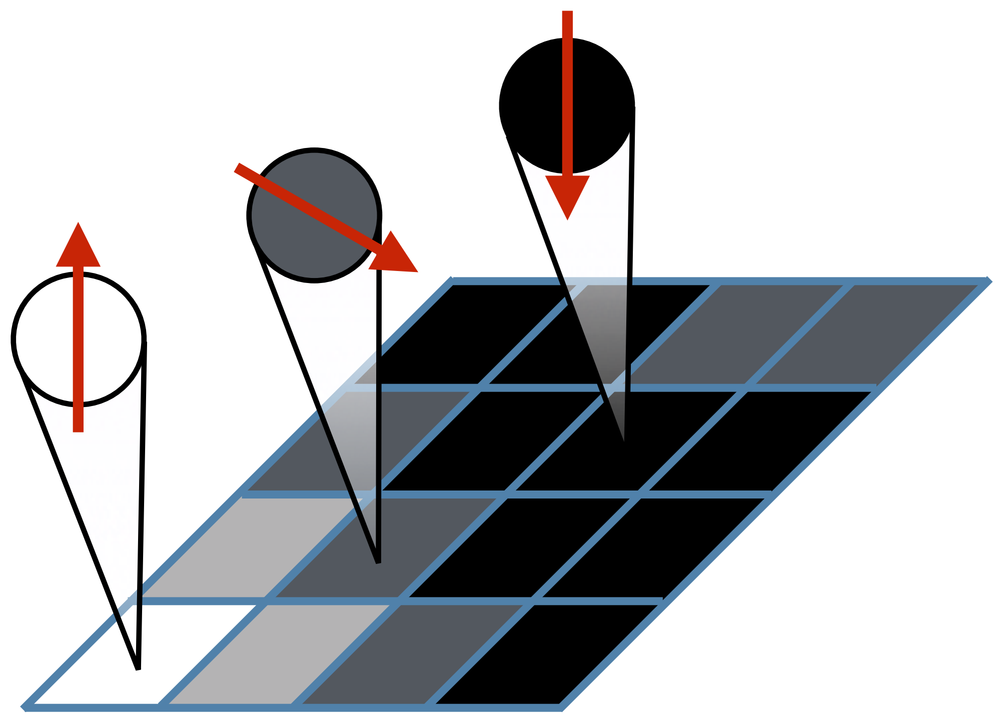
Machine Learning with Quantum-Inspired Tensor Networks



E.M. Stoudenmire and David J. Schwab

Advances in Neural Information Processing **29**

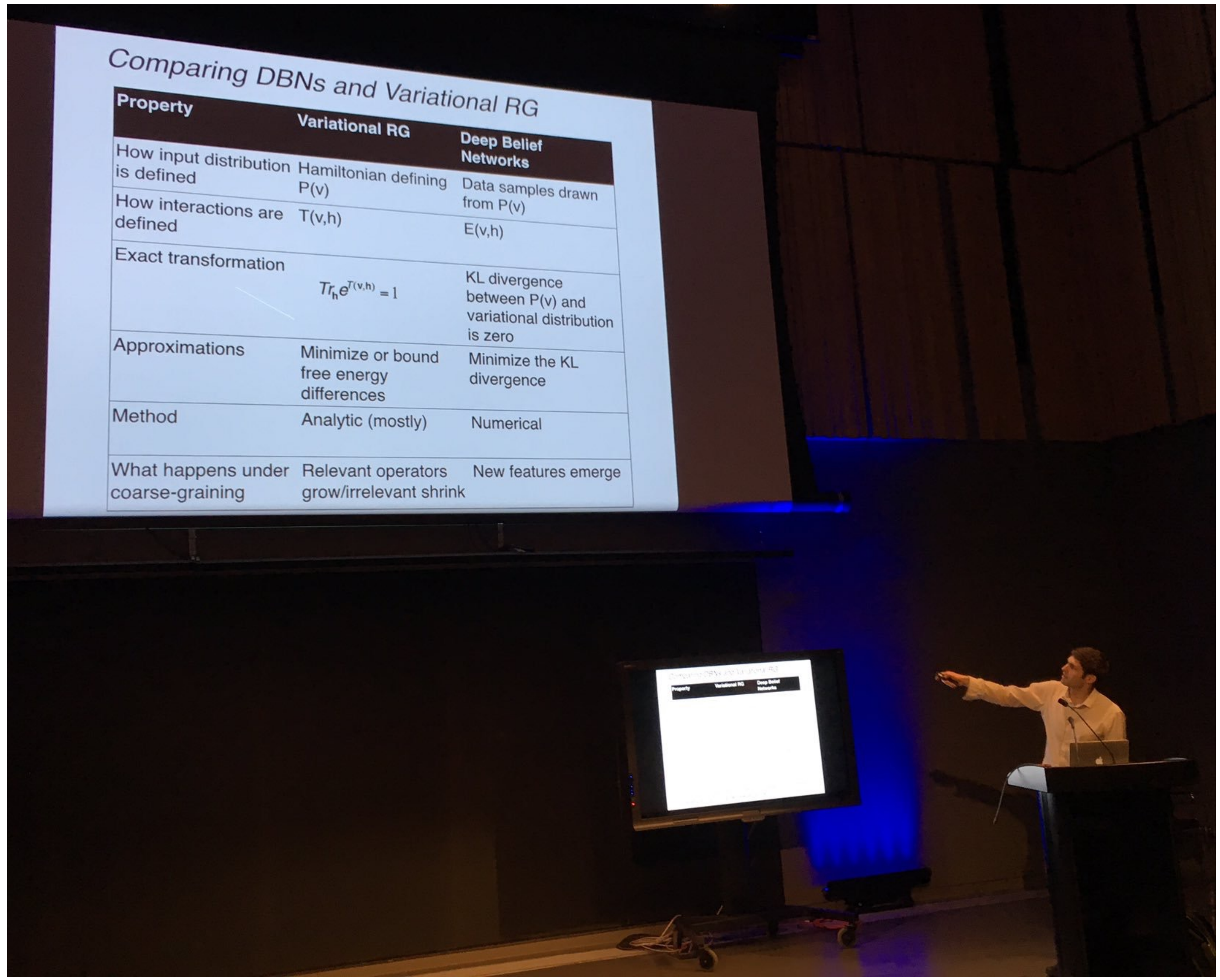
arxiv:1605.05775

RIKEN AICS - Mar 2017

UCIrvine

SIMONS FOUNDATION

Collaboration with **David J. Schwab**, Northwestern and CUNY Graduate Center



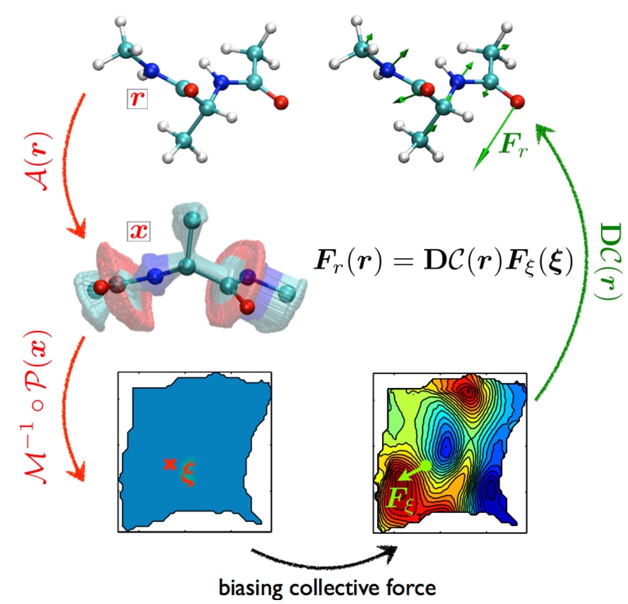
Quantum Machine Learning, Perimeter Institute, Aug 2016

Exciting time for machine learning

Language Processing

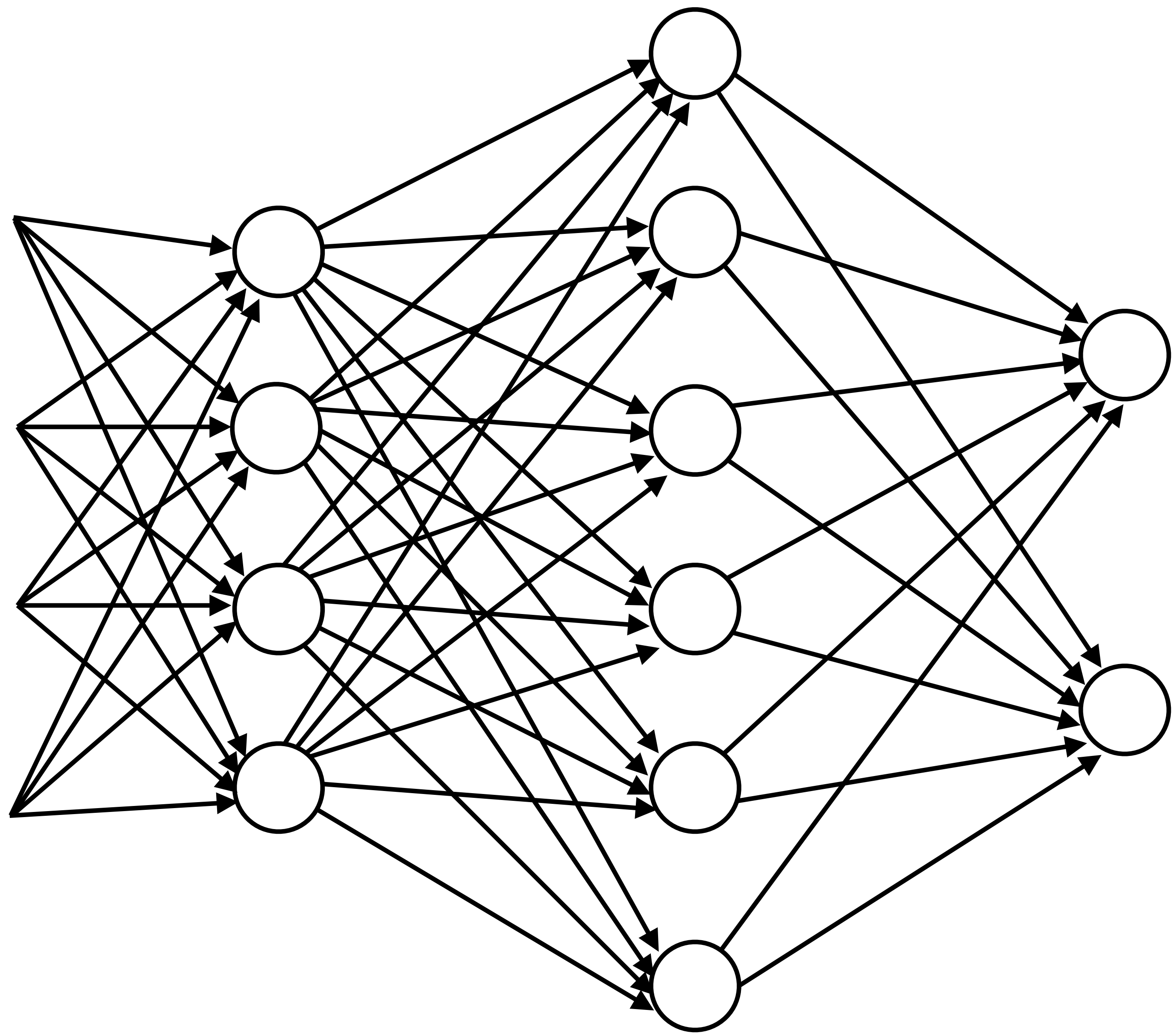
Self-driving cars

Medicine



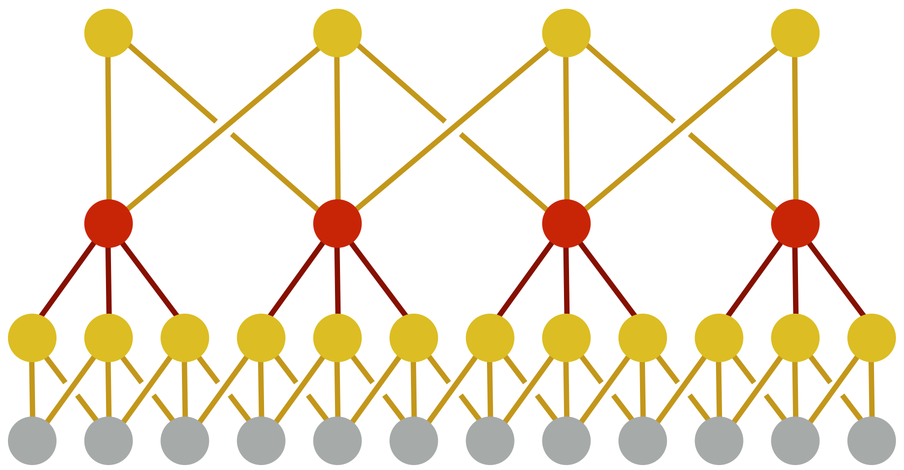
Materials Science / Chemistry

Progress in neural networks and deep learning

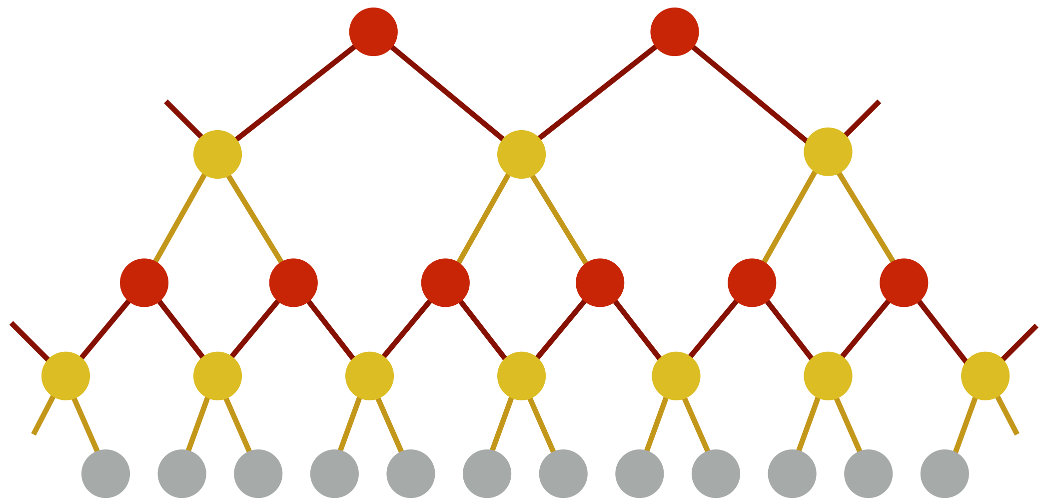


neural network diagram

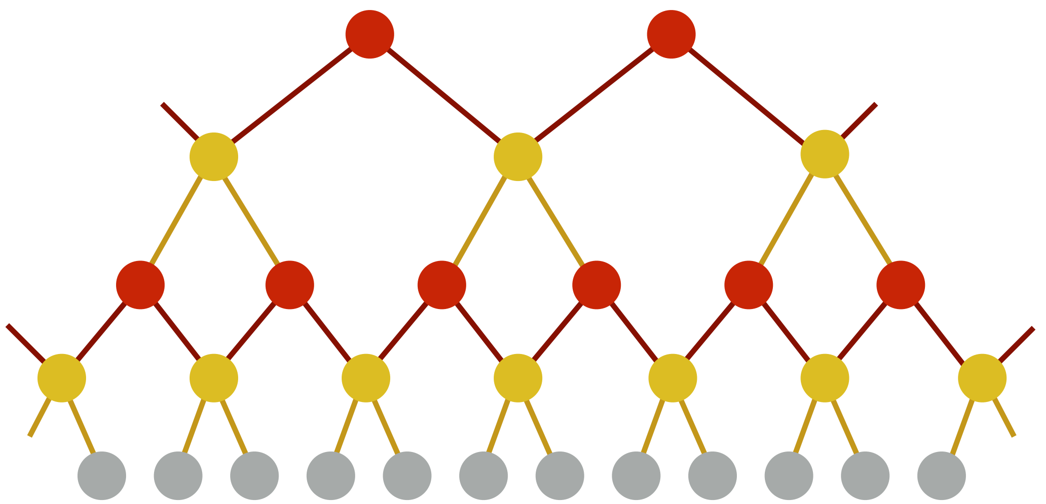
Convolutional neural network



"MERA" tensor network



Are tensor networks useful for machine learning?



