learning

A very high-order polynomial

$f(x_1, x_2, \ldots, x_N) =$



W weight tensor

$$\begin{bmatrix}1\\x_1\end{bmatrix}\begin{bmatrix}1\\x_2\end{bmatrix}\begin{bmatrix}1\\x_3\end{bmatrix}\begin{bmatrix}1\\x_4\end{bmatrix}\begin{bmatrix}1\\x_5\end{bmatrix}\begin{bmatrix}1\\x_6\end{bmatrix}$$

$$= W^{111111} + W^{211111}x_1 + W^{12111}x_2 + W^{112111}x_3 \ldots$$

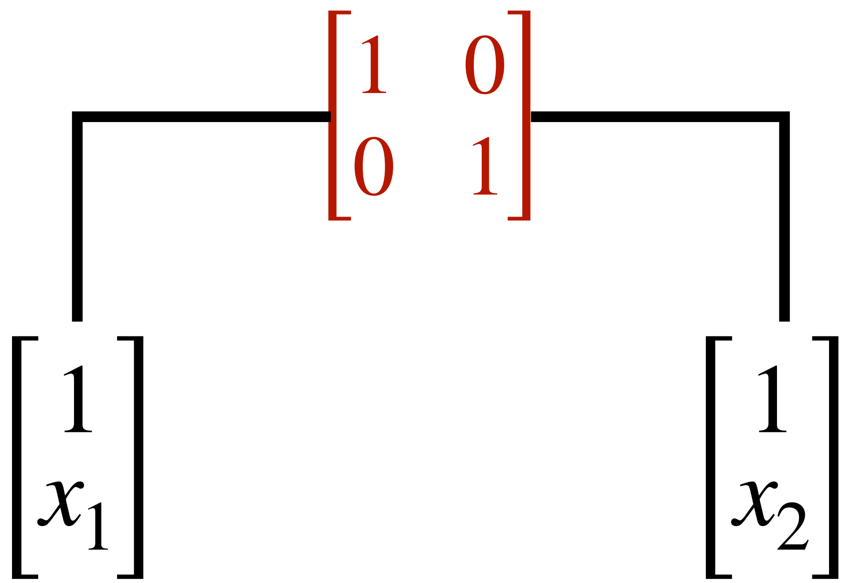
$$+ W^{221111}x_1x_2 + W^{212111}x_1x_3 + W^{12211}x_2x_3 + \ldots$$

$$+ \ldots$$

$$+ W^{222222}x_1x_2x_3x_4x_5x_6$$

Tensor Network Machine Learning

Test your understanding: what function is this?

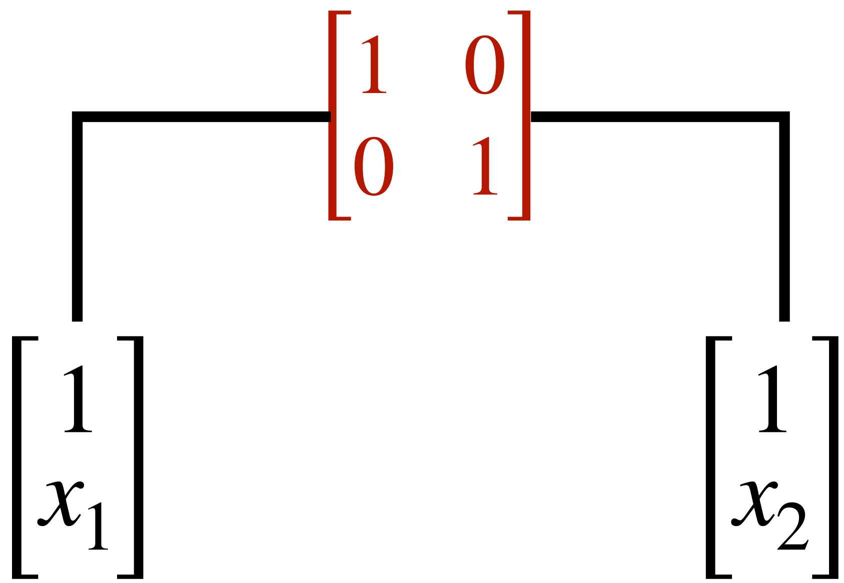


W weight tensor

$$= \ ?$$

Tensor Network Machine Learning

Test your understanding: what function is this?

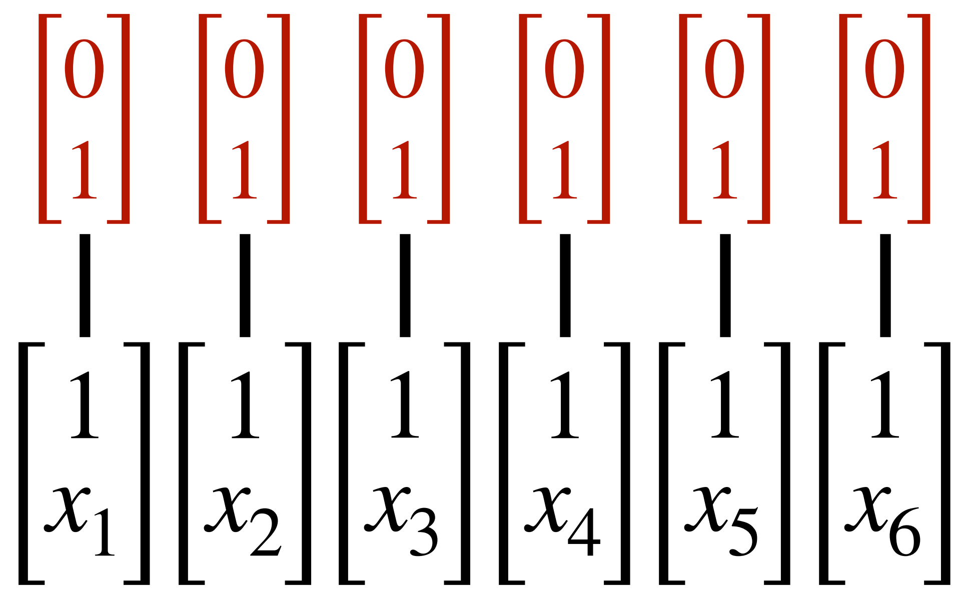


W weight tensor

$$= 1 + x_1 x_2$$

Tensor Network Machine Learning

Test your understanding: what function is this?

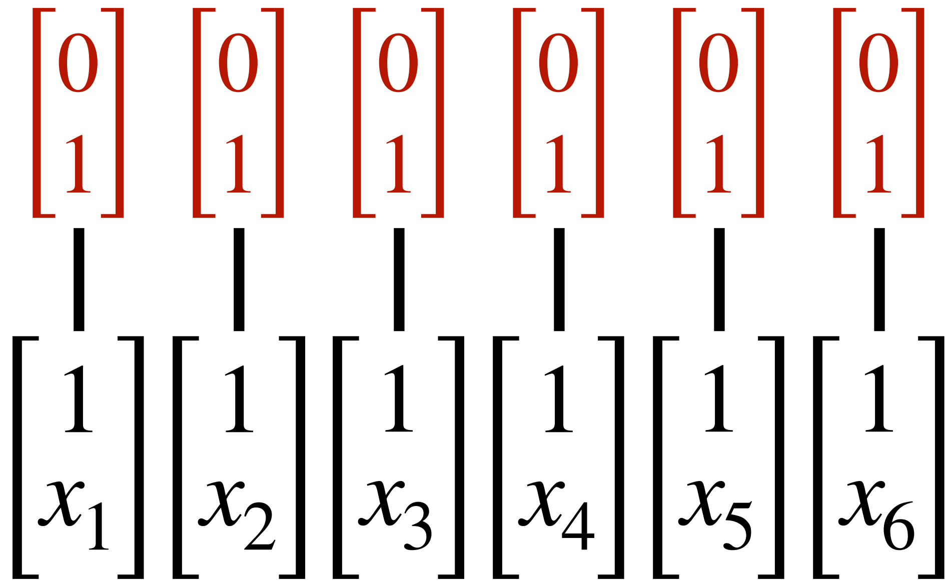


W weight tensor

$$= \ ?$$

Tensor Network Machine Learning

Test your understanding: what function is this?

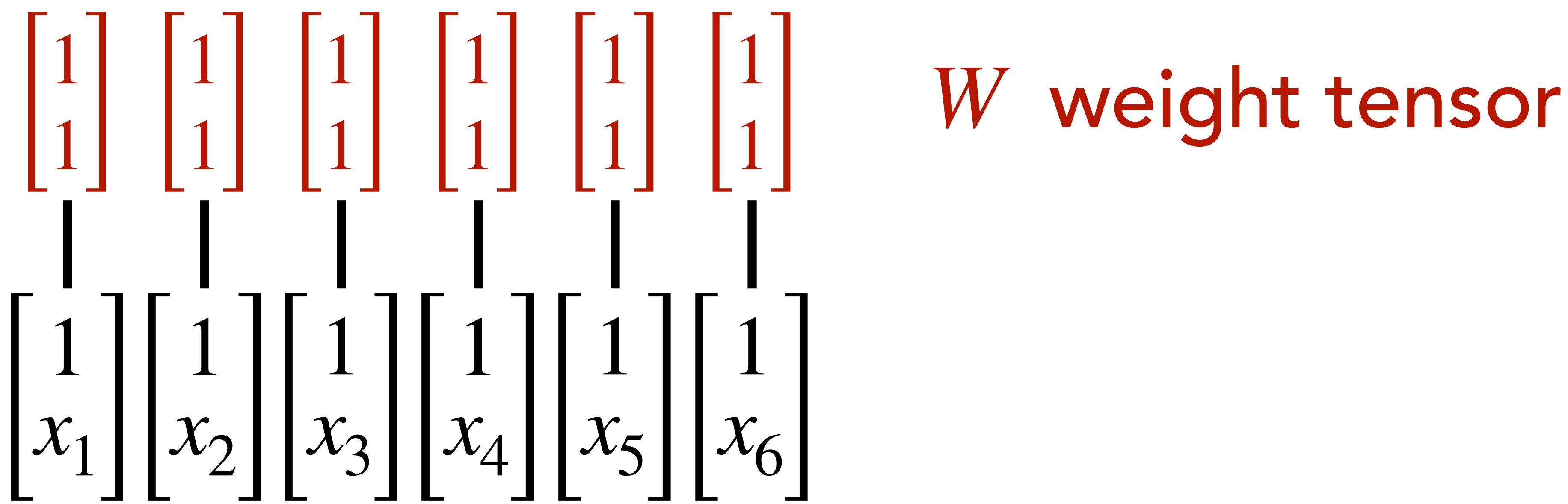


W weight tensor

$$= x_1 x_2 x_3 x_4 x_5 x_6$$

Tensor Network Machine Learning

Test your understanding: what function is this?



$$= \ ?$$

Tensor Network Machine Learning

Test your understanding: what function is this?

$$\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}$$

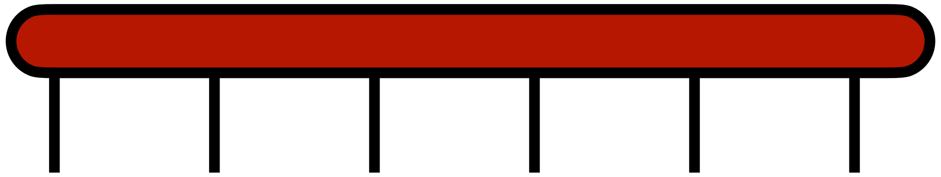
W weight tensor

$$\begin{bmatrix}1\\x_1\end{bmatrix}\begin{bmatrix}1\\x_2\end{bmatrix}\begin{bmatrix}1\\x_3\end{bmatrix}\begin{bmatrix}1\\x_4\end{bmatrix}\begin{bmatrix}1\\x_5\end{bmatrix}\begin{bmatrix}1\\x_6\end{bmatrix}$$

$$= (1 + x_1)(1 + x_2)(1 + x_3)(1 + x_4)(1 + x_5)(1 + x_6)$$

Tensor Network Machine Learning

Exponentially many weights in general

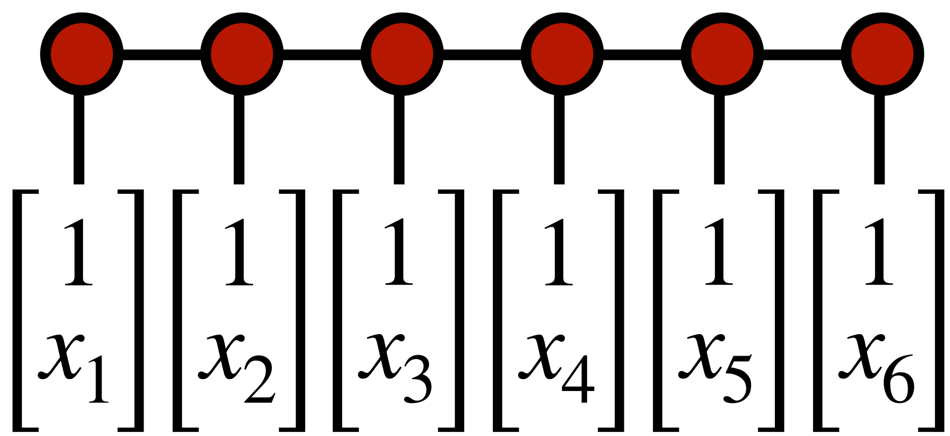
$f(x_1, x_2, \ldots, x_N) =$  W weight tensor

$$\begin{bmatrix}1\\x_1\end{bmatrix}\begin{bmatrix}1\\x_2\end{bmatrix}\begin{bmatrix}1\\x_3\end{bmatrix}\begin{bmatrix}1\\x_4\end{bmatrix}\begin{bmatrix}1\\x_5\end{bmatrix}\begin{bmatrix}1\\x_6\end{bmatrix}$$

Tensor Network Machine Learning

Use tensor network to make efficient

$f(x_1, x_2, \ldots, x_N) =$



W weight MPS

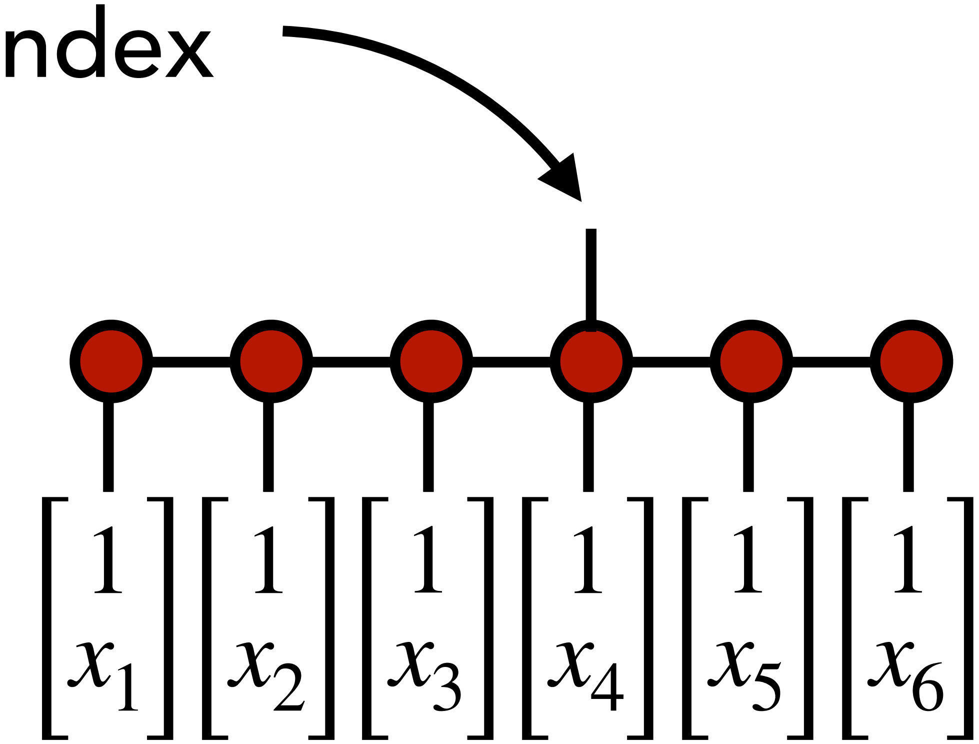
Higher bond dimension χ = more representation power

Tensor Network Machine Learning

For supervised learning,

put extra label index

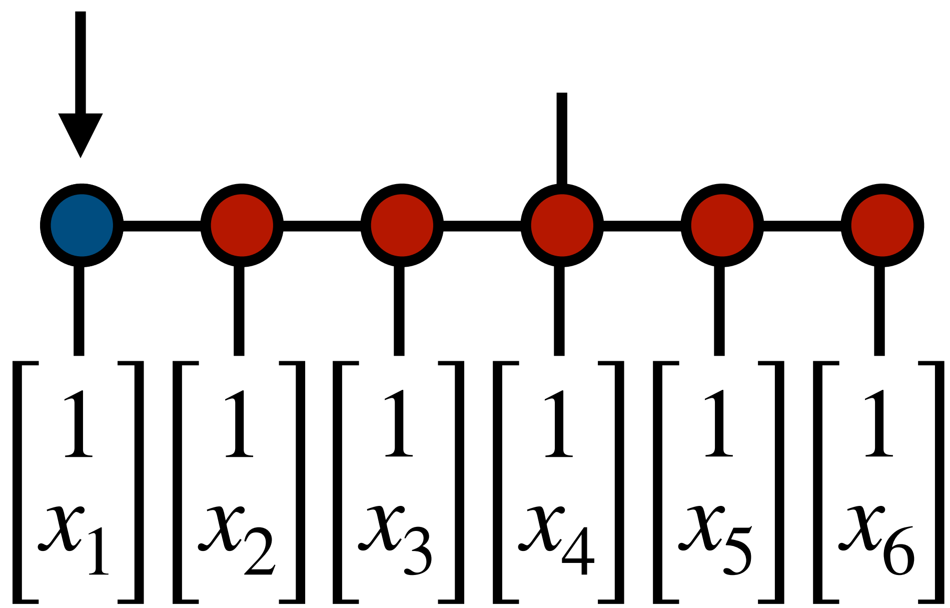
$$f(x_1, x_2, \ldots, x_N) =$$



W weight MPS

Tensor Network Machine Learning

Train using alternating gradient descent

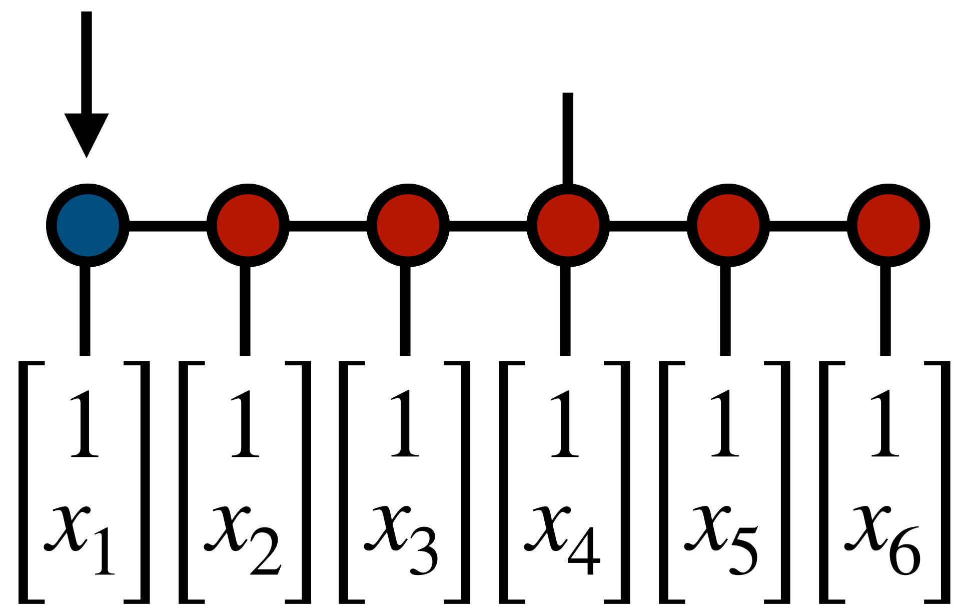
$f(x_1, x_2, \ldots, x_N) =$ 

Could use favorite neural network or auto-differentiation framework (JAX, PyTorch, etc.)

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



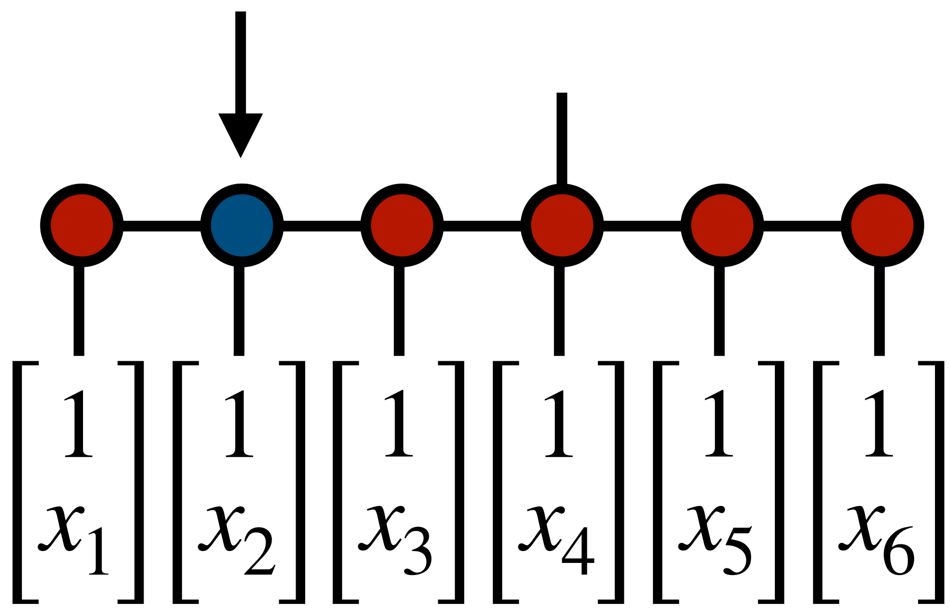
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



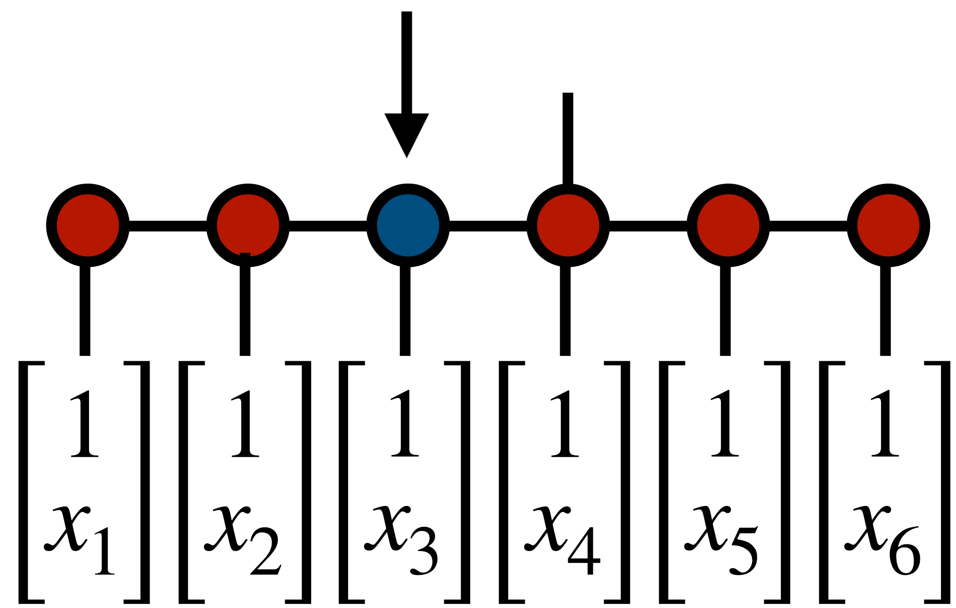
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



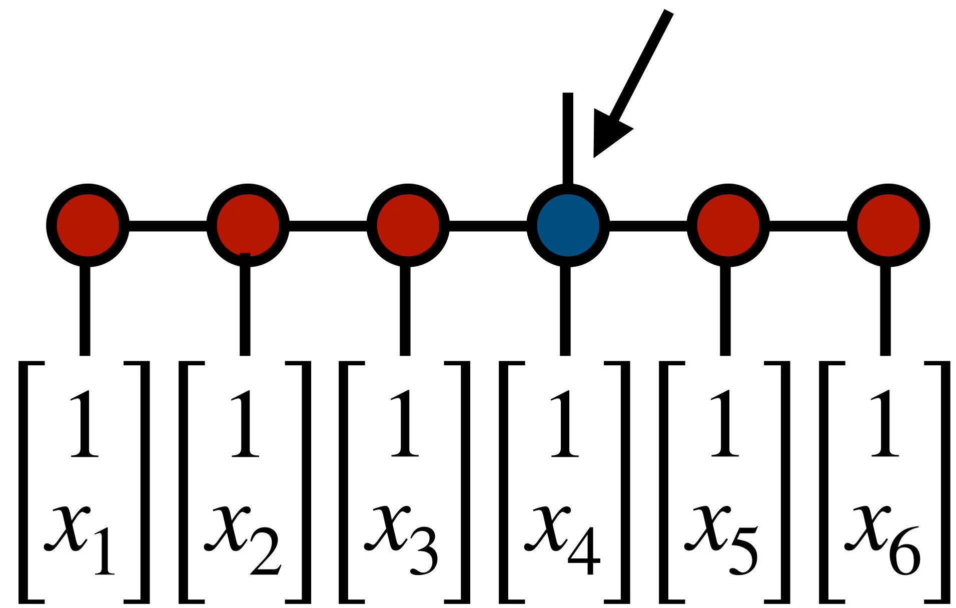
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



