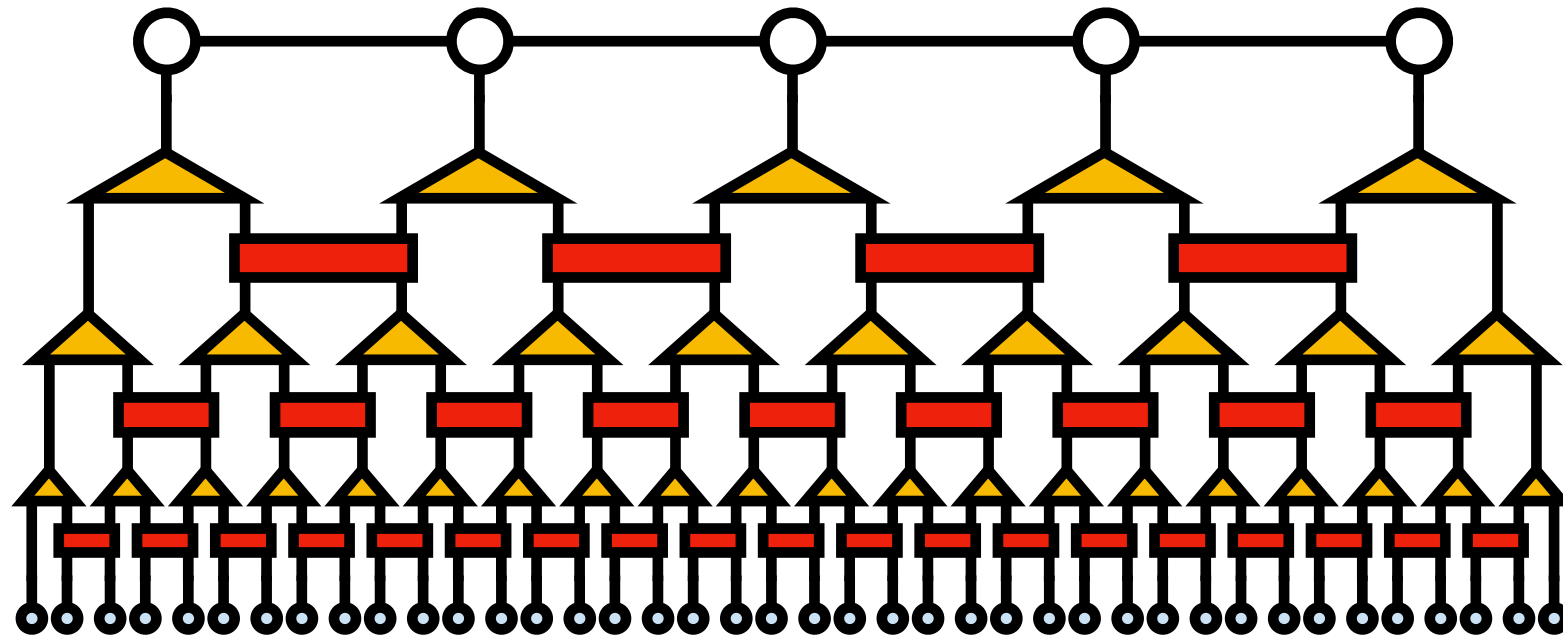


Tensor Networks for Machine Learning: Architectures, Algorithms, and Applications



Talk Advertisement:

Prof. Jens Eisert will speak about tensor networks in machine learning today at 9pm CET (3pm EST)

"Tensor networks as a data structure in probabilistic modelling and for learning dynamical laws from data"

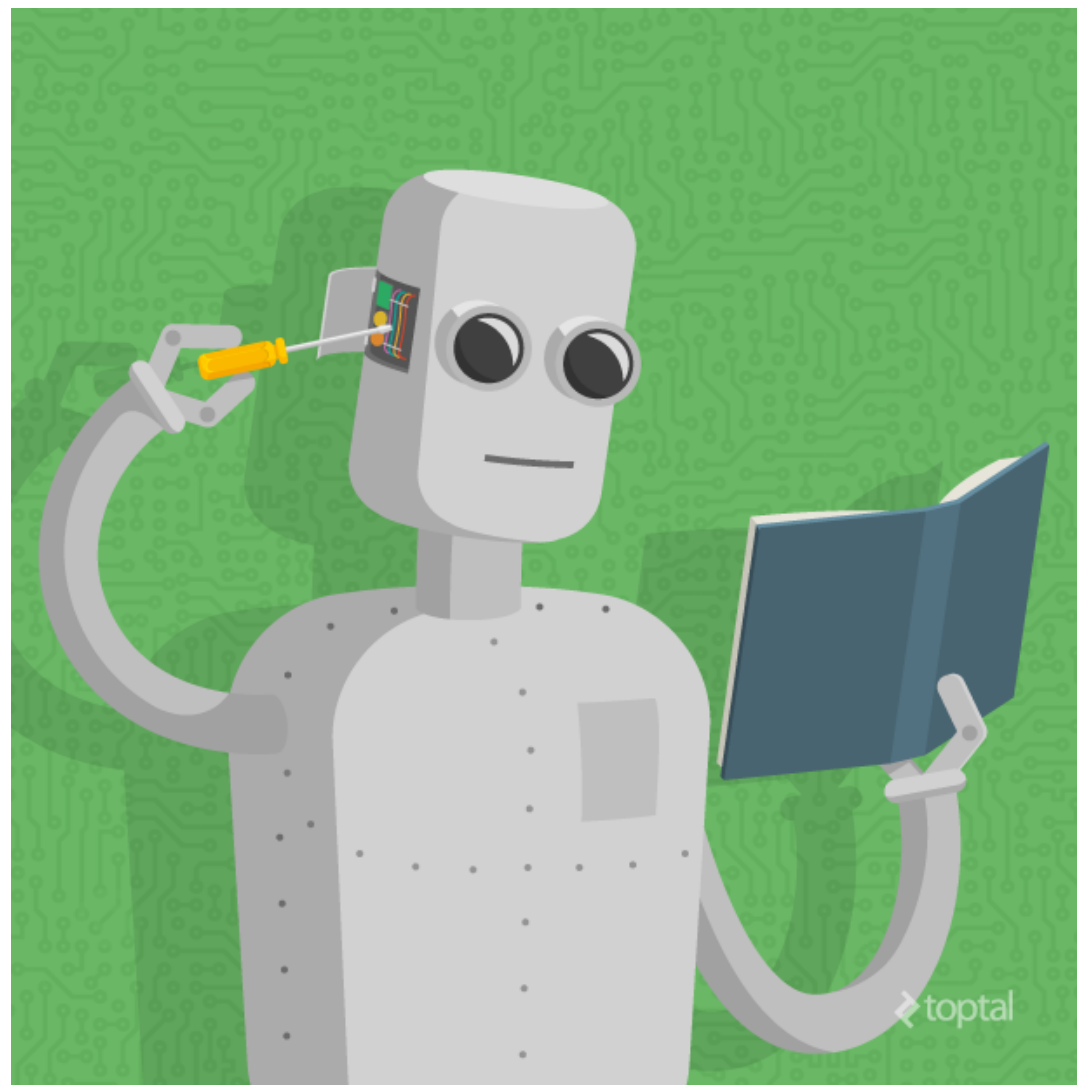
Zoom Meeting ID: 923 9226 6655, Password: 919332



Screen shot this if interested

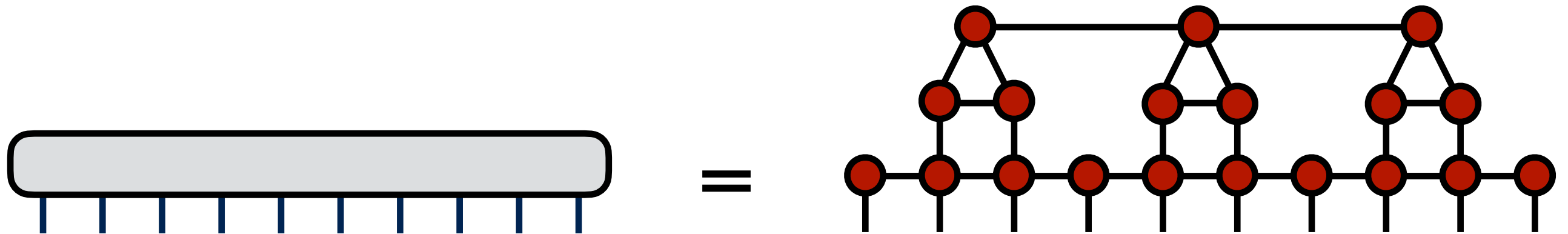
Last Time...

Introduced **machine learning**: system which given more data performs increasingly well at a task



Last Time...

Introduced **tensor networks**: factorization of N-index tensor into contraction of many small tensors



Last Time...

Introduced **tensor network machine learning models**
with weights as tensor network (entering linearly)

$$f(x_1, x_2, \dots, x_6) = \sum_{\mathbf{n}} \begin{array}{c} n_1 \quad n_2 \quad n_3 \quad n_4 \quad n_5 \quad n_6 \\ \text{---} \circ \text{---} \circ \text{---} \circ \text{---} \circ \text{---} \circ \text{---} \circ \text{---} \end{array} x_1^{n_1} x_2^{n_2} \cdots x_6^{n_6}$$

Last Time...

Discussed **applications** to both real-world and physics-specific machine learning tasks

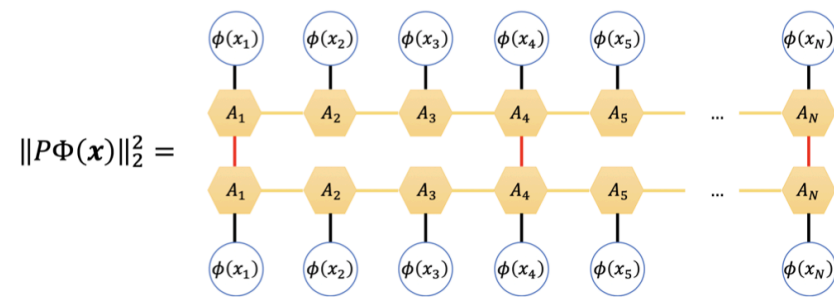
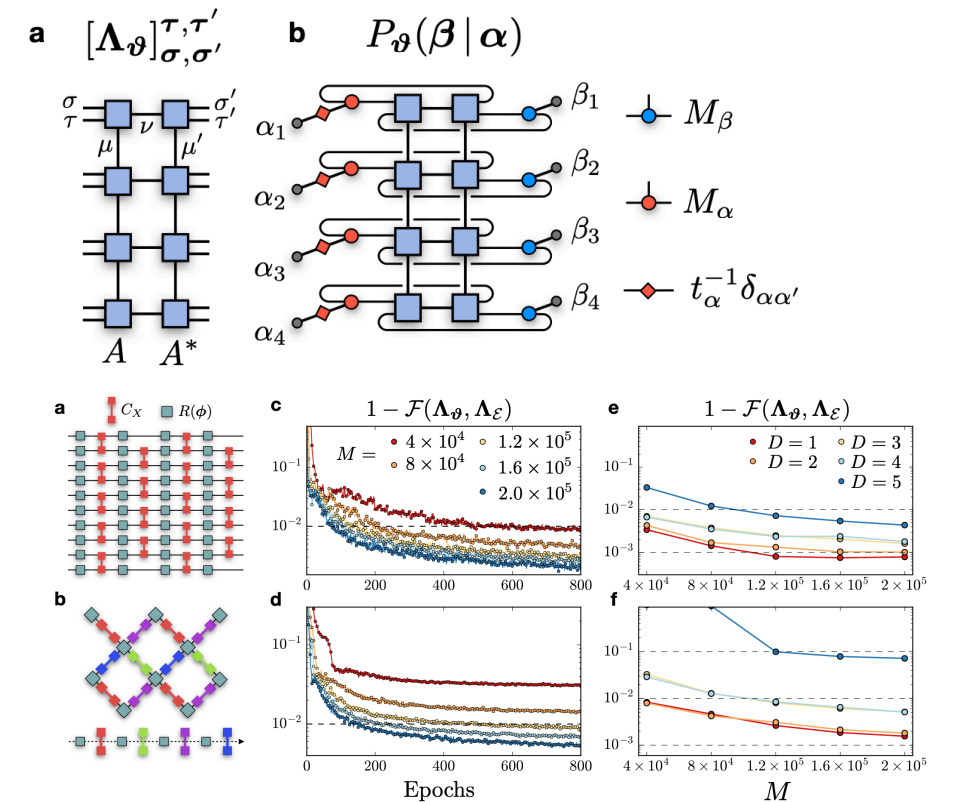


Table 3: Mean AUROC scores (in %) and standard errors on ODDS datasets.