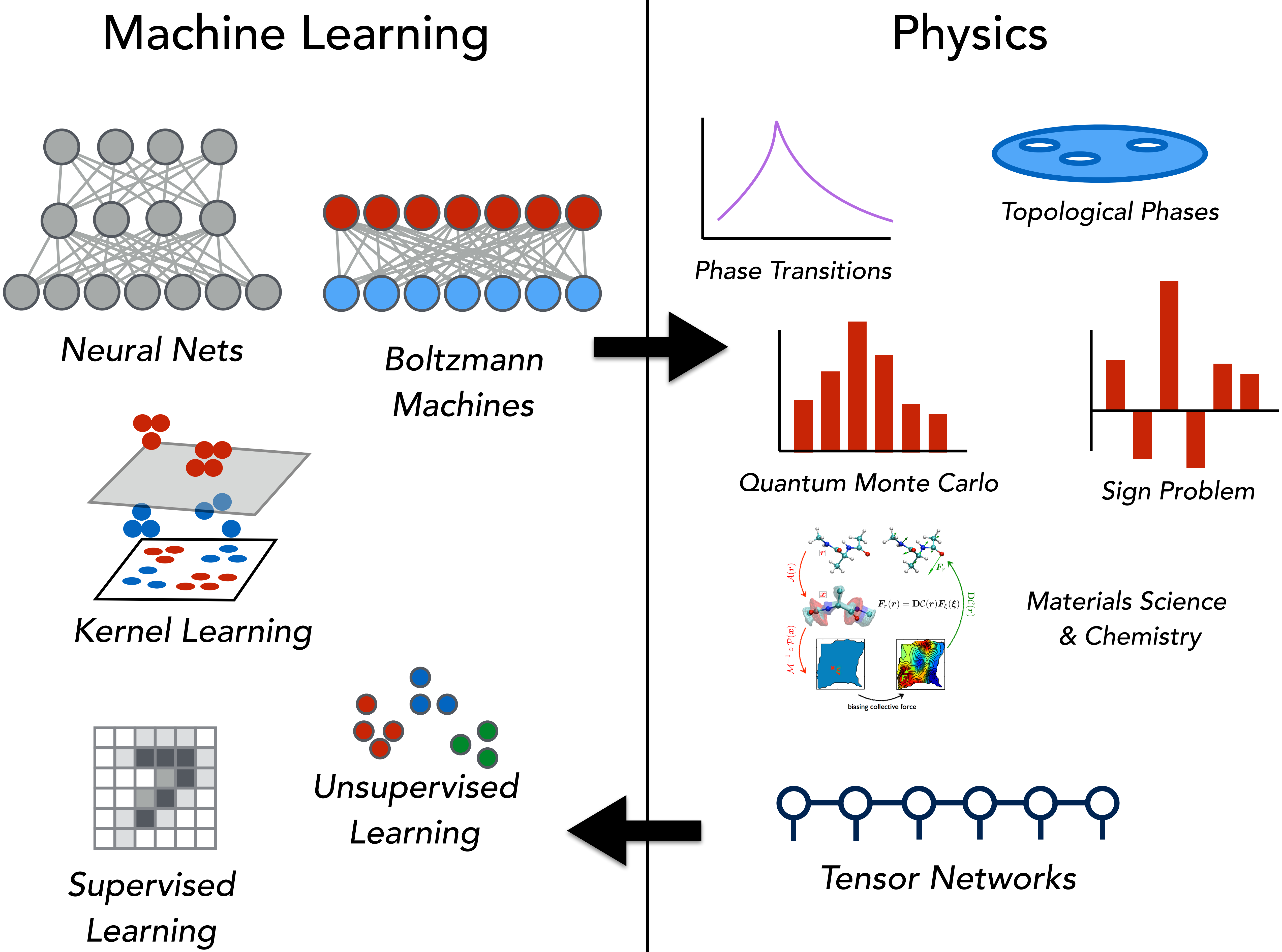
This Talk

Tensor networks fit naturally into *kernel learning*

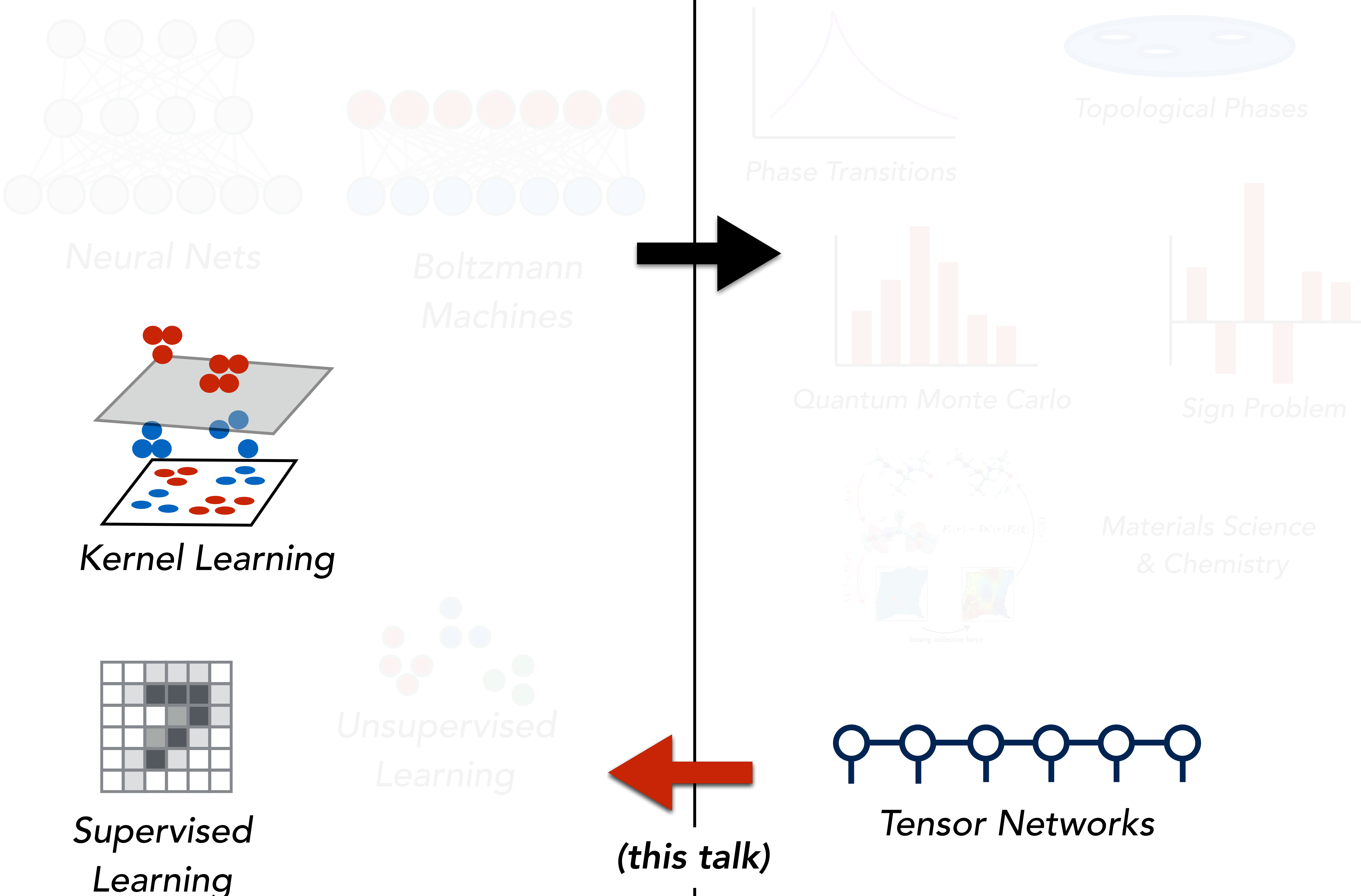
(Also very strong connections to *graphical models*)

Many benefits for learning

- Linear scaling
- Adaptive
- Feature sharing


Machine Learning
Physics
Neural Nets
Boltzmann Machines
Kernel Learning
Supervised Learning
Unsupervised Learning
Phase Transitions
Topological Phases
Quantum Monte Carlo
Sign Problem
Materials Science & Chemistry
biasing collective force
Tensor Networks

Machine Learning | Physics



What are Tensor Networks?

How do tensor networks arise in physics?

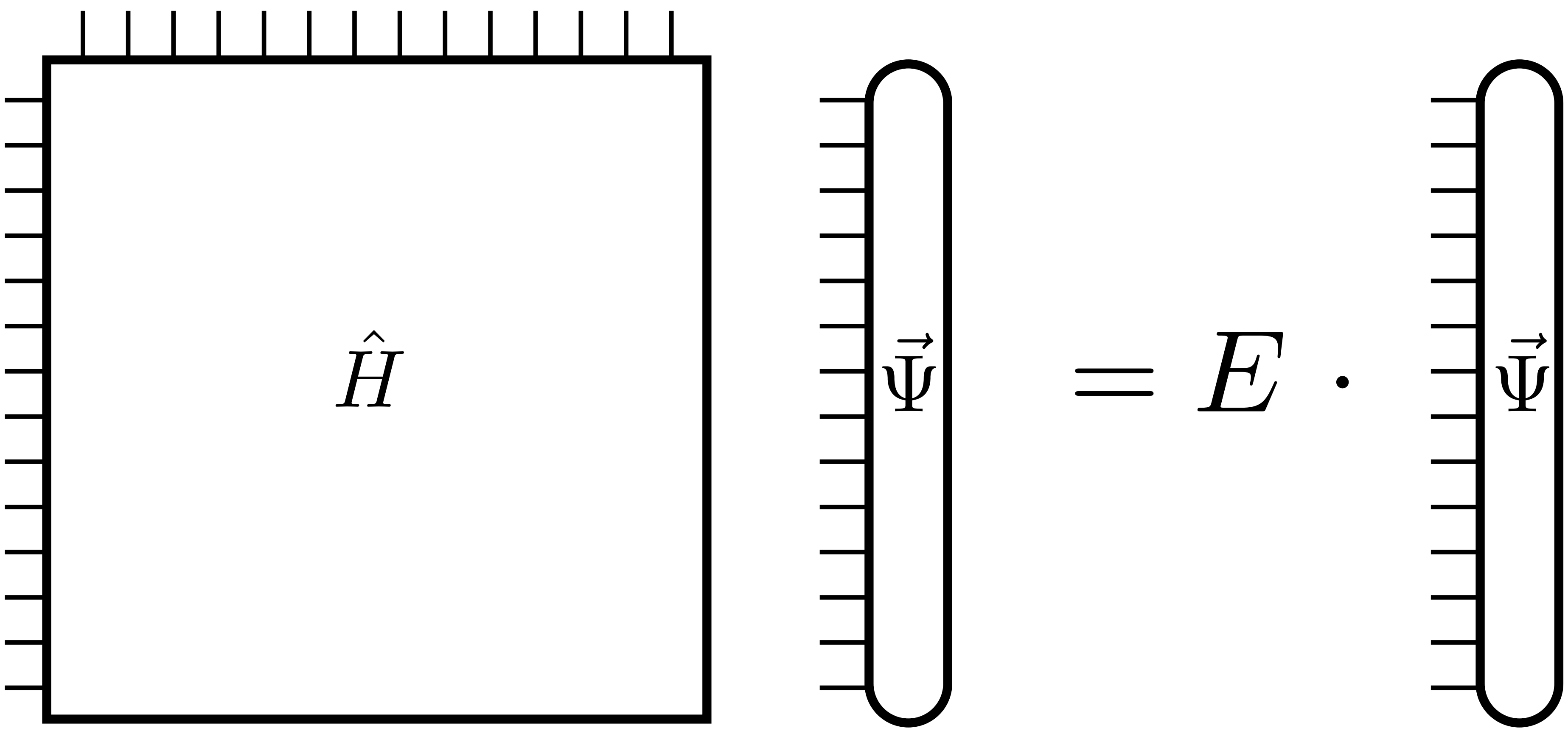
Quantum systems governed by Schrödinger equation:

$$\hat{H}\vec{\Psi} = E\vec{\Psi}$$

It is just an eigenvalue problem.