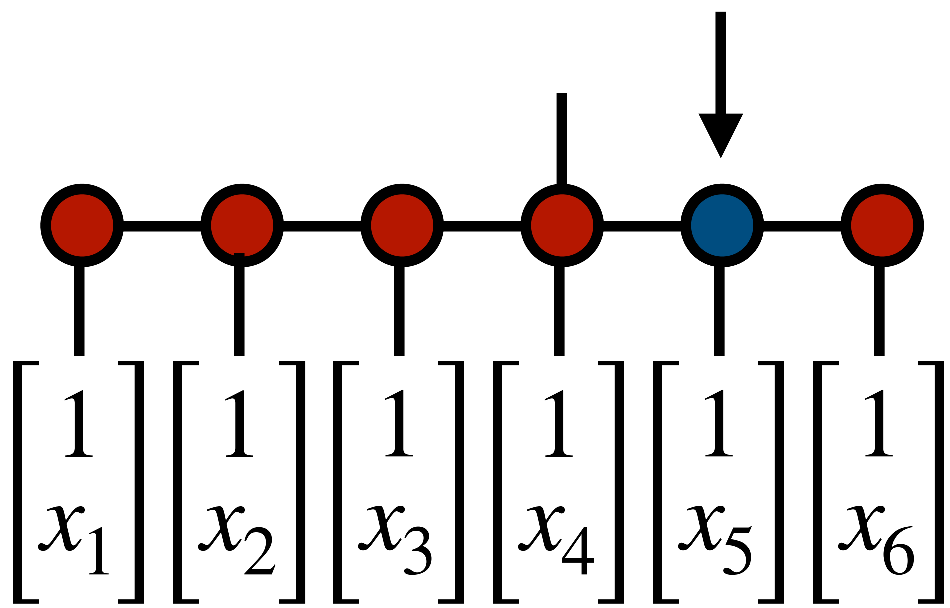
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



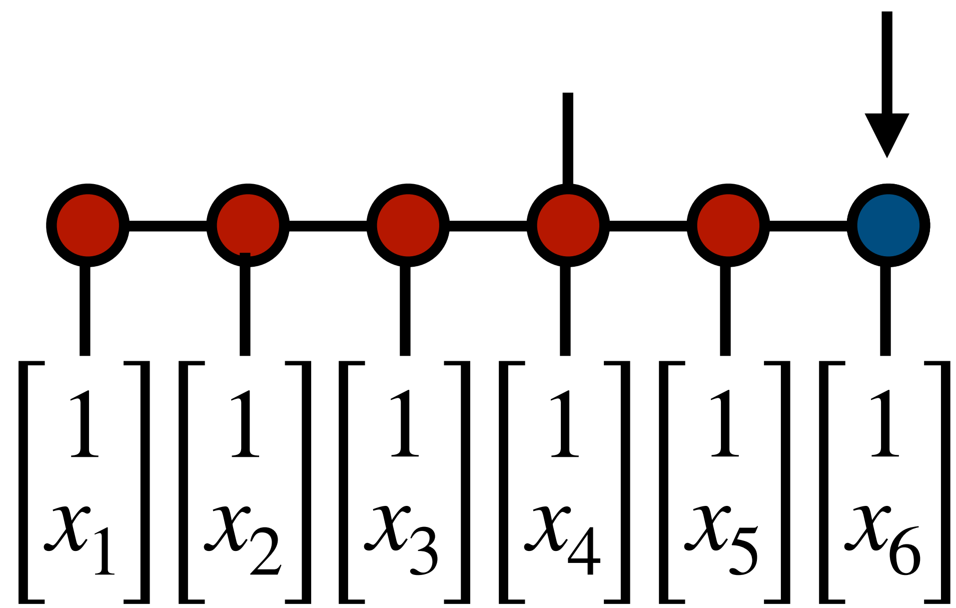
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



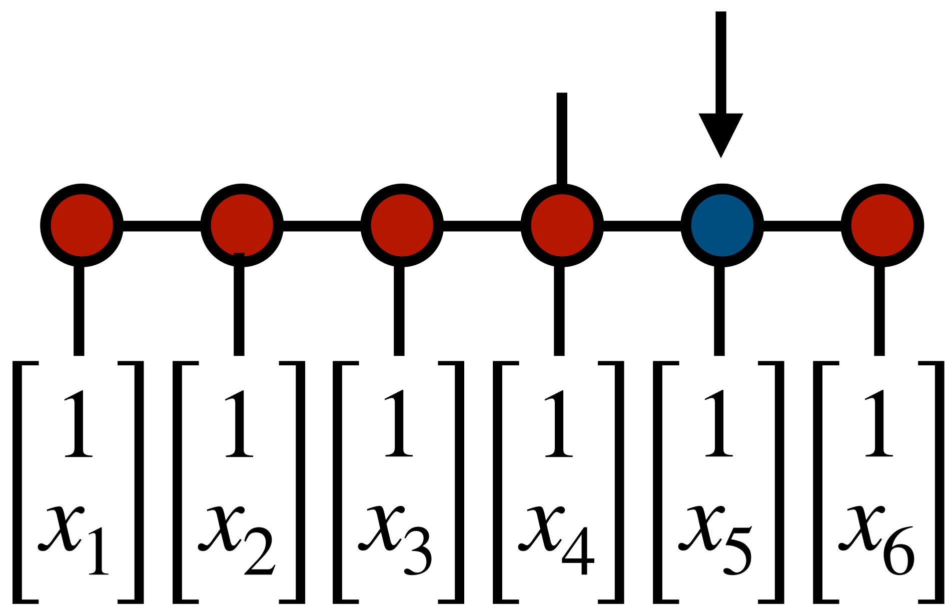
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



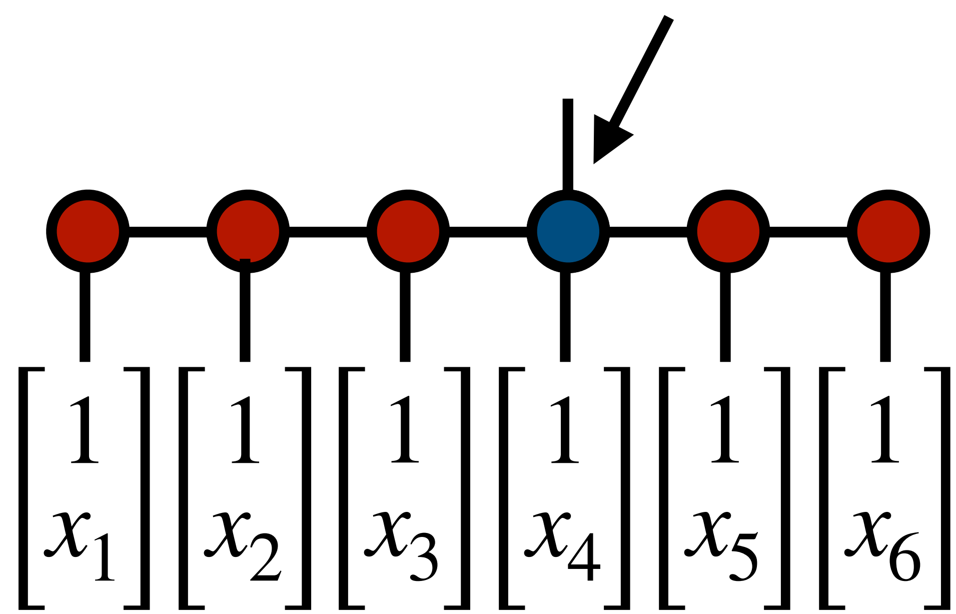
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



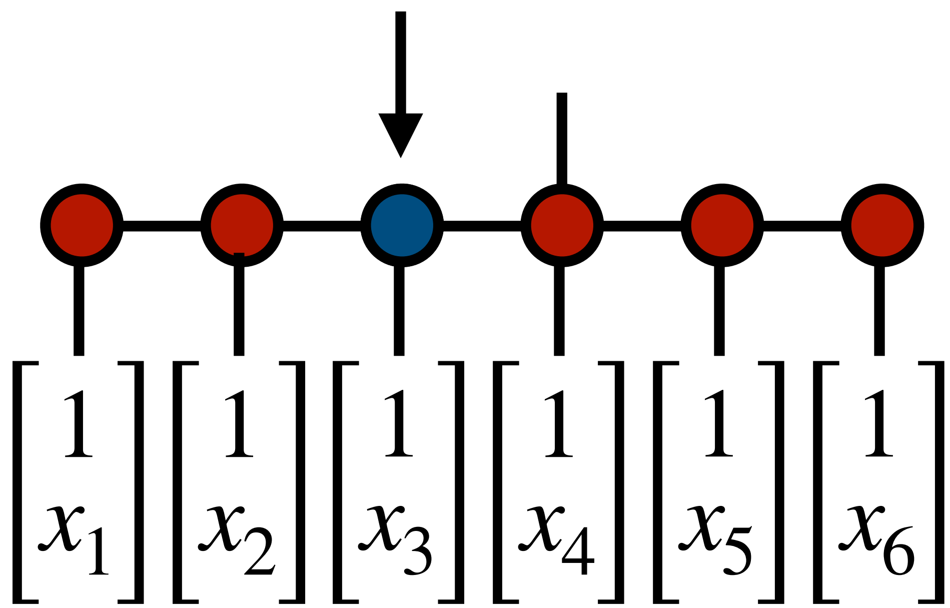
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



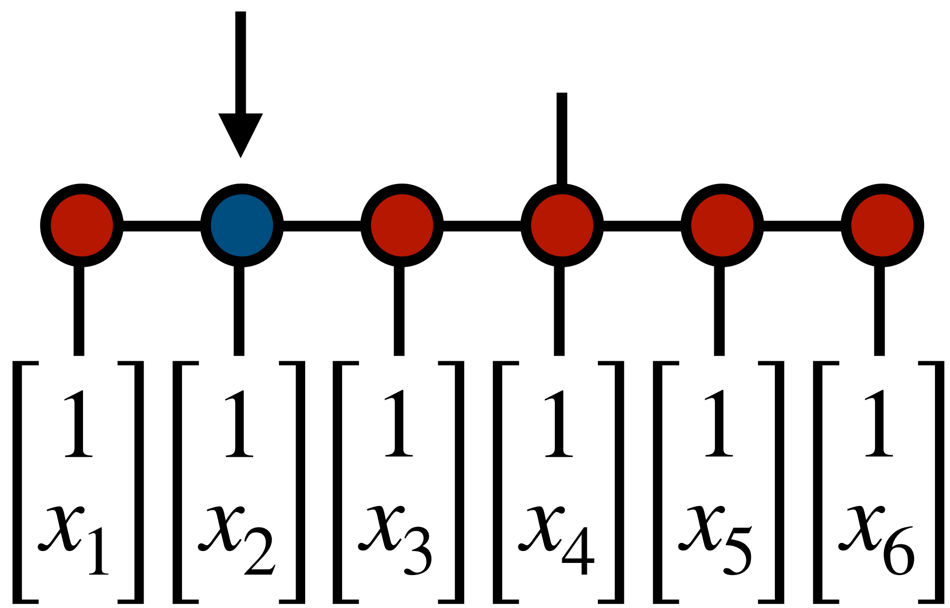
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$$f(x_1, x_2, \ldots, x_N) =$$



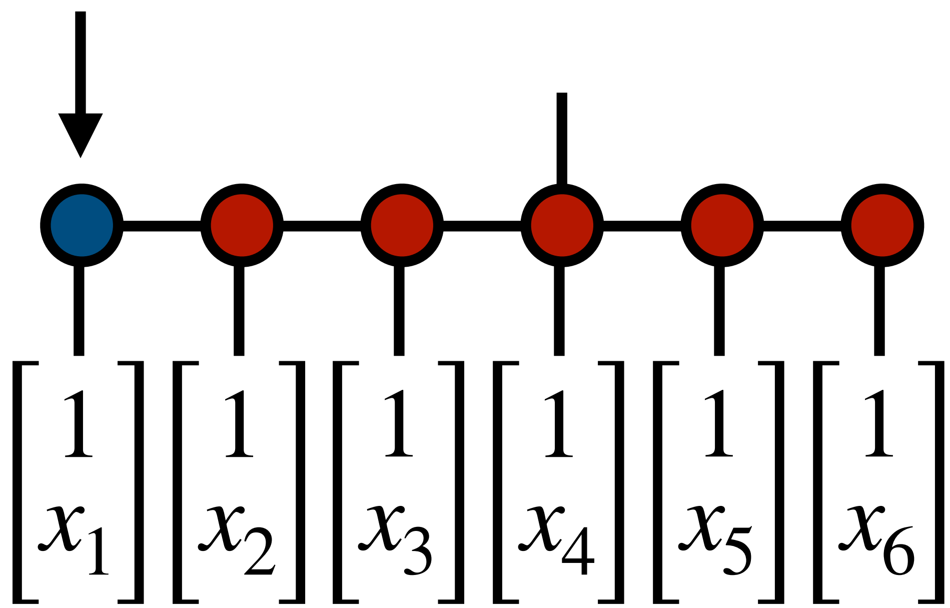
Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Train using alternating gradient descent

$f(x_1, x_2, \ldots, x_N) =$

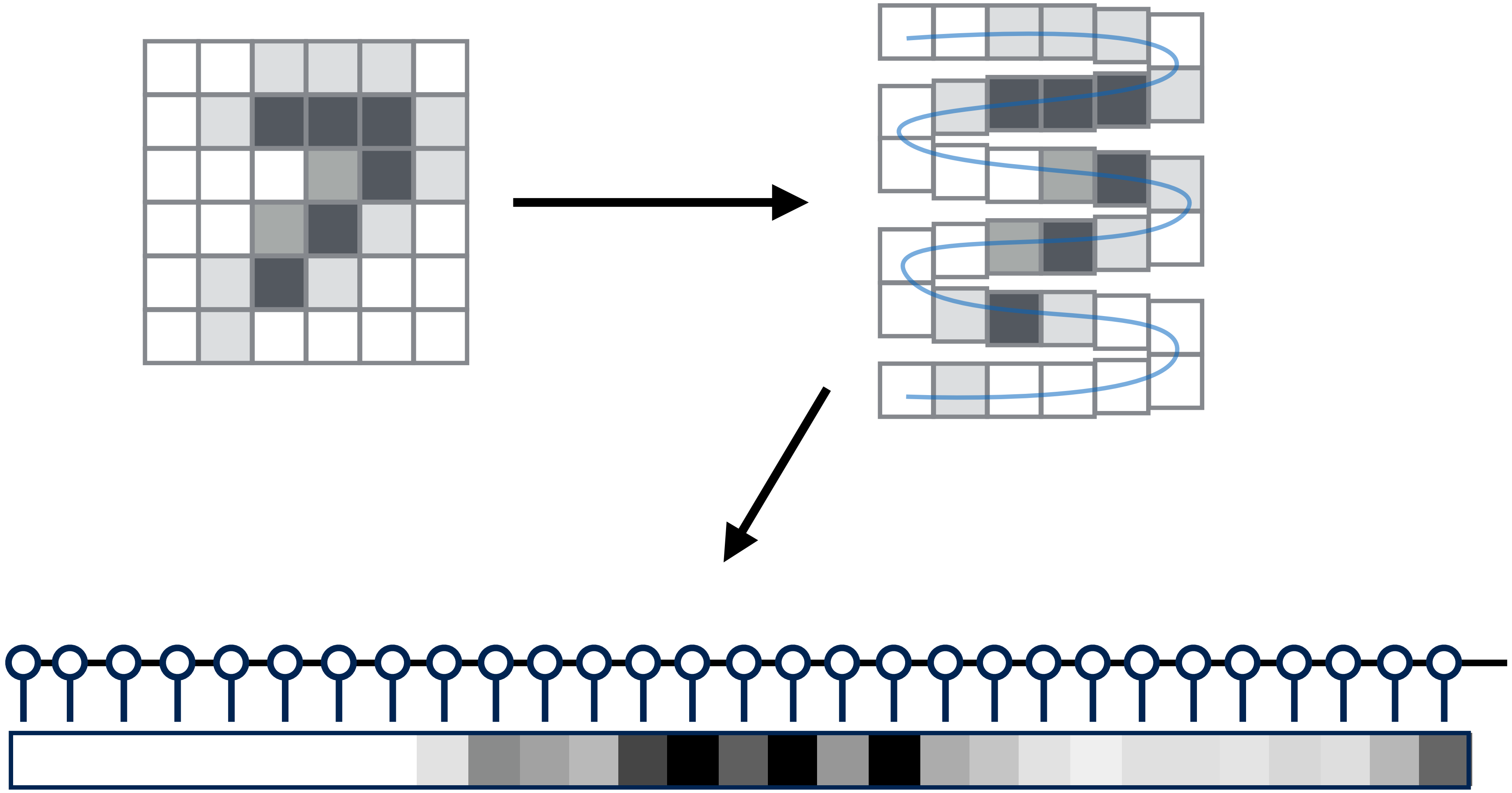


Minimize cost function such as squared error

$$C = \frac{1}{N_T} \sum_{j=1}^{N_T} \left(f(\vec{x}_j) - y_j \right)^2$$

Tensor Network Machine Learning

Example: Supervised learning of MNIST handwriting



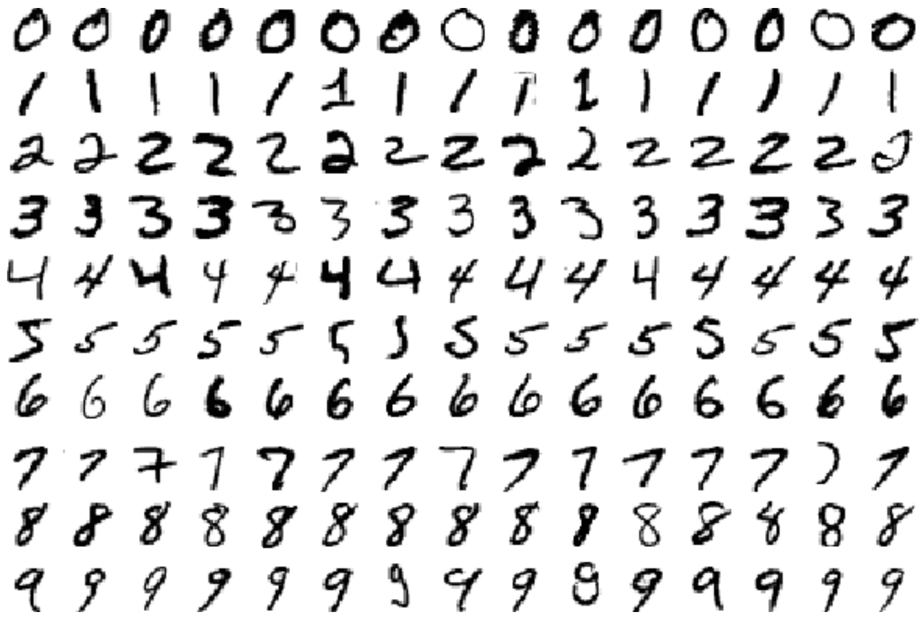
Train to 99.95% accuracy on 60,000 training images

Obtain **99.03%** accuracy on 10,000 test images (only 97 incorrect)

Stoudenmire, Schwab, arxiv:1605.05775

Tensor Network Machine Learning

Example: Supervised learning of MNIST handwriting



Bond dimension	Test Set Error
m = 10	~5% (500/10,000 incorrect)
m = 20	~2% (200/10,000 incorrect)
m = 120	0.97% (97/10,000 incorrect)

Stoudenmire, Schwab, arxiv:1605.05775

Tensor Network Machine Learning

Amplitude encoding

In this representation, indices do not correspond to different features.

Instead, indices "collectively" access each feature.

Let's see how this works...

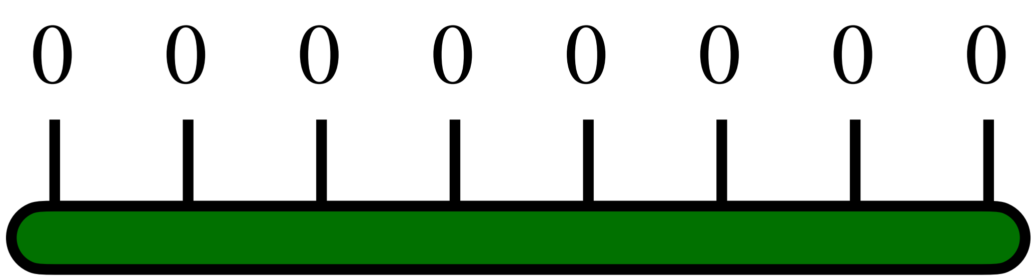
Tensor Network Machine Learning

Amplitude encoding

Say we have data vector $\vec{x}$

$$\vec{x} = [x_0, x_1, x_2, \ldots, x_{N-1}] \qquad \text{(zero indexed)}$$

Define tensor such that



$$= x_0$$

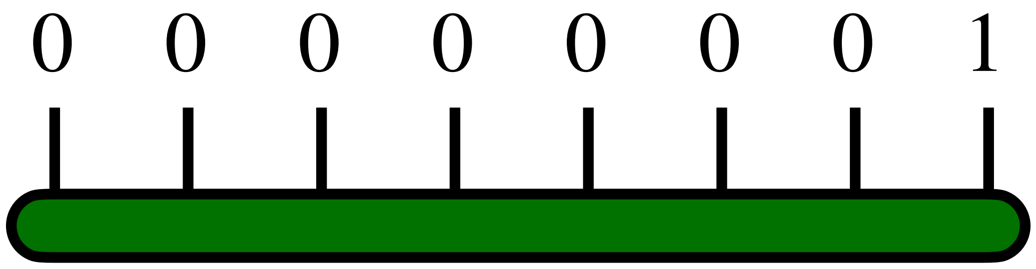
Tensor Network Machine Learning

Amplitude encoding

Say we have data vector $\vec{x}$

$$\vec{x} = [x_0, x_1, x_2, \ldots, x_{N-1}] \qquad \text{(zero indexed)}$$

Define tensor such that



$$= x_1$$

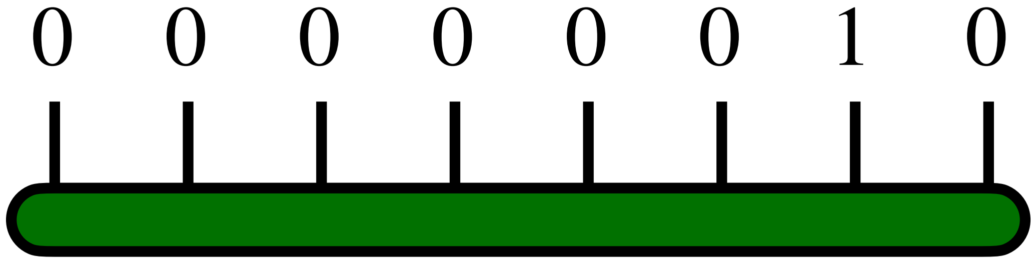
Tensor Network Machine Learning

Amplitude encoding

Say we have data vector $\vec{x}$

$$\vec{x} = [x_0, x_1, x_2, \ldots, x_{N-1}] \qquad \text{(zero indexed)}$$

Define tensor such that



$$= x_2$$

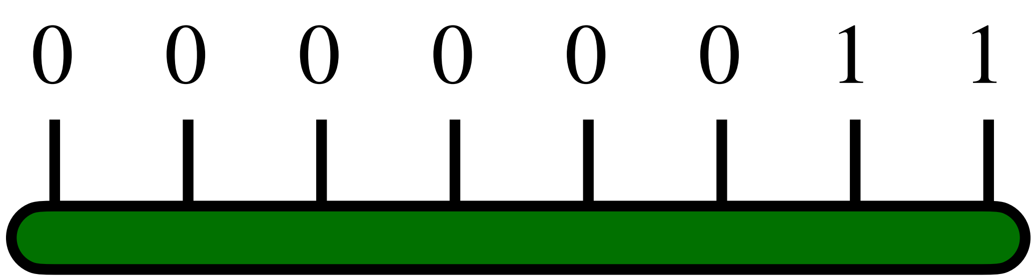
Tensor Network Machine Learning

Amplitude encoding

Say we have data vector $\vec{x}$

$$\vec{x} = [x_0, x_1, x_2, \ldots, x_{N-1}]$$ (zero indexed)

Define tensor such that



$$= x_3$$

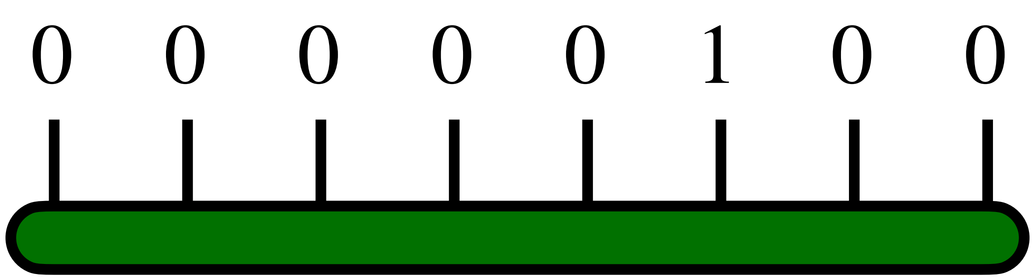
Tensor Network Machine Learning

Amplitude encoding

Say we have data vector $\vec{x}$

$$\vec{x} = [x_0, x_1, x_2, \ldots, x_{N-1}]$$ (zero indexed)