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

Anomaly
Detection with
Tensor Networks

arxiv:2006.02516



Quantum process
tomography with ...
tensor networks

arxiv:2006.02424

Outline of Today's Talk

Review of **tensor network** machine learning

Architectures beyond MPS

Direct learning **algorithm** for training MPS tensor network (for generative modeling)

Theoretical prediction of generalization performance

Typical Ingredients of Machine Learning

Dataset represented as vectors $\{\mathbf{x}_j\}$

Labels are numbers or vectors $\{\mathbf{y}_j\}$

Model function $f_W(\mathbf{x})$

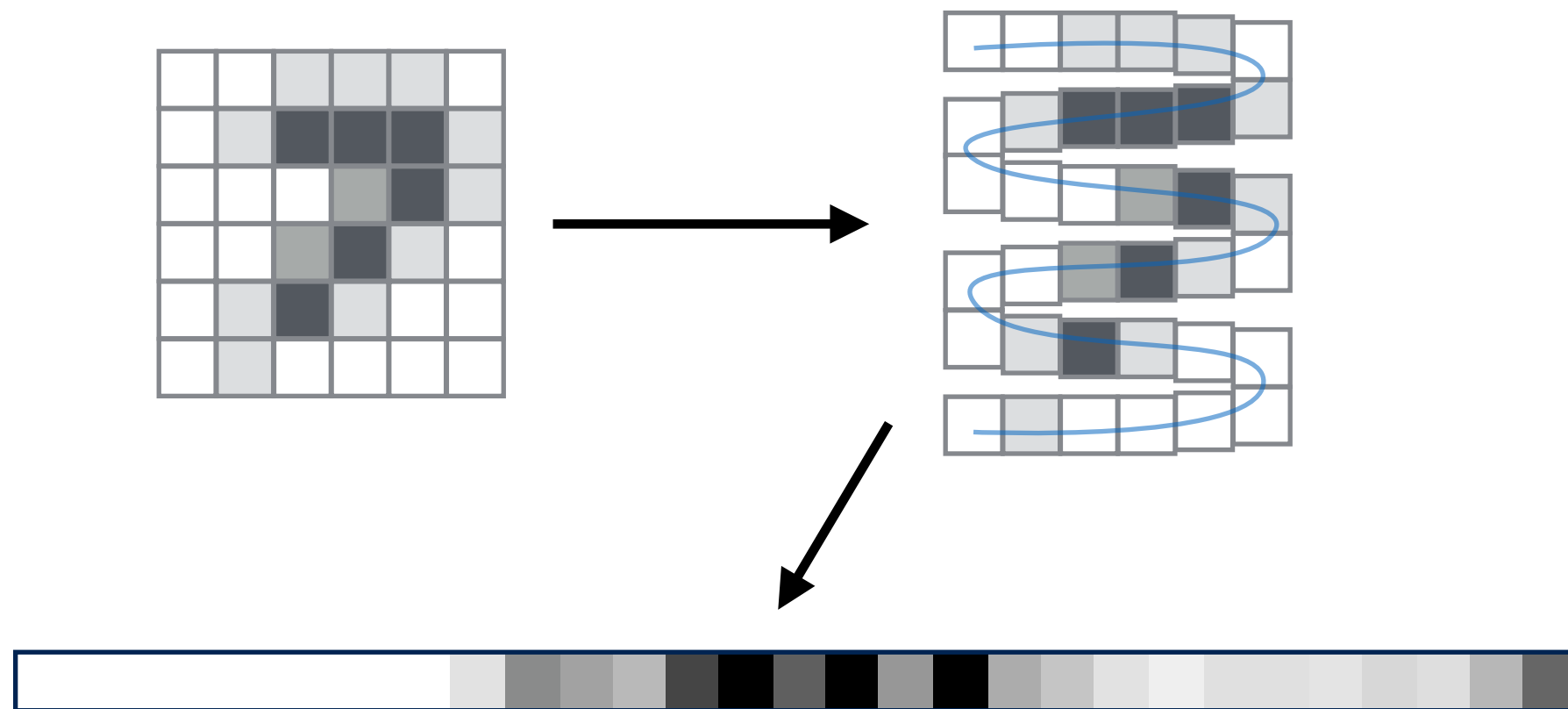
with adjustable weight parameters W

Choose weights to minimize cost function

$$C = \sum_j \mathbb{E}(f_W(\mathbf{x}_j), \mathbf{y}_j)$$

Typical Ingredients of Machine Learning

Dataset represented as vectors $\{\mathbf{x}_j\}$



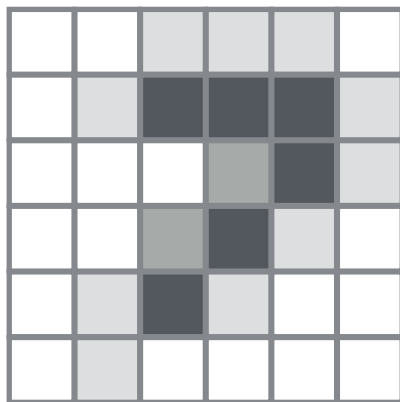
$\mathbf{X} = \{ 0.0, 0.0, 0.1, 0.1, 0.1, 0.0, 0.1, 0.7, 0.7, 0.7, 0.2, \dots \}$

Typical Ingredients of Machine Learning

Dataset represented as vectors $\{\mathbf{x}_j\}$

Labels are numbers (or vectors) $\{y_j\}$

Task: distinguish 7's from 8's



$\mathbf{X} = \{ 0.0, 0.0, 0.1, 0.1, 0.1, 0.0, 0.1, 0.7, 0.7, 0.7, 0.2, \dots$

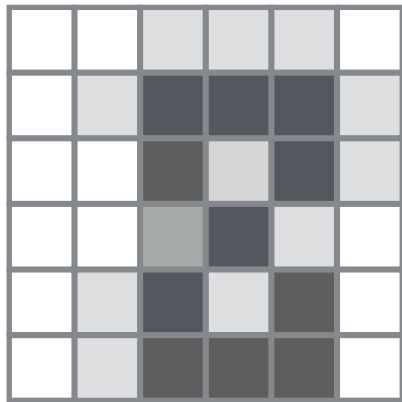
$y = +1$

Typical Ingredients of Machine Learning

Dataset represented as vectors $\{\mathbf{x}_j\}$

Labels are numbers (or vectors) $\{y_j\}$

Task: distinguish 7's from 8's



$\mathbf{X} = \{ 0.0, 0.0, 0.1, 0.1, 0.1, 0.0, 0.1, 0.7, 0.7, 0.7, 0.2, \dots$

$y = -1$

Typical Ingredients of Machine Learning

Model function $f_W(\mathbf{x})$

with adjustable weight parameters W

For example, *linear classifier* model

$$f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + W_0$$

just a dot product of weights with input

Works surprisingly well!

Typical Ingredients of Machine Learning

Choose weights to minimize cost function,
for example *squared error* cost function:

$$C = \frac{1}{N} \sum_{j=1}^N (f_W(\mathbf{x}_j) - y_j)^2$$

$$= \frac{1}{N} \sum_{j=1}^N (\mathbf{w} \cdot \mathbf{x}_j - y_j)^2$$

$$y_j = \begin{cases} +1 & \mathbf{x}_j \in \text{7's} \\ -1 & \mathbf{x}_j \in \text{8's} \end{cases}$$

Typical Ingredients of Machine Learning

Can optimize weights using *gradient descent*

$$C = \frac{1}{N} \sum_{j=1}^N (\mathbf{W} \cdot \mathbf{x}_j - y_j)^2$$

$$-\frac{dC}{dW_n} = \frac{2}{N} \sum_{j=1}^N (y_j - \mathbf{W} \cdot \mathbf{x}_j) x_j^n = \Delta^n$$

$$W^n \rightarrow W^n + \alpha \Delta^n \quad \alpha = \text{learning rate}$$

Project: program this in Julia or Python

Adding Features

What if a linear classifier is not enough?

$$f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + W_0$$

$$= W_0 + W_1x_1 + W_2x_2 + W_3x_3 + \dots$$

Adding Features

What if a linear classifier is not enough?

$$\cancel{f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + W_0}$$

$$\cancel{= W_0 + W_1x_1 + W_2x_2 + W_3x_3 + \dots}$$

Can make extended linear classifier as follows:

$$\begin{aligned} f_{\mathbf{W}, \mathbf{V}}(\mathbf{x}) = & W_0 + W_1x_1 + W_2x_2 + W_3x_3 + \dots \\ & + V_{12}x_1x_2 + V_{13}x_1x_3 + V_{23}x_2x_3 + \dots \end{aligned}$$

Think of $\mathbf{W} \oplus \mathbf{V}$ as weight vector of this 'linear' model
Manifestly non-linear function of $\mathbf{x}$

Adding Features

Why not keep going, adding third-order $x_1x_2x_3$ and fourth-order $x_3x_5x_7x_9$ terms etc. ?

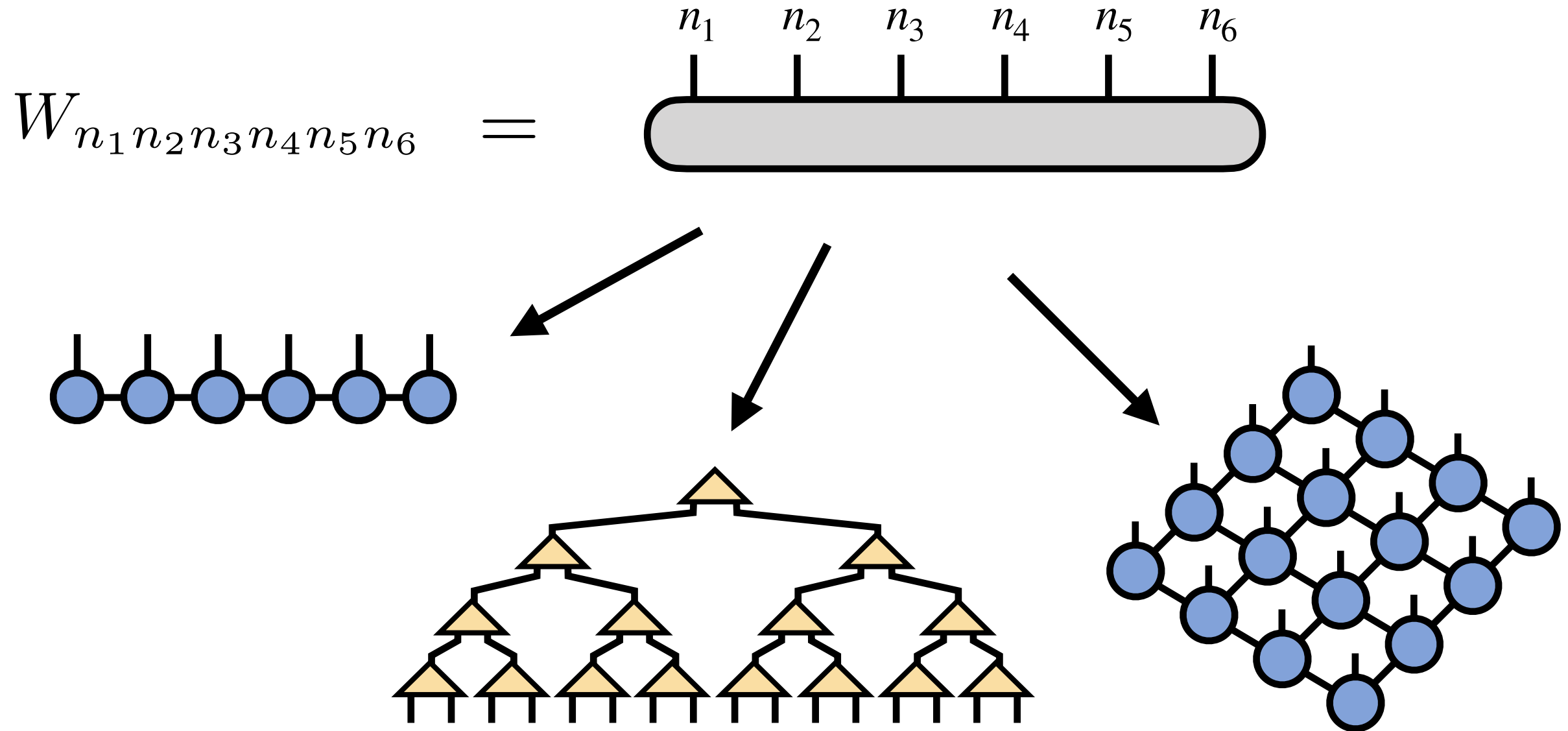
Arrive at model with *weight tensor*

$$f(x_1, x_2, \dots, x_N) = \sum_{\mathbf{n}} W_{n_1 n_2 \dots n_N} x_1^{n_1} x_2^{n_2} \dots x_N^{n_N}$$

Non-linear function of $\mathbf{x}$

Weights enter *linearly* into model

By factorizing exponentially big weight tensor



Into a tensor network, model becomes efficiently trainable & we will see other advantages

Architectures Beyond Matrix Product States

$$W_{n_1 n_2 n_3 n_4 n_5 n_6} = \text{Diagram of a Matrix Product State (MPS) tensor network with 6 blue circular nodes connected horizontally, each with a vertical line above it.}$$

Matrix product state (MPS)

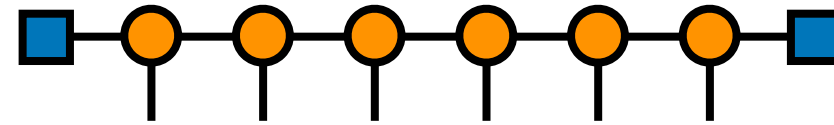
weights already a powerful representation

- **expressive**, especially for one-dimensional correlations
- multiple optimization **algorithms & strategies**
- best understood network **theoretically**

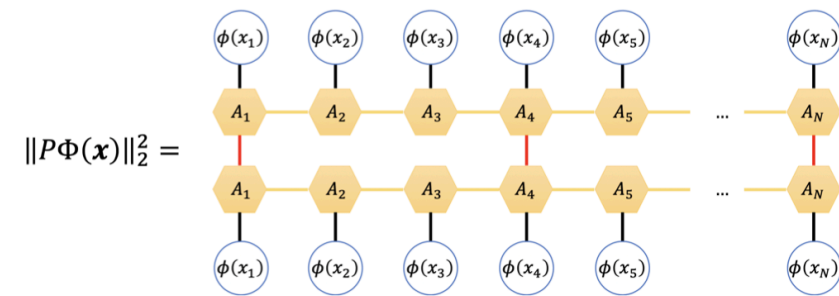
But worthwhile to explore other tensor networks –
let's briefly see why...