The *problem* is that $\hat{H}$ is a 2^N x 2^N matrix

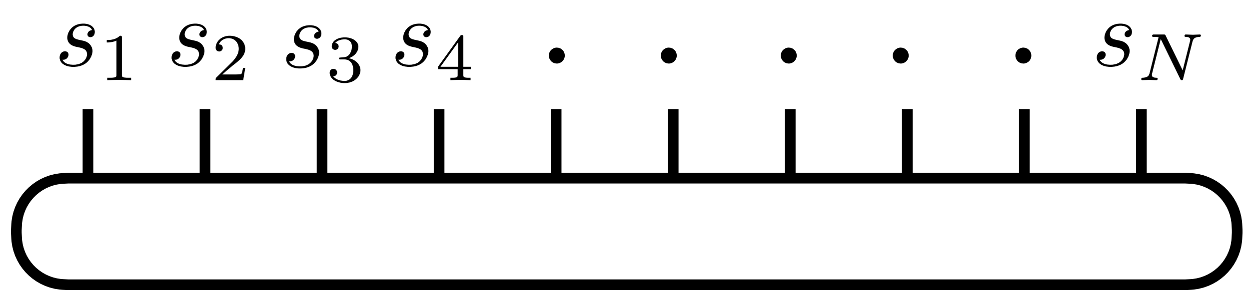
$\Longrightarrow$ wavefunction $\vec{\Psi}$ has 2^N components



Natural to view wavefunction as order-N tensor

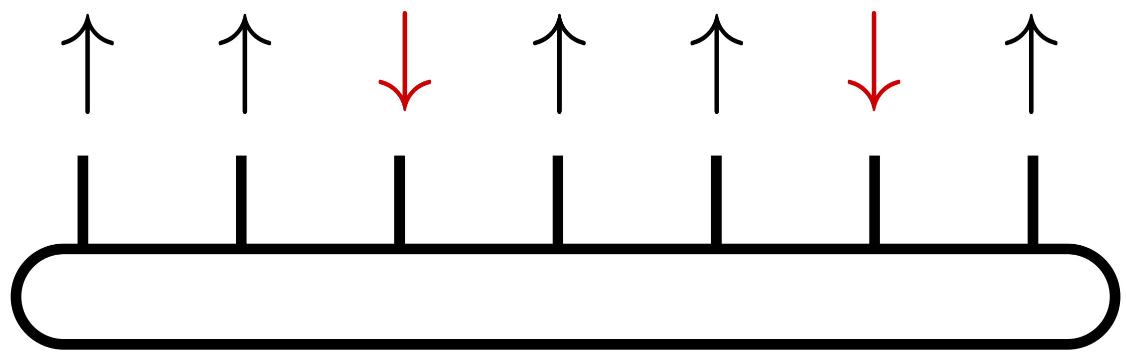
$$|\Psi\rangle = \sum_{\{s\}} \Psi^{s_1 s_2 s_3 \cdots s_N} |s_1 s_2 s_3 \cdots s_N\rangle$$

Natural to view wavefunction as order-N tensor

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

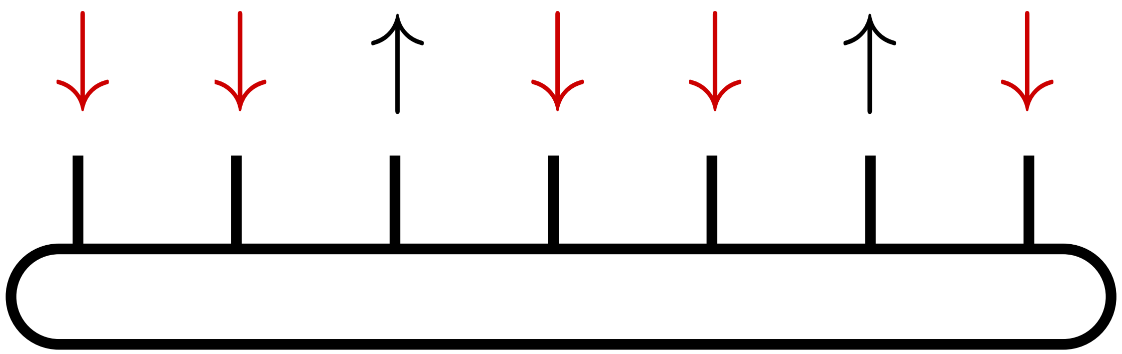
Tensor components related to probabilities of e.g. Ising model spin configurations

$$\Psi^{\uparrow\uparrow\downarrow\uparrow\uparrow\downarrow\uparrow} =$$



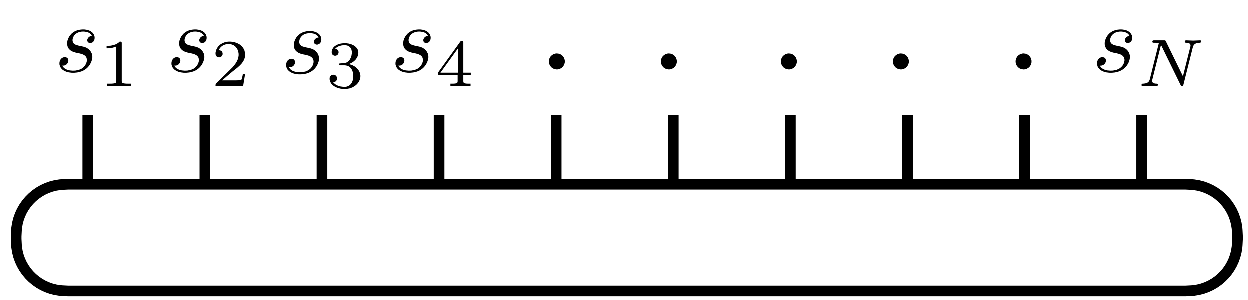
Tensor components related to probabilities of e.g. Ising model spin configurations

$$\Psi^{\downarrow\uparrow\uparrow\downarrow\downarrow\uparrow\downarrow} =$$



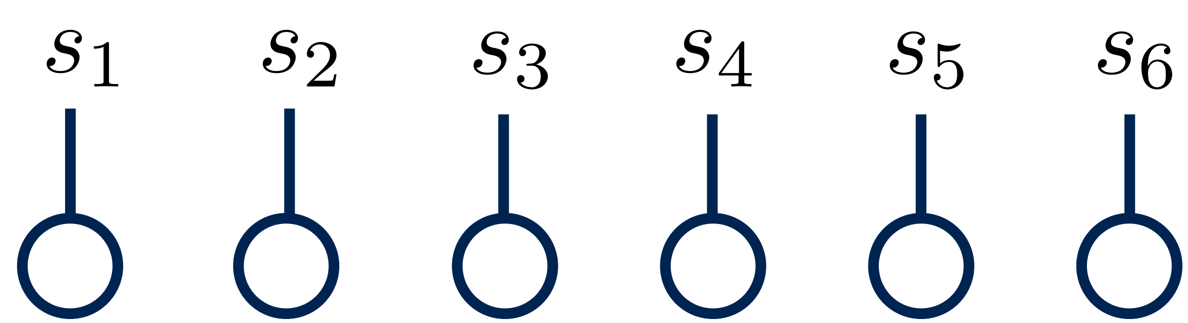
Must find an approximation to this exponential problem

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



Simplest approximation (mean field / rank-1)
Let spins "do their own thing"

$$\Psi^{s_1 s_2 s_3 s_4 s_5 s_6} \simeq \psi^{s_1}\,\psi^{s_2}\,\psi^{s_3}\,\psi^{s_4}\,\psi^{s_5}\,\psi^{s_6}$$

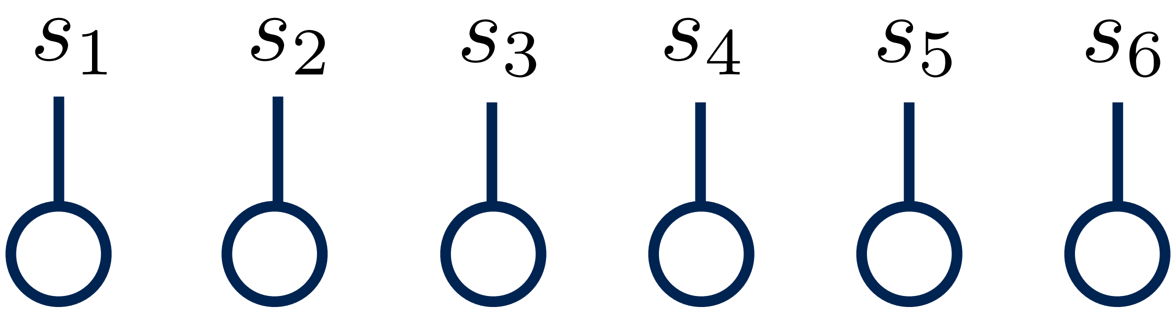


✓ Expected values of individual spins ok

✗ No correlations

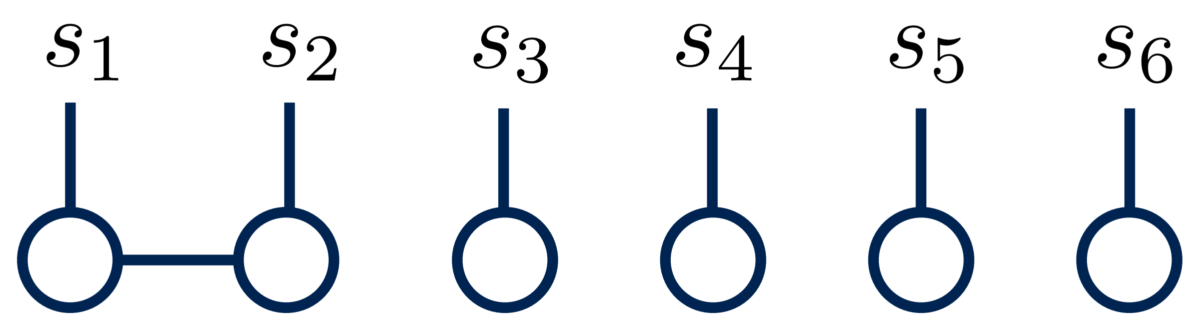
Restore correlations locally

$$\Psi^{s_1 s_2 s_3 s_4 s_5 s_6} \simeq \psi^{s_1}\,\psi^{s_2}\,\psi^{s_3}\,\psi^{s_4}\,\psi^{s_5}\,\psi^{s_6}$$



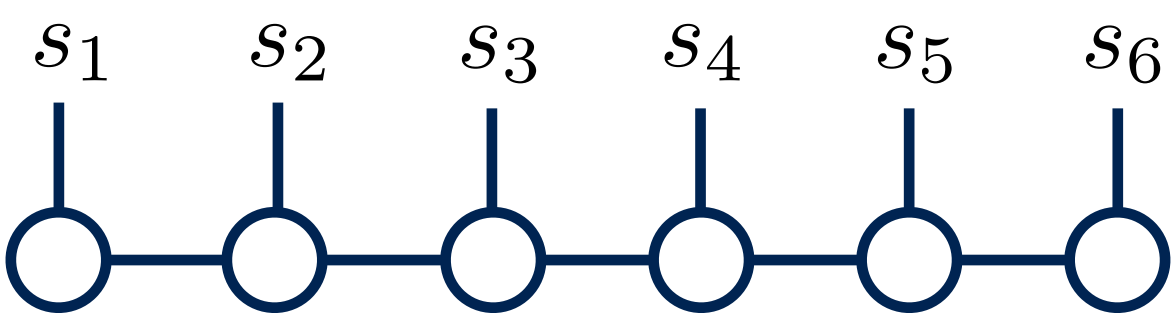
Restore correlations locally

$$\Psi^{s_1 s_2 s_3 s_4 s_5 s_6} \simeq \psi^{s_1}_{i_1}\, \psi^{s_2}_{i_1}\, \psi^{s_3}\, \psi^{s_4}\, \psi^{s_5}\, \psi^{s_6}$$



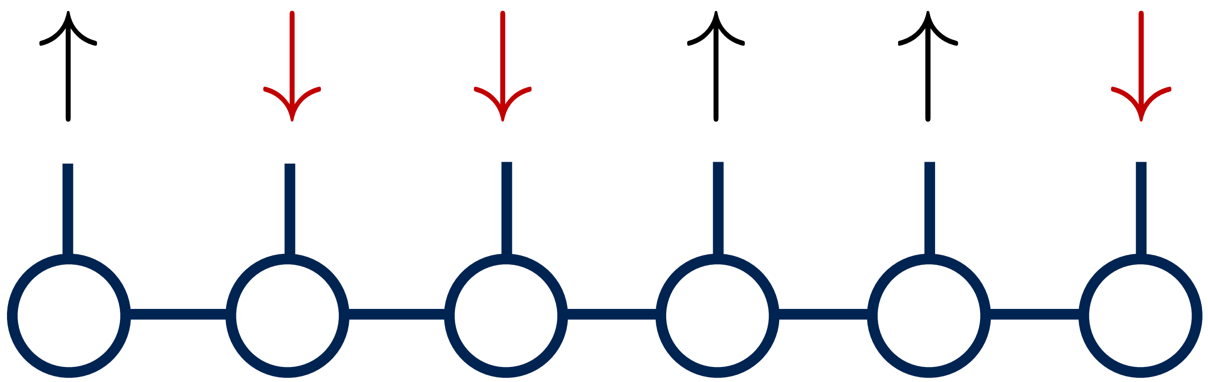
Restore correlations locally

$$\Psi^{s_1 s_2 s_3 s_4 s_5 s_6} \simeq \psi^{s_1}_{i_1} \psi^{s_2}_{i_1 i_2} \psi^{s_3}_{i_2 i_3} \psi^{s_4}_{i_3 i_4} \psi^{s_5}_{i_4 i_5} \psi^{s_6}_{i_5}$$



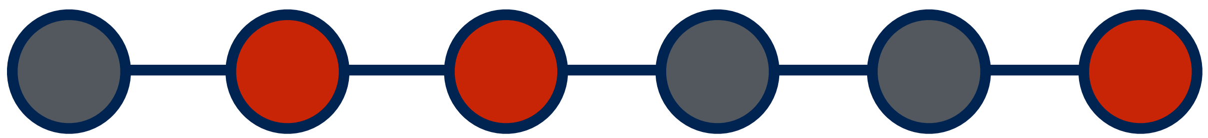
matrix product state (MPS)

- ✓ Local expected values accurate
- ✓ Correlations decay with spatial distance



"Matrix product state" because

retrieving an element $=$ product of matrices



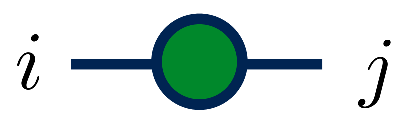
$= \Psi^{\uparrow\downarrow\downarrow\uparrow\uparrow\downarrow}$

"Matrix product state" because

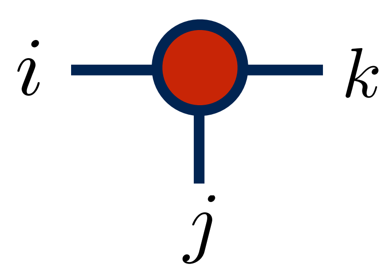
retrieving an element $=$ product of matrices

Tensor diagrams have rigorous meaning

$$v_j$$

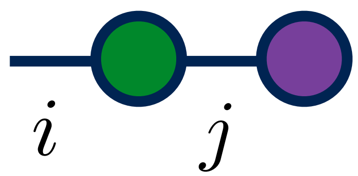


$$M_{ij}$$

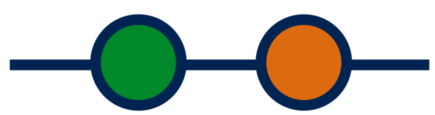


$$T_{ijk}$$

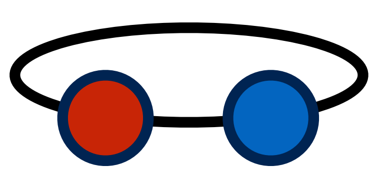
Joining lines implies contraction, can omit names