Define tensor such that



$$= x_4$$

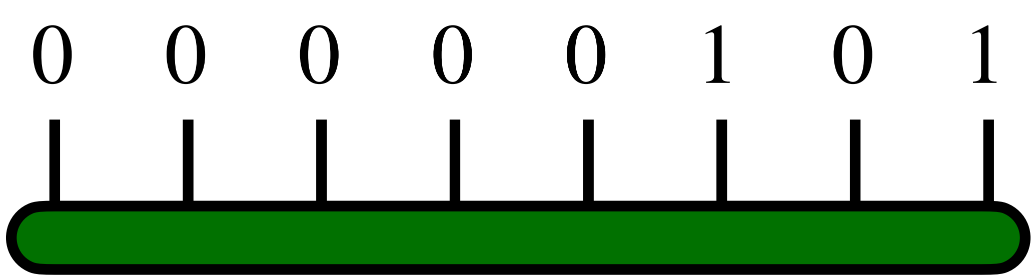
Tensor Network Machine Learning

Amplitude encoding

Say we have data vector $\vec{x}$

$$\vec{x} = [x_0, x_1, x_2, \ldots, x_{N-1}] \qquad \text{(zero indexed)}$$

Define tensor such that



$$= x_5$$

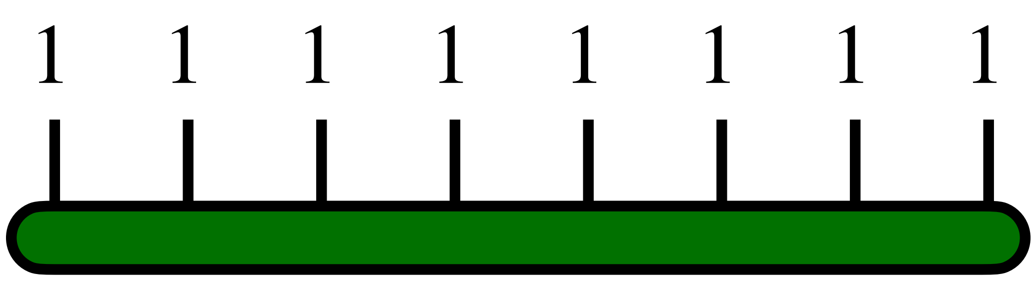
Tensor Network Machine Learning

Amplitude encoding

Say we have data vector $\vec{x}$

$$\vec{x} = [x_0, x_1, x_2, \ldots, x_{N-1}] \qquad \text{(zero indexed)}$$

Define tensor such that



$= x_{N-1}$

Tensor Network Machine Learning

Amplitude encoding

Viewed as a quantum state, it is just

$$\vec{x} = [x_0, x_1, x_2, \ldots, x_{N-1}] \qquad \text{(zero indexed)}$$

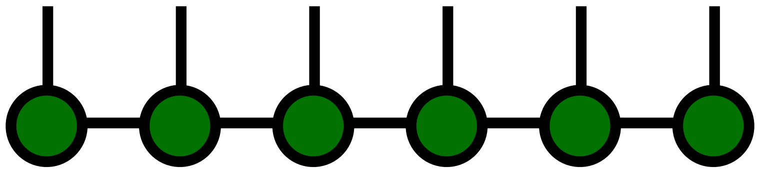
$$\downarrow$$

$$|\vec{x}\rangle = \sum_{i=0}^{2^n-1} x_i \, |i\rangle$$

Tensor Network Machine Learning

Amplitude encoding

To make efficient, again factorize as MPS

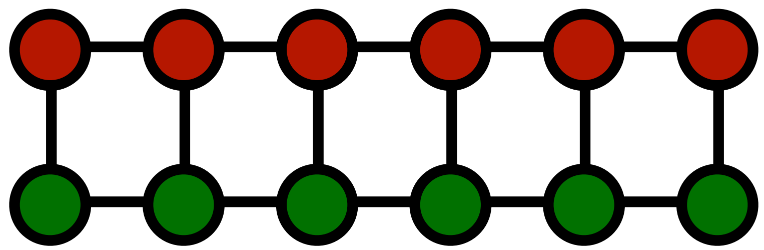
$\vec{x} \rightarrow$  $\approx$ 

Tensor Network Machine Learning

Amplitude encoding

To make into a model, contract with weight MPS

$$f(\vec{x}) =$$

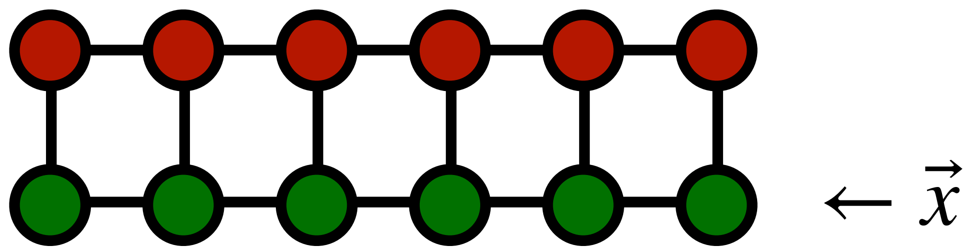


$$\leftarrow \vec{x}$$

Tensor Network Machine Learning

Amplitude encoding

$f(\vec{x}) =$



Since indices enumerate entries of $\vec{x}$ one-by-one, it is just a 'linear classifier' in tensor form

$$f(\vec{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

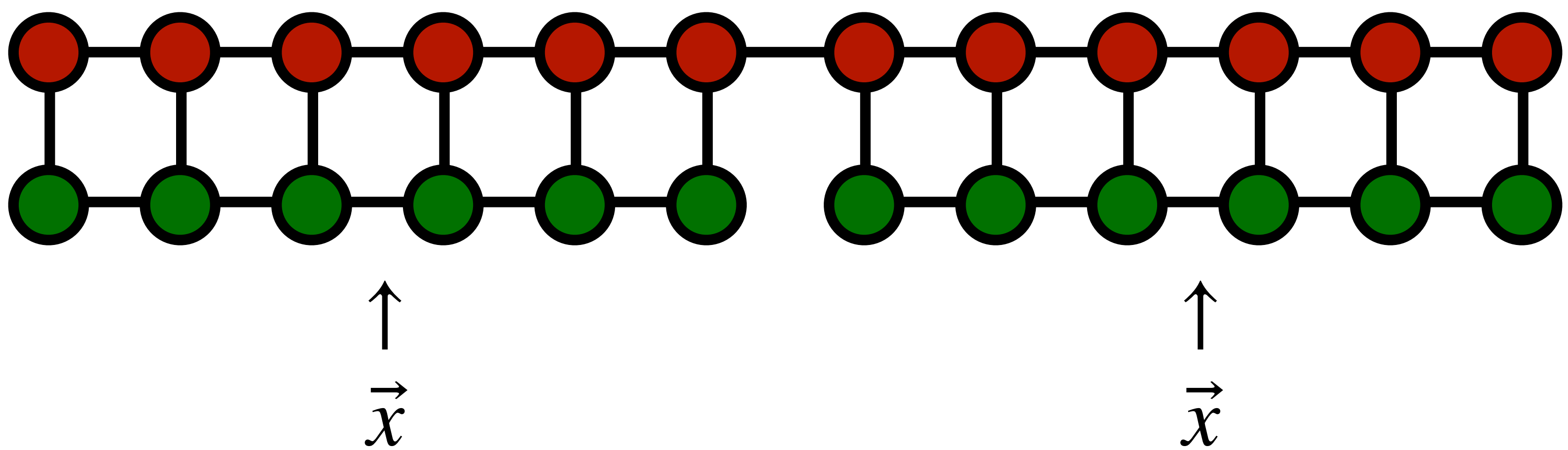
Not very powerful...

Tensor Network Machine Learning

Amplitude encoding

Make more powerful by repeating ("stacking") data input

$$f(\vec{x}) =$$



Now model contains linear + quadratic terms

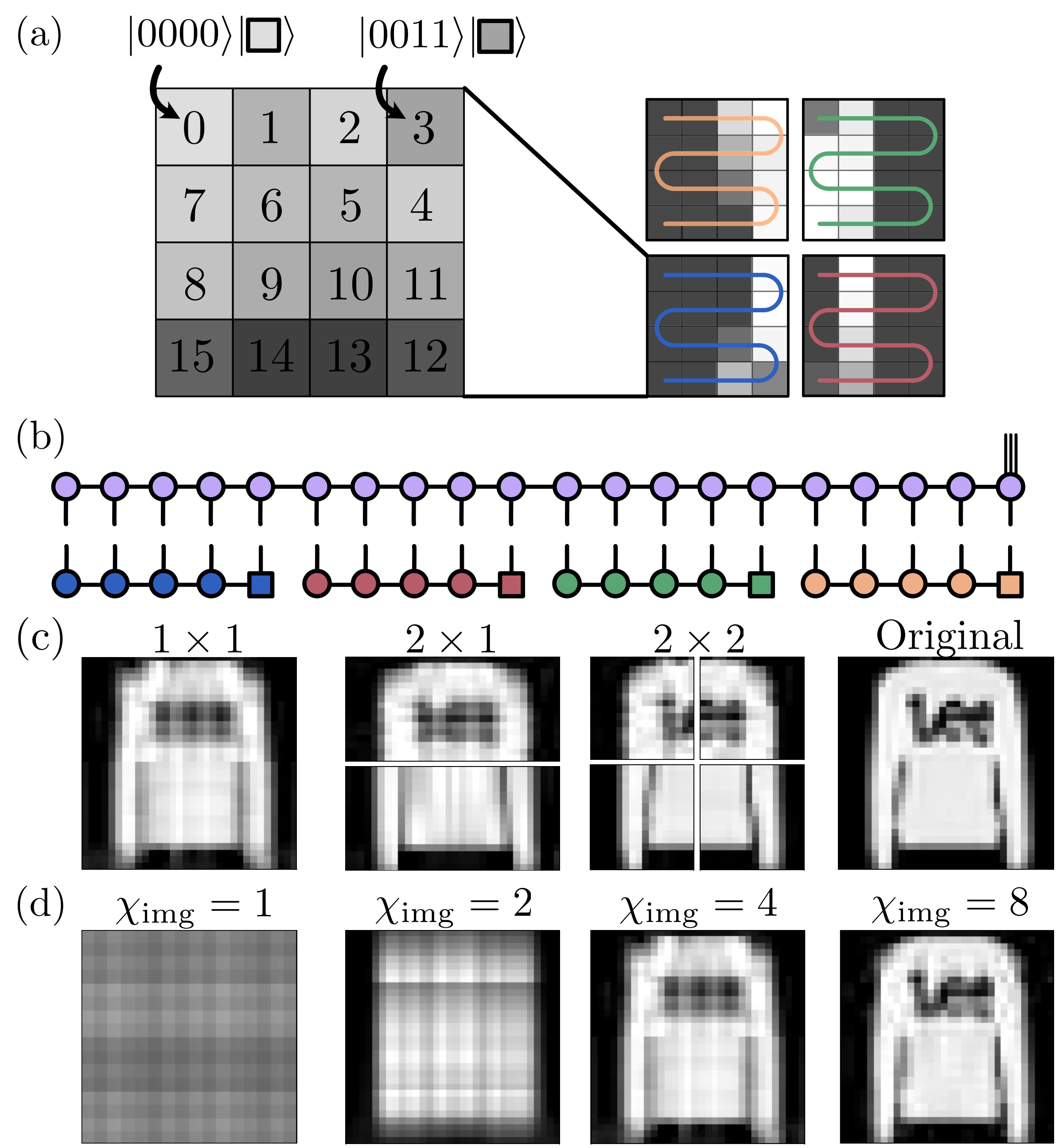
$$f(\vec{x}) = a + w_1 x_1 + \ldots + w_{11}(x_1)^2 + w_{12} x_1 x_2 + \ldots$$

Tensor Network Machine Learning

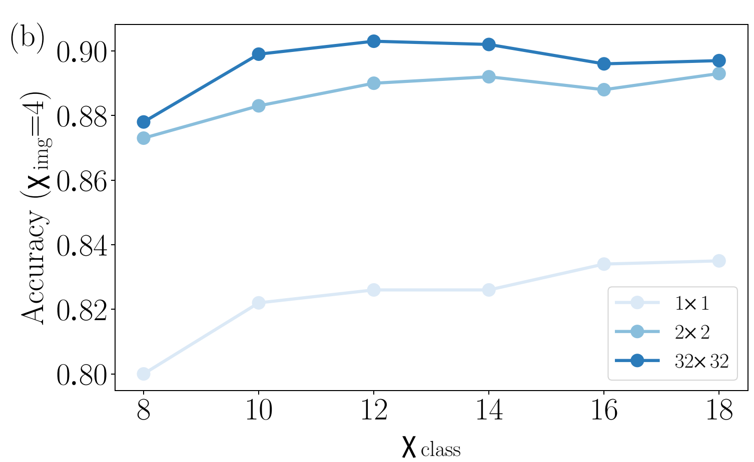
Application: Supervised learning of "Fashion-MNIST"

Dilip, Liu, Smith, Pollmann, PRR 4, 043007 (2022)

Use patches of amplitude and basis encoded data



Obtain 90% test accuracy (!) using $\chi = 10$

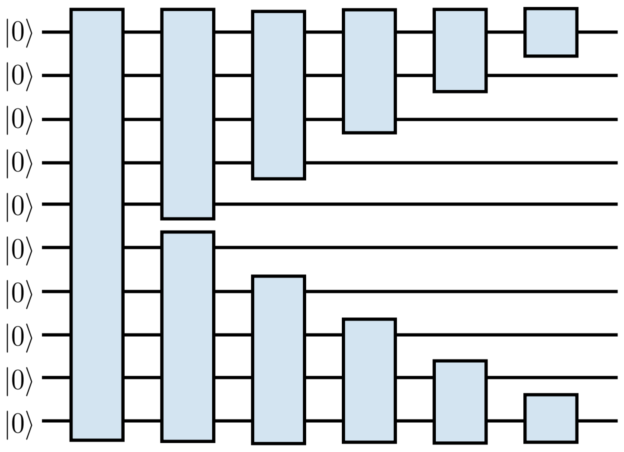


Tensor Network Machine Learning

Application: Quantum Circuit Learning Models

Wright, Barratt, Green, et al., arxiv 2205.09768 (2022)

Using amplitude encoded data, propose circuits equaling MPS



Use "stacking" (inputting data multiple times) to get higher-order functions

Deterministic (no gradient, linear algebra) learning

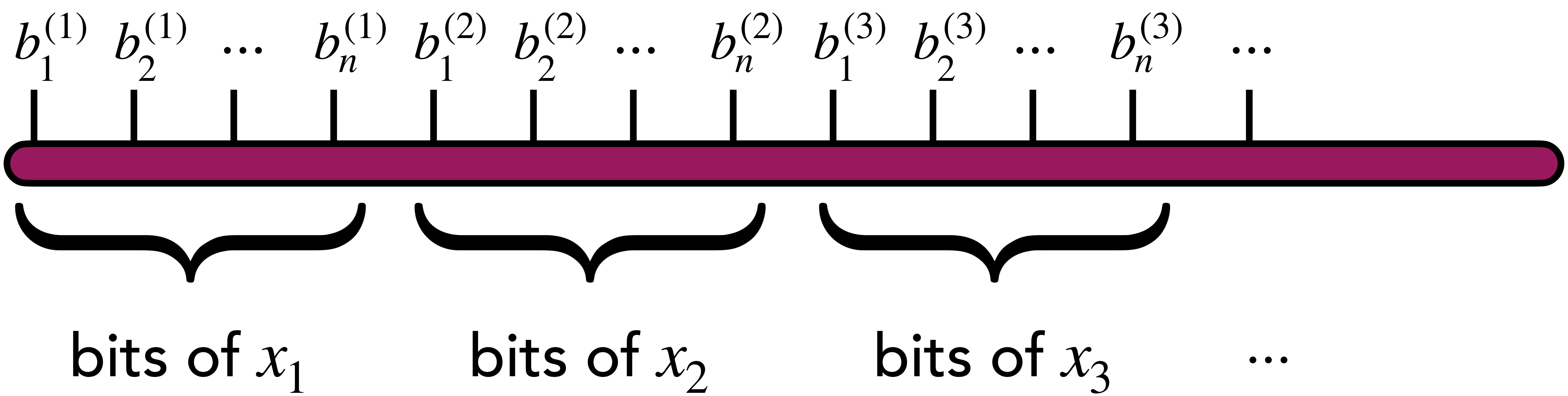
Tensor Network Machine Learning

Let's do some brief
"theory of tensor network machine learning"...

Tensor Network Machine Learning

Mixing high-dimensional and low-dimensional encoding gives "universal approximation theorem" for tensor networks

$$f(x_1, x_2, x_3, \ldots, x_N) =$$

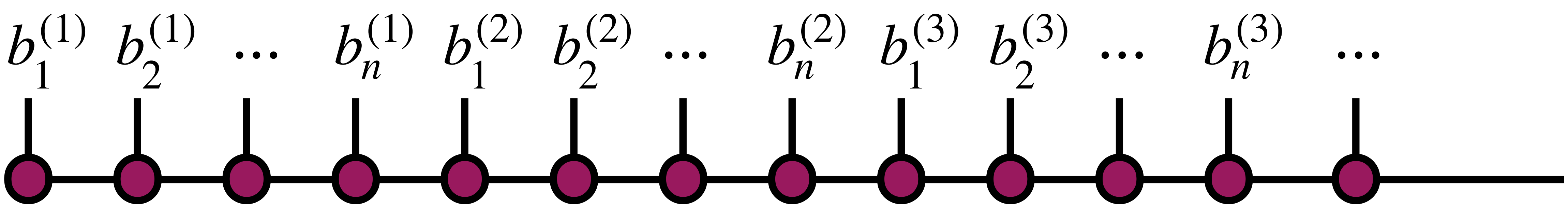


Tensor entries arbitrary, so can store any function on exponentially fine continuum grid

Tensor Network Machine Learning

Mixing high-dimensional and low-dimensional encoding gives "universal approximation theorem" for tensor networks

$$f(x_1, x_2, x_3, \ldots, x_N) \approx$$



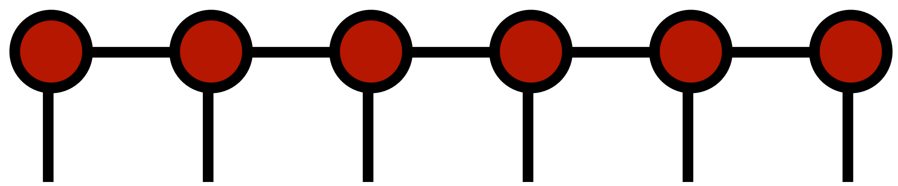
And any tensor is representable by MPS with large enough bond dimension χ

No explicit non-linearities, and yet true

Tensor Network Machine Learning

Tensor network learning is a form of kernel learning

$f(\vec{x}) =$



W

$$\begin{bmatrix}1\\x_1\end{bmatrix}\begin{bmatrix}1\\x_2\end{bmatrix}\begin{bmatrix}1\\x_3\end{bmatrix}\begin{bmatrix}1\\x_4\end{bmatrix}\begin{bmatrix}1\\x_5\end{bmatrix}\begin{bmatrix}1\\x_6\end{bmatrix} \quad \phi(\vec{x})$$

$$= W \cdot \phi(\vec{x})$$

Yet training scales **linearly** with data set size

Does not use "kernel trick" which scales quadratically

The Future of Tensor Network Machine Learning

Future Direction #1:
Continuum Functions

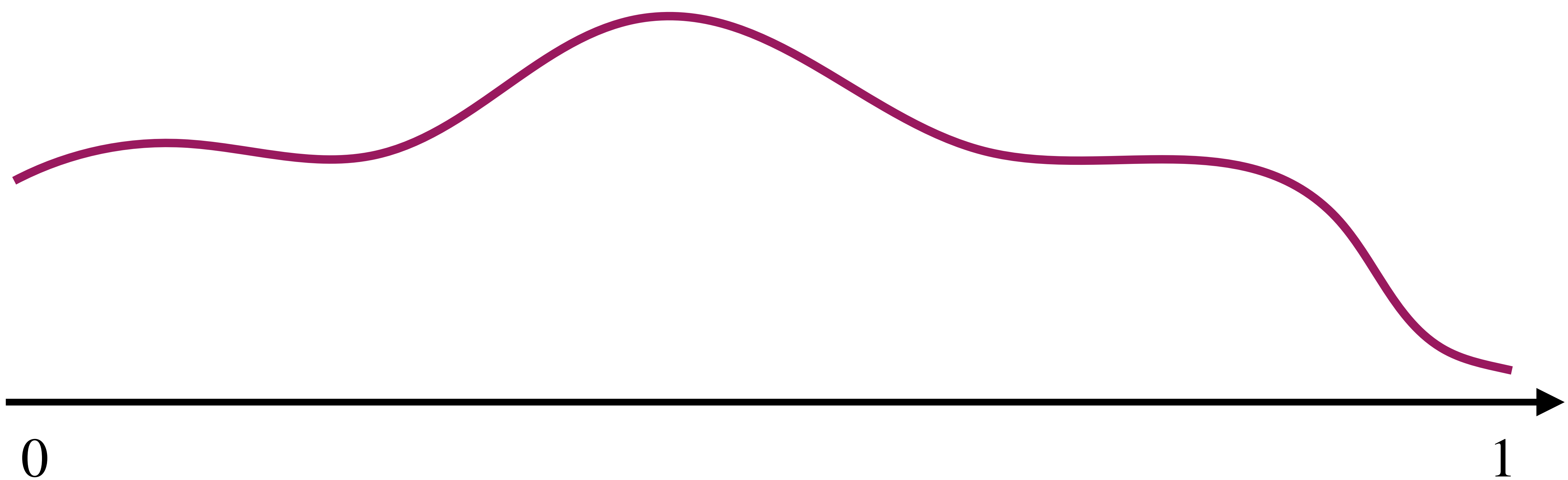
Future Directions – Continuum Functions

Continuum amplitude encoding

Especially useful for **continuous inputs** to tensors

For a function f, evaluate on fine grid of spacing $1/2^n$

$$\vec{f} = [f(0.0000),\ f(0.0001),\ f(0.0010),\ \ldots,\ f(0.1111)]$$

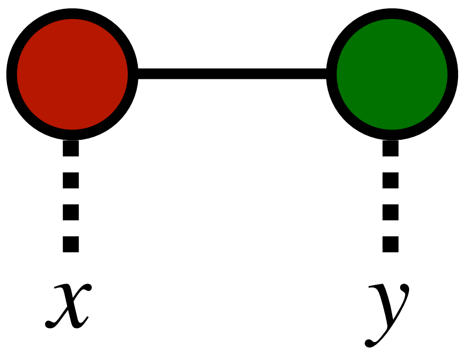


Future Directions – Continuum Functions

One powerful technique is

Continuum amplitude encoding ("quantics tensor train")