Since 2016, tensor network machine learning now successfully "ported" to other tensor net architectures

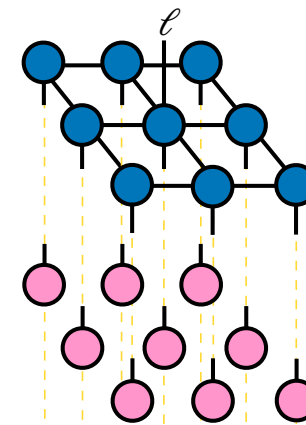
Infinite MPS



Locally purified states

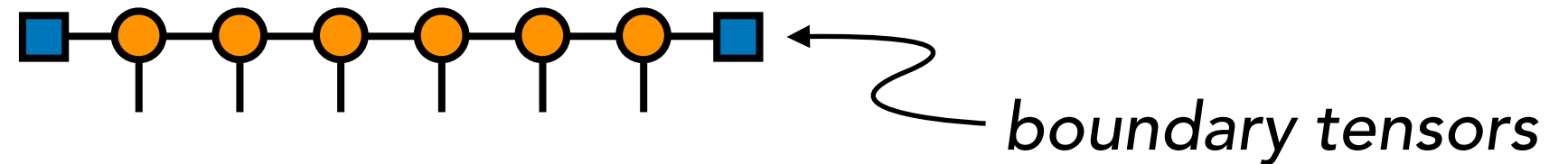


PEPS



Infinite MPS

Miller, Rabusseau, Terilla, "Tensor Networks for Probabilistic Sequence Modeling", arxiv:2003.01039



- used to generate model languages with various grammars
- very few parameters and parallel optimization
- superior results to LSTM in many cases, equal in most others
- can *generalize* from training on shorter sequences to correct results on longer sequences (so really learning the grammar)

Locally purified states

Anomaly Detection with Tensor Networks

arxiv:2006.02516

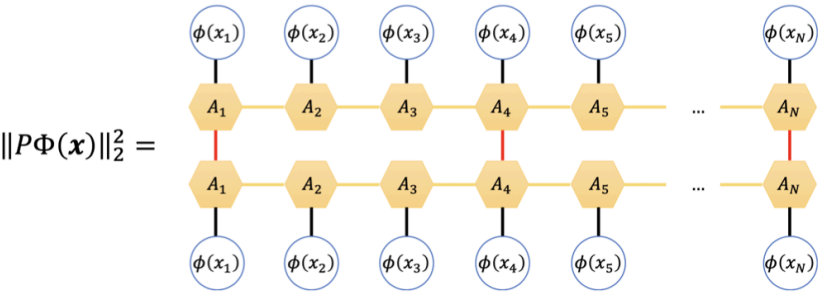


Table 3: Mean AUROC scores (in %) and standard errors on ODDS datasets.

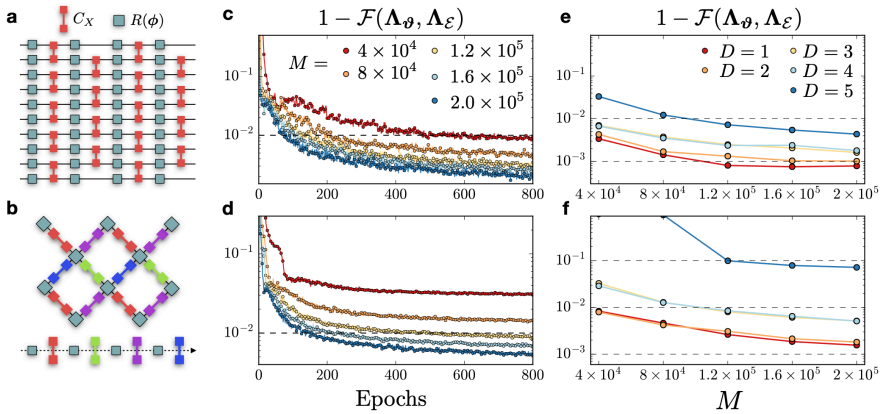
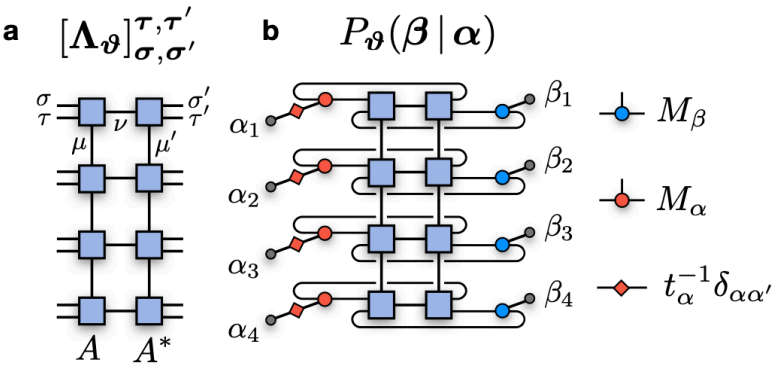
Dataset	OC-SVM	IF	GOAD	DAGMM	TNAD
Wine	60.0	46.0 ± 8.4	48.2 ± 24.7	51.7 ± 19.3	97.3 ± 4.5
Glass	62.0	57.2 ± 1.6	53.5 ± 13.6	52.5 ± 12.9	81.8 ± 7.3
Thyroid	98.8	99.0 ± 0.1	95.8 ± 1.3	88.8 ± 6.8	99.0 ± 0.1
Satellite	79.9	77.2 ± 0.9	60.6 ± 5.3	72.1 ± 4.7	81.3 ± 0.5
Forest	97.7	71.7 ± 2.6	64.6 ± 4.7	60.9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

Novel anomaly detection framework

Results better than neural
networks for tabular data

Quantum process tomography with ... tensor networks

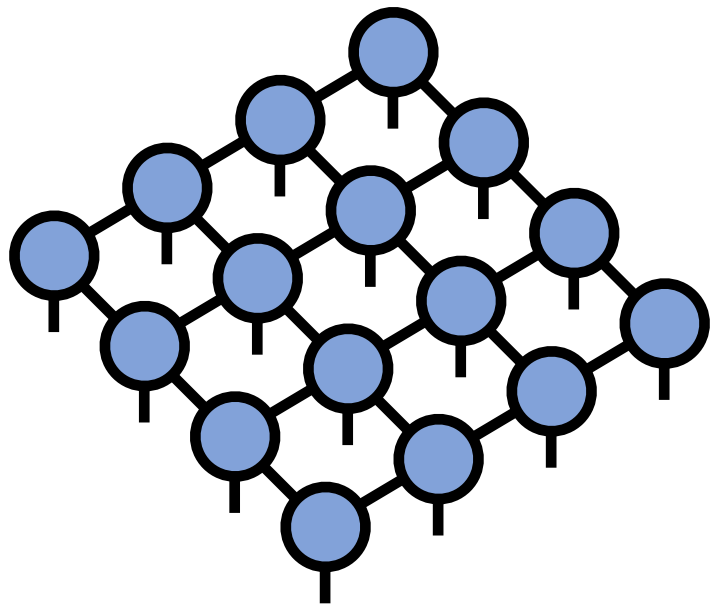
arxiv:2006.02424



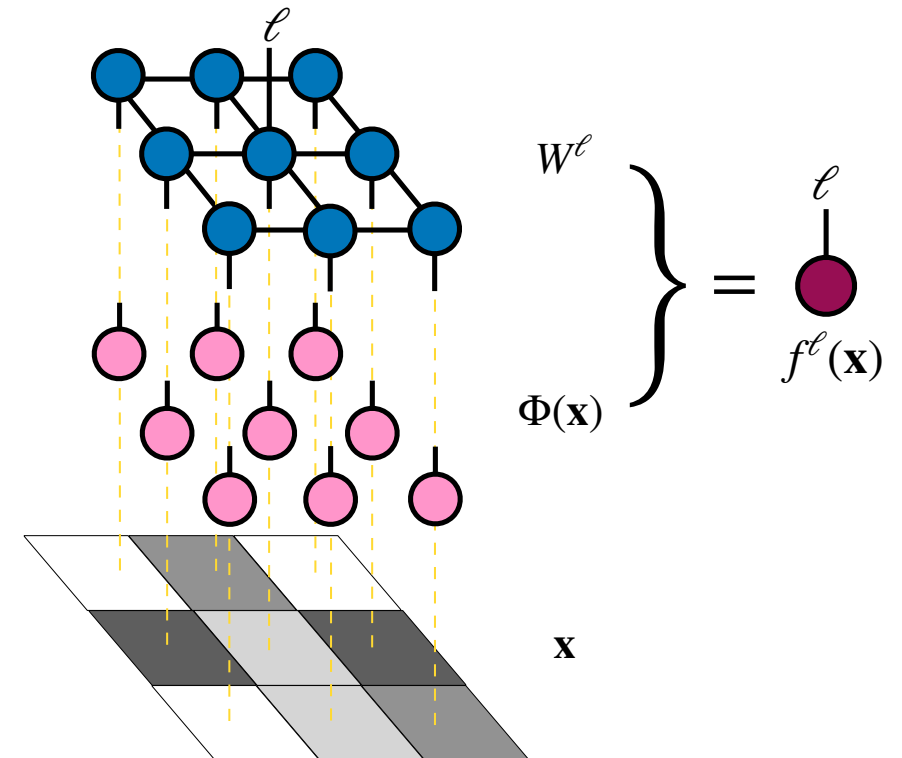
Scalable learning of noisy
quantum "channels" or
processes from experiments

PEPS (Projected Entangled Pair States)

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

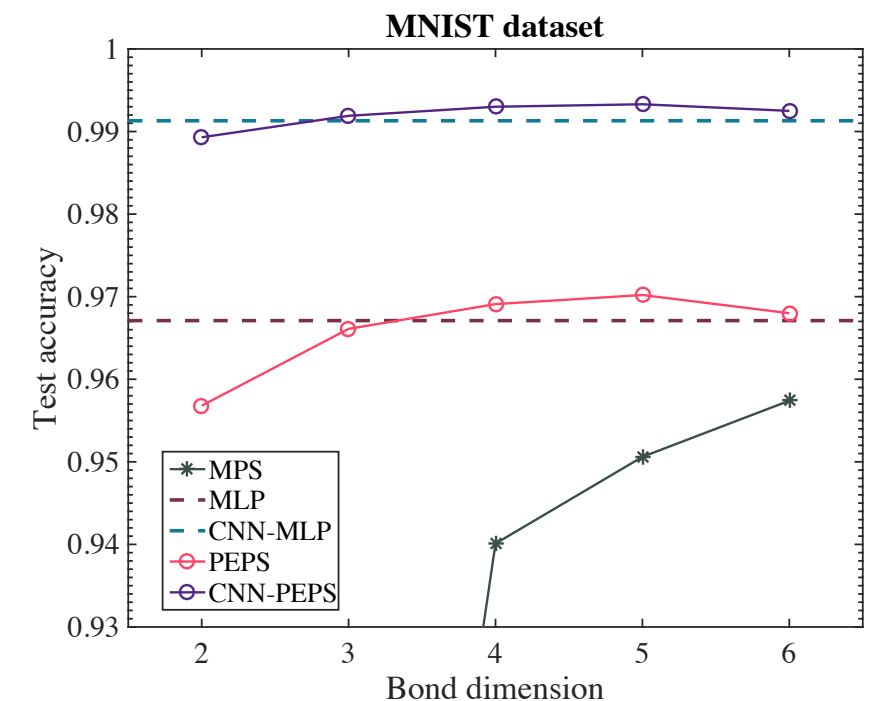


PEPS = 2D analogue of MPS



Framework for learning

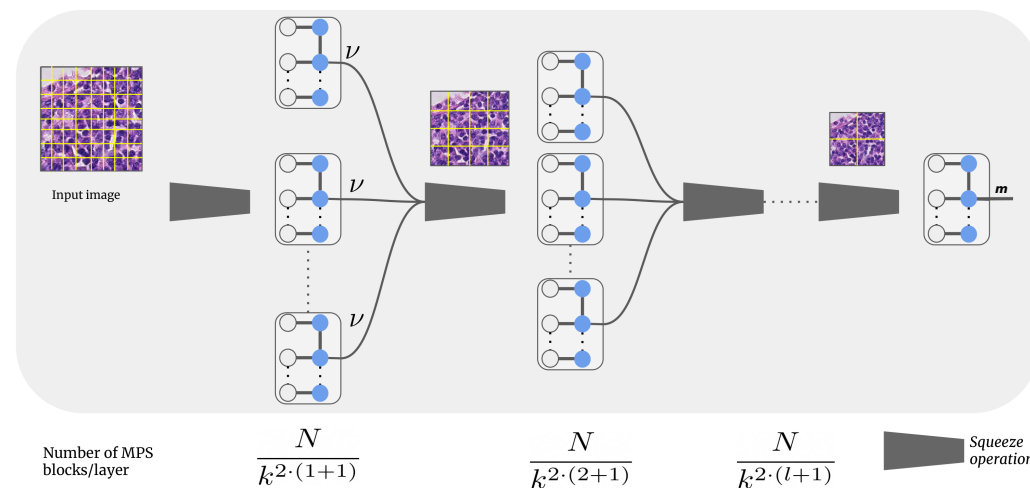
- comparable results to NN with fewer parameters
- starting to see overfitting at larger D? →
- convolutional layers help (91% test correct on FashionMNIST using CNN+PEPS!)
- cost to train is high