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

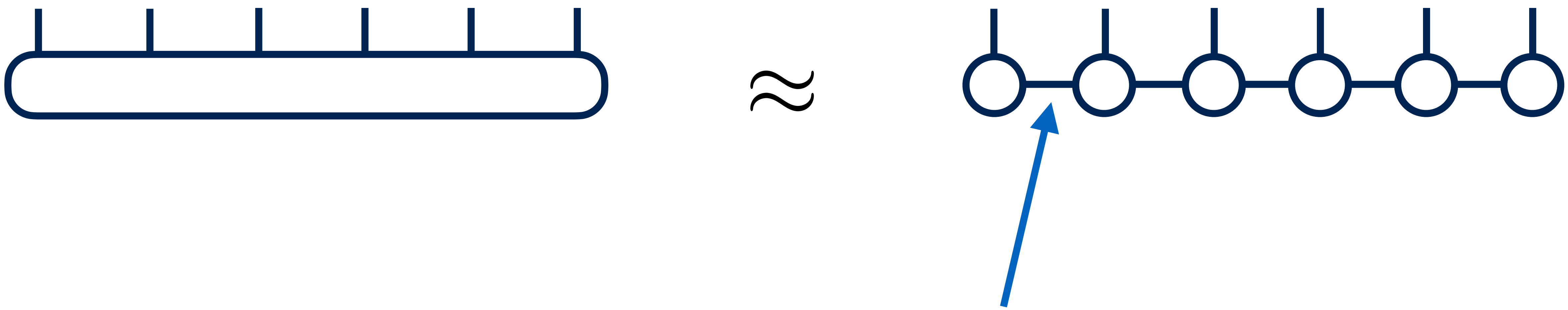


$$\longleftrightarrow \quad A_{ij} B_{jk} = AB$$



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

MPS = matrix product state

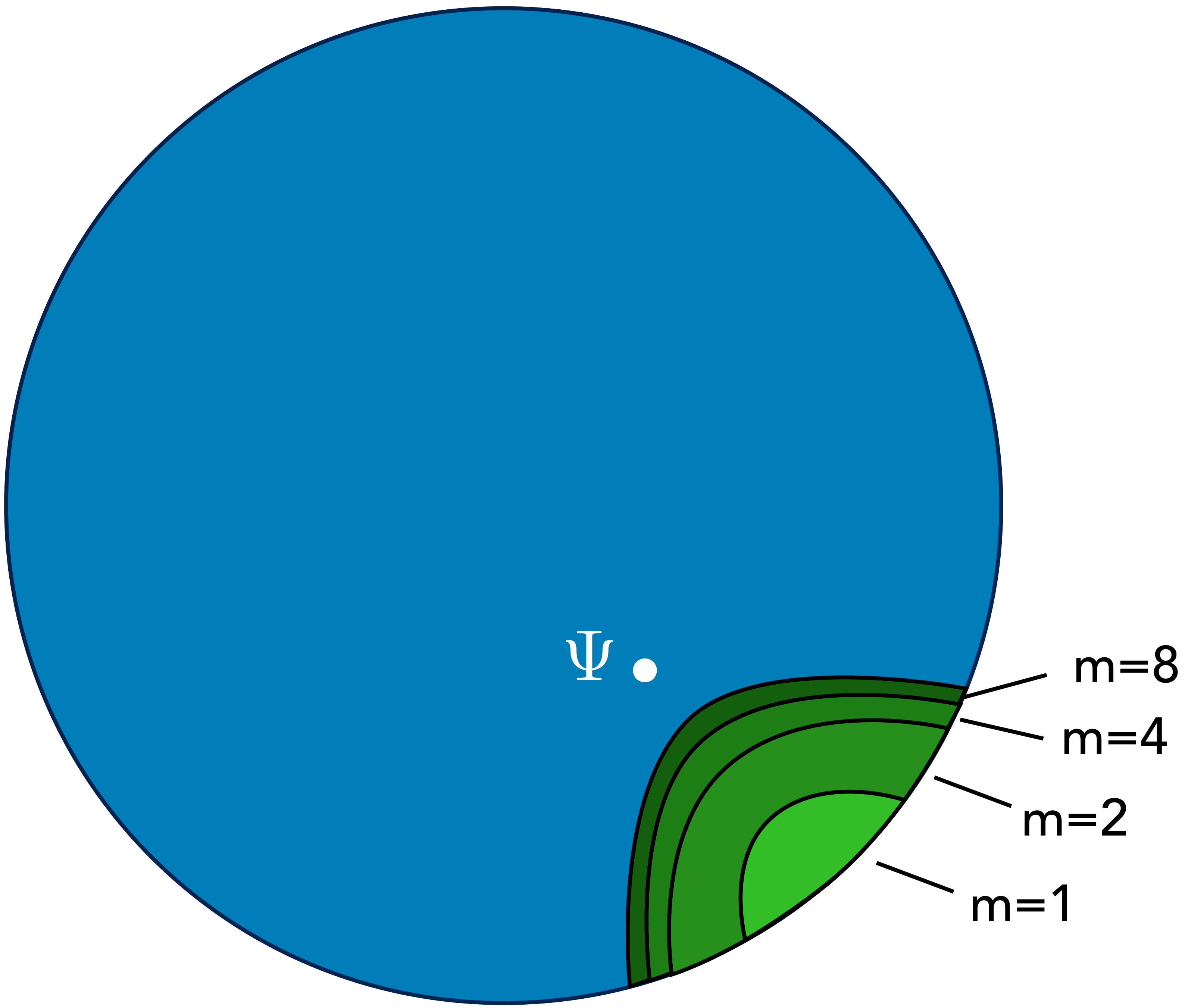


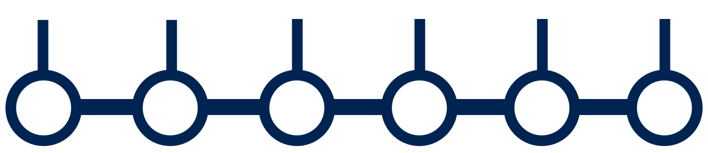
MPS approximation controlled by
bond dimension "m" (like SVD rank)

Compress 2^N parameters into
$N \cdot 2 \cdot m^2$ parameters

$m \sim 2^{\frac{N}{2}}$ can represent any tensor

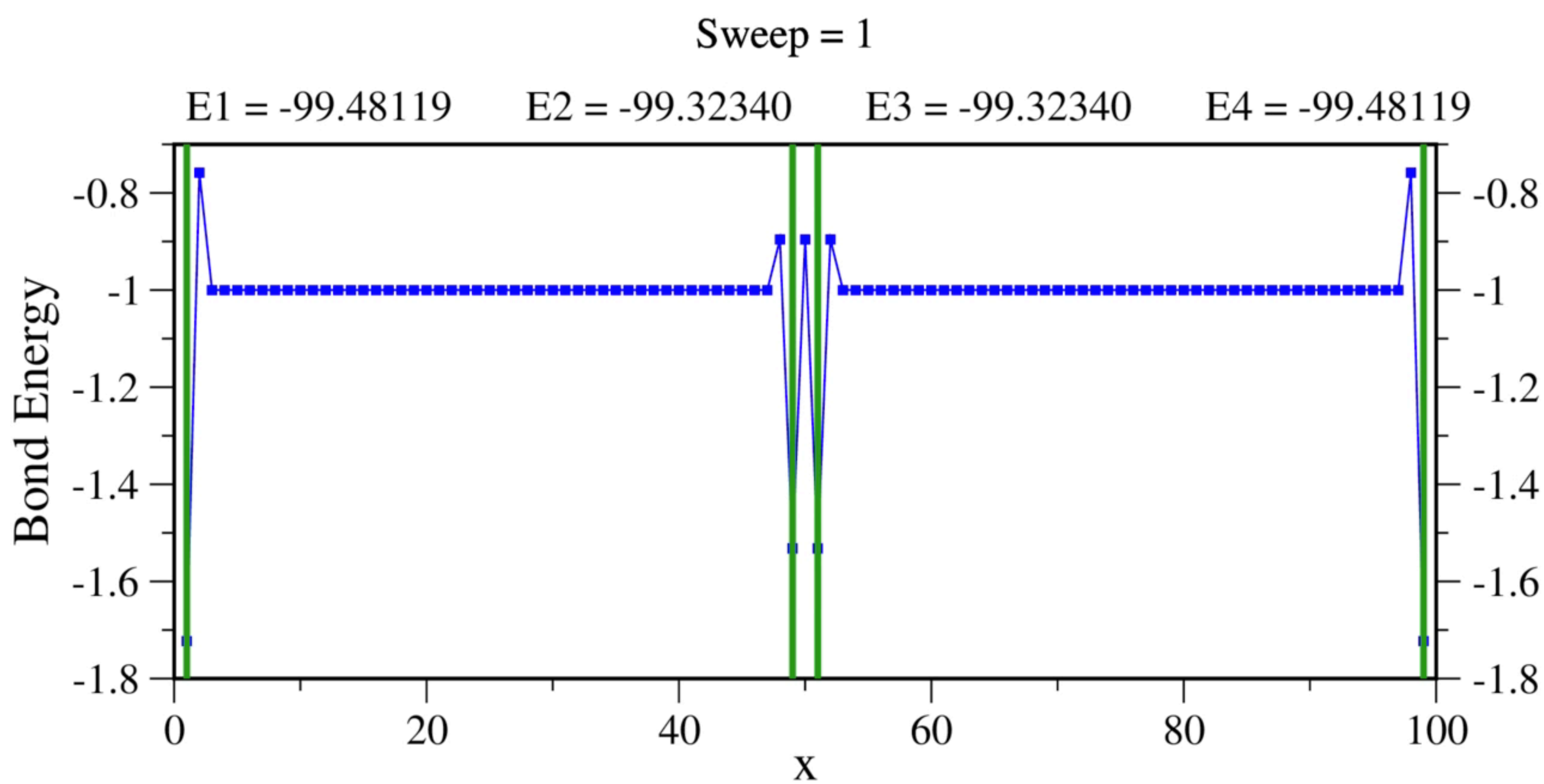
Friendly neighborhood of "quantum state space"





MPS = matrix product state

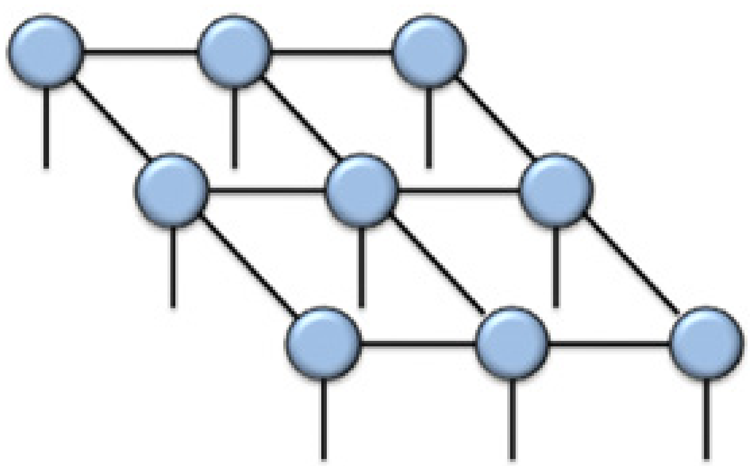
MPS lead to powerful optimization techniques (*DMRG algorithm*)

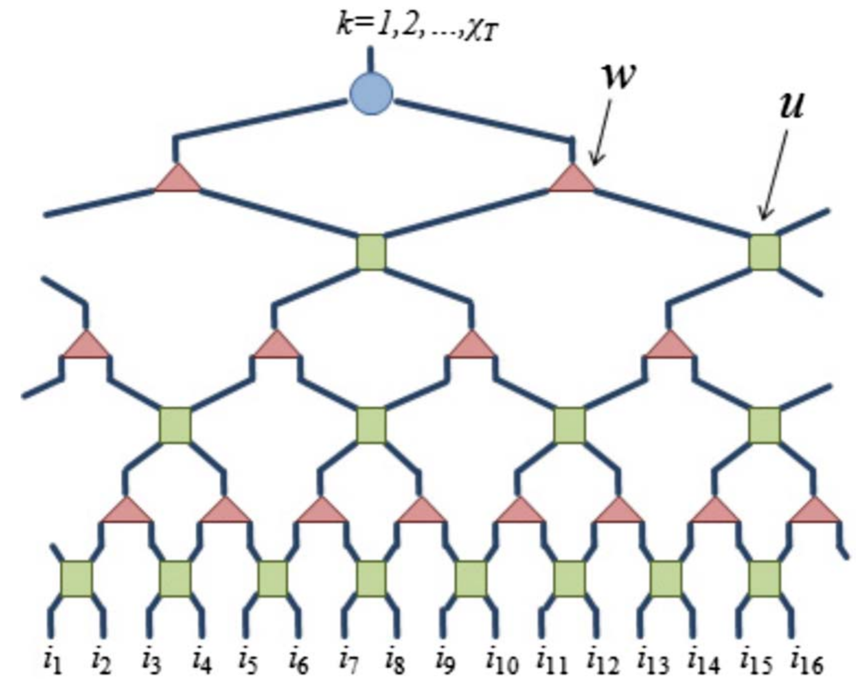


White, PRL **69**, 2863 (1992)

Stoudenmire, White, PRB **87**, 155137 (2013)

Besides MPS, other successful tensor are PEPS and MERA





PEPS

(2D systems)

MERA

(critical systems)

Evenbly, Vidal, PRB **79**, 144108 (2009)

Verstraete, Cirac, cond-mat/0407066 (2004)

Orus, Ann. Phys. **349**, 117 (2014)

Supervised Kernel Learning

Supervised Learning

Very common task:

Labeled training data (= supervised)

Find *decision function* $f(\mathbf{x})$

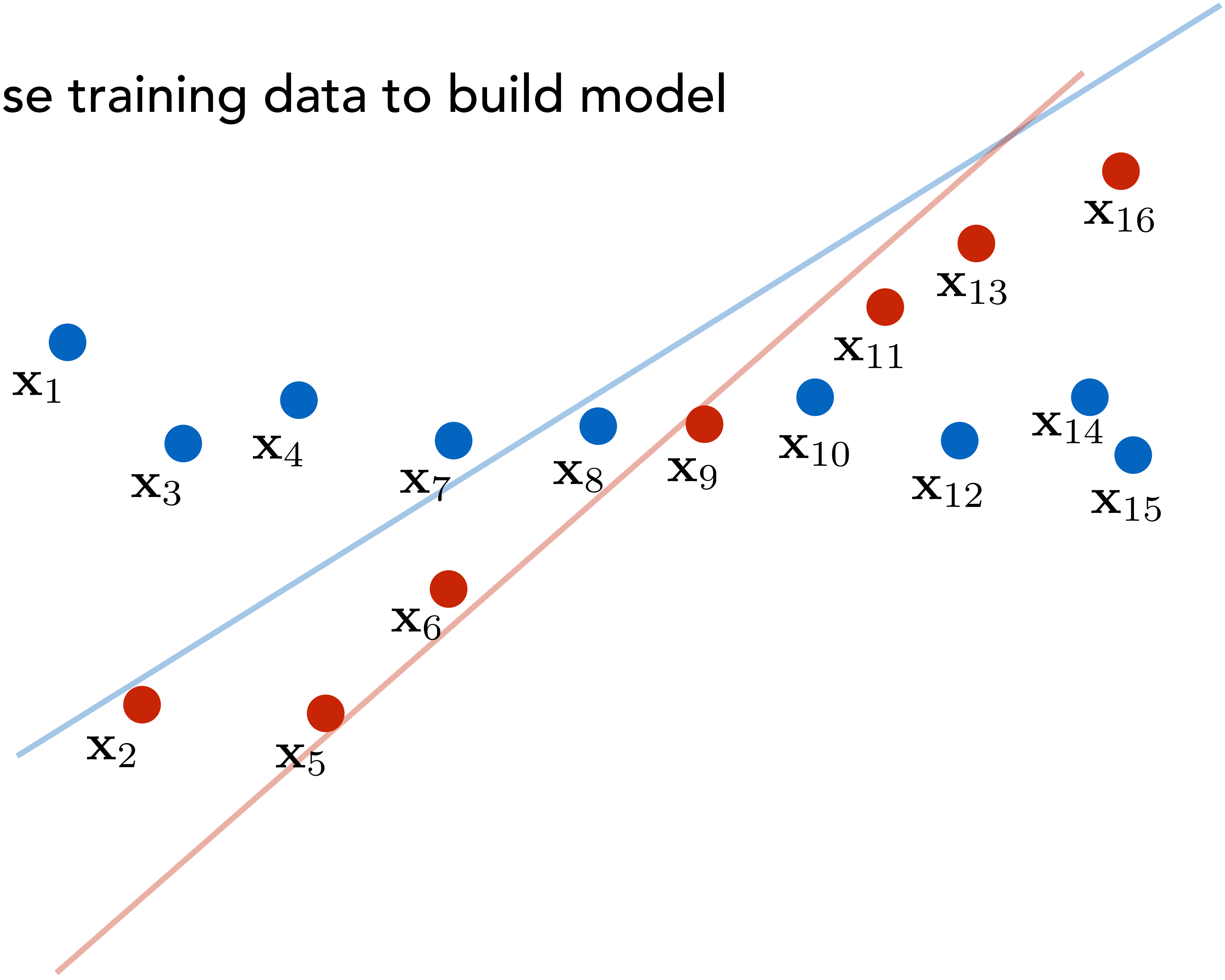
$$f(\mathbf{x}) > 0 \qquad \mathbf{x} \in A$$