Especially useful for **continuous inputs** to tensors



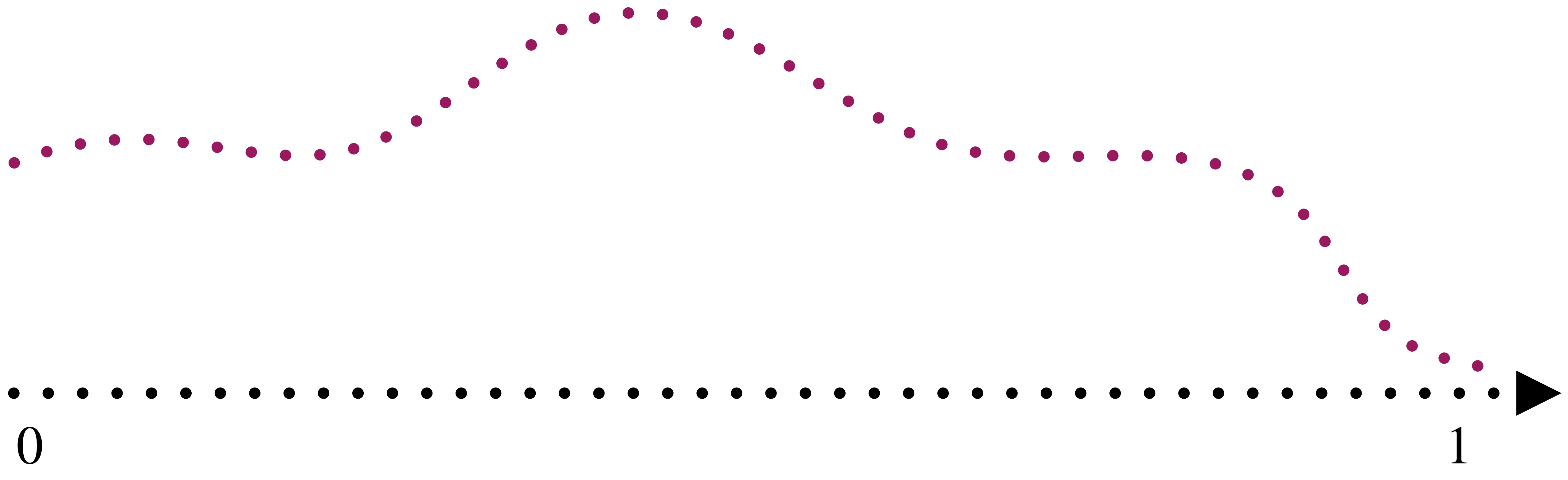
$$\sum_i A^i(x) B^i(y)$$

Future Directions – Continuum Functions

Continuum amplitude encoding

For a function f, evaluate on fine grid of spacing $1/2^n$

$$\vec{f} = [f(0.0000),\ f(0.0001),\ f(0.0010),\ \ldots,\ f(0.1111)]$$



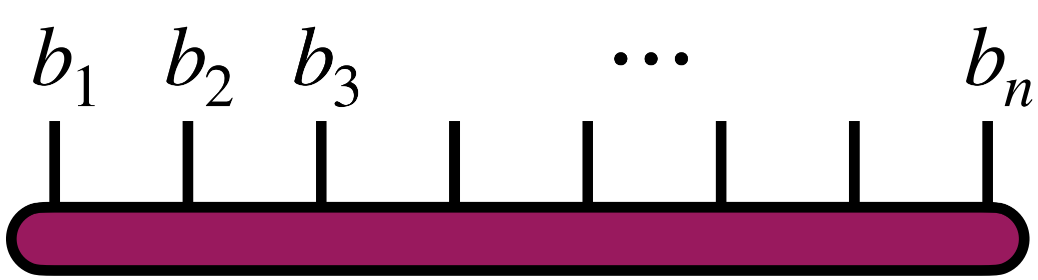
Future Directions – Continuum Functions

Continuum amplitude encoding

For a function f, evaluate on fine grid of spacing $1/2^n$

$$\vec{f} = [f(0.0000),\ f(0.0001),\ f(0.0010),\ \ldots,\ f(0.1111)]$$

Define tensor such that



$$= f(0.b_1 b_2 b_3 \cdots b_n)$$

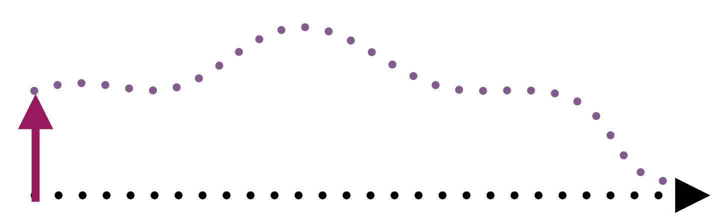
Tensor Network Machine Learning

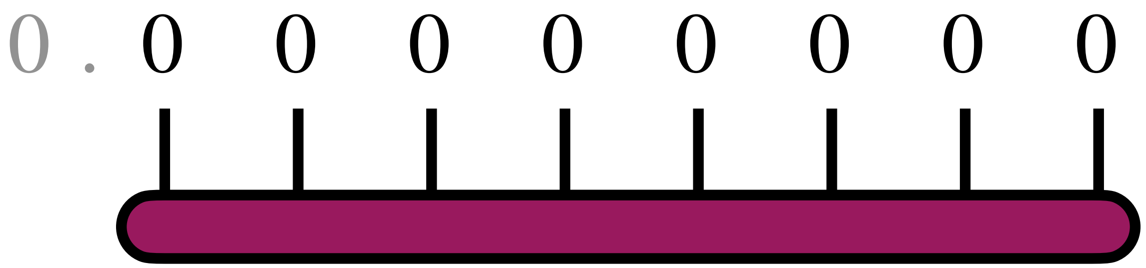
Continuum amplitude encoding

For a function f, evaluate on fine grid of spacing $1/2^n$

$$\vec{f} = [f(0.0000),\ f(0.0001),\ f(0.0010),\ \ldots,\ f(0.1111)]$$

Define tensor such that





$$= f(0.000000)$$

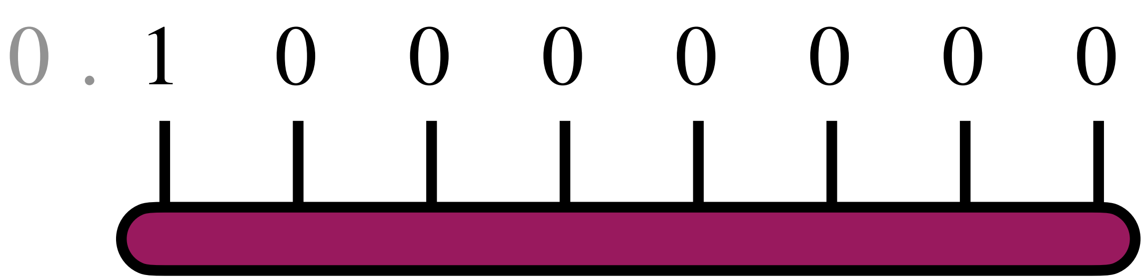
Tensor Network Machine Learning

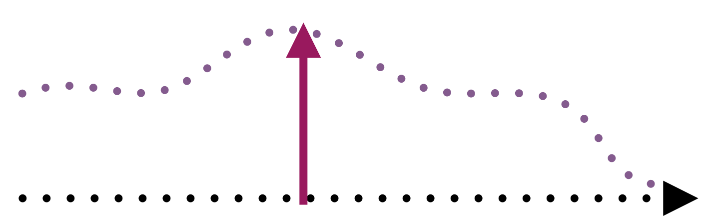
Continuum amplitude encoding

For a function f, evaluate on fine grid of spacing $1/2^n$

$$\vec{f} = [f(0.0000),\ f(0.0001),\ f(0.0010),\ \ldots,\ f(0.1111)]$$

Define tensor such that





$$= f(0.100000)$$

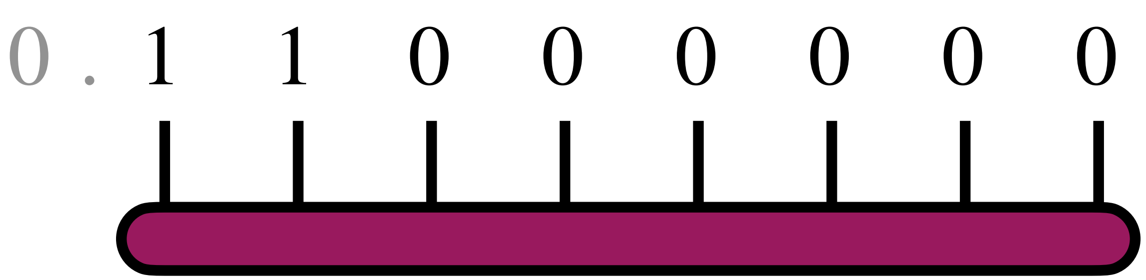
Tensor Network Machine Learning

Continuum amplitude encoding

For a function f, evaluate on fine grid of spacing $1/2^n$

$$\vec{f} = [f(0.0000),\ f(0.0001),\ f(0.0010),\ \ldots,\ f(0.1111)]$$

Define tensor such that



$$= f(0.110000)$$

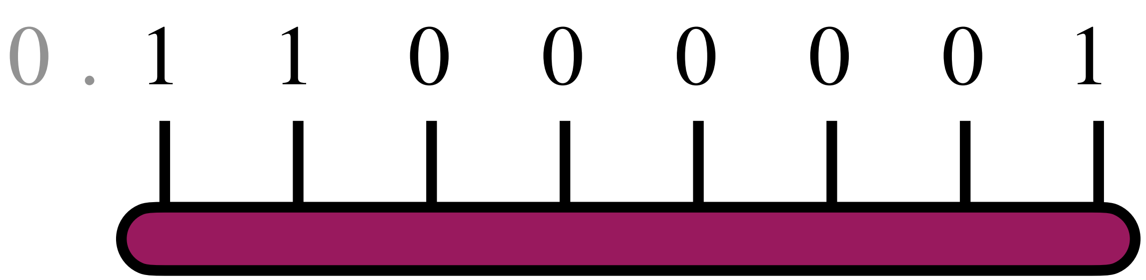
Tensor Network Machine Learning

Continuum amplitude encoding

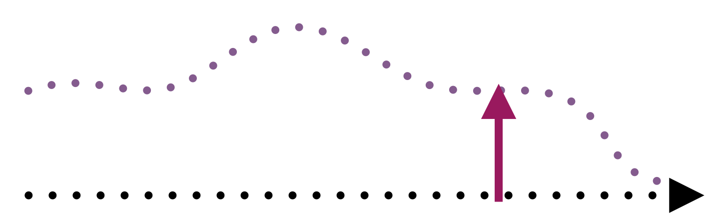
For a function f, evaluate on fine grid of spacing $1/2^n$

$$\vec{f} = [f(0.0000),\ f(0.0001),\ f(0.0010),\ \ldots,\ f(0.1111)]$$

Define tensor such that



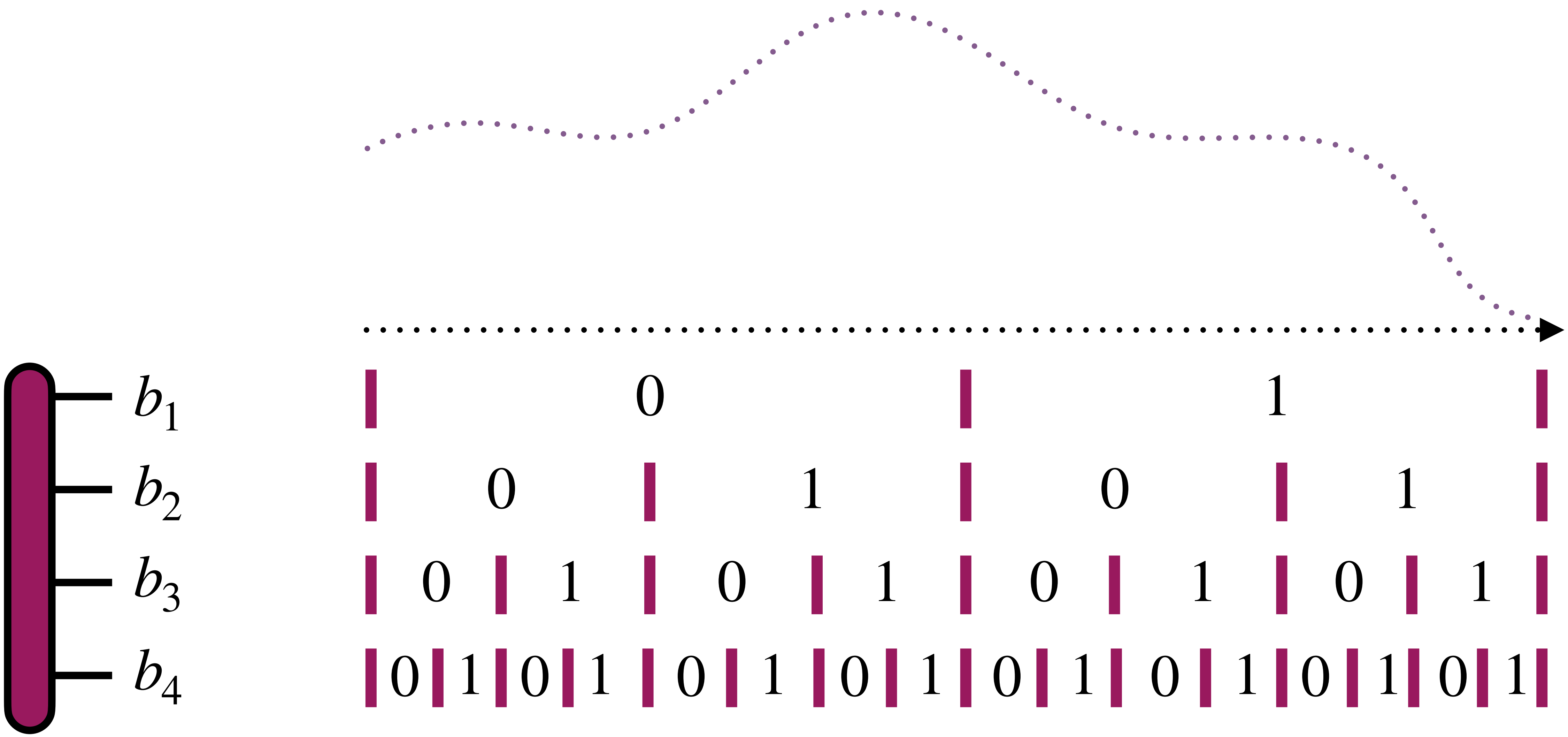
$$= f(0.110001)$$



Future Directions – Continuum Functions

Continuum amplitude encoding

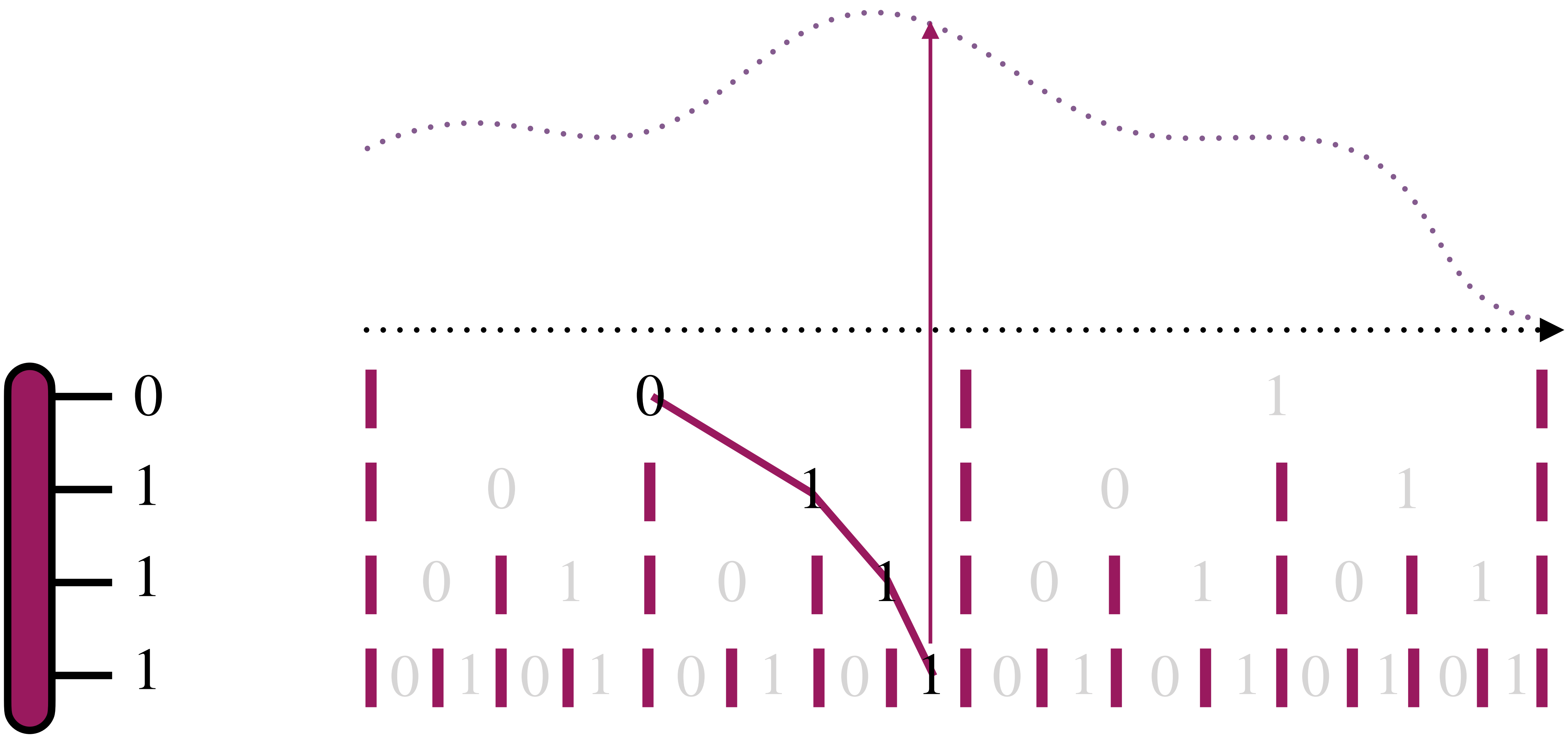
It is a **hierarchical** representation of data



Tensor Network Machine Learning

Continuum amplitude encoding

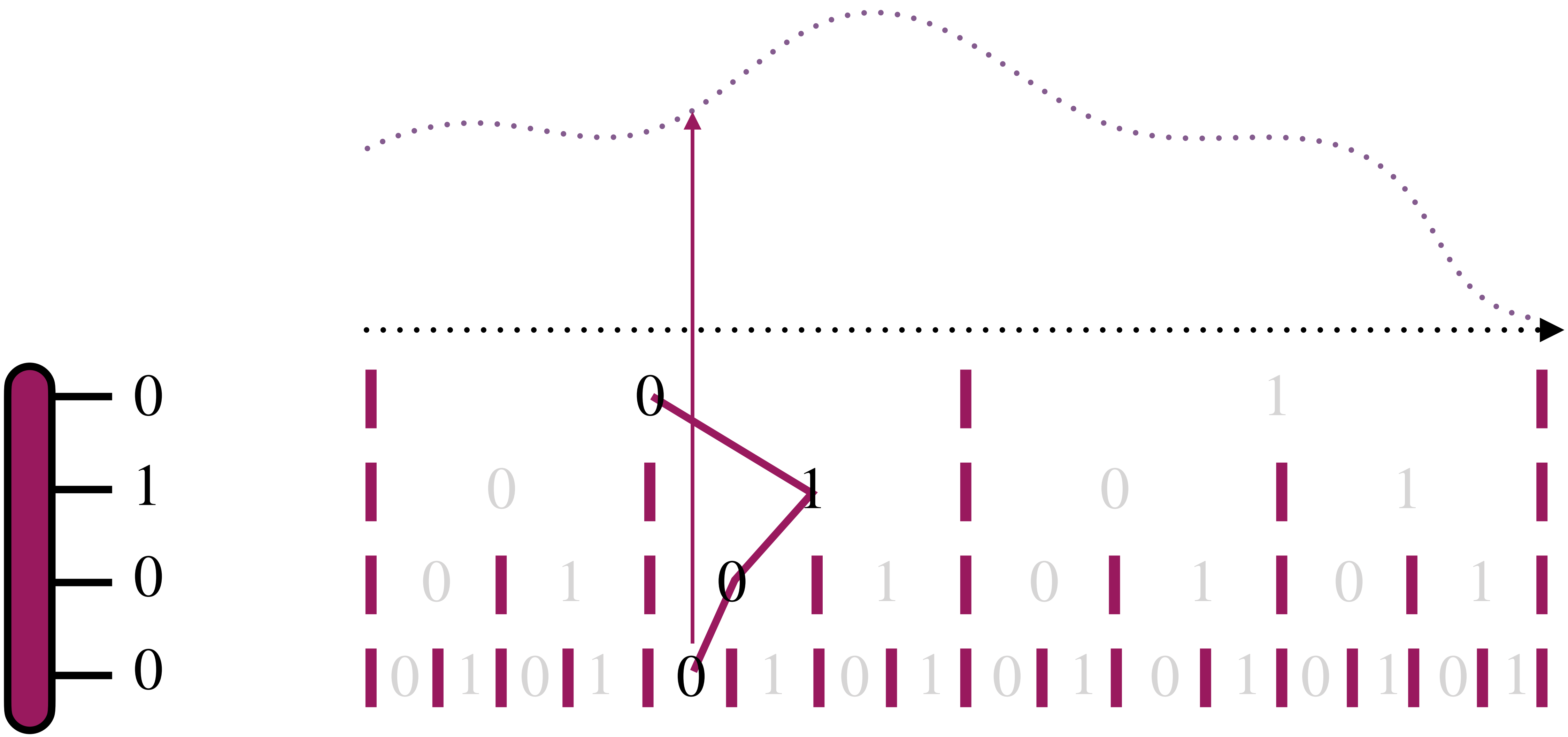
It is a **hierarchical** representation of data



Tensor Network Machine Learning

Continuum amplitude encoding

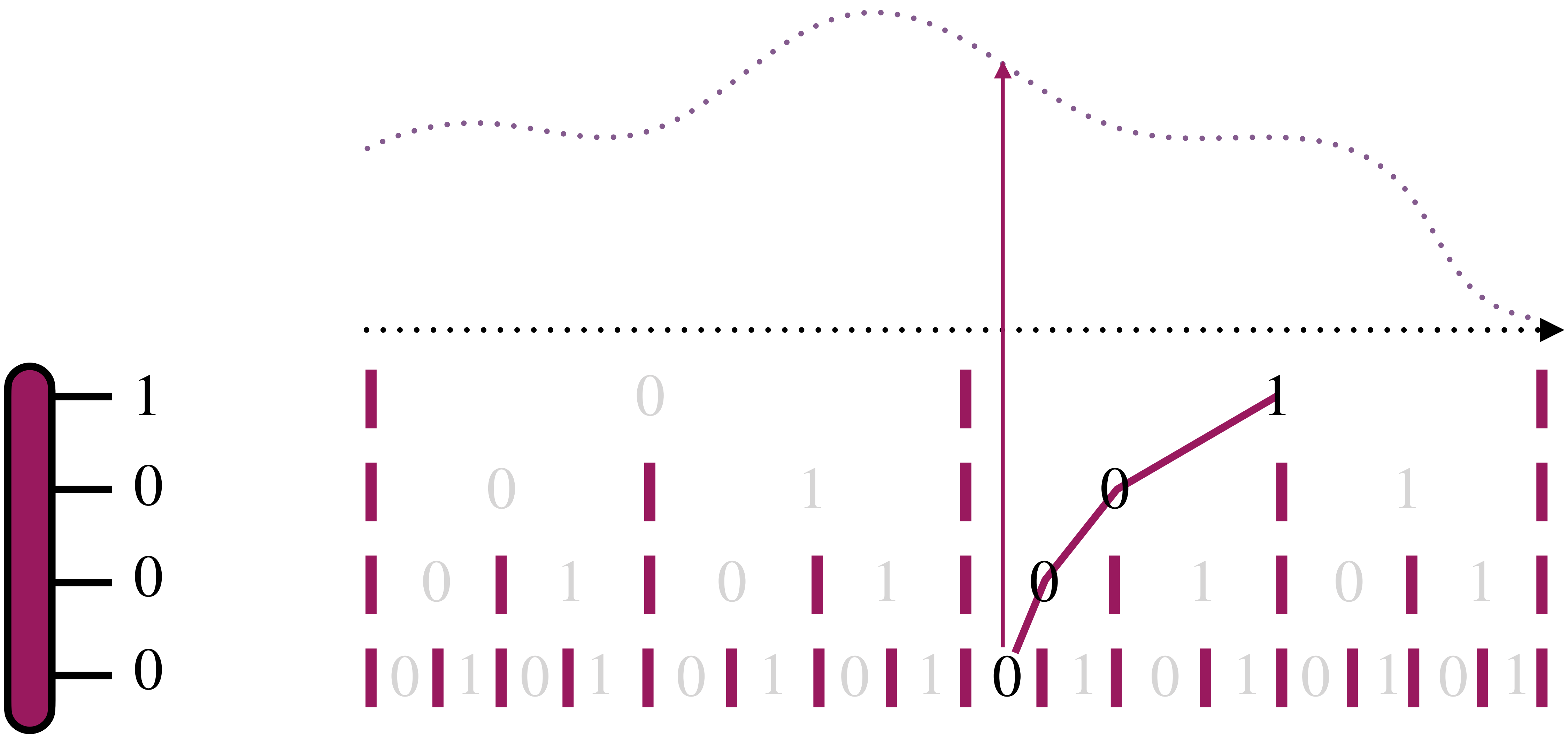
It is a **hierarchical** representation of data



Tensor Network Machine Learning

Continuum amplitude encoding

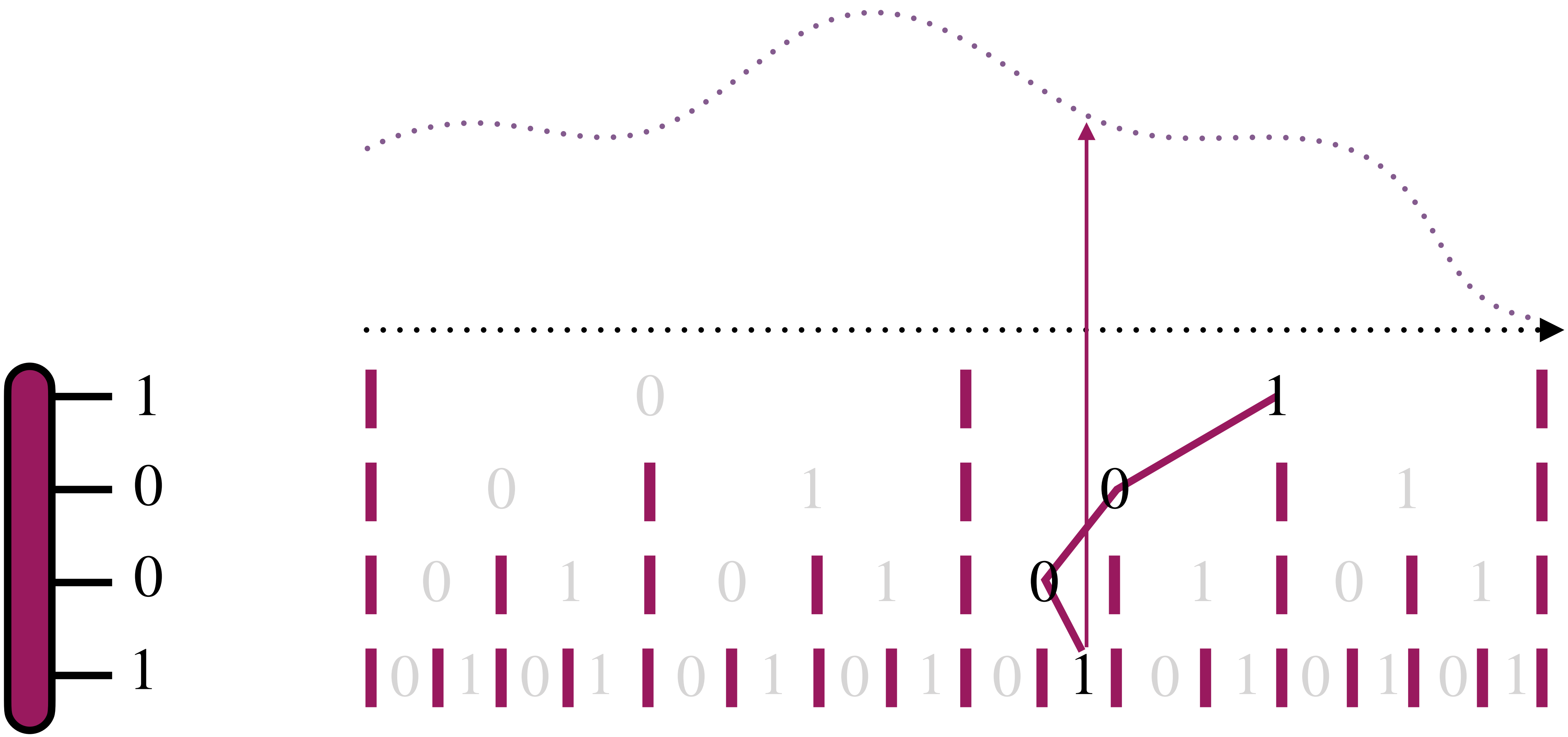
It is a **hierarchical** representation of data



Tensor Network Machine Learning

Continuum amplitude encoding

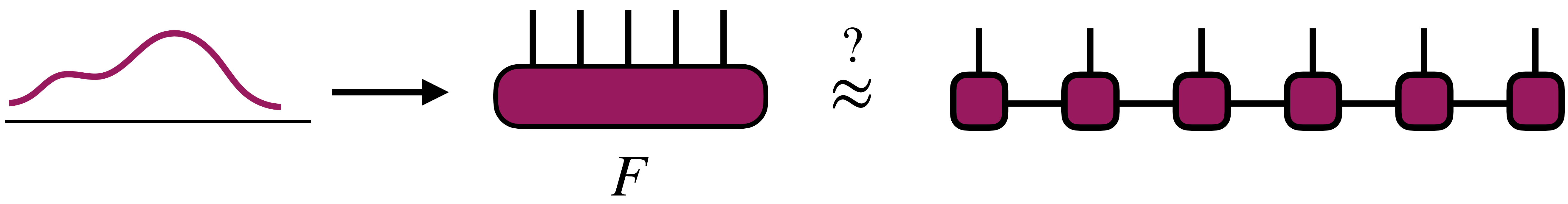
It is a **hierarchical** representation of data



Future Directions – Continuum Functions

Key question: for a given $f(x)$

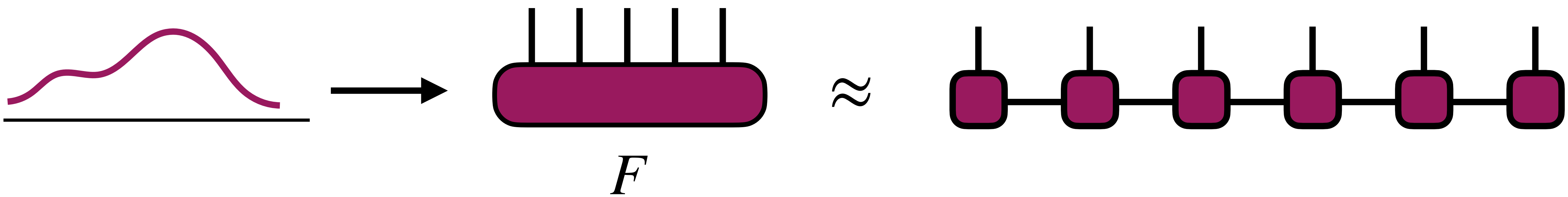
is F low-rank as a tensor network?