Architectures & Applications

Many other applications & architectures could be mentioned:

- Selvan, Dam, "Tensor Networks for Medical Image Classification"



arxiv:2011.06982

- Wall et al. "Generative machine learning with tensor networks: benchmarks on near-term quantum computers" arxiv:2010.03641

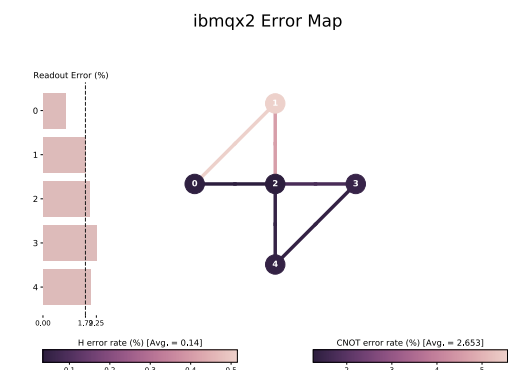
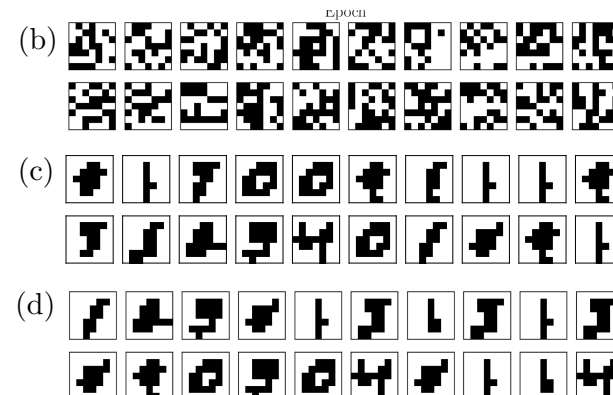
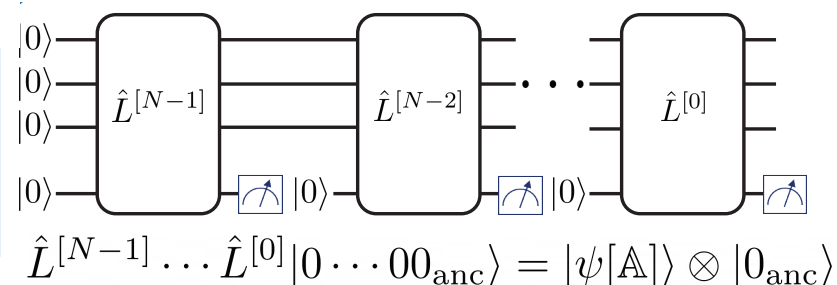


FIG. 4. Exemplar NISQ hardware architecture. The qubit layout

Algorithms for Optimizing Tensor Network Models

Exploring *performance* of tensor networks is interesting

But also seek

theory of training algorithms

theory of generalization

understanding which models best for certain data

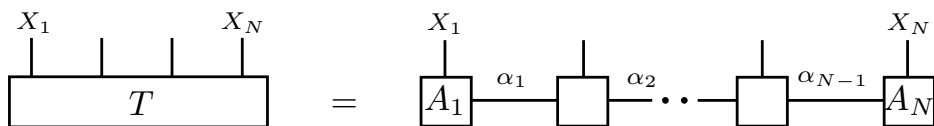
....

even if it means trading away performance

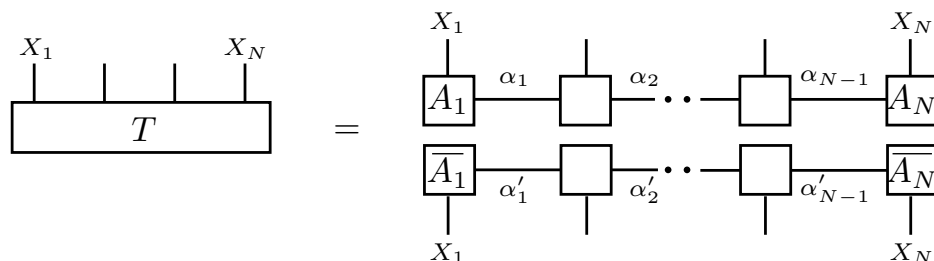
Theory is Already Being Developed

Theory of expressive power of tensor architectures:

TT/MPS



Born Machine (BM)



Locally Purified State (LPS)

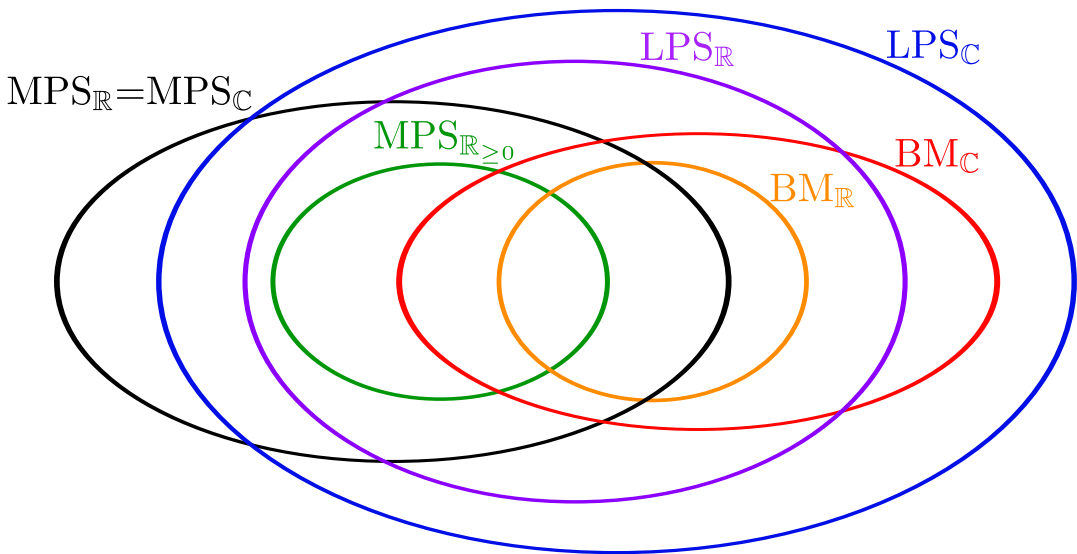
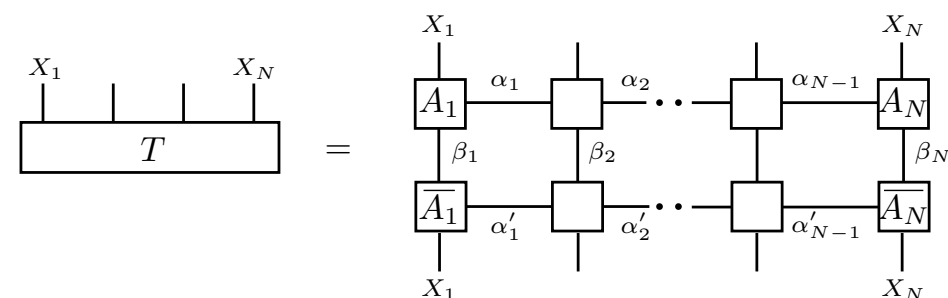


Table 2: Results of Proposition 2

	TT-rank $_{\mathbb{R}}$	TT-rank $_{\mathbb{R} \geq 0}$	Born-rank $_{\mathbb{R}}$	Born-rank $_{\mathbb{C}}$	puri-rank $_{\mathbb{R}}$	puri-rank $_{\mathbb{C}}$
TT-rank $_{\mathbb{R}}$	=	<	<, >	<, >	<, >	<, >
TT-rank $_{\mathbb{R} \geq 0}$	>	=	<, >	<, >	>	>
Born-rank $_{\mathbb{R}}$	<, >	<, >	=	>	>	>
Born-rank $_{\mathbb{C}}$	<, >	<, >	<	=	<, >	>
puri-rank $_{\mathbb{R}}$	<, >	<	<	<, >	=	>
puri-rank $_{\mathbb{C}}$	<, >	<	<	<	<	=

Plan for This Section

Let's study training **algorithms** more systematically

Use **synthetic data** (even-parity dataset) to make evaluation of model well-posed

Contrast two training algorithms

Theoretical prediction of generalization performance of second algorithm

For the synthetic data set, we consider *even-parity data set*

Bit strings of length L with even number of 1 bits

Even-parity data set ($L = 10$):

0 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 1 1 1

0 1 0 1 0 0 1 0 0 1

0 0 1 1 0 0 0 1 1 0

⋮

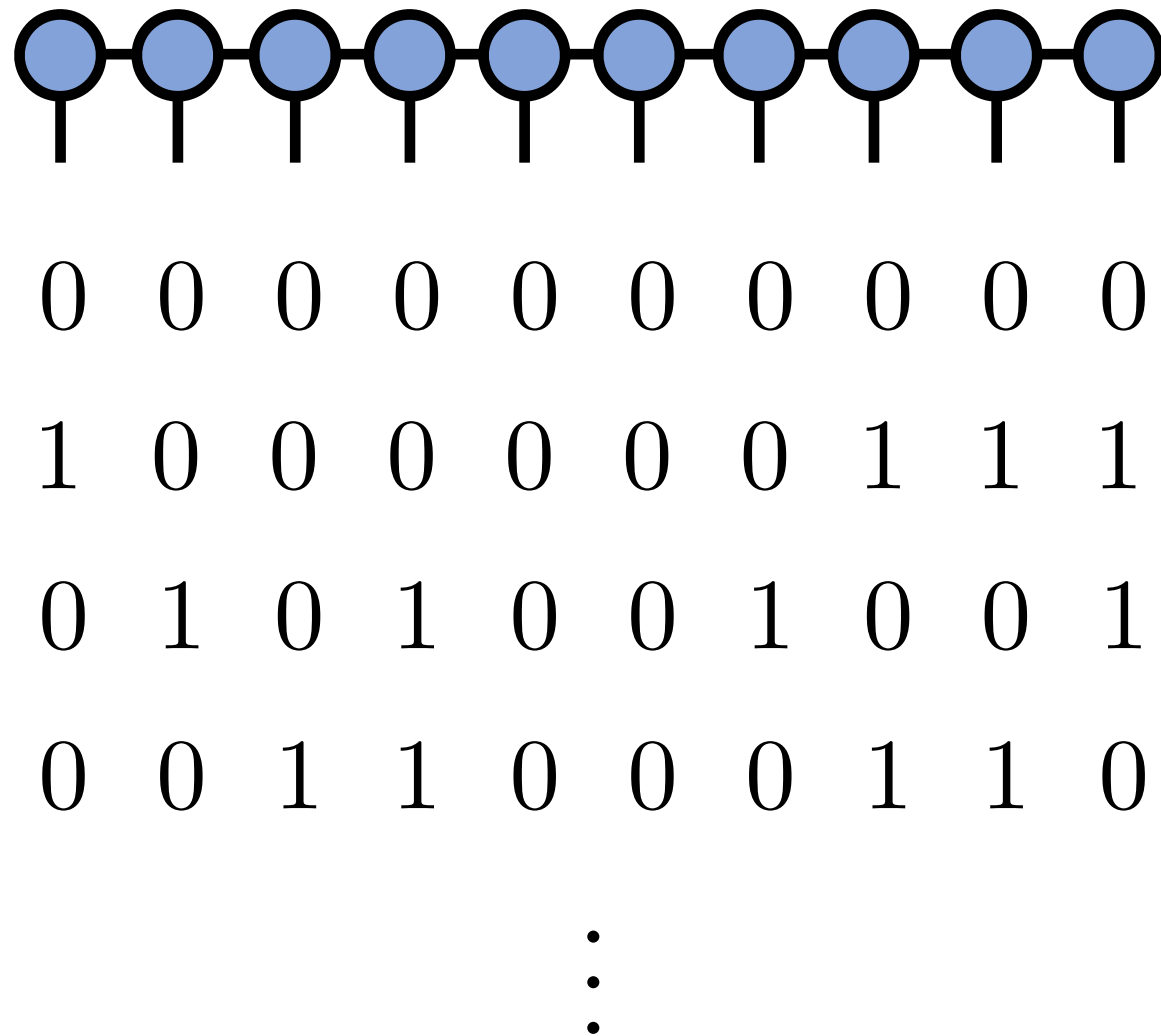
Two ways of representing probability distribution through a tensor:

$$p(s_1, s_2, s_3, s_4, s_5, s_6) = \text{[yellow bar with labels } s_1, s_2, s_3, s_4, s_5, s_6 \text{]} \quad \text{1-norm formalism}$$

$$p(s_1, s_2, s_3, s_4, s_5, s_6) = \left| \text{[purple bar with labels } s_1, s_2, s_3, s_4, s_5, s_6 \text{]} \right|^2 \quad \text{2-norm formalism}$$