$$f(\mathbf{x}) < 0 \qquad \mathbf{x} \in B$$

Input vector $\mathbf{x}$ e.g. image pixels

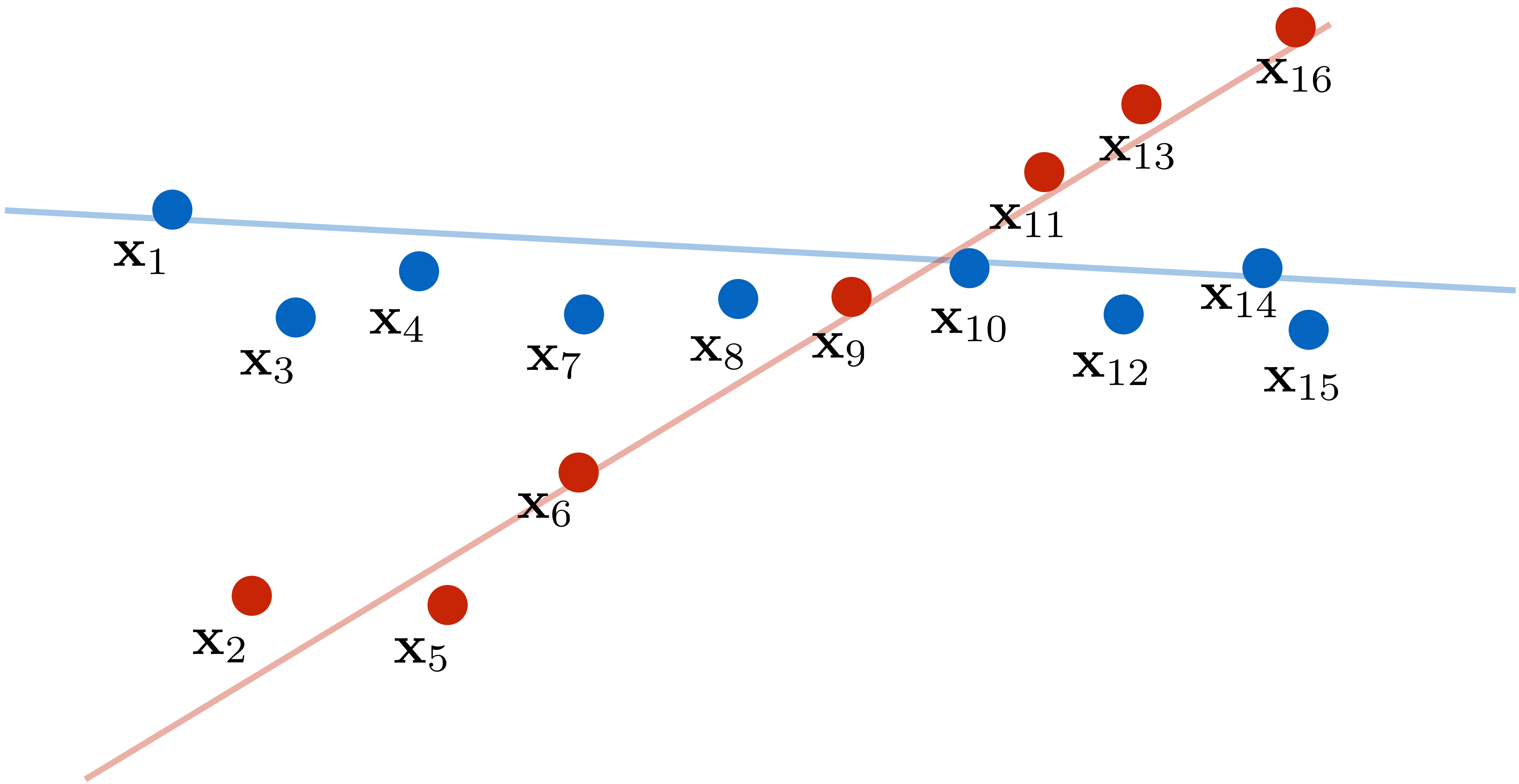
ML Overview

Use training data to build model



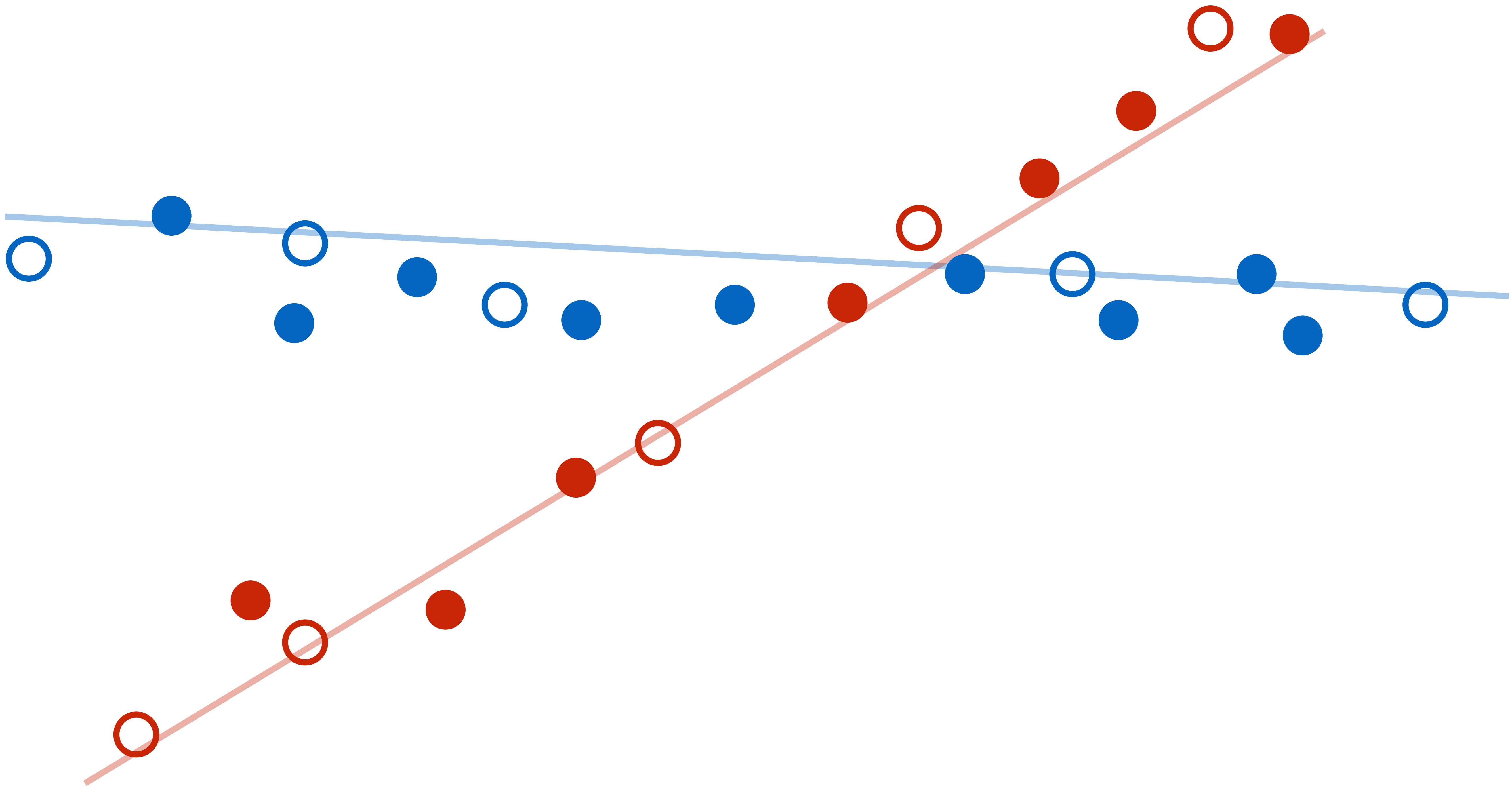
ML Overview

Use training data to build model



ML Overview

Use training data to build model



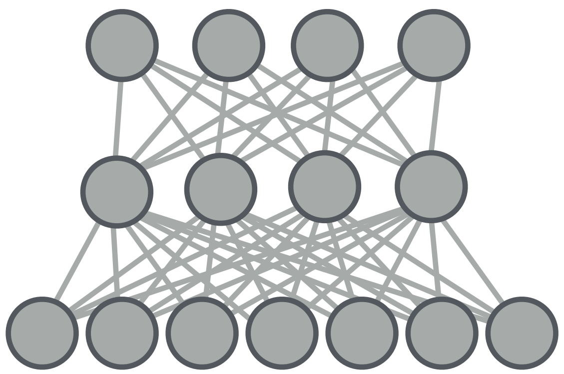
Generalize to unseen test data

ML Overview

Popular approaches

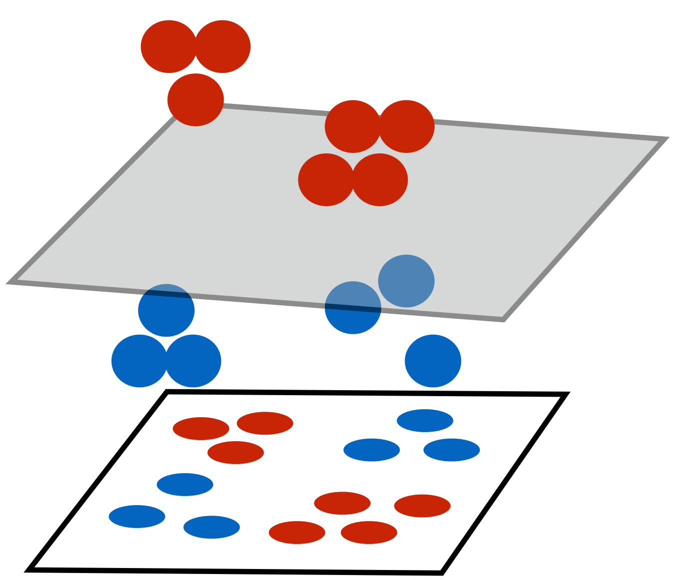
Neural Networks

$$f(\mathbf{x}) = \Phi_2\Big(M_2\Phi_1\big(M_1\mathbf{x}\big)\Big)$$



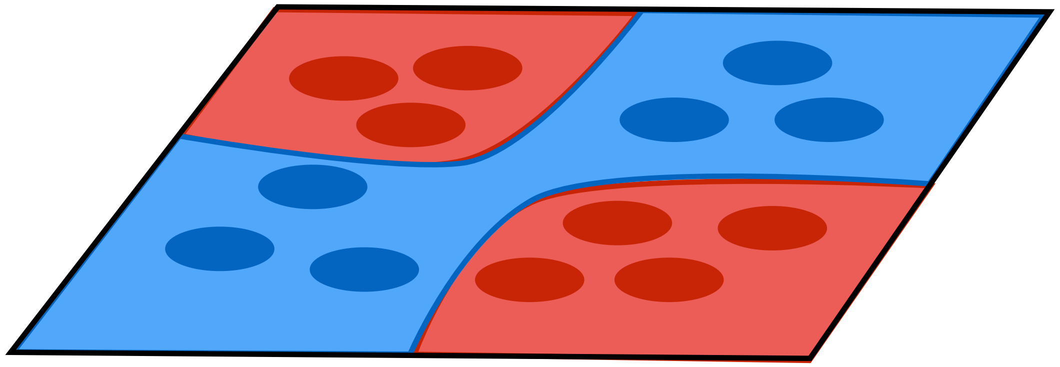
Non-Linear Kernel Learning

$$f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})$$



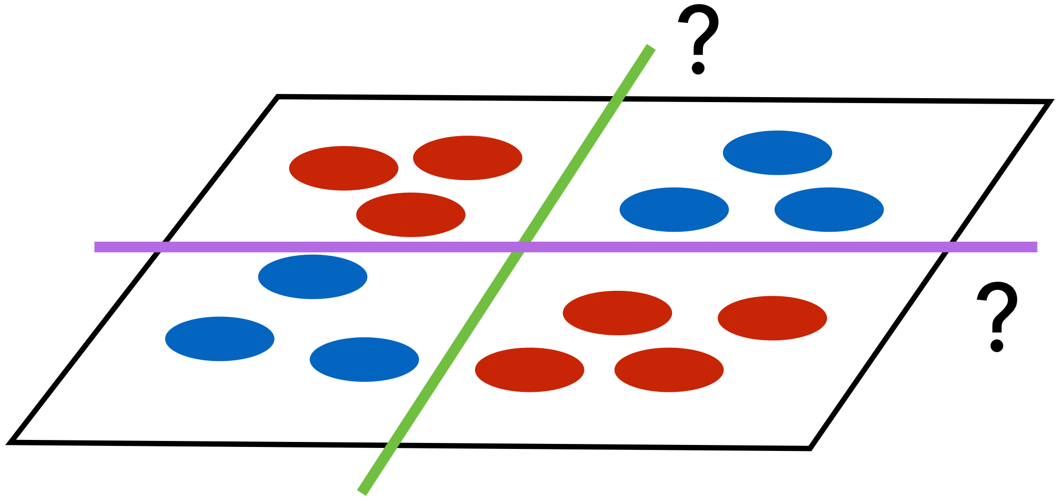