Future Directions – Continuum Functions

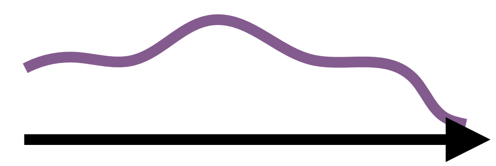
[1] Mazen Ali, Anthony Nouy, Constr Approx
[2] Mazen Ali, Anthony Nouy, arxiv:2101.11932
[3] Chen, EMS, White, PRX Quantum, arxiv:2210.08468

Low rank for many cases

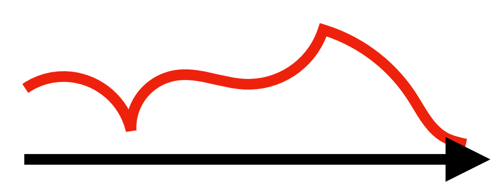


Tensor network low rank for:

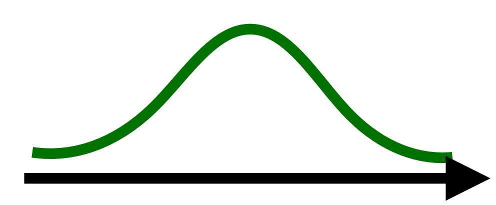
- all smooth enough functions [1,2]



- functions with finite number of cusps or discontinuities [1,2]



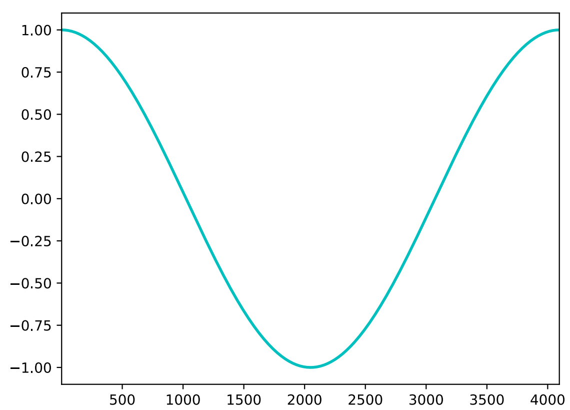
- any Fourier transform of these [3]

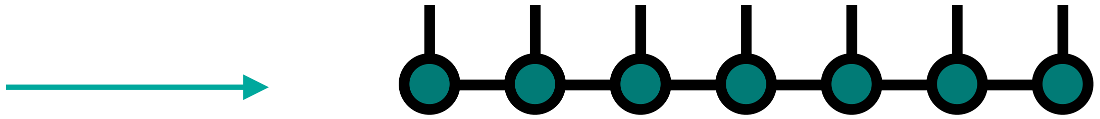


Future Directions – Continuum Functions

Examples:

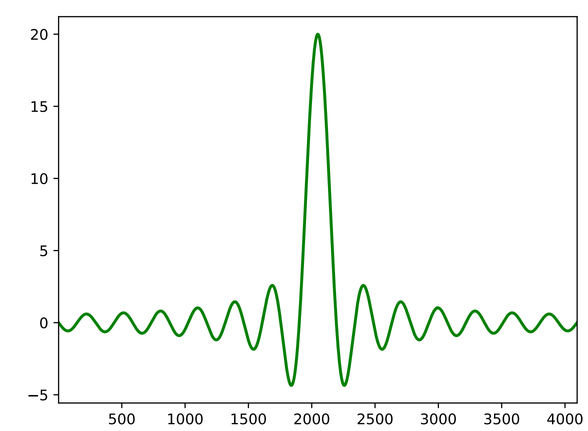
$$\cos\left[x - \frac{1}{2}\right]$$

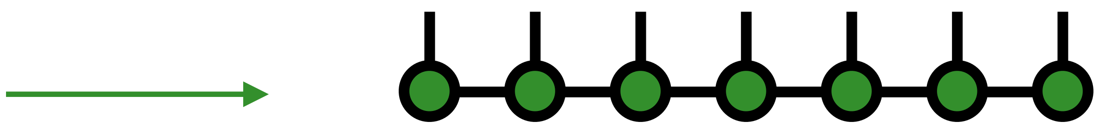




$\chi = 2$

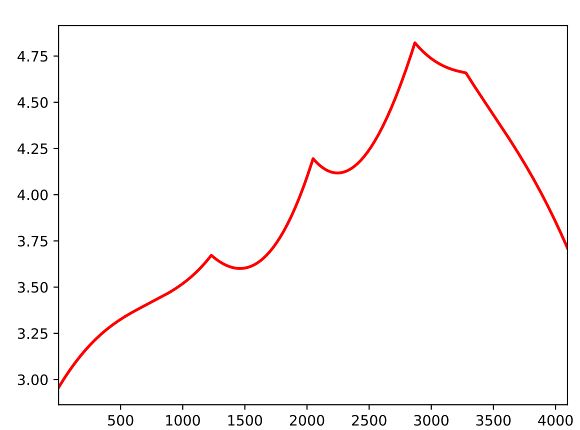
sum of 20 cosines

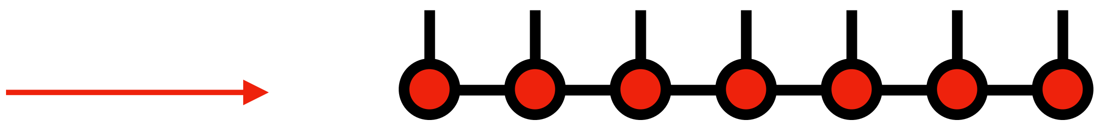




$\chi = 18$

cosine + cusps



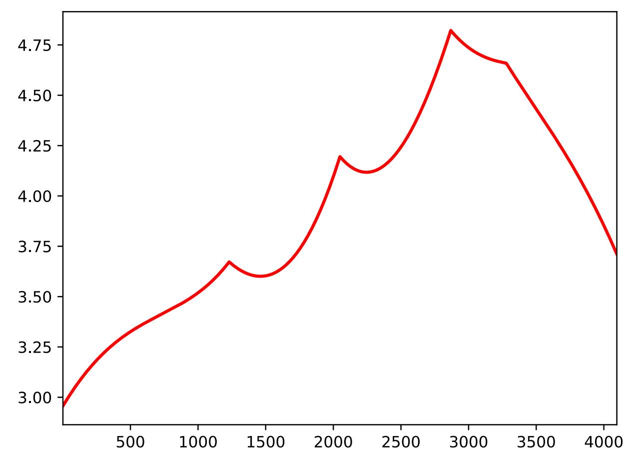


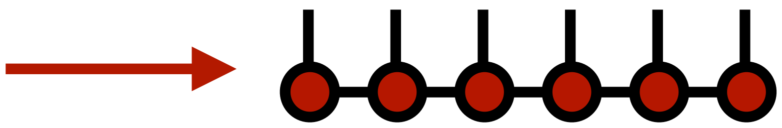
$\chi = 9$

Future Directions – Continuum Functions

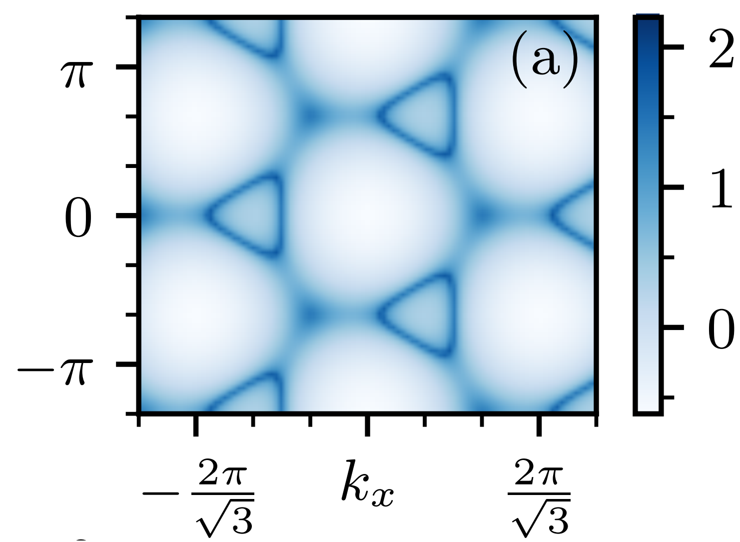
Works in 1D, 2D, …

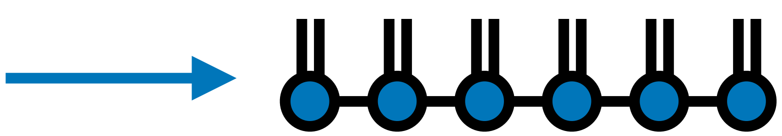
1D





2D





Future Directions – Continuum Functions

Continuum amplitude encoding

Payoff: can machine learn continuous functions

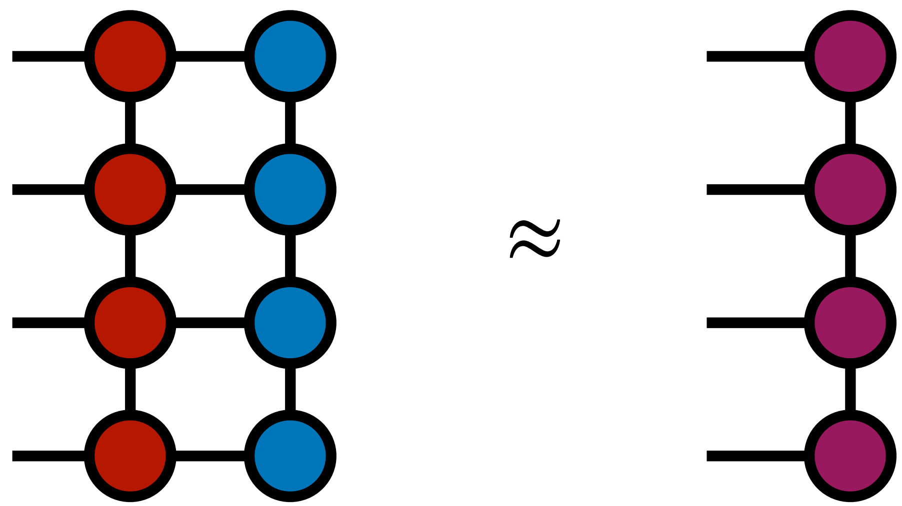
See https://tensornetwork.org/functions
for more details and key references

Future Direction #2:
Tensor Train Recursive Sketching Algorithm

Future Directions – Recursive Sketching Algorithm

Talked a lot about representations

But real power is in algorithms

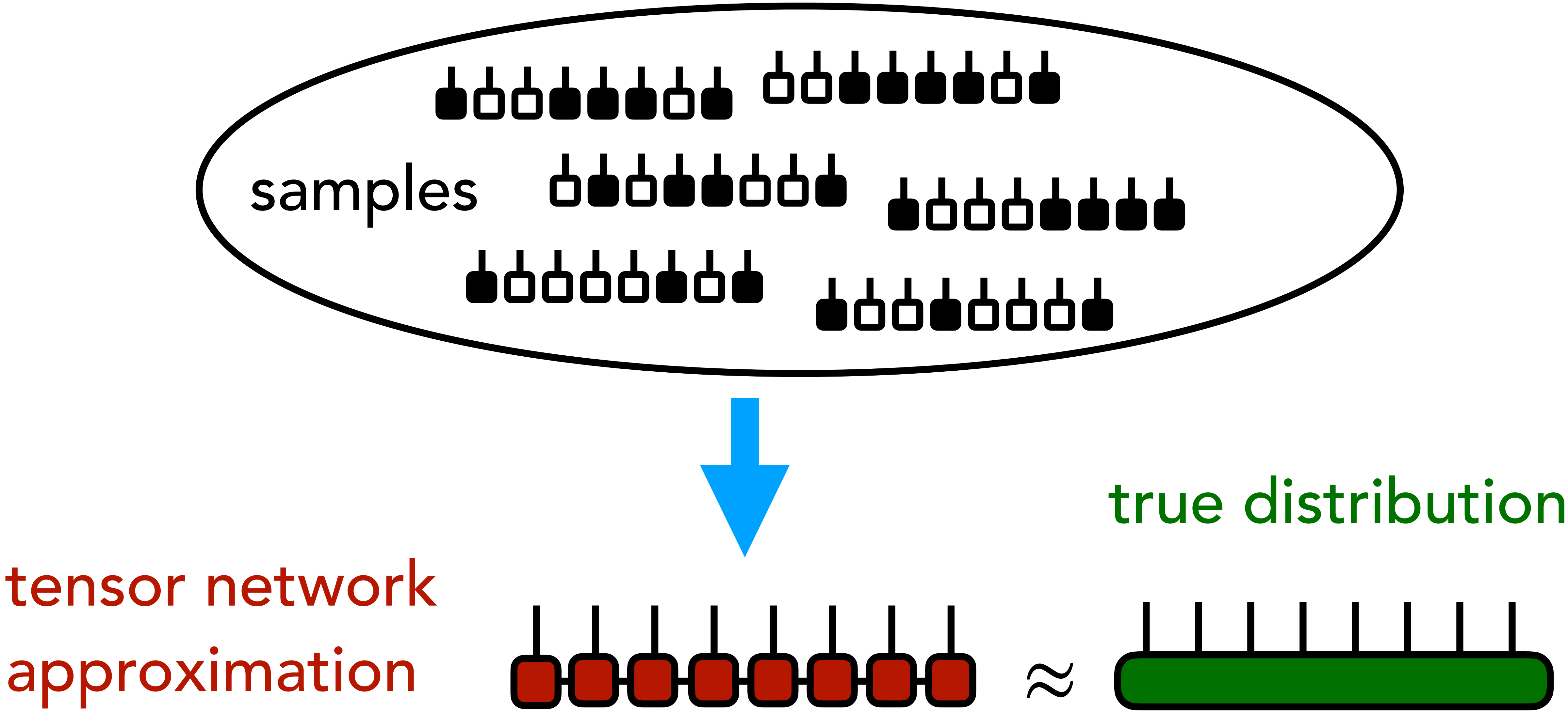


Tensor networks = high dimensional linear algebra

Should have **deterministic** algorithms (like QR, SVD, ...)

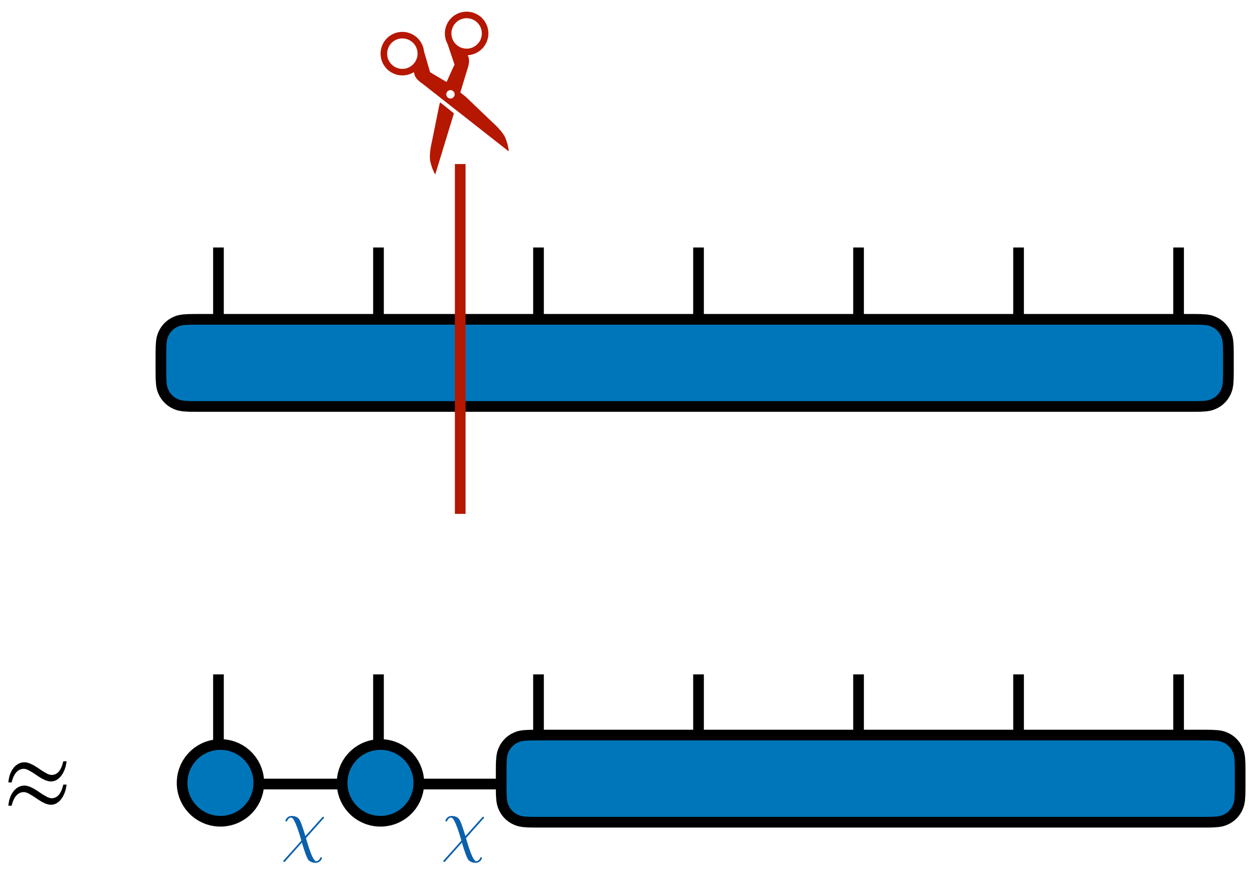
Future Directions – Recursive Sketching Algorithm

The "**tensor train recursive sketching**" (TTRS) algorithm estimates a probability distribution from samples



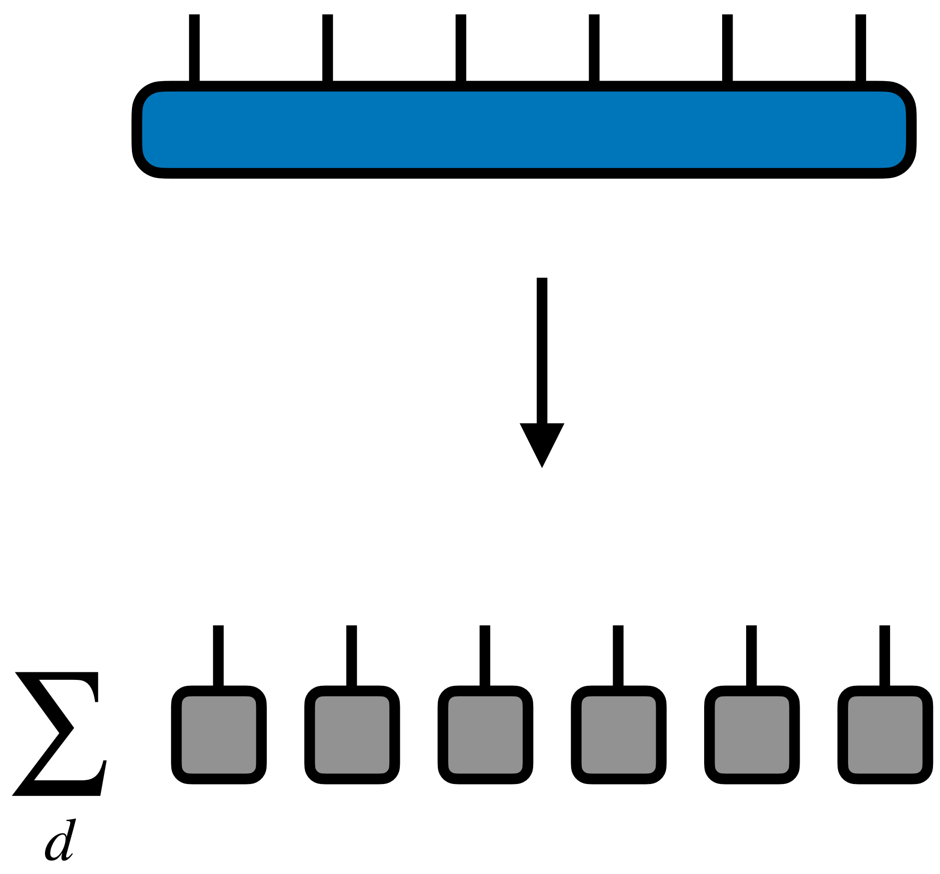
Future Directions – Recursive Sketching Algorithm