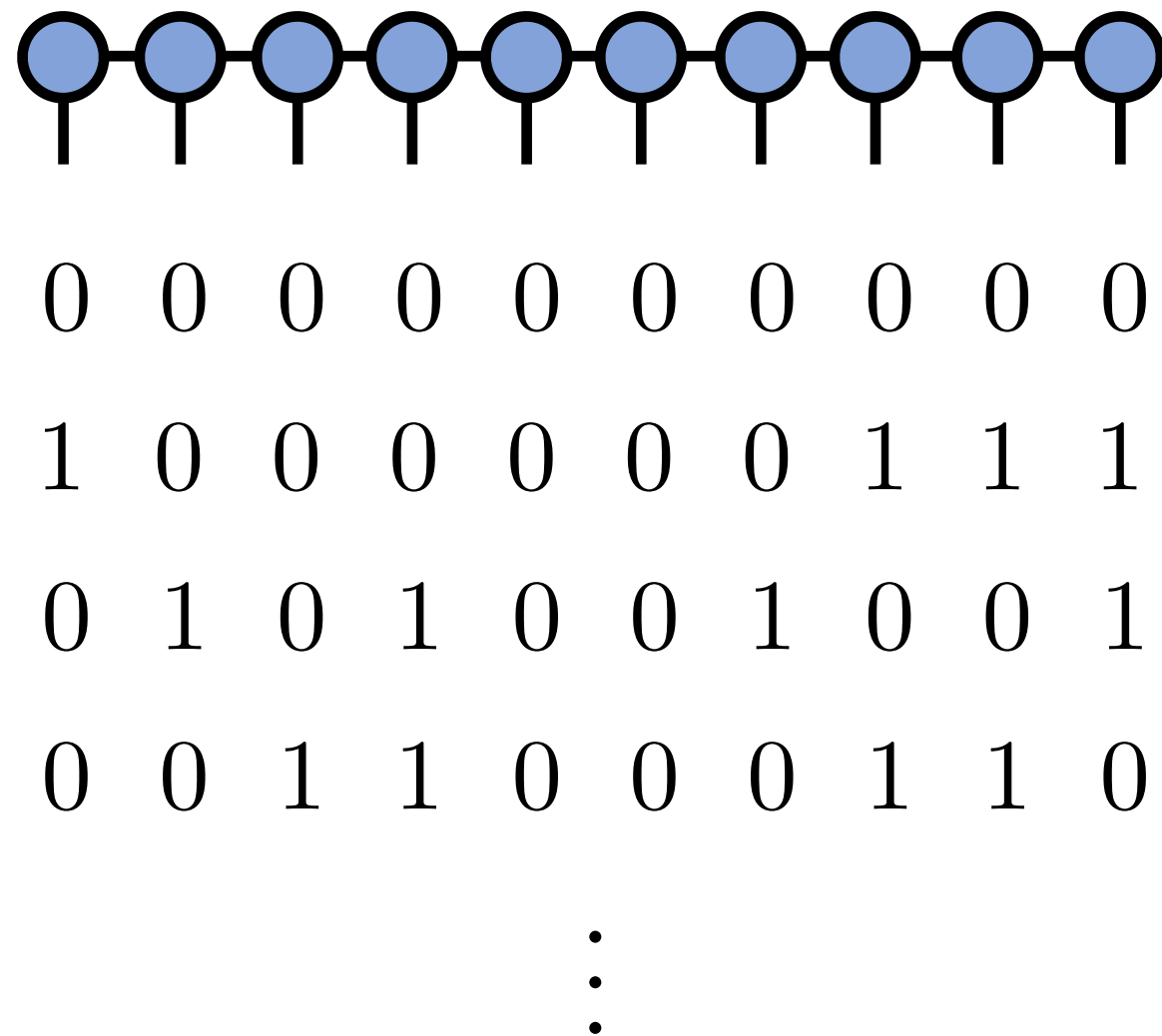
Advantages & disadvantages of each
(will use **2-norm** in what follows)

Seek an MPS which generates this data with uniform probability



Entire even-parity dataset can be fit by
MPS of bond dimension 2

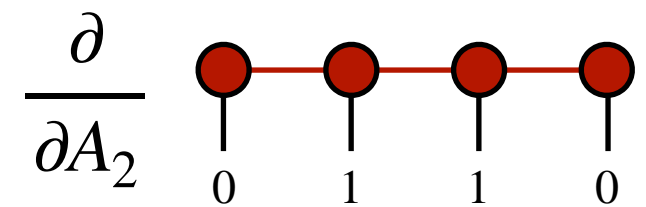
So we already know model is in the right class



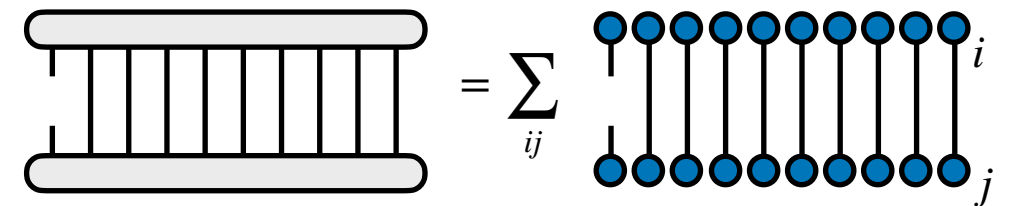
There are many ways one can devise to optimize MPS for machine learning objectives

Let's discuss two:

1. Alternating Optimization of Tensors



2. Density matrix algorithm

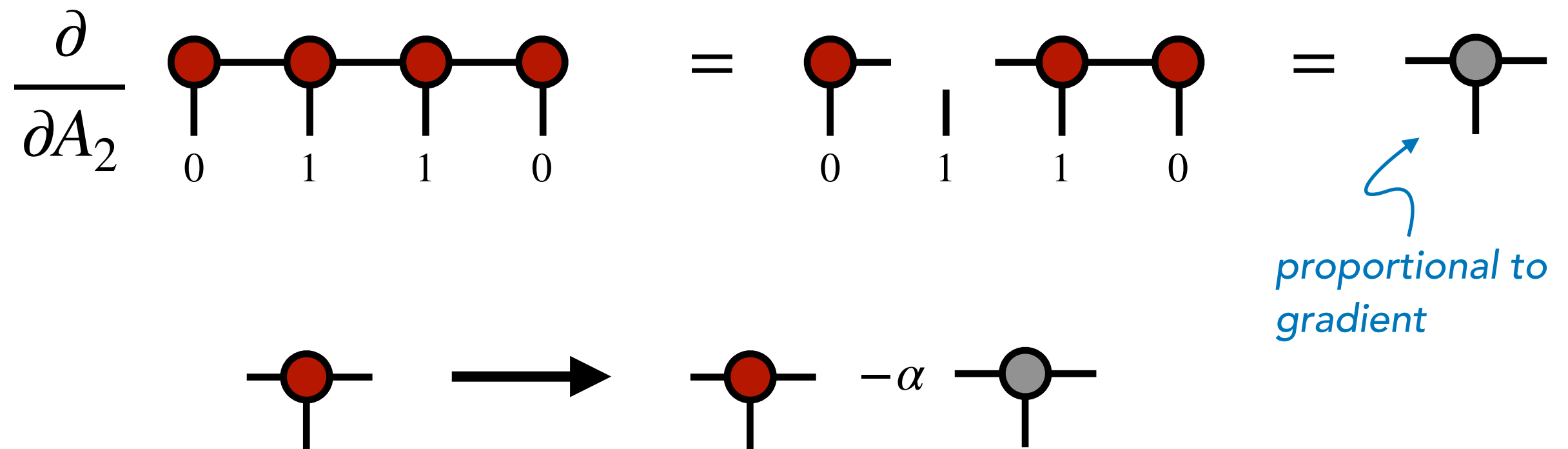


1. Alternating Optimization of MPS Tensors

Alternating Optimization of MPS Tensors

Optimize one MPS tensor at a time

Project data samples through MPS tensors to obtain gradient:

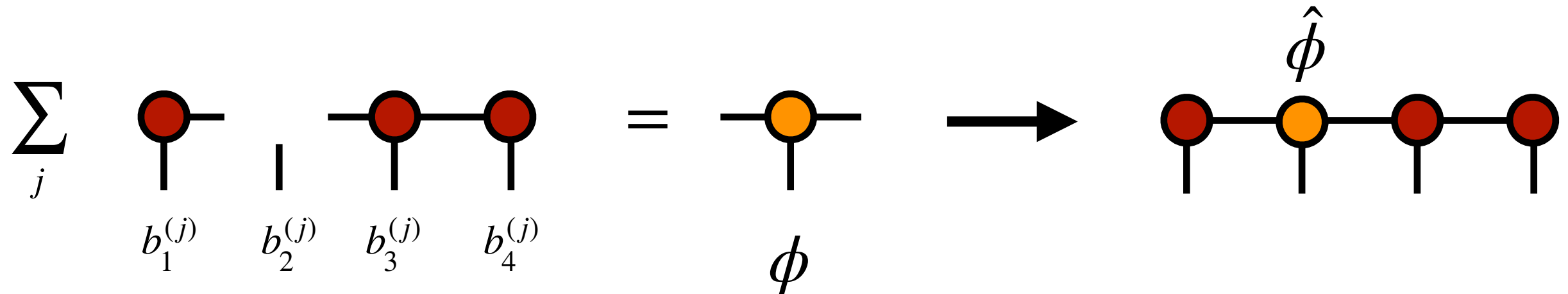


Scaling **linear** in training set size & **linear** in length of MPS

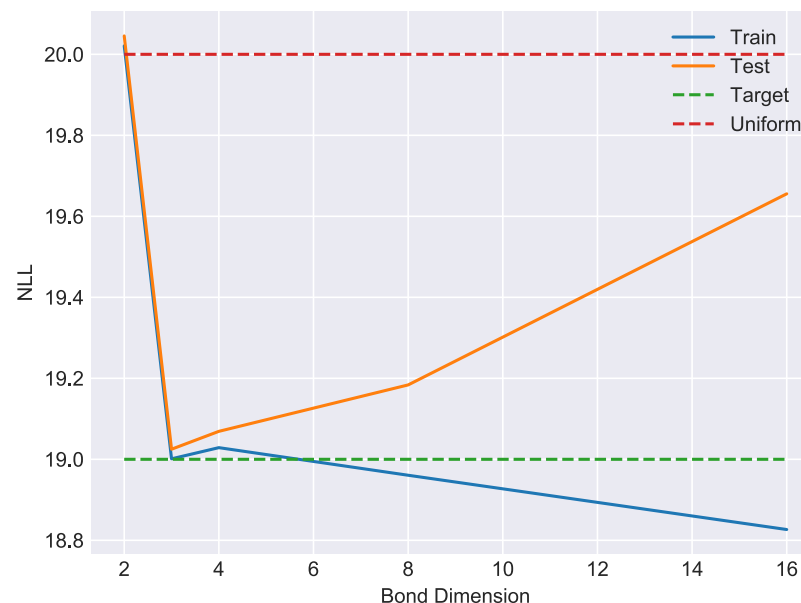
Alternating Optimization of MPS Tensors

Used by Stokes and Terilla to study **even-parity** data set

Locally optimal update for generative training:



Bond dimension related to generalization gap:



Alternating Optimization of MPS Tensors

But alternating optimization has some drawbacks:

- can get **stuck** in local minimum
- requires good **initialization** to work
- results depend on how **long** algorithm is run

Alternatives?