Non-linear kernel learning

Want $f(\mathbf{x})$ to separate classes



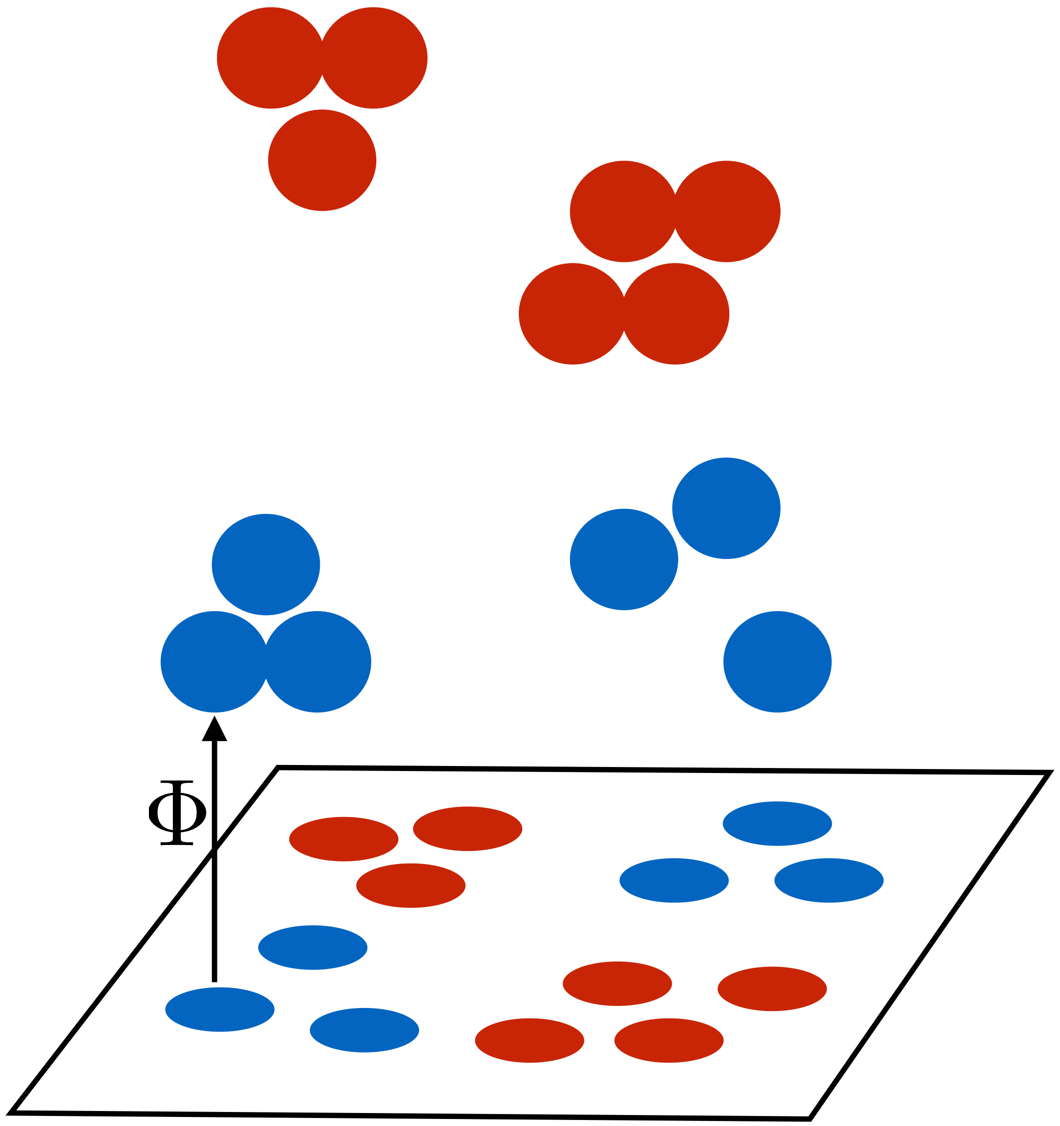
Linear classifier often insufficient

$$f(\mathbf{x}) = W \cdot \mathbf{x}$$



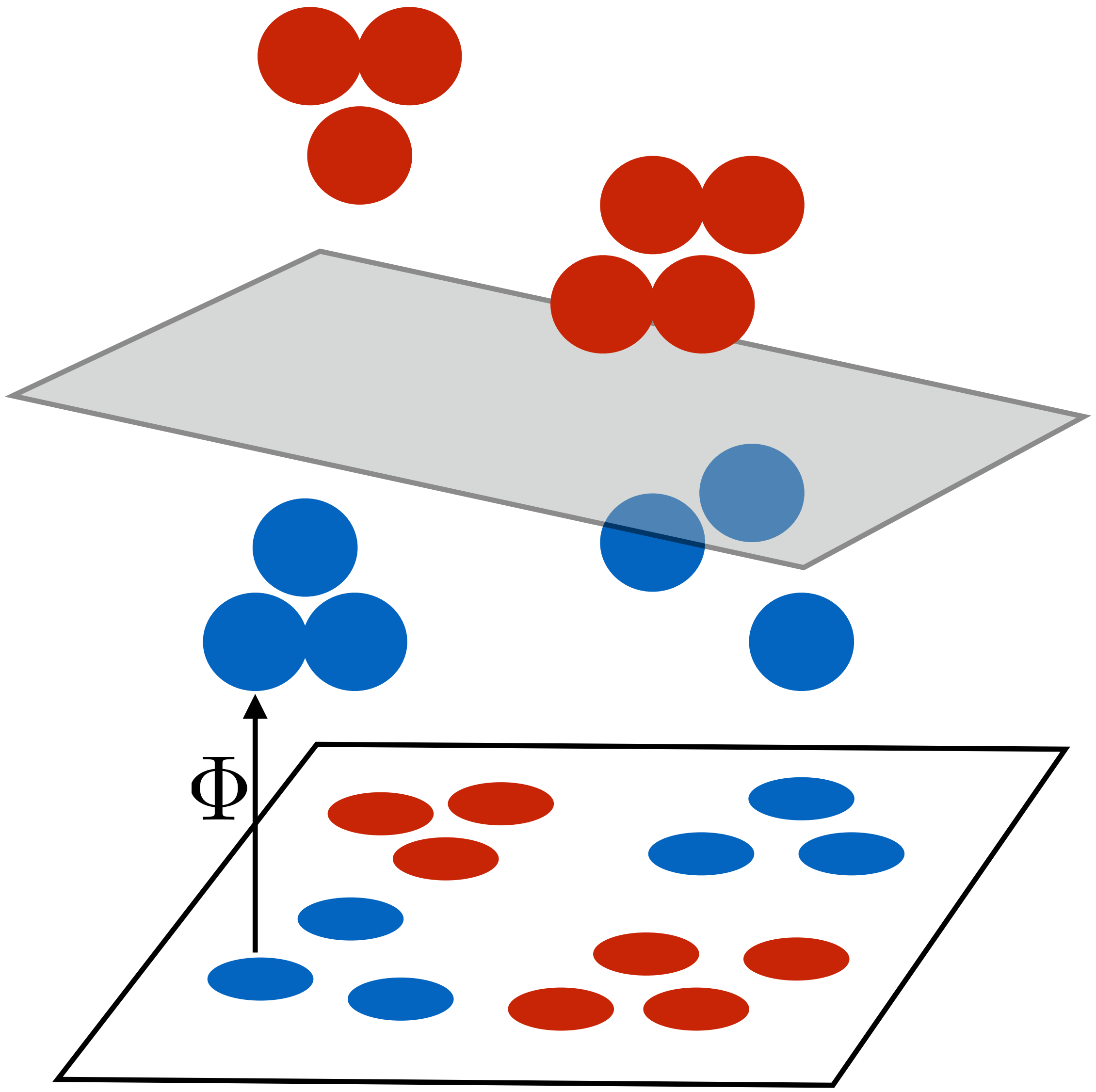
Non-linear kernel learning

Apply non-linear "feature map" $\mathbf{x} \rightarrow \Phi(\mathbf{x})$



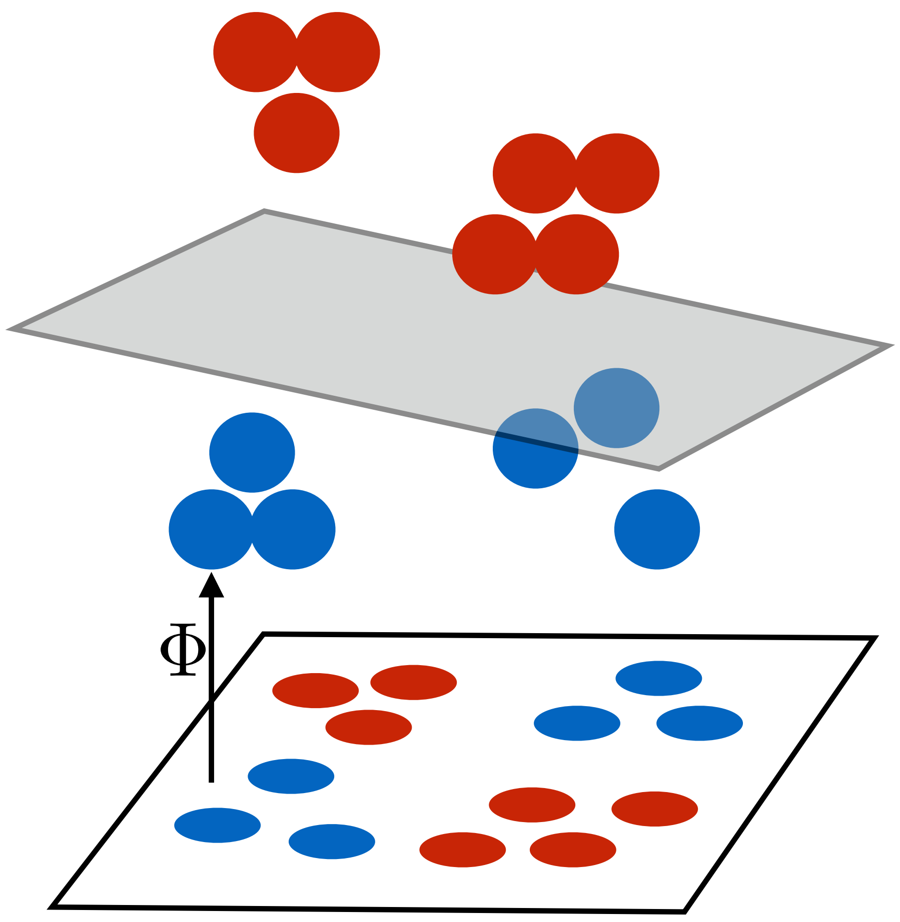
Non-linear kernel learning

Apply non-linear "feature map" $\mathbf{x} \to \Phi(\mathbf{x})$



Decision function $\boxed{f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})}$

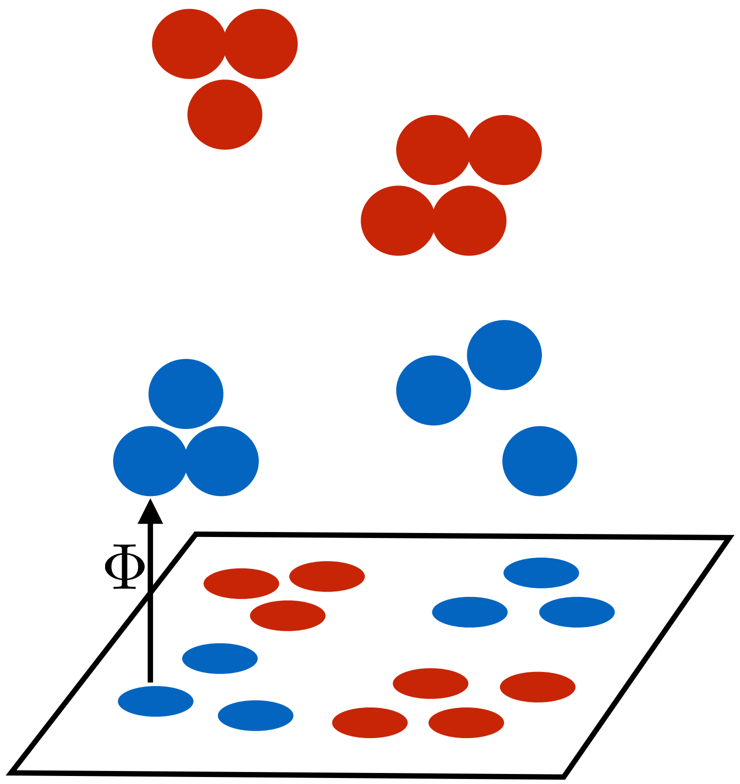
Non-linear kernel learning



Decision function $$\boxed{f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})}$$