Very similar to "recursive SVD" for making MPS



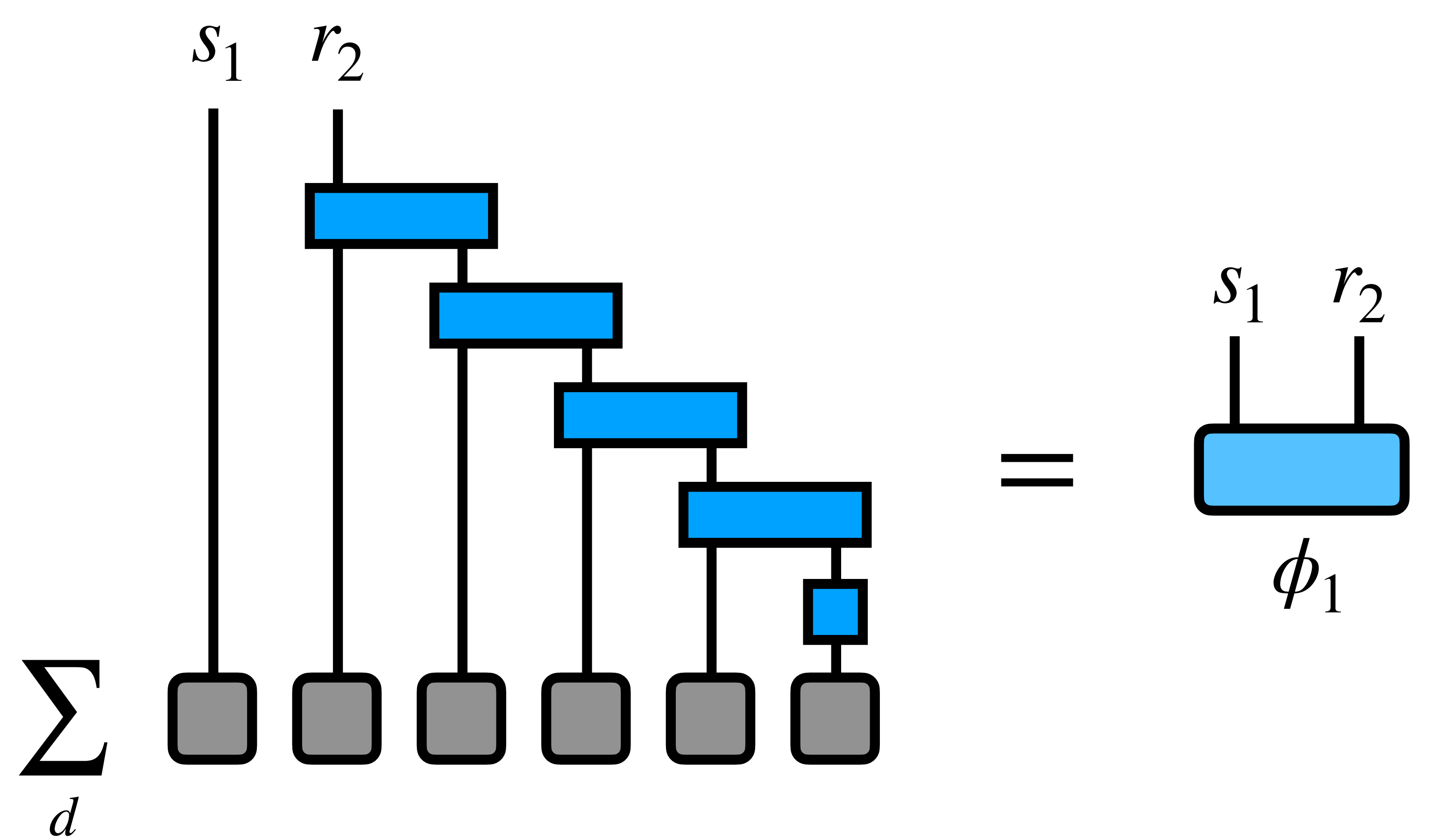
Future Directions – Recursive Sketching Algorithm

But replace tensor with
sum over training data



Future Directions – Recursive Sketching Algorithm

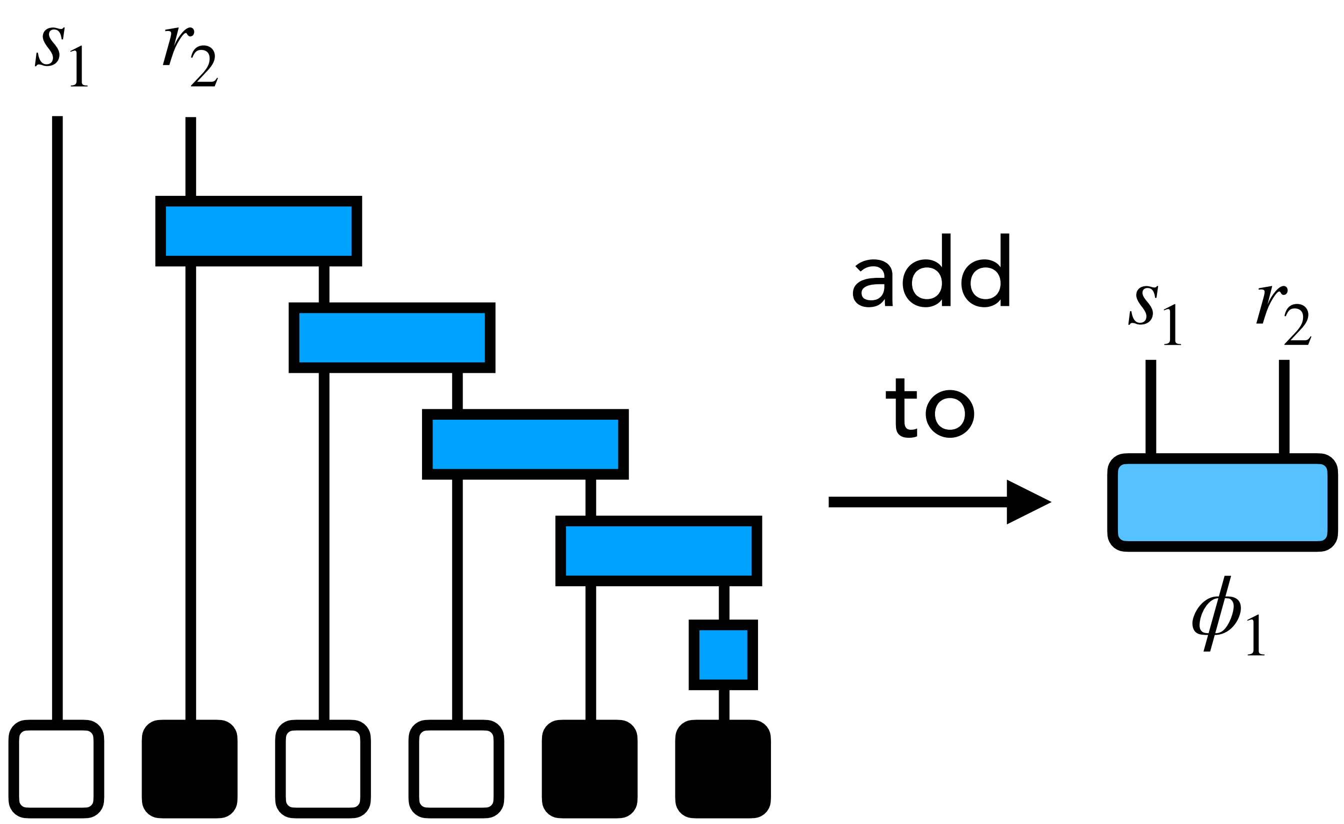
And apply "sketch" tensors to right-hand side



when performing sum, to "broaden" data

Future Directions – Recursive Sketching Algorithm

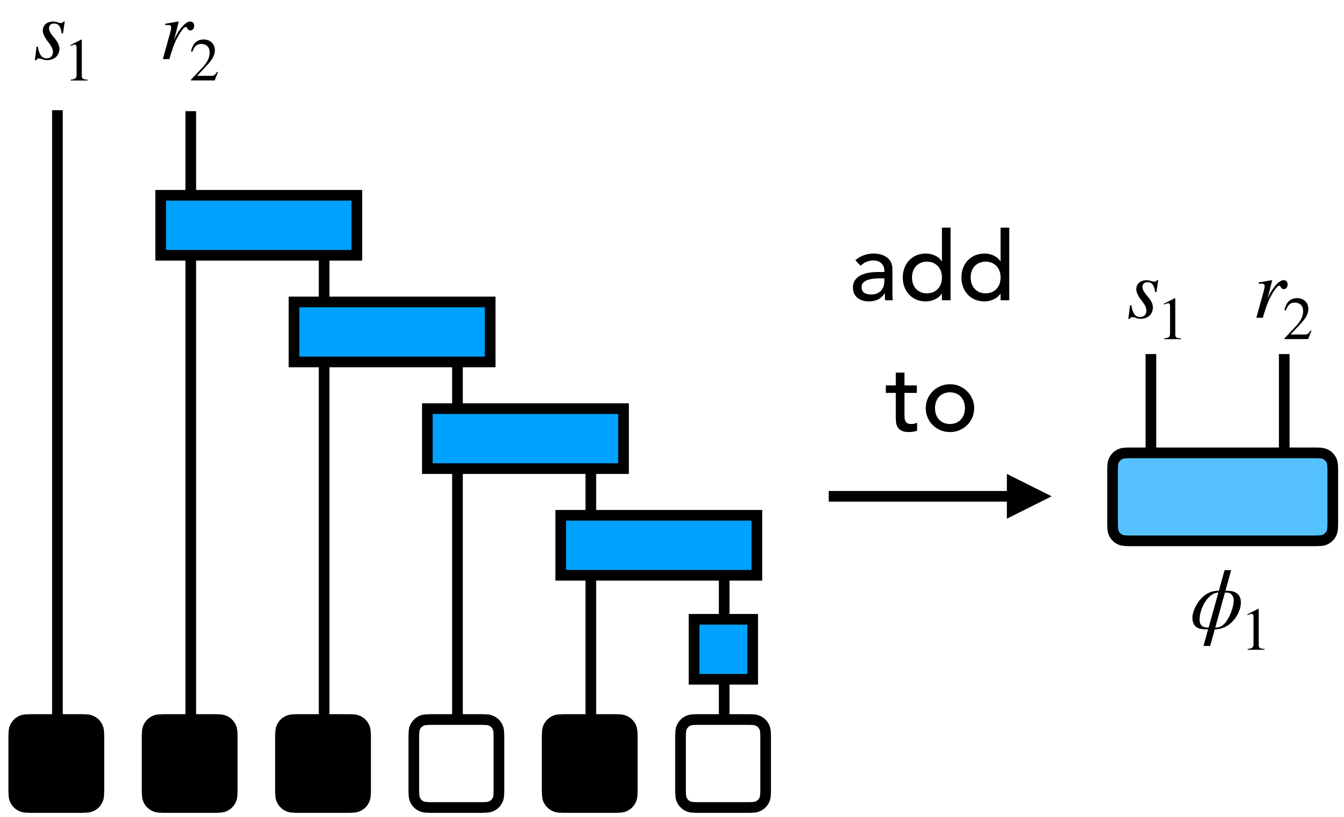
And apply "sketch" tensors to right-hand side



when performing sum, to "broaden" data

Future Directions – Recursive Sketching Algorithm

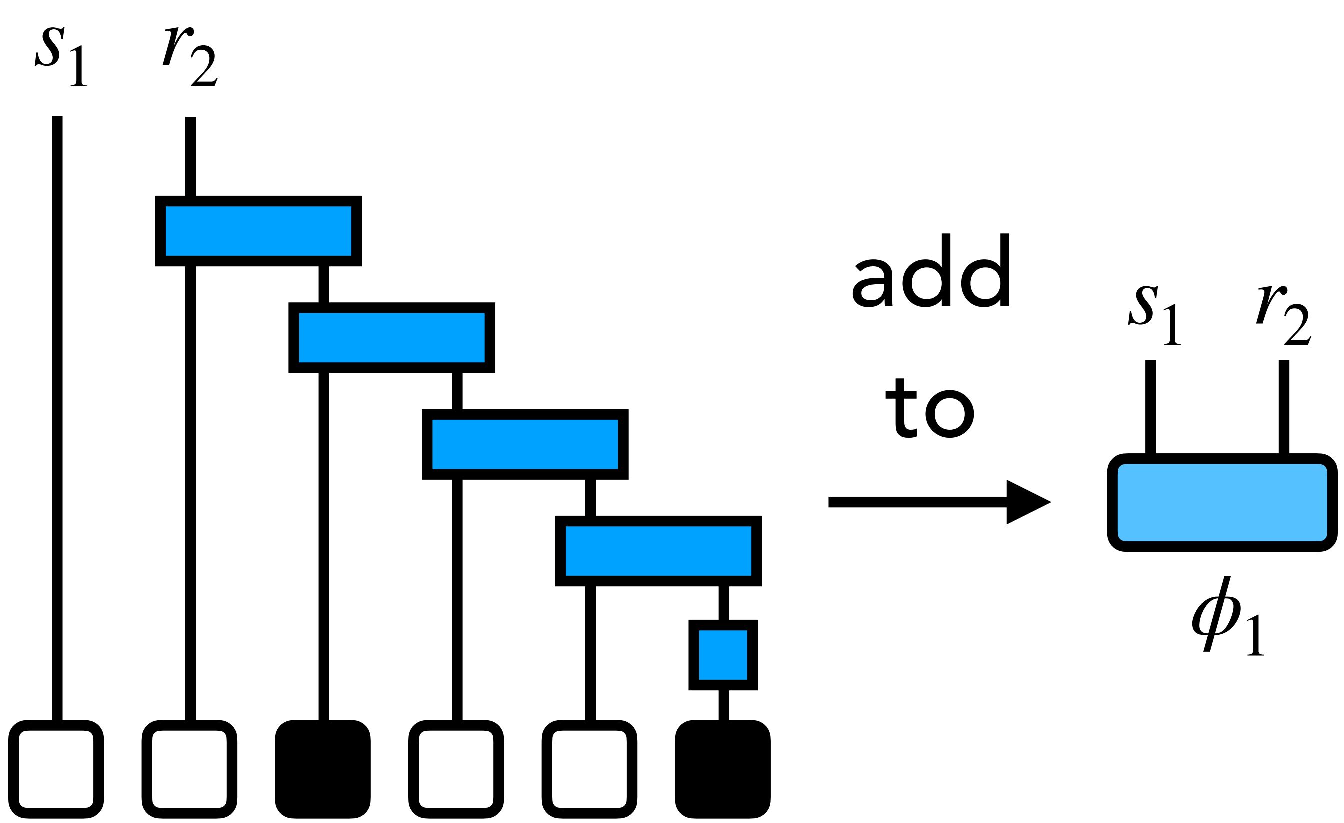
And apply "sketch" tensors to right-hand side



when performing sum, to "broaden" data

Future Directions – Recursive Sketching Algorithm

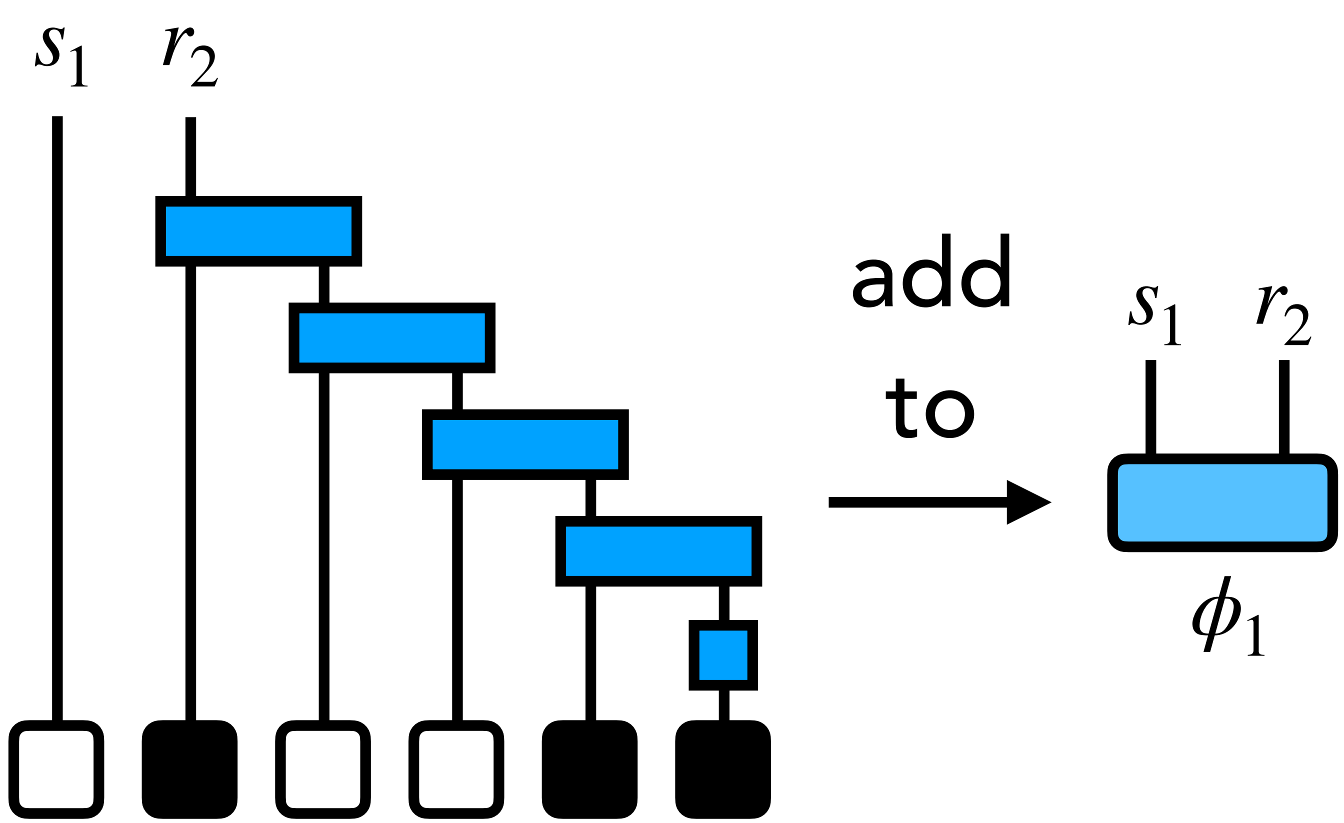
And apply "sketch" tensors to right-hand side



when performing sum, to "broaden" data

Future Directions – Recursive Sketching Algorithm

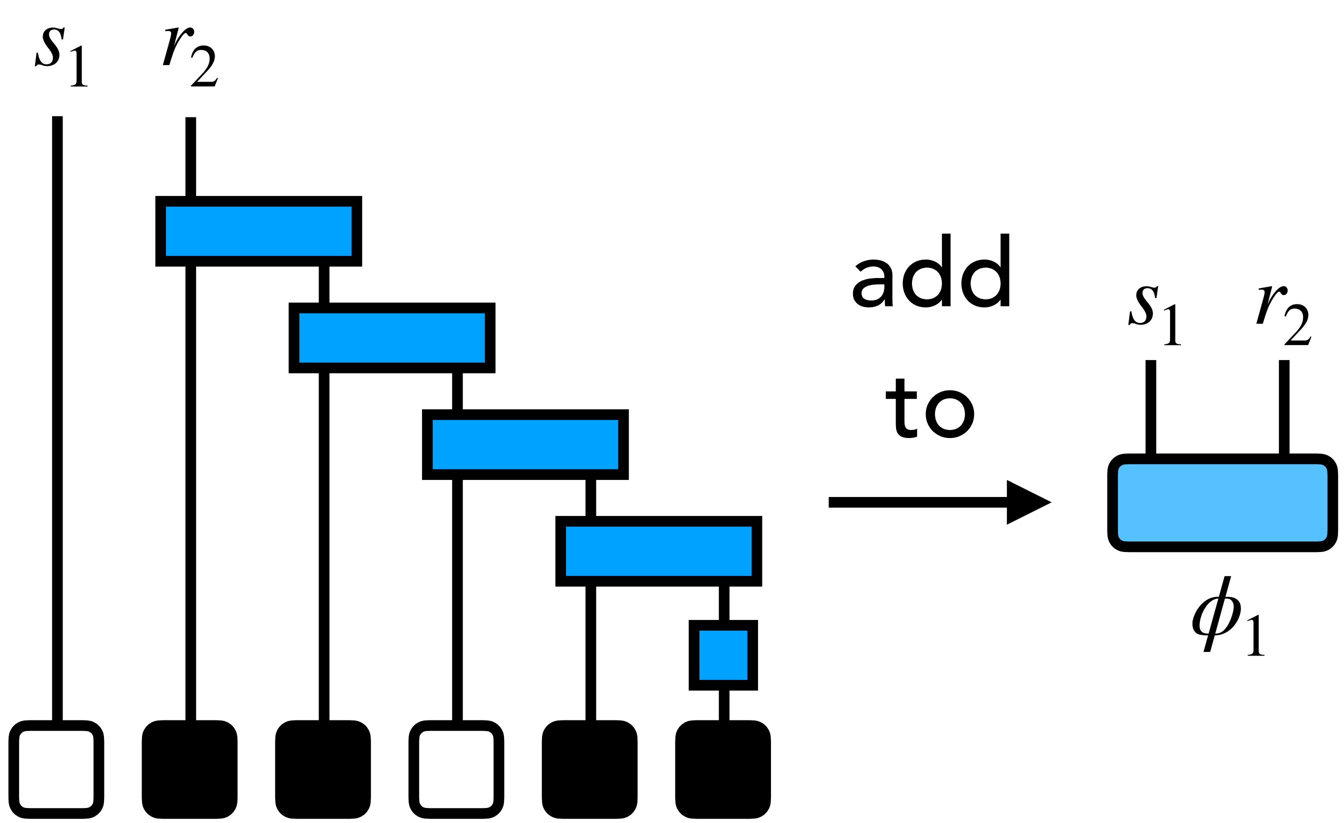
And apply "sketch" tensors to right-hand side



when performing sum, to "broaden" data

Future Directions – Recursive Sketching Algorithm

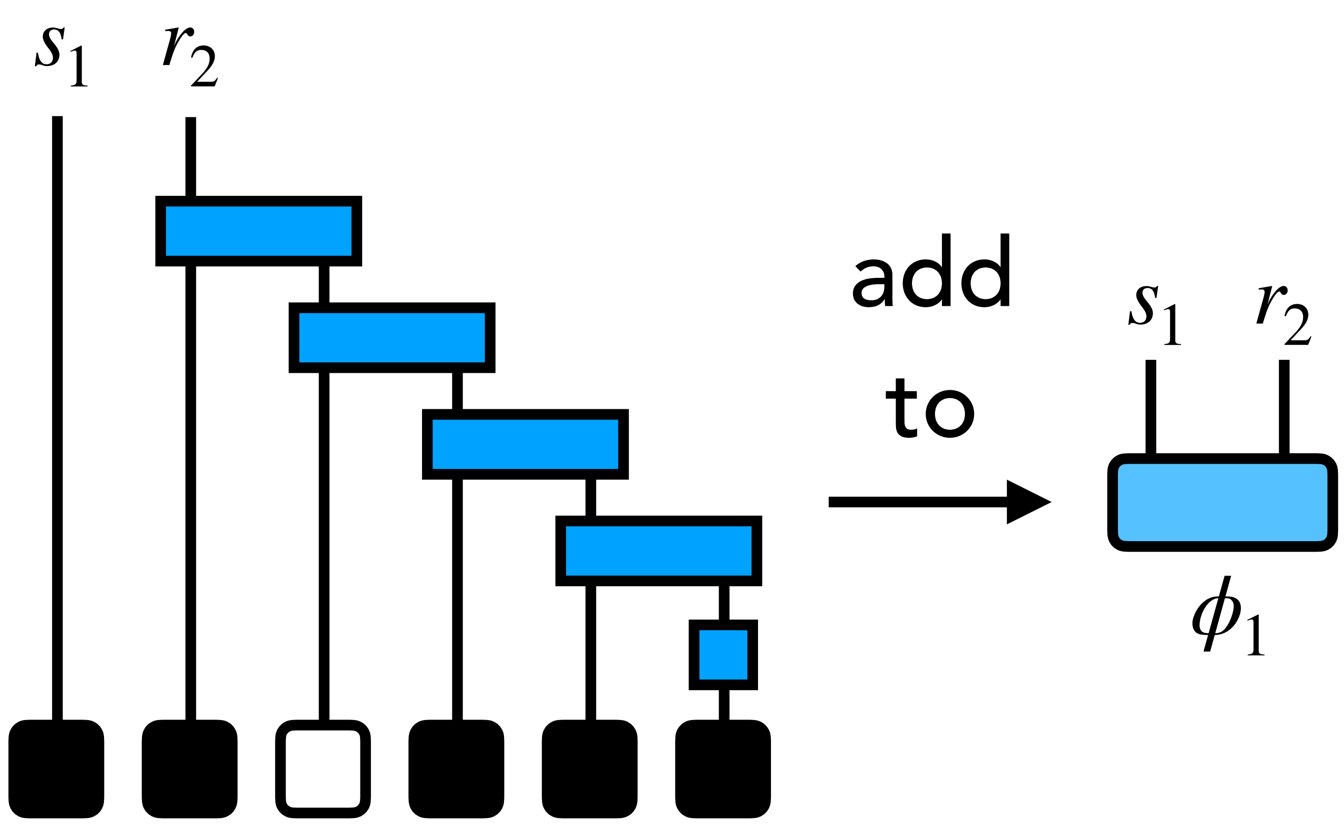
And apply "sketch" tensors to right-hand side



when performing sum, to "broaden" data

Future Directions – Recursive Sketching Algorithm

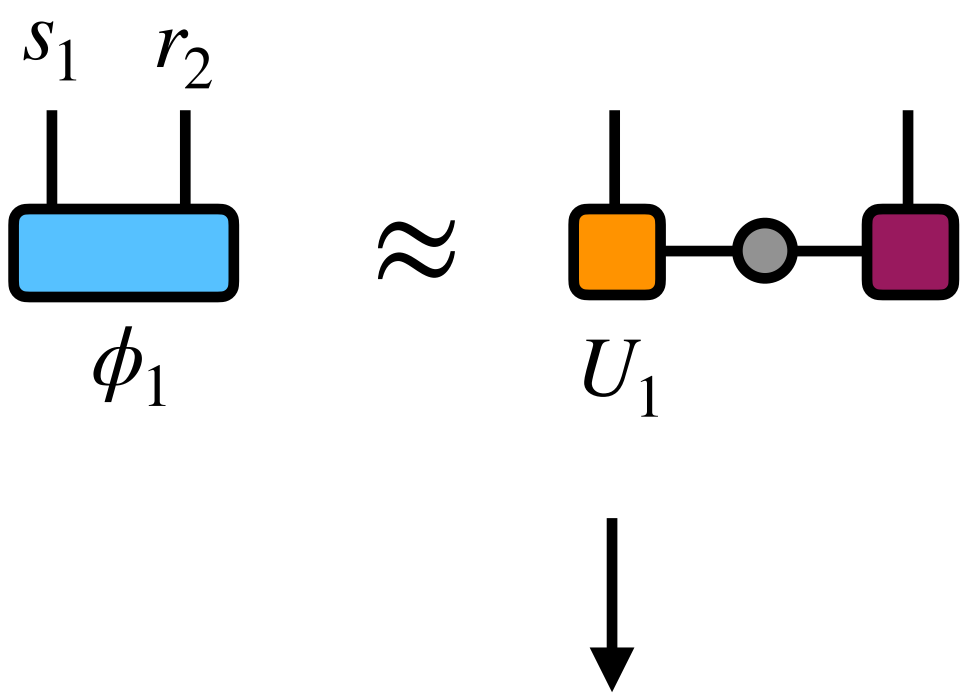
And apply "sketch" tensors to right-hand side