2. Density Matrix Algorithm

Density Matrix Algorithm

This algorithm involves a **deterministic** strategy:

write 'perfect' answer then compress into MPS

Because every step of the algorithm involves **linear algebra**, we could product a theoretical **prediction of generalization** performance



Tai-Danae Bradley



John Terilla

First let's review the learning algorithm

Then give a summary of the theoretical prediction

Density Matrix MPS Learning Algorithm

Map data to rank-one tensors

$$0 \longrightarrow \text{light gray circle with top line} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$1 \longrightarrow \text{dark gray circle with top line} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{array}{ccccccccc} 0 & 1 & 1 & 0 & 1 & 1 & \longrightarrow & \text{light gray circle with top line} & \text{dark gray circle with top line} & \text{dark gray circle with top line} & \text{light gray circle with top line} & \text{dark gray circle with top line} & \text{dark gray circle with top line} \\ & & & & & & & \Phi(\mathbf{x}_i) \end{array}$$

$\mathbf{x}_i$ $\Phi(\mathbf{x}_i)$

Density Matrix MPS Learning Algorithm

Define "target tensor" which perfectly overfits training data

$$N = |\mathcal{T}|$$

$$\psi^{s_1 s_2 s_3 s_4 s_5 s_6} = \frac{1}{\sqrt{N}} \sum_{j \in \mathcal{T}} \text{Diagram 1}$$

Diagram 1: A horizontal row of six red circles, each with a vertical line extending upwards from its center.

$$= \text{Diagram 2}$$

Diagram 2: A horizontal red bar with rounded ends and a dark blue outline. Six vertical lines extend upwards from the top edge of the bar, evenly spaced.

(Do not actually perform sum yet!)

See also: Stokes, Terilla, arxiv:1902.06888

Density Matrix MPS Learning Algorithm

Perfectly overfits because data vectors orthonormal

$$\begin{aligned} p(\mathbf{x}_i) &= \left| \begin{array}{c} \text{Diagram: A red horizontal bar with six vertical lines extending upwards to six circles. The circles alternate in shading: light gray, dark gray, dark gray, light gray, light gray, light gray.} \end{array} \right|^2 \\ &= \left| \frac{1}{\sqrt{N}} \sum_{j \in \mathcal{T}} \begin{array}{c} \text{Diagram: A pair of circles (one light gray, one red) connected by a vertical line. This is repeated six times in a row, corresponding to the six data vectors in the set T.} \end{array} \right|^2 \\ &= \frac{1}{N} \quad \text{if } i \in \mathcal{T} \\ &\quad \text{otherwise zero} \end{aligned}$$

Density Matrix MPS Learning Algorithm

Compute closest MPS to this tensor
(restricting to bond dimension 2)

$$\psi^{s_1 s_2 s_3 s_4 s_5 s_6} = \frac{1}{\sqrt{N}} \sum_{j \in \mathcal{T}} \text{Diagram 1}$$

Diagram 1: A horizontal row of six red circles. Each circle has a vertical line extending upwards to a label $s_1, s_2, s_3, s_4, s_5, s_6$ respectively.

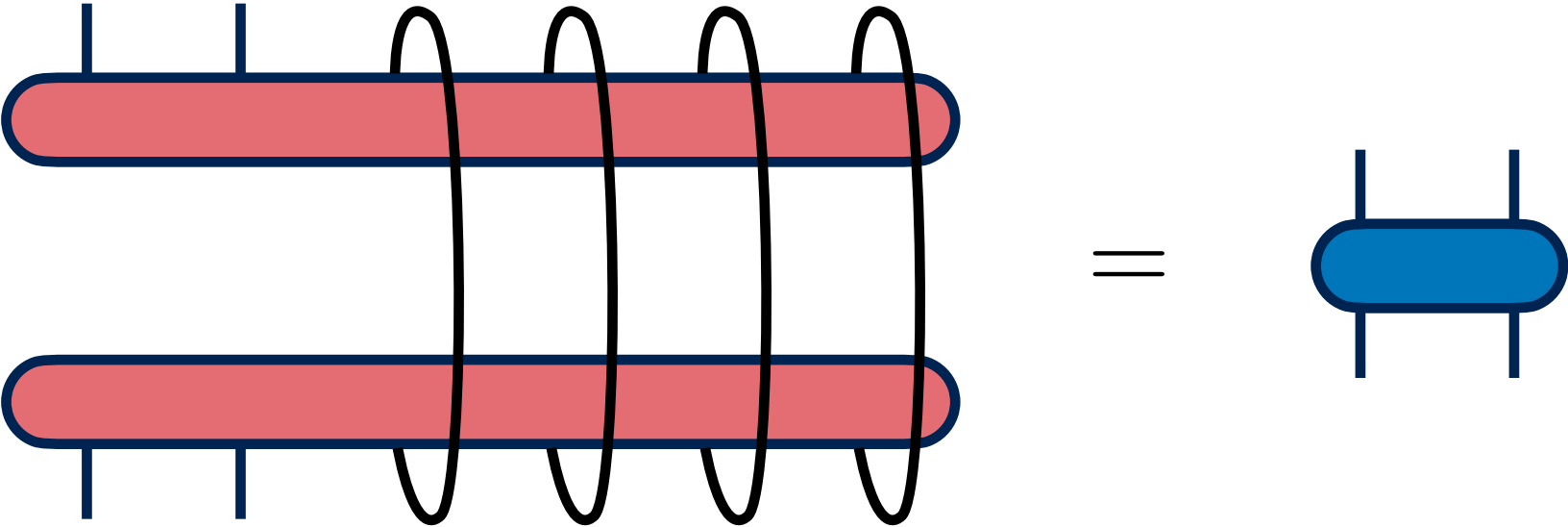
$$\approx \text{Diagram 2}$$

Diagram 2: A horizontal chain of six blue circles connected by horizontal lines. Each circle has a vertical line extending upwards to a label $s_1, s_2, s_3, s_4, s_5, s_6$ respectively.

Density Matrix MPS Learning Algorithm

Algorithm is *density matrix algorithm* for compressing tensor to MPS form (closely related to TT-SVD and DMRG)

Compute Hermitian, positive semi-definite matrix

$$\rho_{12} =$$


The diagram illustrates the contraction of a density matrix tensor into an MPS form. On the left, a density matrix tensor is represented by two horizontal red rounded rectangles. Each rectangle has four vertical lines extending from its top and bottom edges. Four black loops connect the top and bottom lines of the two rectangles, representing the contraction of the density matrix. To the right of this is an equals sign, followed by a single blue rounded rectangle with four vertical lines extending from its top and bottom edges, representing the resulting MPS form.

Density Matrix MPS Learning Algorithm

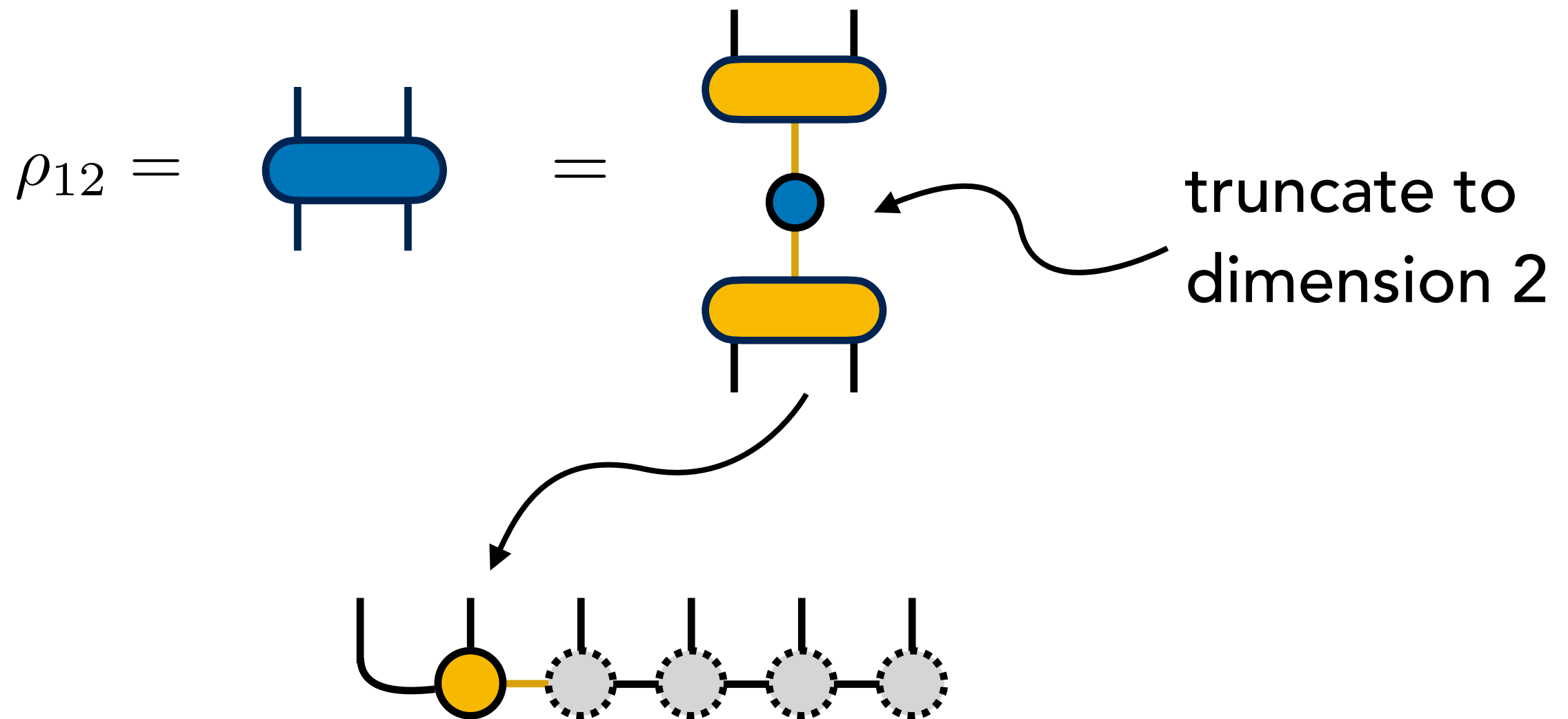
Can compute density matrix with
summed form of ψ inserted

$$\rho_{12} = \sum_{j,j' \in \mathcal{T}} \text{Diagram} = \text{Diagram}$$

The diagram on the left represents the density matrix ρ_{12} as a sum over indices $j, j' \in \mathcal{T}$. It consists of two rows of six red circles. The top row circles have vertical lines extending upwards, and the bottom row circles have vertical lines extending downwards. Pairs of circles in the top and bottom rows are connected by vertical arcs, forming three loops. The diagram on the right is a single blue rounded rectangle with four vertical lines (two on the left, two on the right) representing the density matrix.

Density Matrix MPS Learning Algorithm

Diagonalizing ρ_{12} gives first & second MPS tensors



Density Matrix MPS Learning Algorithm

Use diagonalizing unitary to compress data

$$\psi^{s_1 s_2 s_3 s_4 s_5 s_6} = \frac{1}{\sqrt{N}} \sum_{j \in \mathcal{T}} \text{Diagram 1}$$
$$= \frac{1}{\sqrt{N}} \sum_{j \in \mathcal{T}} \text{Diagram 2}$$

The diagrams illustrate the compression of a 6-site density matrix MPS tensor into a single-site tensor. Diagram 1 shows a yellow rounded rectangle (representing a unitary) connected to two red circles (representing the original sites). Diagram 2 shows a blue circle (representing the compressed site) connected to one red circle (representing the original site).