Linear classifier in *feature space*

Non-linear kernel learning



Example of *feature map*

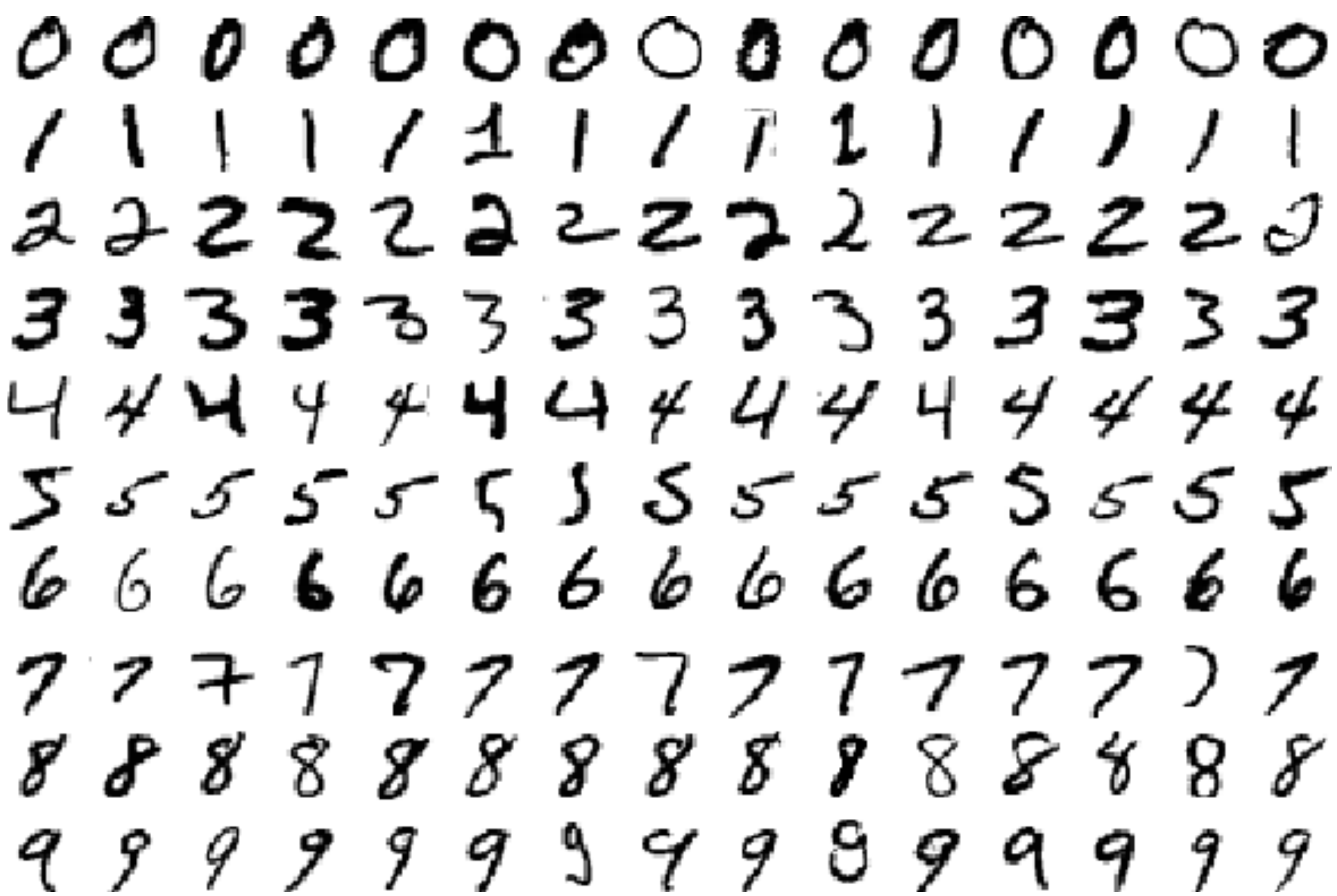
$$\mathbf{x} = (x_1,\, x_2,\, x_3)$$

$$\Phi(\mathbf{x}) = (1,\, x_1,\, x_2,\, x_3,\, x_1 x_2,\, x_1 x_3,\, x_2 x_3)$$

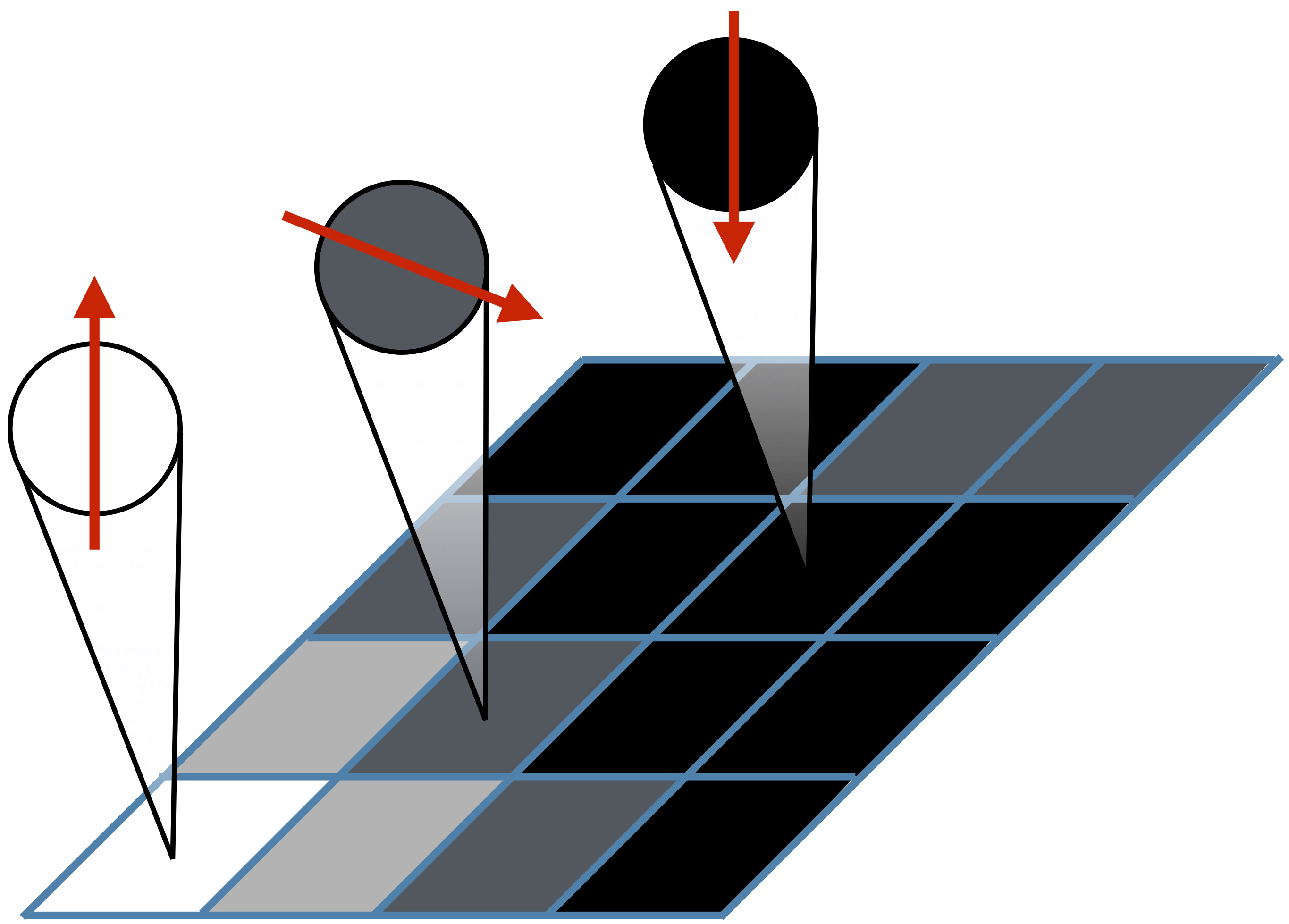
$\mathbf{x}$ is "lifted" to feature space

Proposal for Learning

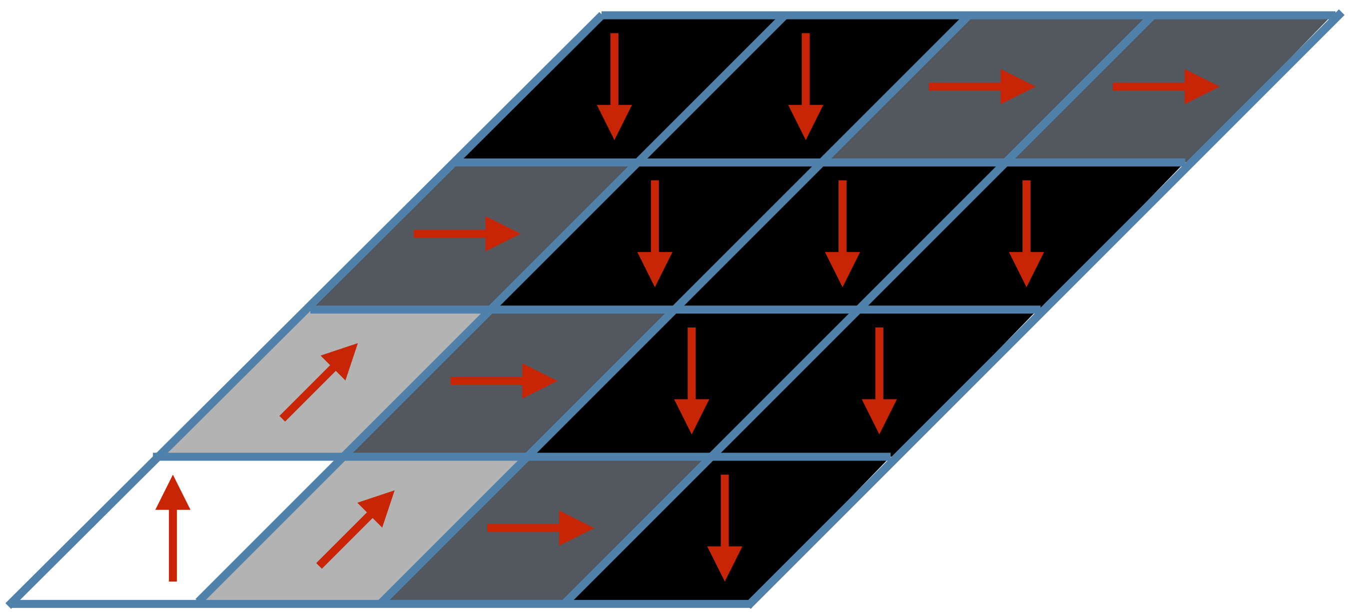
Grayscale image data



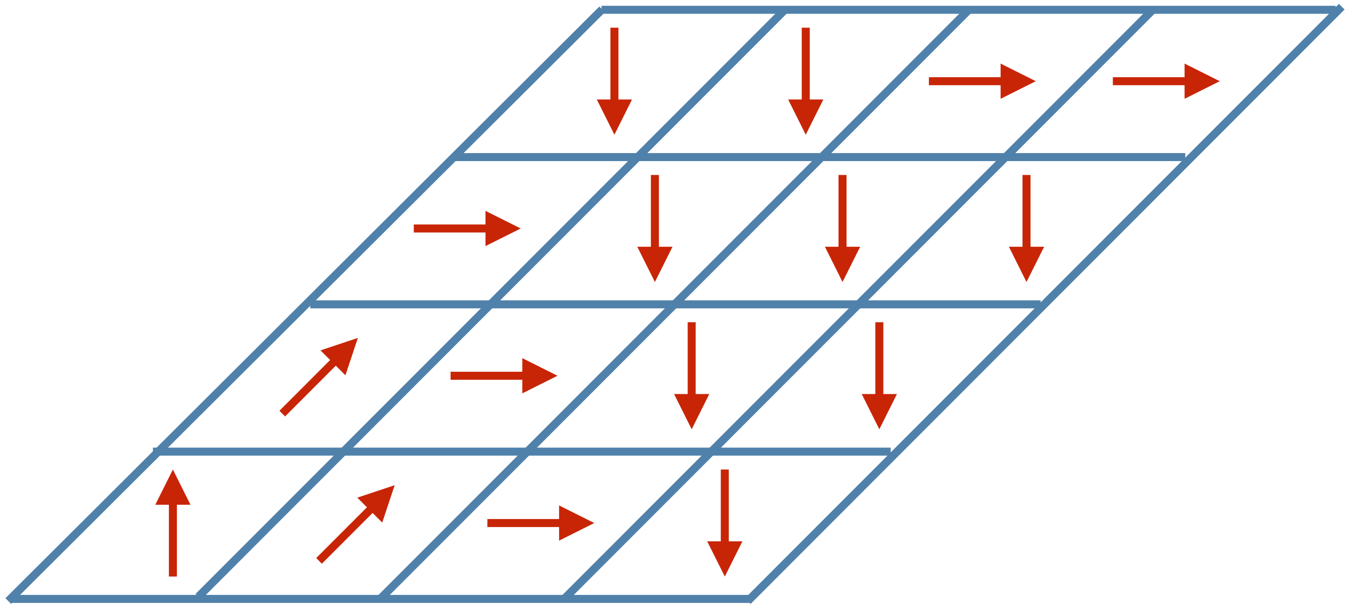
Map pixels to "spins"



Map pixels to "spins"



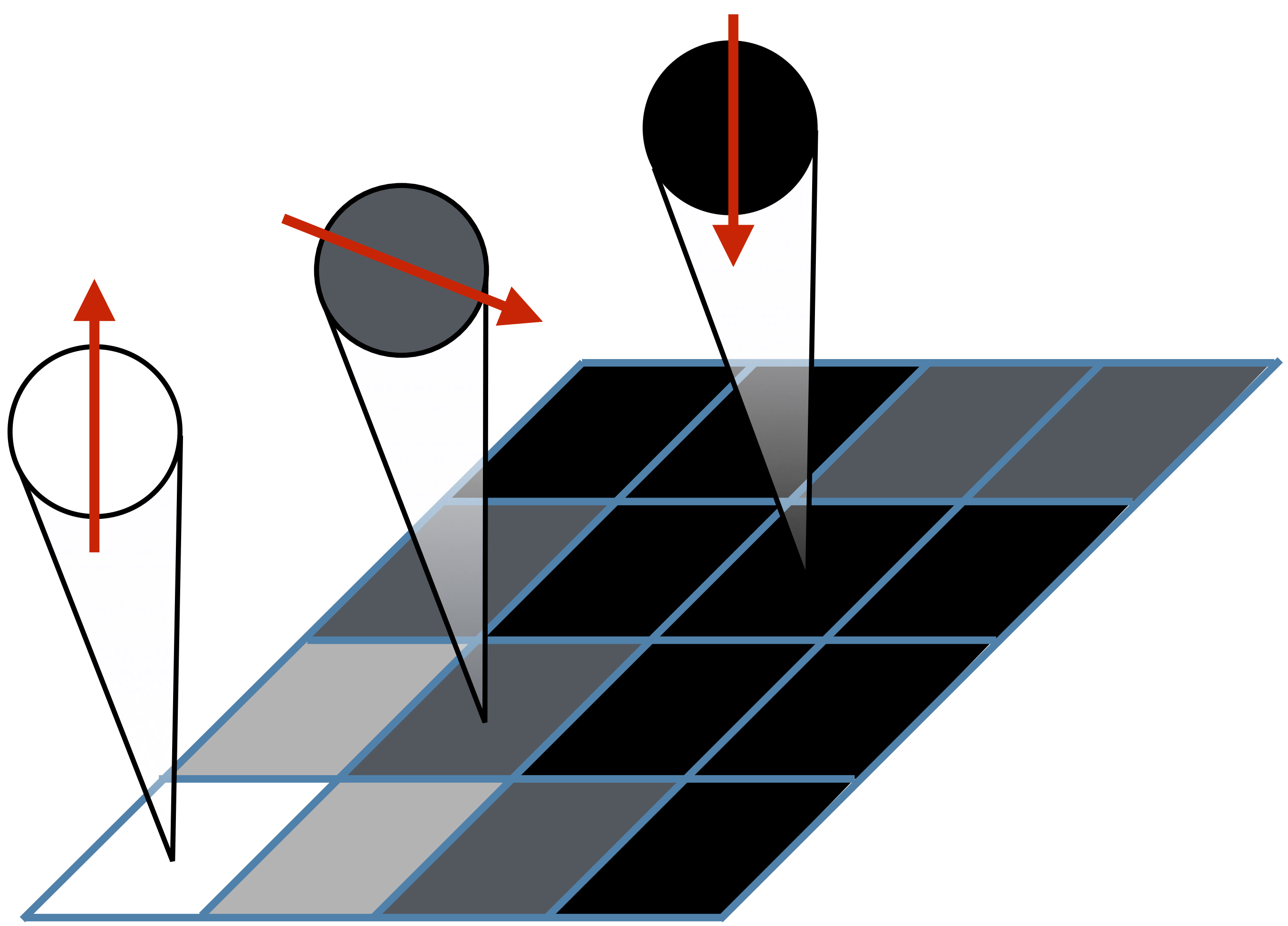
Map pixels to "spins"



$\mathbf{x} =$ input

Local feature map, dimension d=2

$$\phi(x_j) = \left[\cos\left(\frac{\pi}{2}x_j\right), \sin\left(\frac{\pi}{2}x_j\right)\right] \qquad x_j \in [0,1]$$



Crucially, grayscale values not orthogonal

$\mathbf{x} =$ input

$\phi =$ local feature map

Total feature map $\Phi(\mathbf{x})$

$$\Phi^{s_1 s_2 \cdots s_N}(\mathbf{x}) = \phi^{s_1}(x_1) \otimes \phi^{s_2}(x_2) \otimes \cdots \otimes \phi^{s_N}(x_N)$$

- Tensor product of local feature maps / vectors
- Just like product state wavefunction of spins
- Vector in 2^N dimensional space