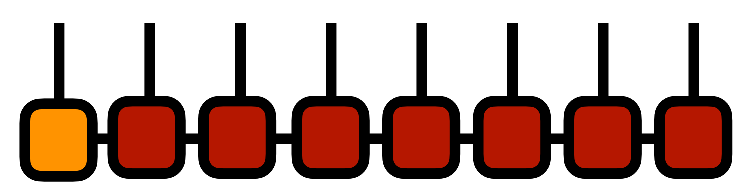
when performing sum, to "broaden" data

Future Directions – Recursive Sketching Algorithm

SVD of ϕ_1 recovers first MPS tensor



First tensor of distribution want to learn:

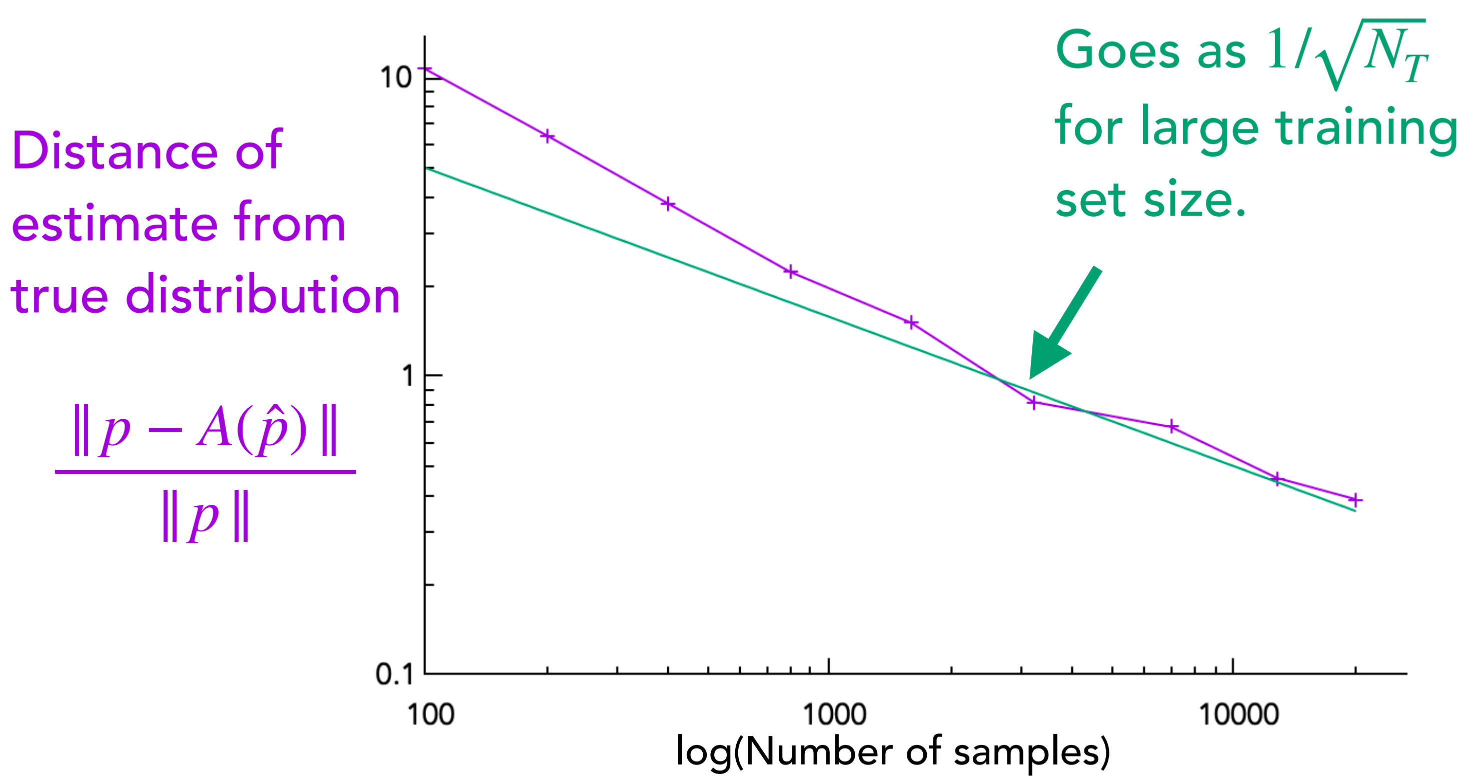


Then repeat recursively…

Future Directions – Recursive Sketching Algorithm

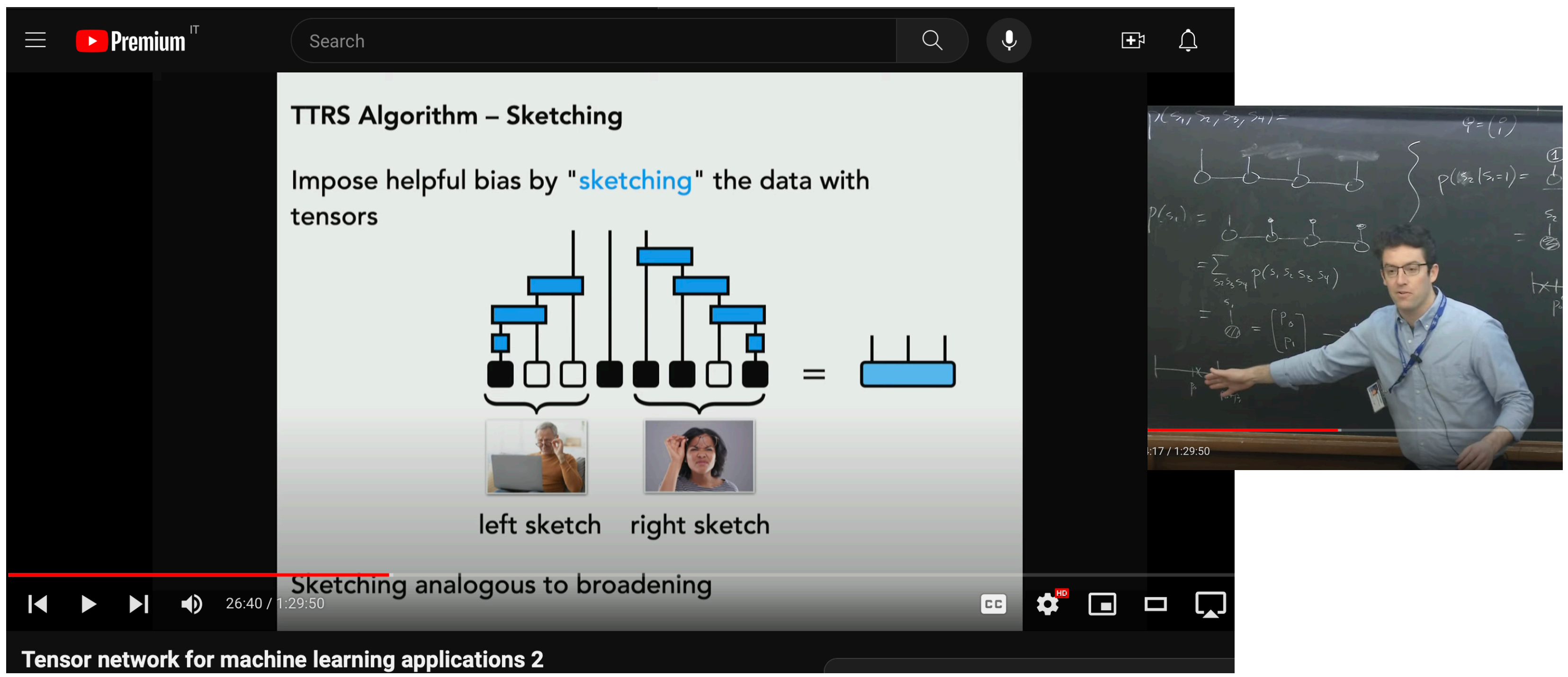
TTRS can accurately reconstruct probability distribution of ***disordered*** Ising chain

Provably optimal sketches known



Future Directions – Recursive Sketching Algorithm

For more details of TTRS algorithm,
see following lecture and slides



[YouTube Link](https://youtu.be/Qbnek0yjZrg) (https://youtu.be/Qbnek0yjZrg)

[Slides Link](https://itensor.org/miles/Trieste02TTRS.pdf) (https://itensor.org/miles/Trieste02TTRS.pdf)

References:
Generative Modeling via Tensor Train Sketching, arxiv: 2202.11788
Generative Modeling via Hierarchical Tensor Sketching, arxiv: 2304.05305

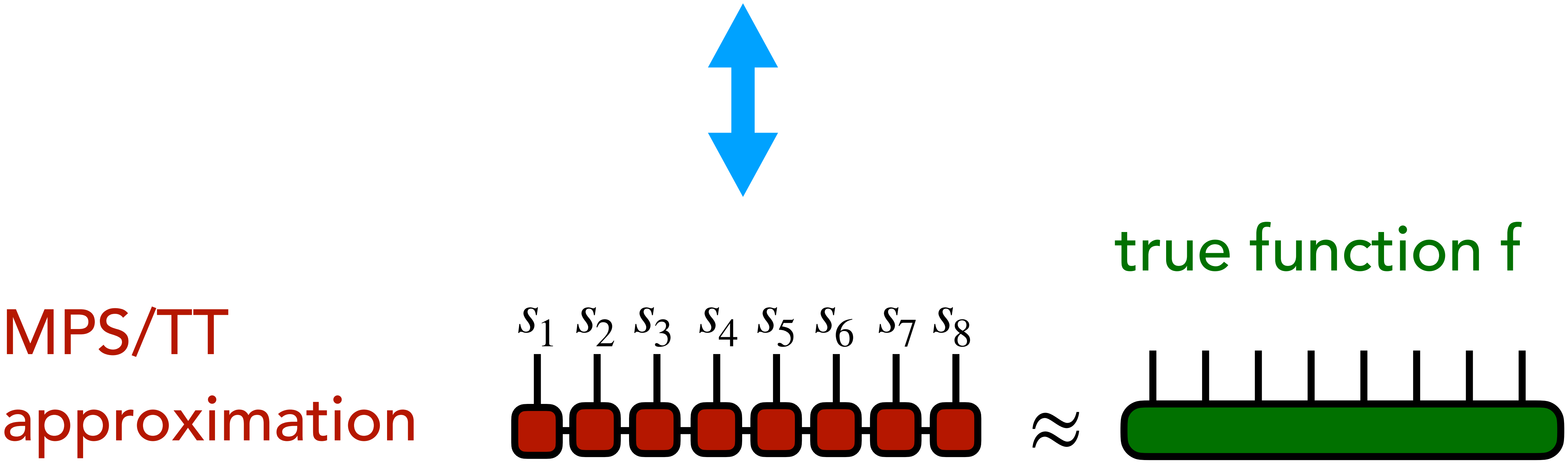
Future Direction #3:
Tensor Cross Interpolation Algorithm

Future Directions – Tensor Cross Interpolation

The "**tensor cross interpolation**" (TCI) algorithm learns a function from calls to that function

"black box" function

$$f(s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8)$$



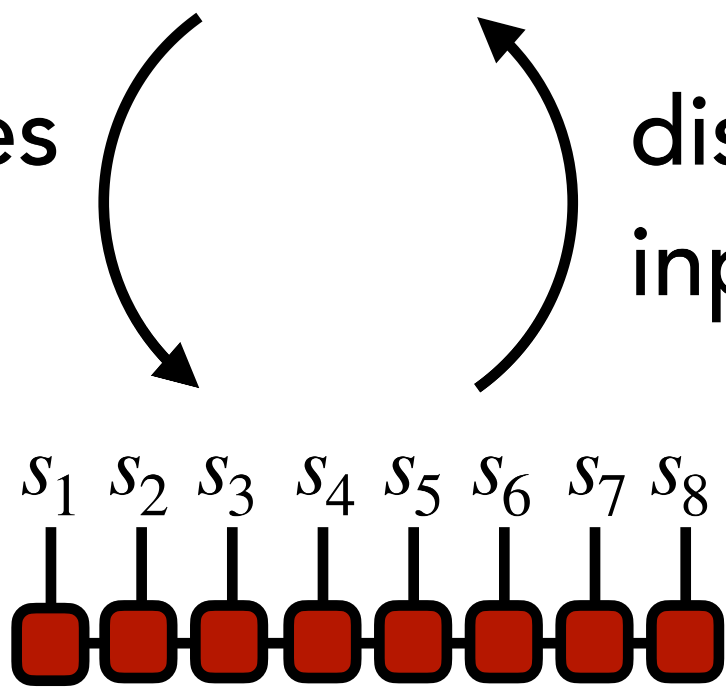
Future Directions – Tensor Cross Interpolation

It is an "active learning" algorithm

$$f(s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8)$$

get values from f

discover important inputs to f



Future Directions – Tensor Cross Interpolation

Invented and refined in following papers:

- I. Oseledets and E. Tyrtyshnikov, *TT-Cross Approximation for Multidimensional Arrays*, Linear Algebra Appl. 432, 70 (2010).
- D. V. Savostyanov, *Quasioptimality of Maximum- Volume Cross Interpolation of Tensors*, Linear Algebra Appl. 458, 217 (2014)
- S. Dolgov and D. Savostyanov, *Parallel Cross Interpolation* for High-Precision Calculation of High-Dimensional *Integ*rals, Comput. Phys. Commun. 246, 106869 (2020)
- Núnẽz Fernández, et al., *Learning Feynman Diagrams with Tensor Trains*, PRX 12, 041018 (2022)


Ivan Oseledets

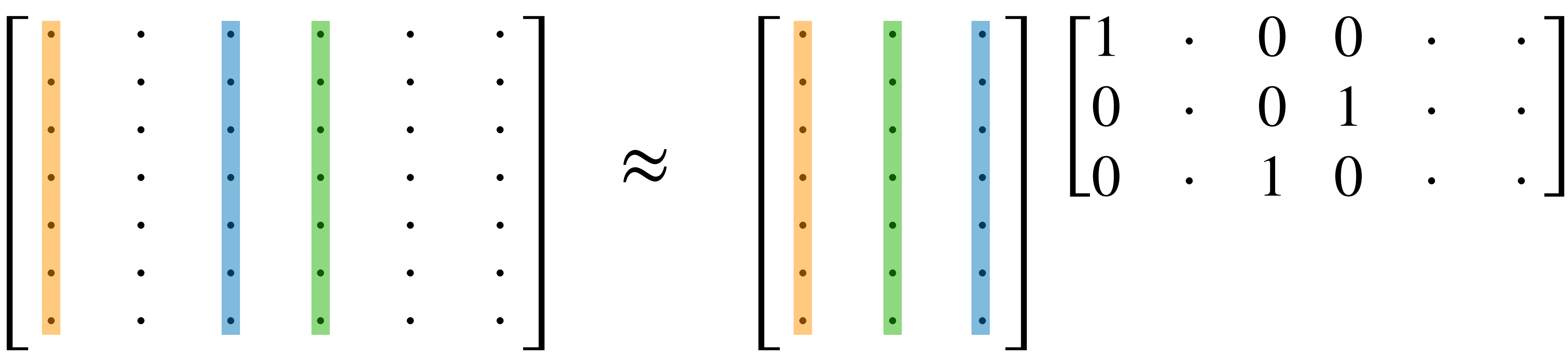

Dmitry Savostyanov


Yuriel Núnẽz Fernández

Future Directions – Tensor Cross Interpolation

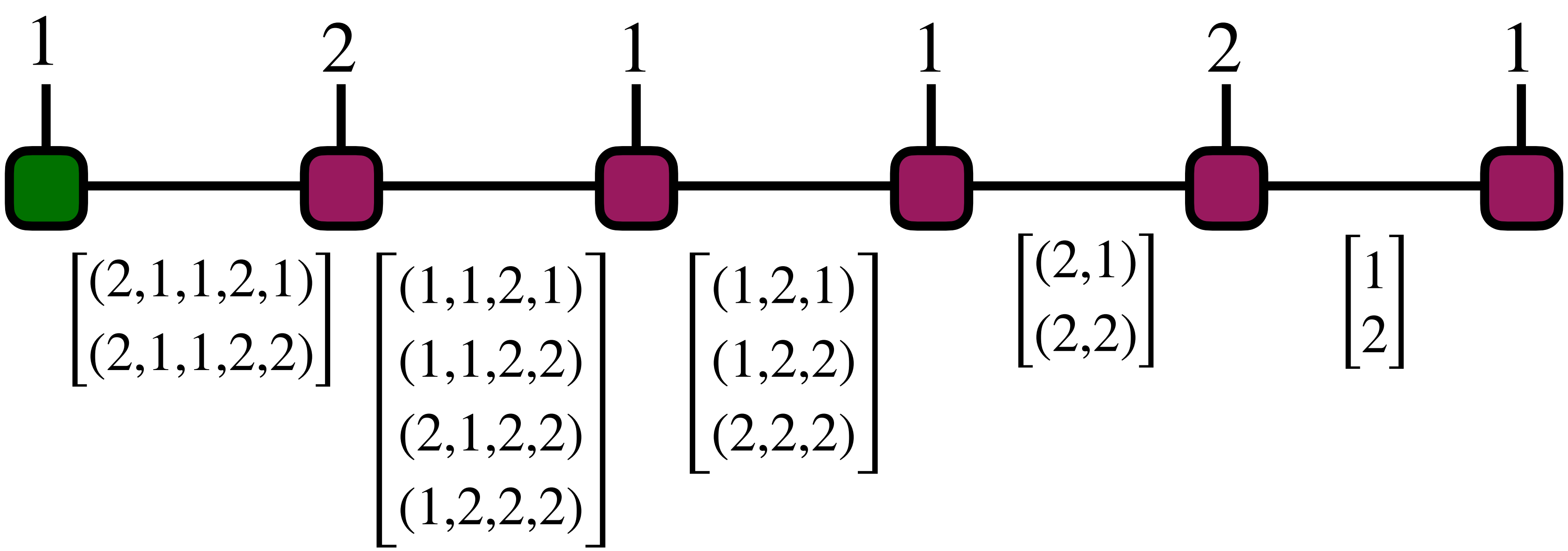
Lifts the idea of the "interpolative decomposition" of a matrix to tensor networks

Given some matrix M, can reconstruct from a subset of columns

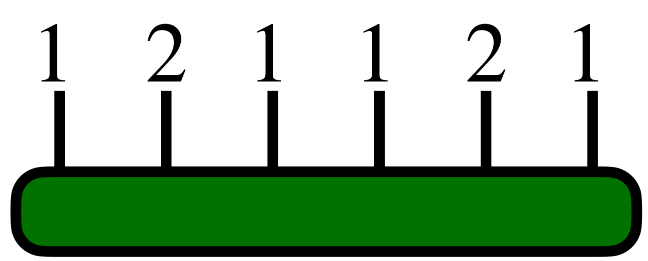


Future Directions – Tensor Cross Interpolation

Leads to new MPS "gauge" with interpolation property



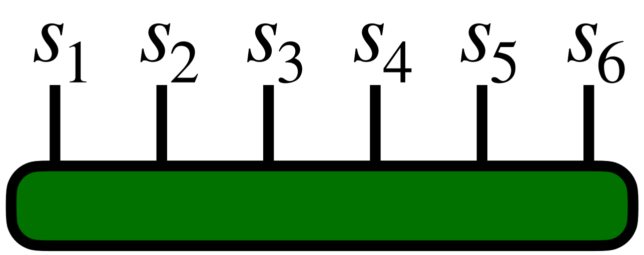
=



exact on 1,2,1,1,2,1 entry

Future Directions – Tensor Cross Interpolation

Instead of viewing original tensor as an array, think of it as a **callable function**



$$= f(s_1, s_2, s_3, s_4, s_5, s_6) =$$

```
function f(s1,s2,s3,s4,s5,s6)
    return 3*(s1+s2)+exp(s3*s4+s5+s6/7)
end
```

Can just be a piece of code

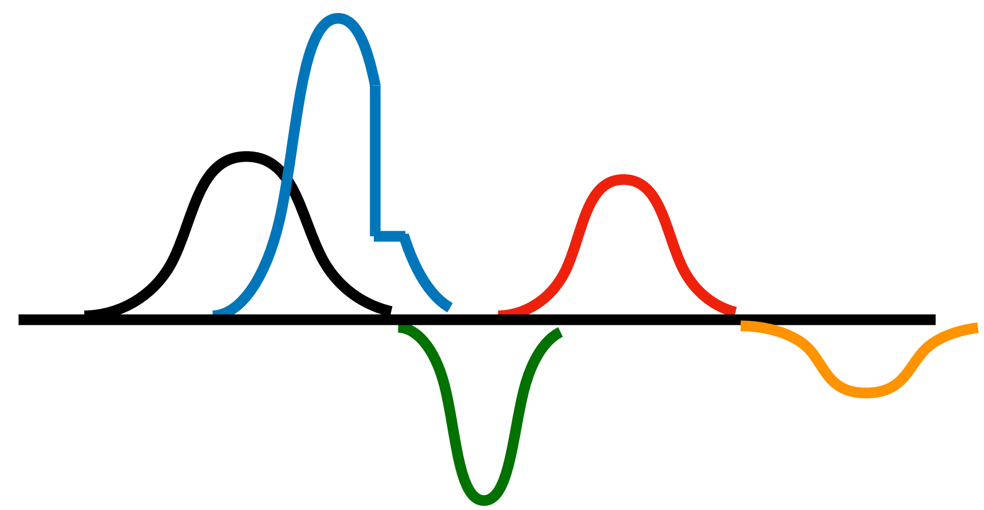
Must be able to ask function for any element the TCI algorithm wants

Future Directions – Tensor Cross Interpolation

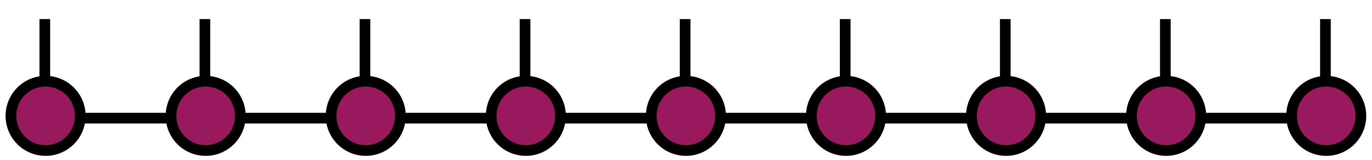
We already saw yesterday the demo of learning

40 Gaussians, random location, width, & height
+ a sharp step at 0.4

$$f(x) = \sum_{g=1}^{N_g} a_g e^{-w_g (x - x_g)^2}$$
$$+ 0.4 \cdot \Theta(x)$$



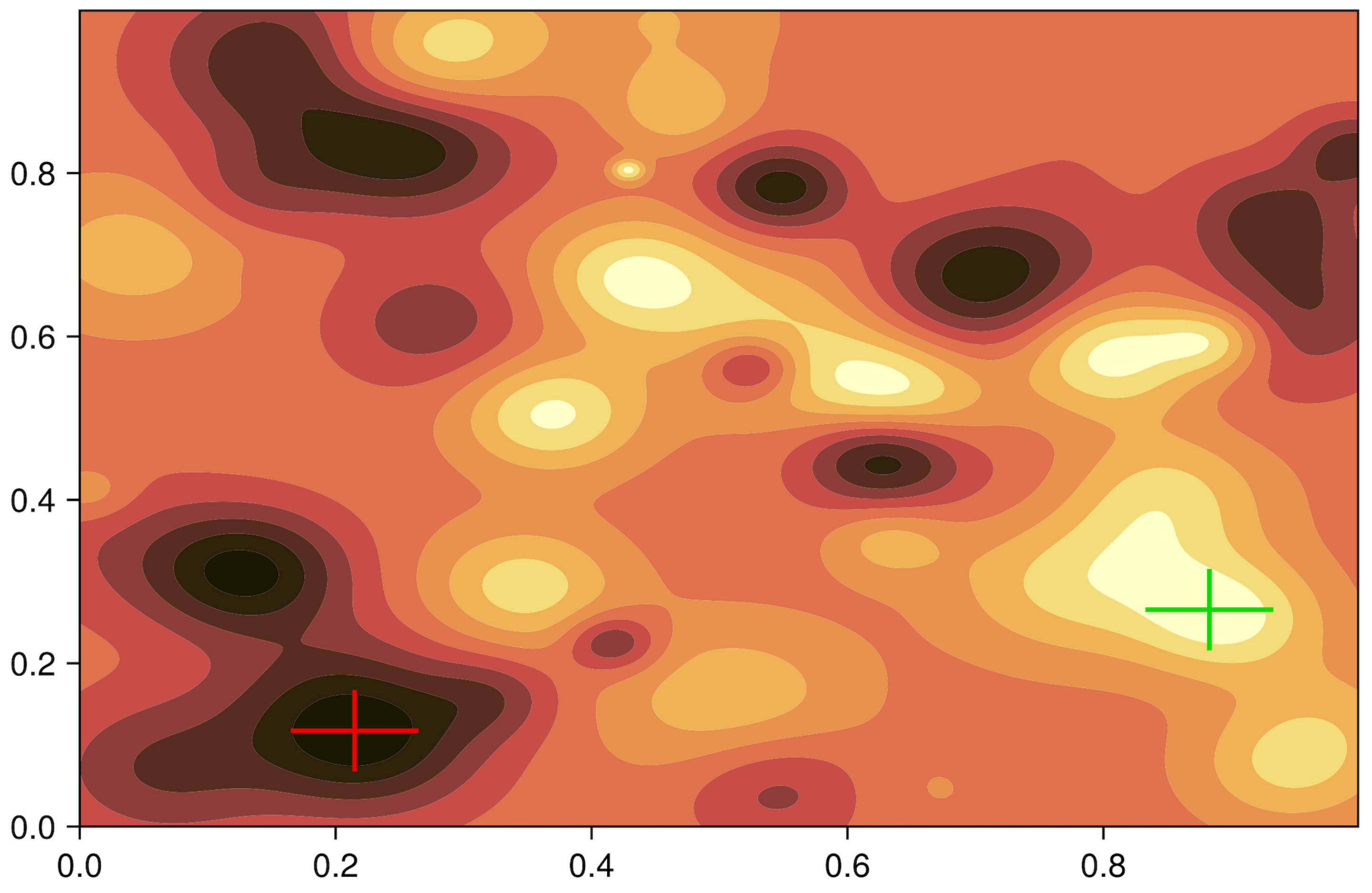
$$f(x) \approx$$



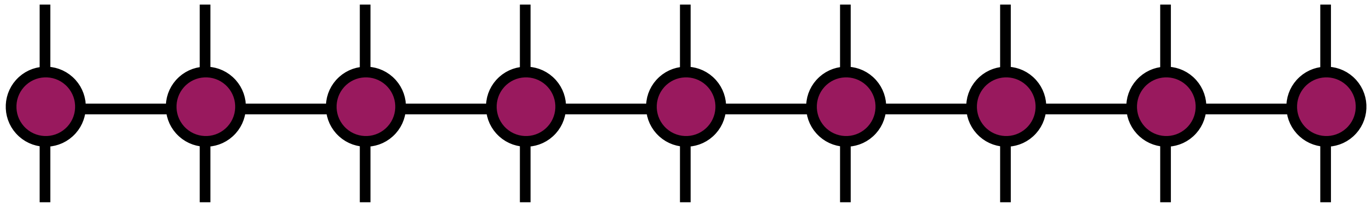
Future Directions – Tensor Cross Interpolation