& recursively repeat algorithm to generate all MPS tensors

Density Matrix MPS Learning Algorithm

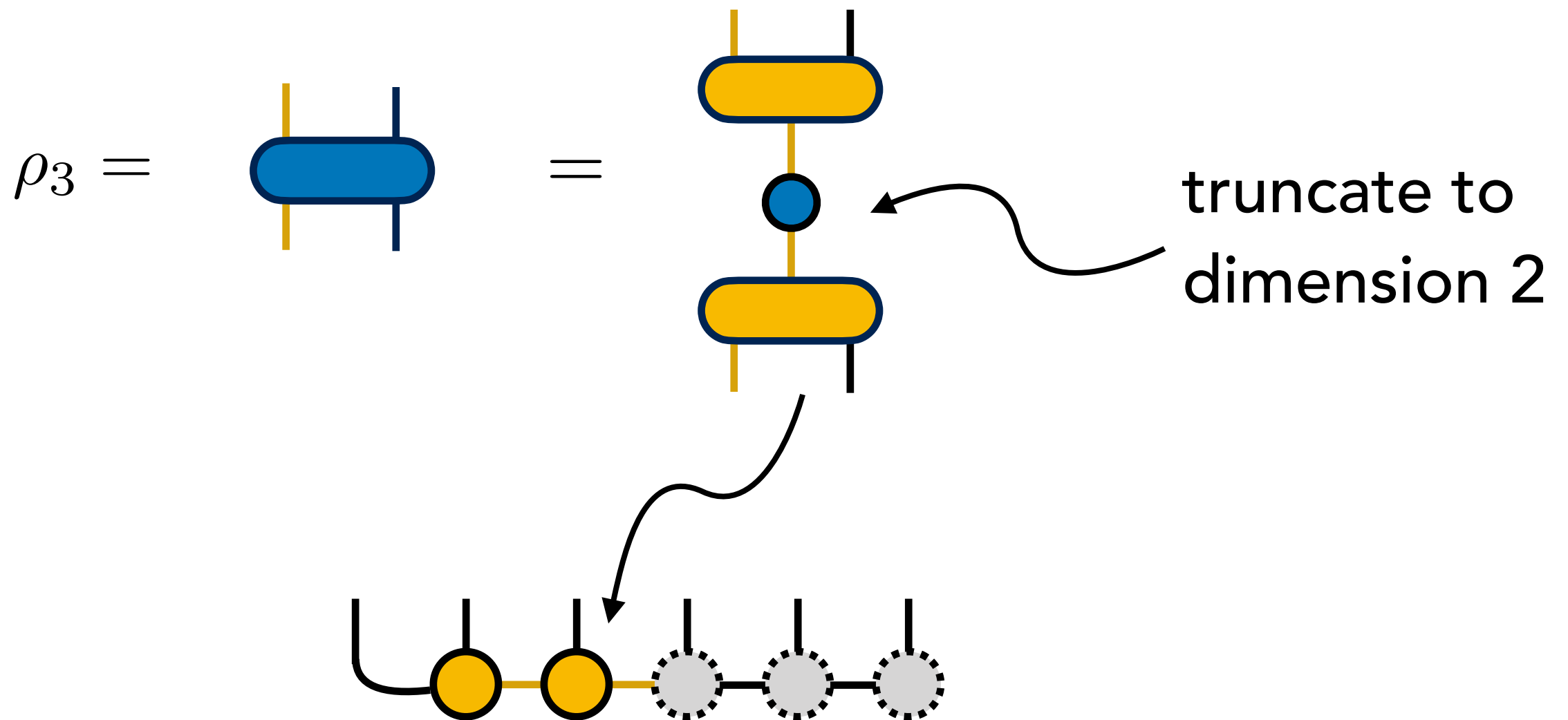
Computing next density matrix ρ_3

$$\rho_3 = \sum_{j,j' \in \mathcal{T}} \text{Diagram} = \text{Diagram}$$

The diagram illustrates the computation of the next density matrix ρ_3 in the Density Matrix MPS Learning Algorithm. It shows a summation over indices $j, j' \in \mathcal{T}$ of a tensor network contraction. The contraction involves a sequence of tensors: two blue circles (representing the initial state), followed by three pairs of red circles (representing the evolution and measurement process). The red circles are connected by vertical lines, indicating the contraction of indices. The result is a single blue circle, representing the final density matrix ρ_3 .

Density Matrix MPS Learning Algorithm

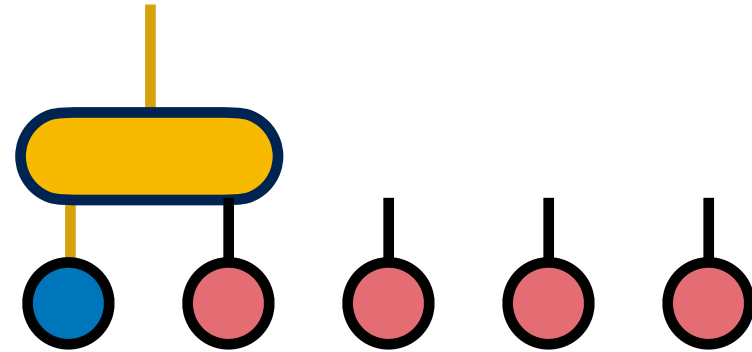
Diagonalizing ρ_3 gives third MPS tensor



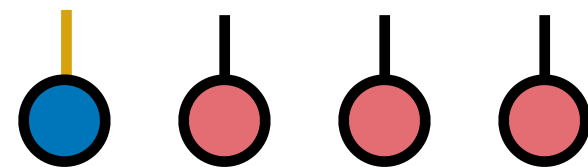
Density Matrix MPS Learning Algorithm

Use diagonalizing unitary to compress data

$$\psi^{s_1 s_2 s_3 s_4 s_5 s_6} = \frac{1}{\sqrt{N}} \sum_{j \in \mathcal{T}}$$



$$= \frac{1}{\sqrt{N}} \sum_{j \in \mathcal{T}}$$



& recursively repeat algorithm to generate all MPS tensors

Density Matrix MPS Learning Algorithm

Deterministic algorithm for generative modeling

Truncation of density matrix determines
bond dimension of MPS

Intentionally make 'error' from overfitted model
in order to generalize

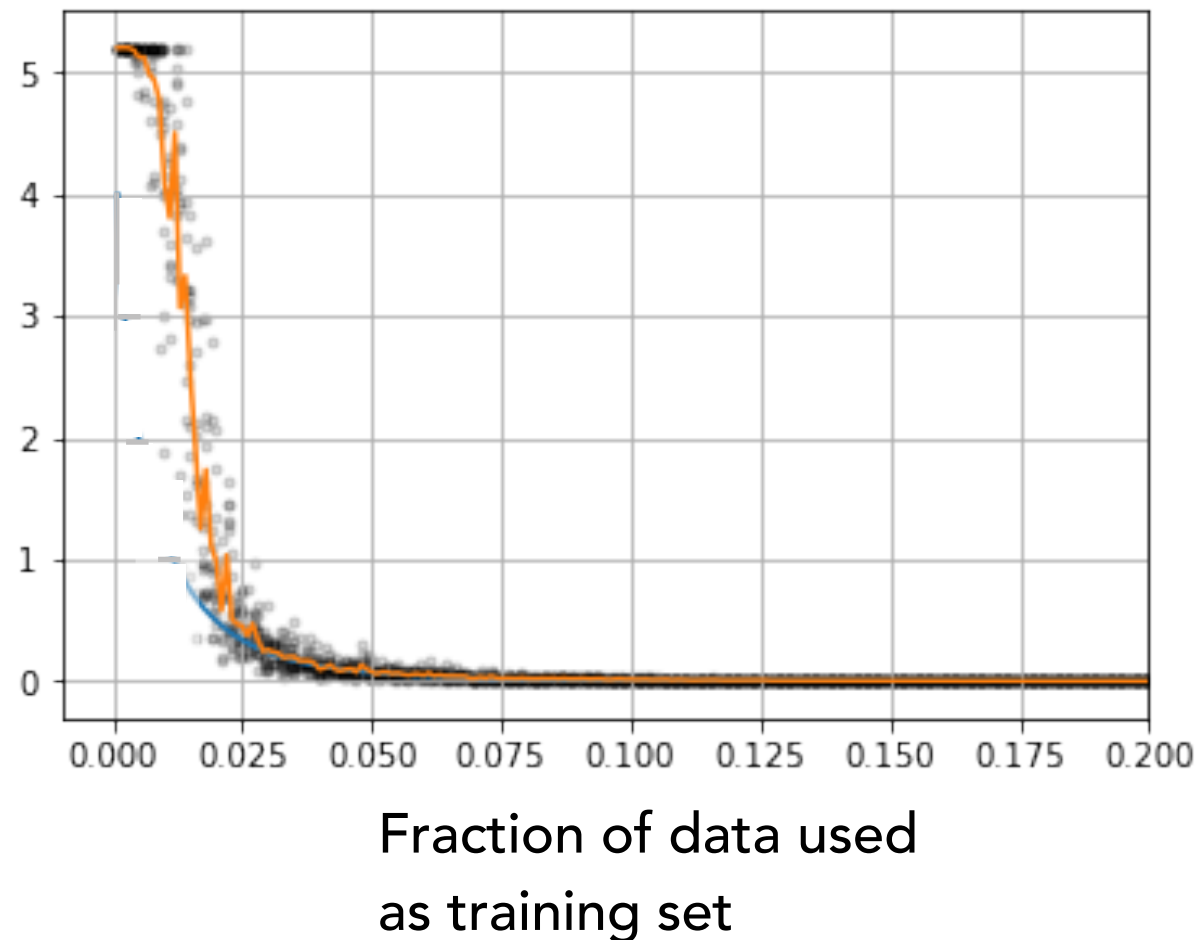
compression = learning

Density Matrix MPS Learning Algorithm

How well does it work?

L=16 results:

Bhattacharyya distance
to true distribution



— experimental avg.

Bhattacharyya distance:

$$D_B = -\ln \left(\sum_x \sqrt{p(x)q(x)} \right)$$

can evaluate exactly for
parity data set

Sketch of the theoretical prediction



Tai-Danae Bradley



John Terilla

Observe that density matrices have a block-diagonal form (blocks for even & odd parity 'sectors')

$$\rho_{12} = \sum_{j,j' \in \mathcal{T}} \begin{array}{c} \text{Diagram with 6 red circles in two rows. The first two columns have single circles. The next four columns have pairs of circles connected by vertical loops.} \end{array} = \begin{array}{c} \text{Diagram of a blue rounded rectangle with four vertical lines extending from its top and bottom edges.} \end{array}$$

$$\rho_{12} = \begin{bmatrix} d_1 & s_e & & \\ s_e & d_2 & & \\ & & d_3 & s_o \\ & & s_o & d_4 \end{bmatrix}$$

Reason is trace over bits 3,4,...,N pairs identical suffixes (& identical parity), otherwise zero

Assume two most dominant eigenvectors come from the two blocks (one from each)

$$\rho_{12} = \begin{array}{c} \begin{array}{cc} & \begin{array}{cc} 00 & 11 \end{array} \\ \begin{array}{c} 00 \\ 11 \end{array} & \begin{bmatrix} d_1 & s_e \\ s_e & d_2 \end{bmatrix} \end{array} \begin{array}{c} \begin{array}{cc} 01 & 10 \end{array} \\ \begin{bmatrix} d_3 & s_o \\ s_o & d_4 \end{bmatrix} \end{array} \end{array}$$

$$|E_2\rangle = \cos \theta_2 |00\rangle + \sin \theta_2 |11\rangle$$

$$|O_2\rangle = \cos \phi_2 |01\rangle + \sin \phi_2 |10\rangle$$

(Notation $|\mathbf{v}\rangle$ means a vector $\mathbf{v}$)

Easily find expressions for angles in terms of entries of density matrix

$$\rho_{12} = \begin{array}{c} \begin{array}{cc} & \begin{array}{cc} 00 & 11 \end{array} \\ \begin{array}{c} 00 \\ 11 \\ 01 \\ 10 \end{array} & \begin{bmatrix} d_1 & s_e & & \\ s_e & d_2 & & \\ & & d_3 & s_o \\ & & s_o & d_4 \end{bmatrix} \end{array}$$

$$|E_2\rangle = \cos \theta_2 |00\rangle + \sin \theta_2 |11\rangle$$

$$|O_2\rangle = \cos \phi_2 |01\rangle + \sin \phi_2 |10\rangle$$

$$\theta_2 = \arctan \left(\frac{2s_e}{\sqrt{G_e^2 + 4s_e^2} + G_e} \right) \quad \phi_2 = \arctan \left(\frac{2s_o}{\sqrt{G_o^2 + 4s_o^2} + G_o} \right)$$

$$G_e = |d_1 - d_2|$$

$$G_o = |d_3 - d_4|$$

$$\rho_{12} = \begin{array}{c} \begin{array}{cc} & \begin{array}{cccc} & 00 & 11 & 01 & 10 \end{array} \\ \begin{array}{c} 00 \\ 11 \\ 01 \\ 10 \end{array} & \begin{bmatrix} d_1 & s_e & & \\ s_e & d_2 & & \\ & & d_3 & s_o \\ & & s_o & d_4 \end{bmatrix} \end{array} \end{array}$$

$$\theta_2 = \arctan \left(\frac{2s_e}{\sqrt{G_e^2 + 4s_e^2} + G_e} \right) \quad \phi_2 = \arctan \left(\frac{2s_o}{\sqrt{G_o^2 + 4s_o^2} + G_o} \right)$$

$$G_e = |d_1 - d_2|$$

$$G_o = |d_3 - d_4|$$

Importantly, if gaps G_e, G_o zero, perfect learning occurs as long as s_e, s_o non-zero

Non-zero gaps are what drive errors in learning
(coming from strings with unique suffixes)

$$\rho_{12} = \begin{array}{c} \begin{matrix} & 00 & 11 & 01 & 10 \end{matrix} \\ \begin{matrix} 00 \\ 11 \\ 01 \\ 10 \end{matrix} \left[\begin{array}{cc} \mathbf{d}_1 & \mathbf{s}_e \\ \mathbf{s}_e & \mathbf{d}_2 \\ & & \mathbf{d}_3 & \mathbf{s}_o \\ & & \mathbf{s}_o & \mathbf{d}_4 \end{array} \right] \end{array}$$

$$\theta_2 = \arctan \left(\frac{2s_e}{\sqrt{G_e^2 + 4s_e^2} + G_e} \right) \quad \phi_2 = \arctan \left(\frac{2s_o}{\sqrt{G_o^2 + 4s_o^2} + G_o} \right)$$

$$G_e = |d_1 - d_2|$$

$$G_o = |d_3 - d_4|$$

Quantities d_1, d_2, d_3, d_4 are frequency of observing a certain prefix (00,11,01,10). So "gaps" are from imbalance in symmetry-related training examples.

Use combinatorics of even-parity data set (taking a finite fraction as training set)

Determine facts such as:

$s_e =$ number bitstrings with even prefix (00,11)
having same suffix

$$\mathbb{E}[s_e] = \frac{f \cdot |\mathcal{T}|}{4}$$

$$\mathbb{E}[G_e] = \sum_{d_1=0}^r |2d_1 - r| \frac{\binom{n}{d_1} \binom{n}{r-d_1}}{\binom{2n}{r}}$$

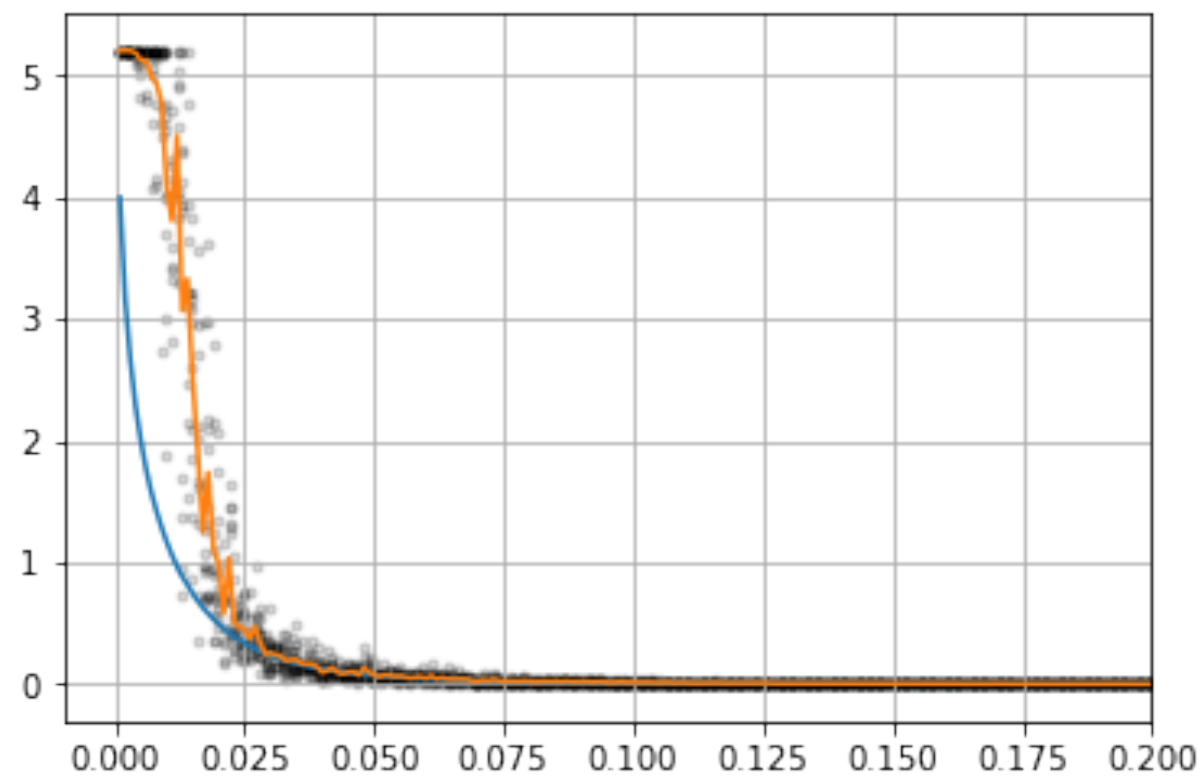
$$r = |\mathcal{T}|/2$$

$$n = 2^{L-3}$$

From combinatorial formulas, can estimate
typical MPS tensors as function of training set fraction

Encouraging agreement with experimental results ($L=16$)

Bhattacharyya distance
to true distribution



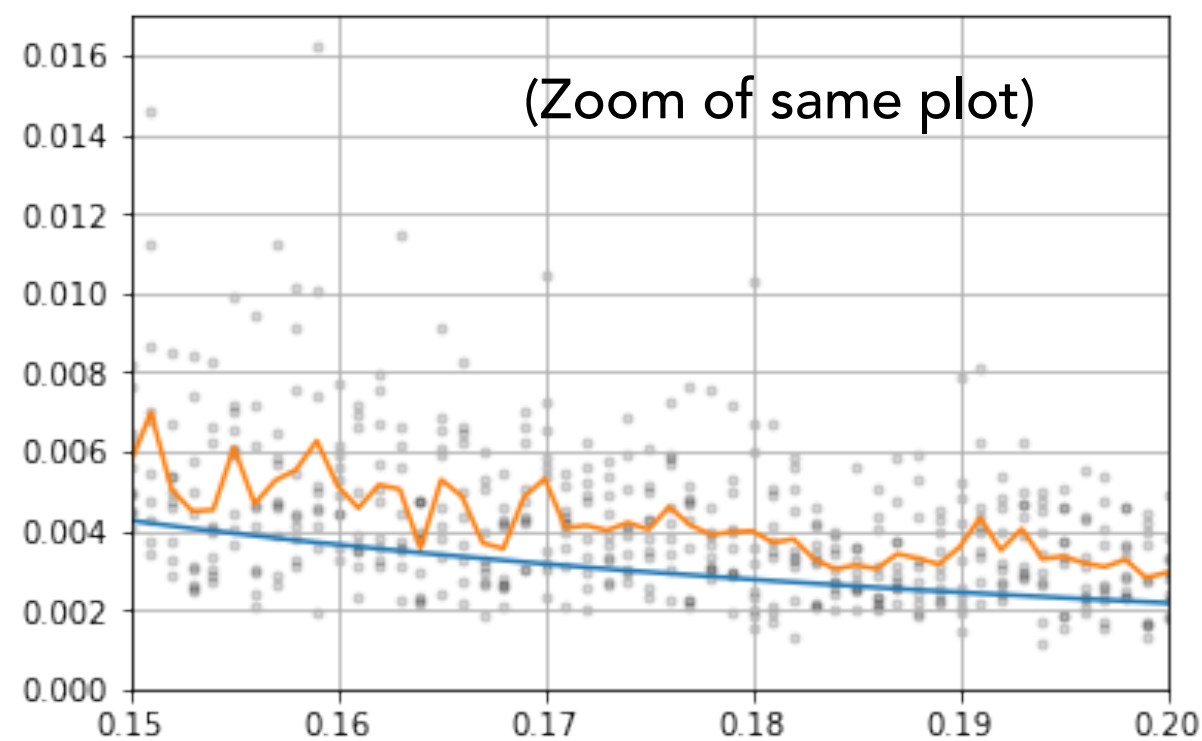
— experimental avg.
— theory prediction

Fraction of data used
as training set

From combinatorial formulas, can estimate
typical MPS tensors as function of training set fraction

Encouraging agreement with experimental results ($L=16$)

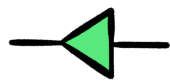
Bhattacharyya distance
to true distribution



— experimental avg.
— theory prediction

Fraction of data used
as training set

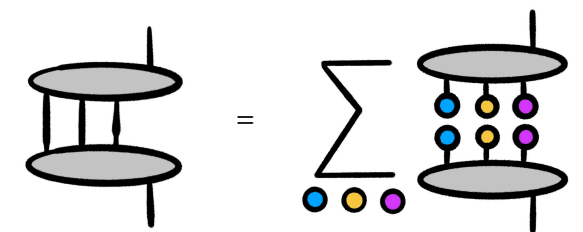
For an expanded discussion, see Tai-Danae's recent thesis *"At the Interface of Algebra and Statistics"* [arxiv:2004.05631](https://arxiv.org/abs/2004.05631)



An **isometric embedding** U is a linear map from a space V to a space W of larger dimension that preserves the lengths of vectors. Such a map satisfies $U^\dagger U = \text{id}_V$ but $UU^\dagger \neq \text{id}_W$. In words,



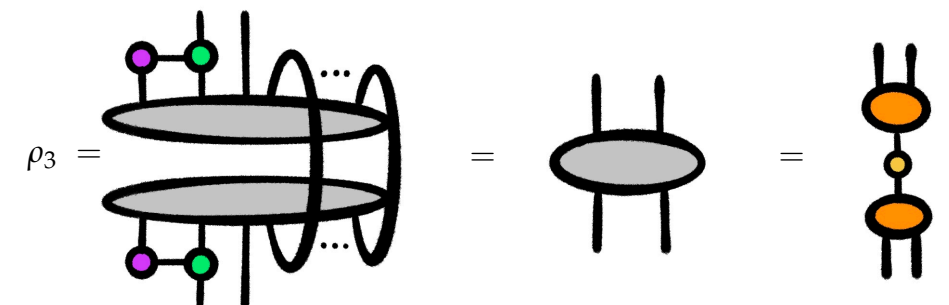
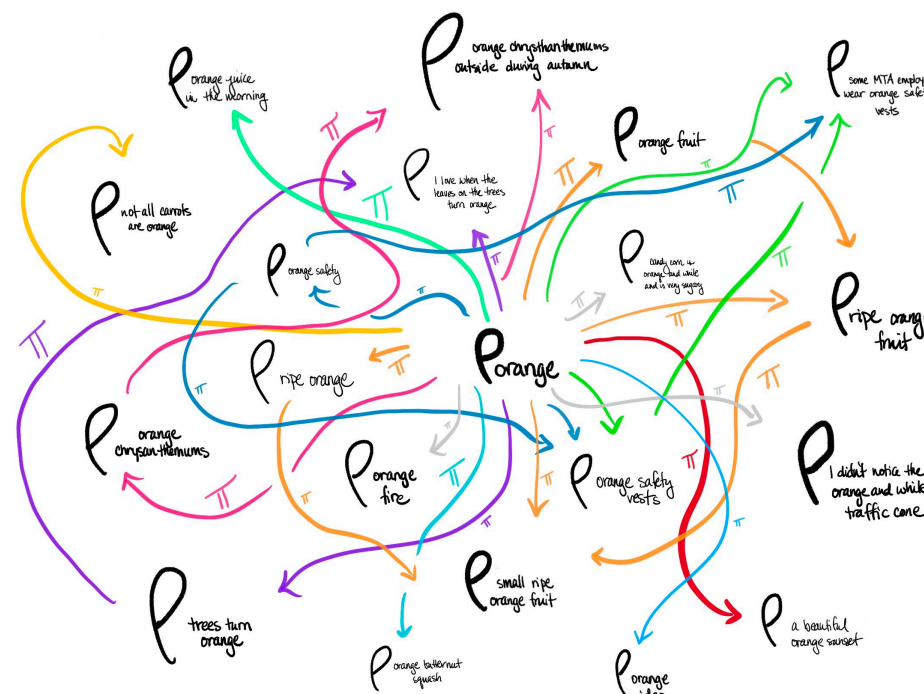
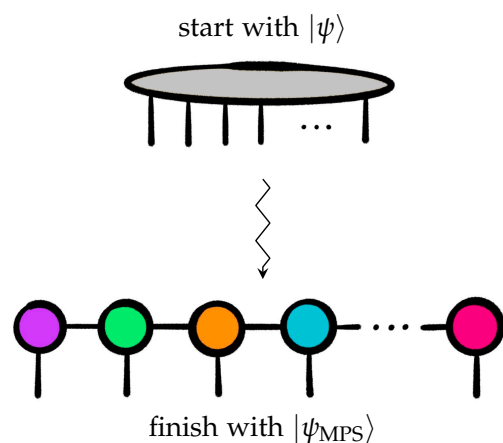
Tai-Danae Bradley
 @math3ma (on Twitter)



In symbols,

$$\rho_Y = \sum_{(a,b,c) \in A \times B \times C} M|abc\rangle \langle abc|M^\dagger$$

where we are using the fact that $A_i|\psi\rangle = M|x_i\rangle$ for each i , as remarked in the main text.



Future Directions to Pursue

- Develop theoretical prediction into rigorous, probabilistic bounds
- Apply this algorithm & theory for other synthetic data
- Find ways to estimate, just from summary stats of data, how likely learning is to succeed for what complexity of model

Summary

- Tensor networks can be used as machine learning models with interesting training algorithms, offering theoretical insights
- Can apply to variety of tasks with good results, sometimes state-of-the-art
- Importantly, opportunity to investigate deeper questions about learning from data

Recommended resource for learning about machine learning:



Physics Reports
Volume 810, 30 May 2019, Pages 1-124



A high-bias, low-variance introduction to Machine Learning for physicists

Pankaj Mehta ^a✉, Marin Bukov ^b✉, Ching-Hao Wang ^a, Alexandre G.R. Day ^a, Clint Richardson ^a, Charles K. Fisher ^c, David J. Schwab ^d

Show more ✓

<https://doi.org/10.1016/j.physrep.2019.03.001>

Under a Creative Commons [license](#)

[Get rights and content](#)

[open access](#)

<https://www.sciencedirect.com/science/article/pii/S0370157319300766>