Next demo: 2D function and optima

50 **2D** Gaussians, random location, width, & height



$f(x) \approx$



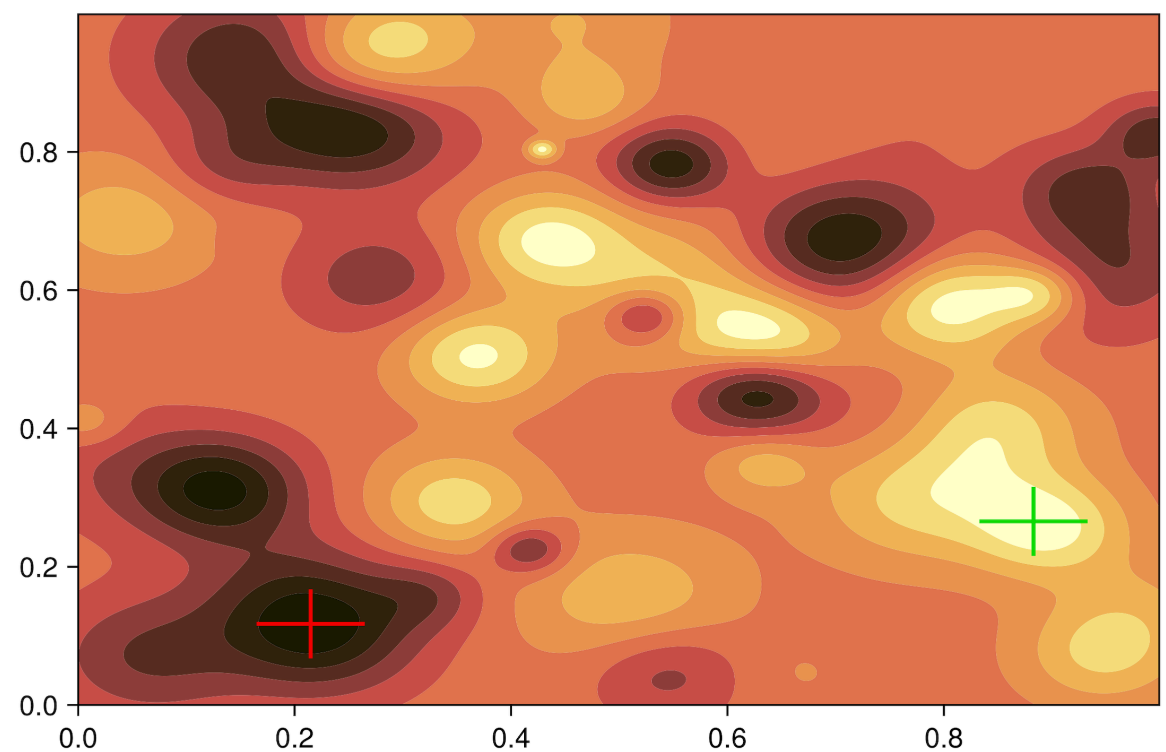
Future Directions – Tensor Cross Interpolation

How were optima (global min and max) found?

Very interesting proposal called "**TTOpt**" [1]

Claim: TCI usually "encounters" global min and max

Alternatively can use MPS perfect sampling strategies [2]

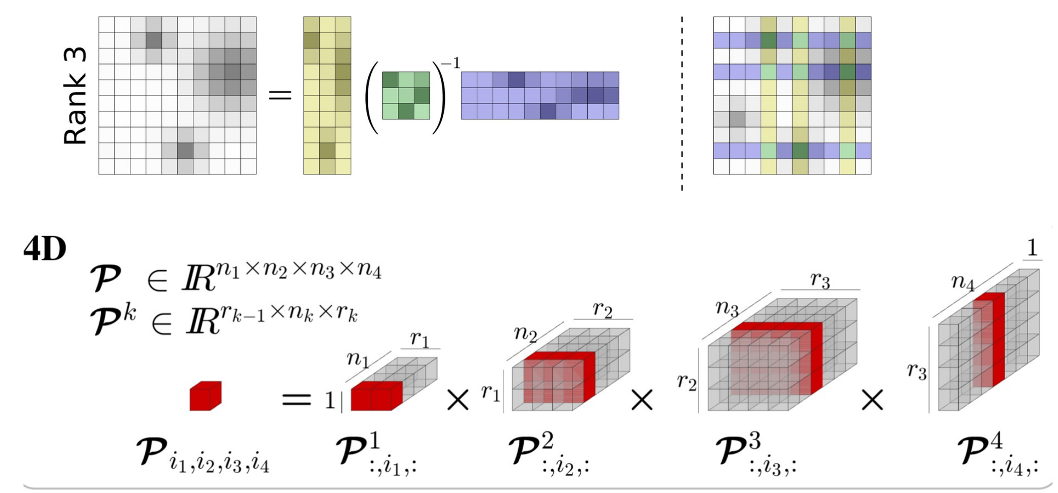


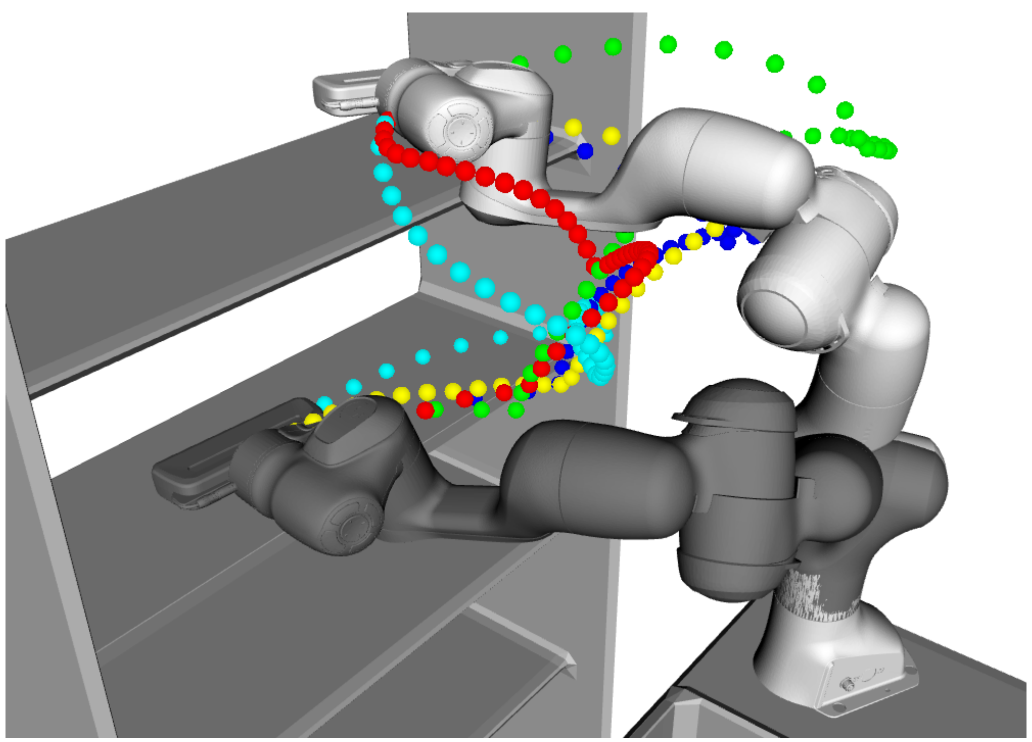
[1] Sozykin, Chertkov, Schutski, Phan, Cichocki, Oseledets, TTOpt, NeurIPS (2022)
[2] Chertkov, Ryzhakov, Novikov, Oseledets, arxiv:2209.14808

Future Directions – Tensor Cross Interpolation

Fun paper using TTOpt to control a real robot arm! *

TCI (= TT-Cross) algorithm





(a)

Figure 1. Solutions from TTGO for motion planning of a manipulator from a given initial configuration (white) to a final configuration (dark). The obtained joint angle trajectories result in different paths for the end.effector which are highlighted by dotted curves in different colors. The multimodality is clearly visible from these solutions.

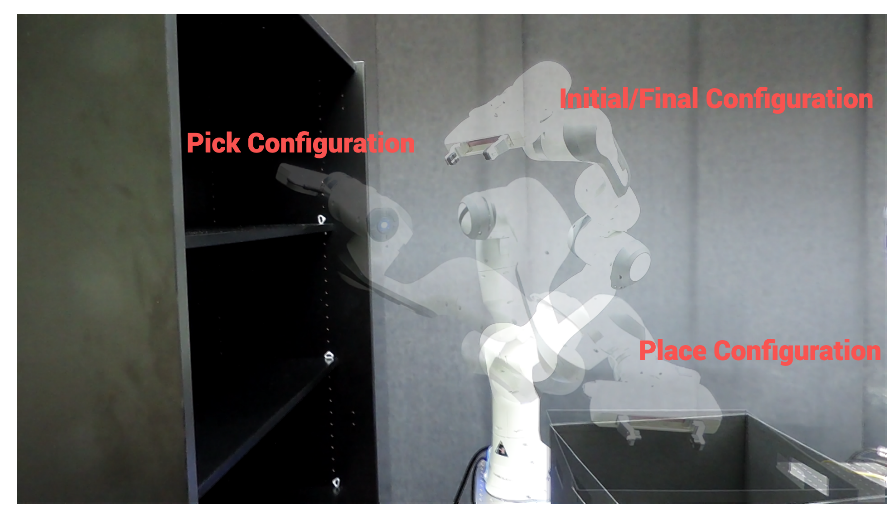
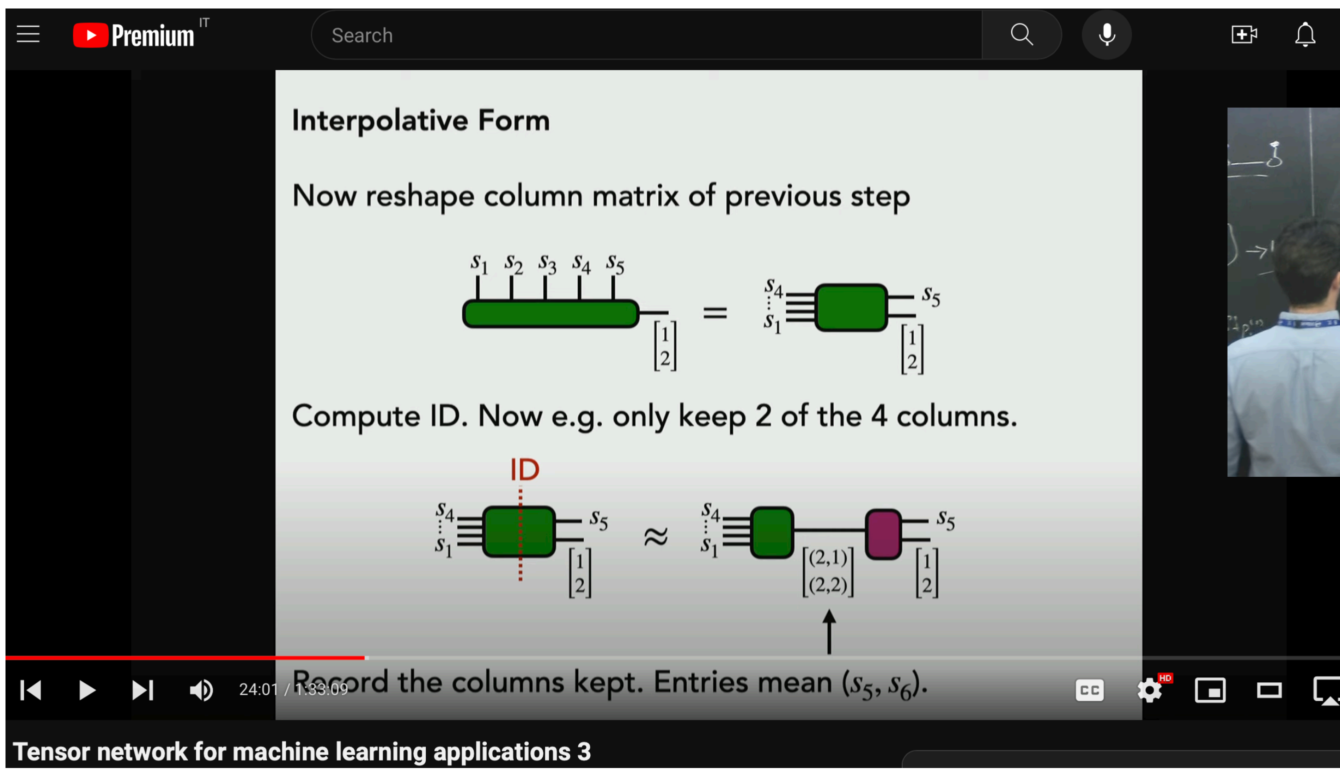


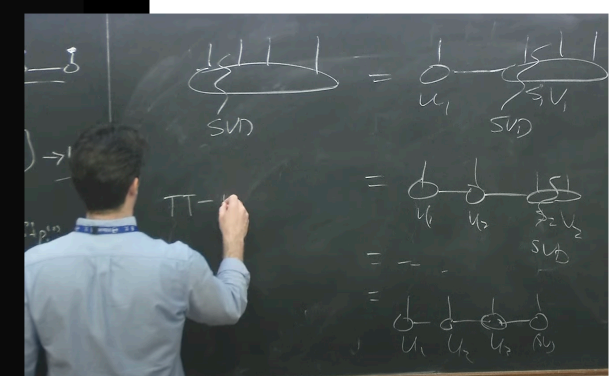
Figure 12. Real robot implementation of one of the TTGO solutions for the pick-and-place task. The motion from the initial configuration to the final configuration (same as the initial configuration in this case) via the picking configuration and placing configuration is depicted.

* Shetty, Lembono, Loew, and Calinon, Tensor Train for Global Optimization Problems in Robotics

Future Directions – Tensor Cross Interpolation

For more details of Tensor Cross algorithm,
see following lecture and slides





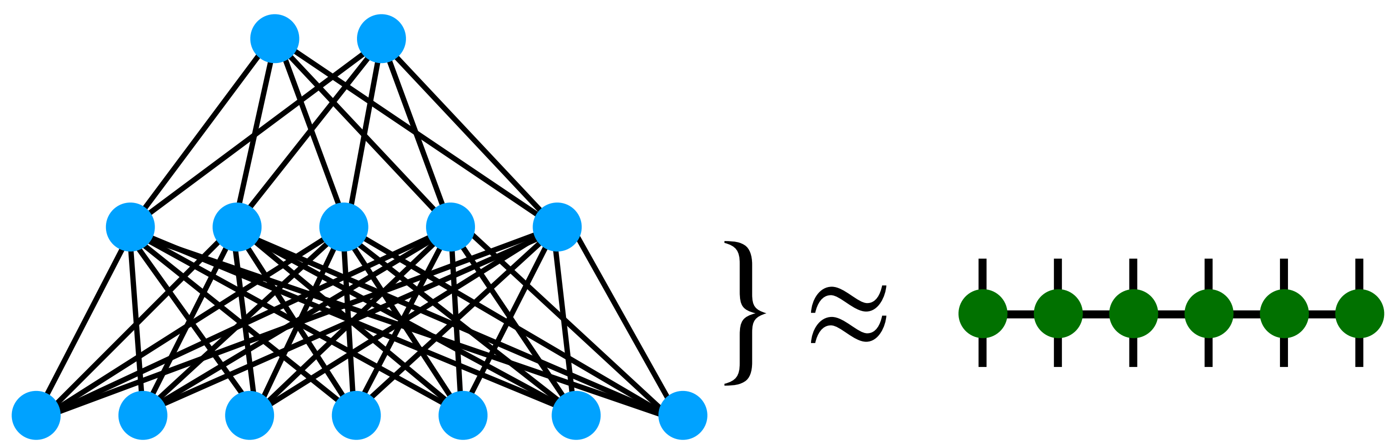
[YouTube Link](https://youtu.be/PFijeMaRGUc) (https://youtu.be/PFijeMaRGUc)

[Slides Link](https://itensor.org/miles/Trieste03TCI.pdf) (https://itensor.org/miles/Trieste03TCI.pdf)

Tensor Network Machine Learning

Many other works I did not have time to cover e.g.

- Compressing neural network weights with tensor networks

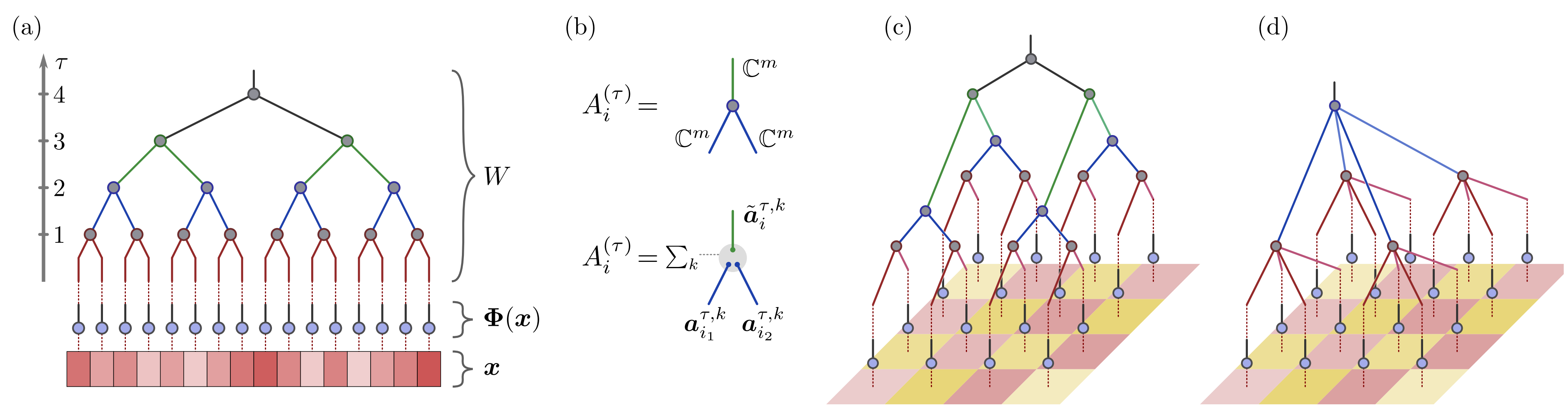


Novikov et al., Advances in Neural Information Processing (2015) (arxiv:1509.06569)

Garipov, Podoprikhin, Novikov, arxiv:1611.03214

- Tree networks of low-rank (CP rank) tensors

Chen, Barthel, arxiv:2305.19440

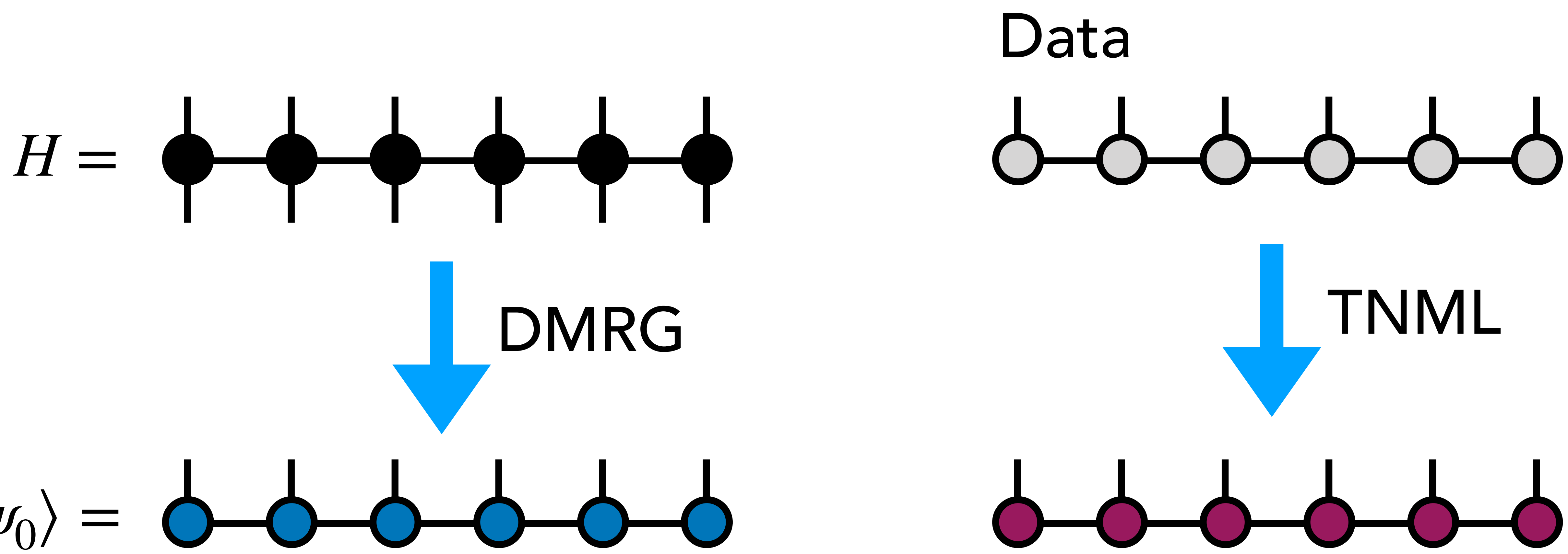


Outlook and Future Directions

Outlook

We saw a ***framework*** for machine learning using tensor networks

Goal of capturing power of tensor networks for physics but for broader applications



Outlook

Does it really deliver? It depends… (as of 2024)

For images / computer vision, promising but not yet competitive:

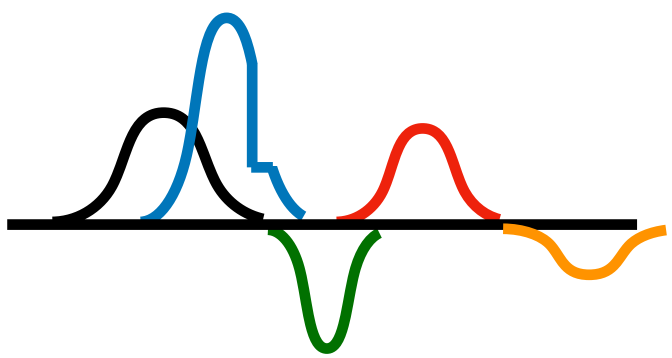
- tensor networks have lots of parameters (relatively)
- gradient descent is therefore slow
- but linear algebra algorithms much faster than gradient
- tensor networks are just linear algebra in high dimensions

Outlook

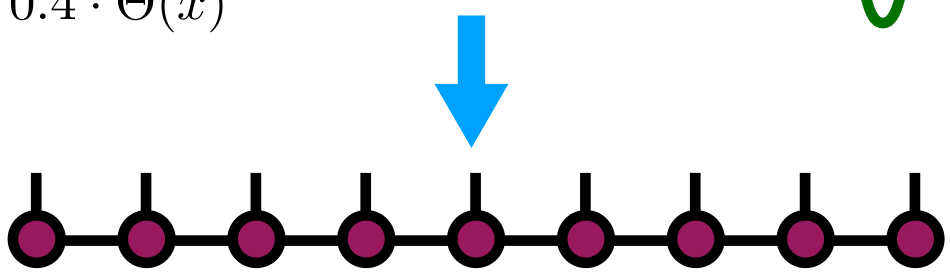
Does it really deliver? It depends… (as of 2024)

For low-to-medium dimensional functions,
very powerful

$$f(x) = \sum_{g=1}^{N_g} a_g e^{-w_g(x-x_g)^2} + 0.4 \cdot \Theta(x)$$



$$f(x) \approx$$



Two novel algorithms for TN machine learning

What promise might they hold?