$\mathbf{x} =$ input

$\phi =$ local feature map

Total feature map $\Phi(\mathbf{x})$

More detailed notation

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

raw inputs

$$\Phi(\mathbf{x}) = \begin{bmatrix} \phi_1(x_1) \\ \phi_2(x_1) \end{bmatrix} \otimes \begin{bmatrix} \phi_1(x_2) \\ \phi_2(x_2) \end{bmatrix} \otimes \begin{bmatrix} \phi_1(x_3) \\ \phi_2(x_3) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \phi_1(x_N) \\ \phi_2(x_N) \end{bmatrix}$$

feature vector

$\mathbf{x} =$ input

$\phi =$ local feature map

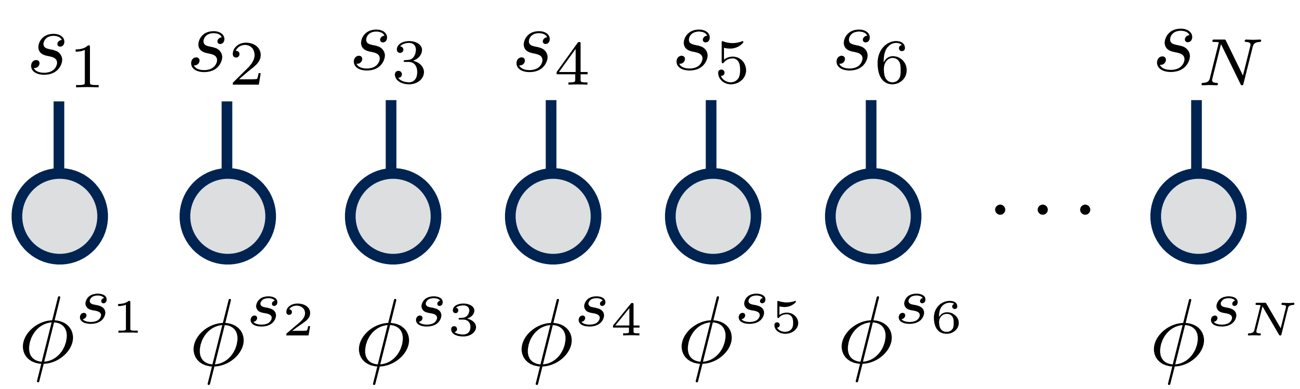
Total feature map $\Phi(\mathbf{x})$

Tensor diagram notation

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

raw inputs

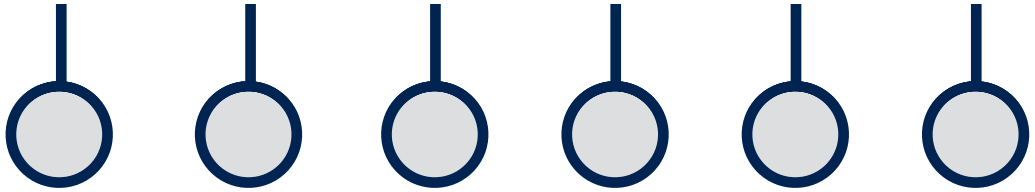
$\Phi(\mathbf{x}) =$



feature vector

Construct decision function

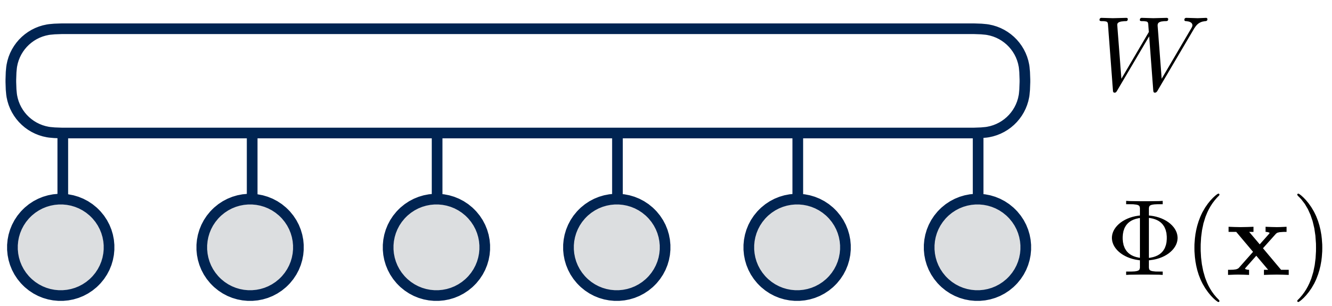
$$\boxed{f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})}$$



$\Phi(\mathbf{x})$

Construct decision function

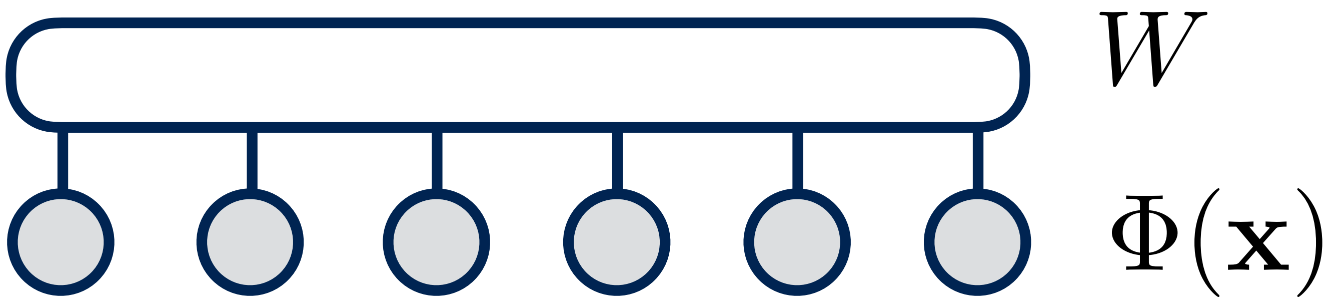
$$\boxed{f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})}$$



Construct decision function

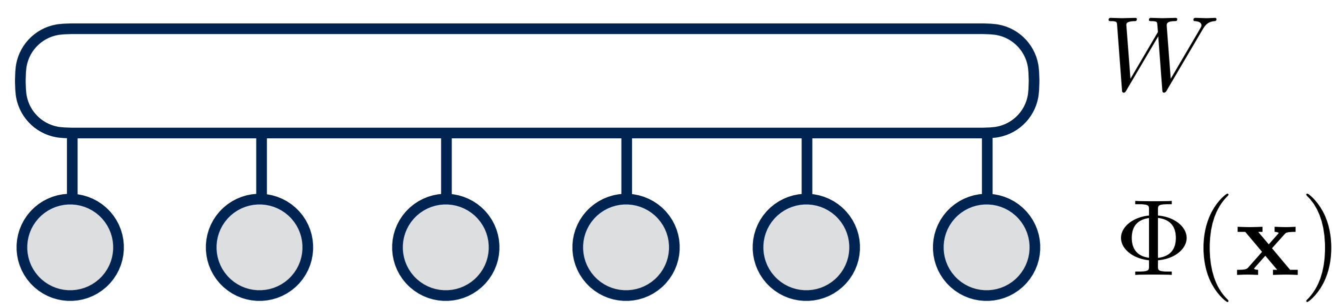
$$\boxed{f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})}$$

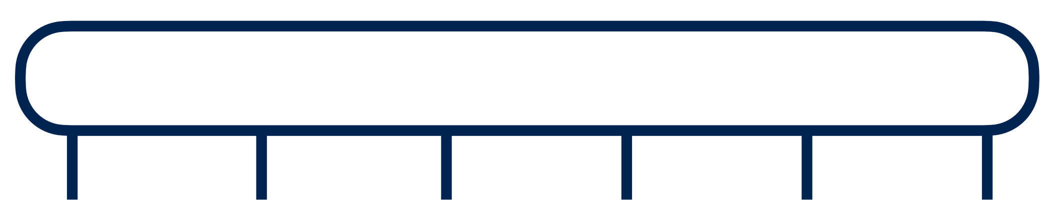
$$f(\mathbf{x}) =$$



Construct decision function

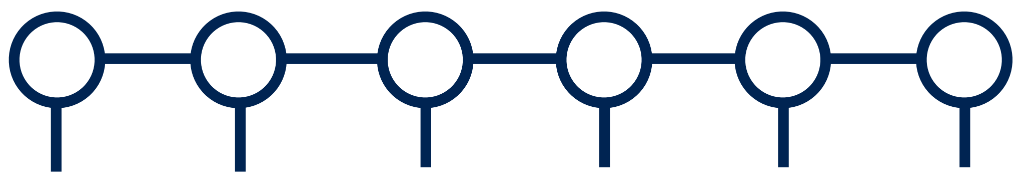
$$\boxed{f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})}$$

$f(\mathbf{x}) =$ 

$W =$ 

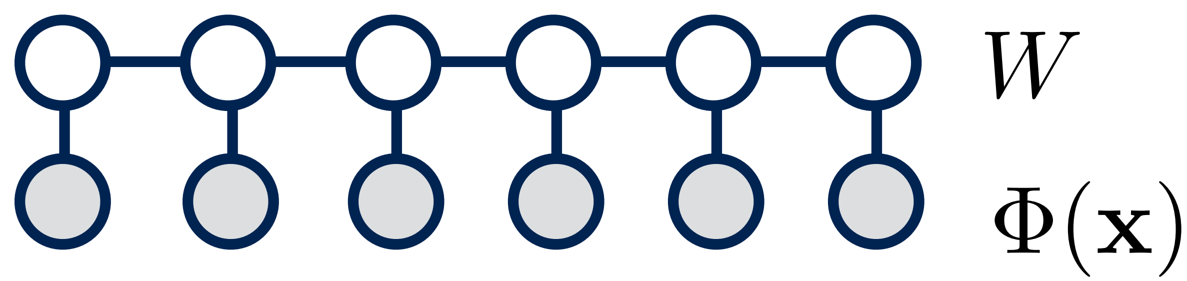
Main approximation

$W =$  *order-N tensor*

$\approx$  *matrix product state (MPS)*

MPS form of decision function

$$f(\mathbf{x}) \; =$$



Linear scaling

Can use algorithm similar to DMRG to optimize

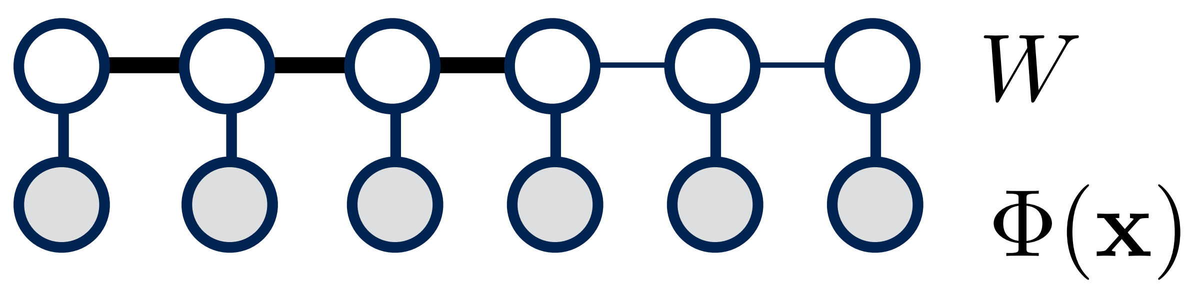
Scaling is $N \cdot N_T \cdot m^3$

$N =$ size of input

$N_T =$ size of training set

$m =$ MPS bond dimension

$f(\mathbf{x}) =$



Linear scaling

Can use algorithm similar to DMRG to optimize

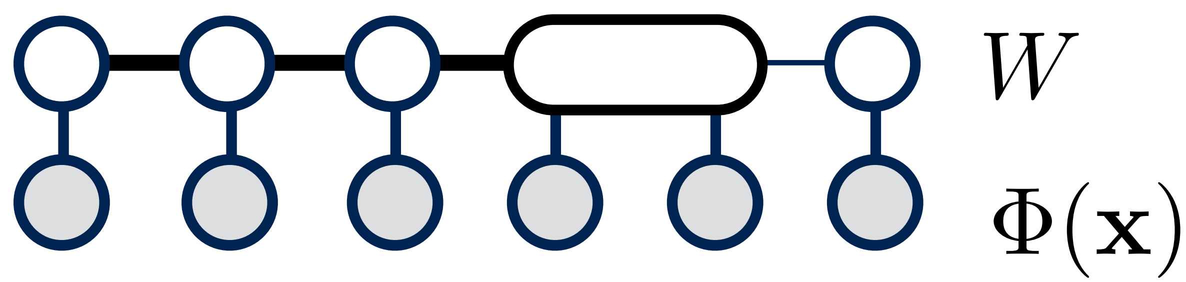
Scaling is $N \cdot N_T \cdot m^3$

$N =$ size of input

$N_T =$ size of training set

$m =$ MPS bond dimension

$f(\mathbf{x}) =$



Linear scaling

Can use algorithm similar to DMRG to optimize

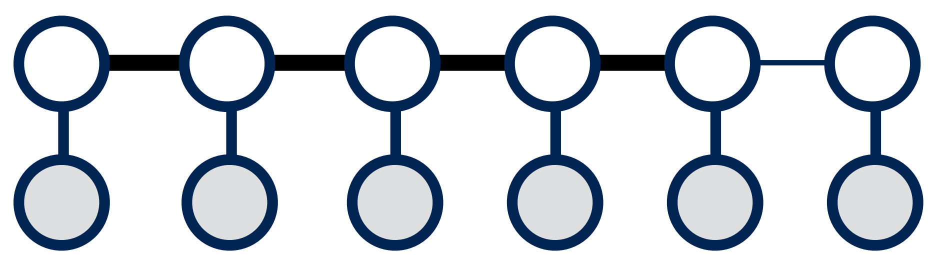
Scaling is $N \cdot N_T \cdot m^3$

$N =$ size of input

$N_T =$ size of training set

$m =$ MPS bond dimension

$f(\mathbf{x}) =$



W

$\Phi(\mathbf{x})$

Linear scaling

Can use algorithm similar to DMRG to optimize

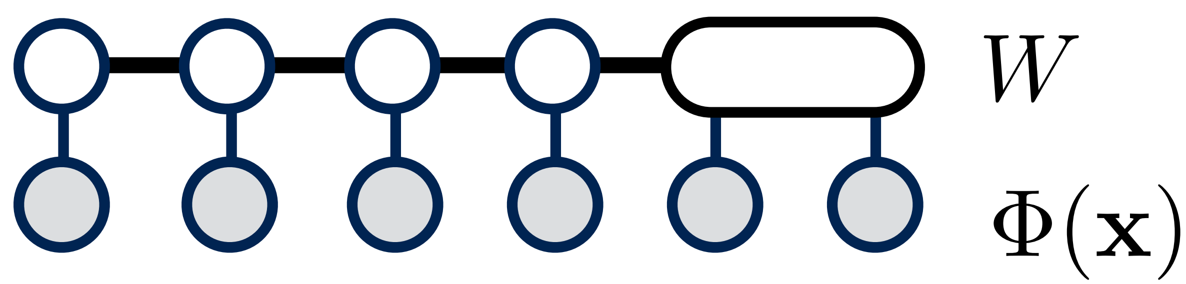
Scaling is $N \cdot N_T \cdot m^3$

$N =$ size of input

$N_T =$ size of training set

$m =$ MPS bond dimension

$f(\mathbf{x}) =$



Linear scaling

Can use algorithm similar to DMRG to optimize

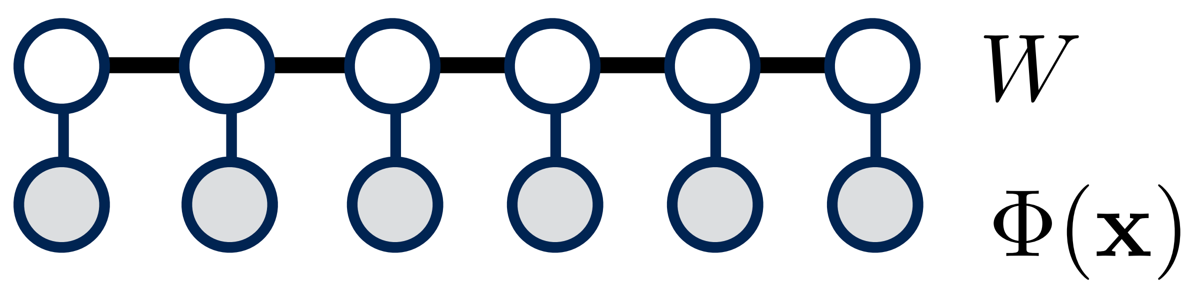
Scaling is $N \cdot N_T \cdot m^3$

$N =$ size of input

$N_T =$ size of training set

$m =$ MPS bond dimension

$f(\mathbf{x}) =$

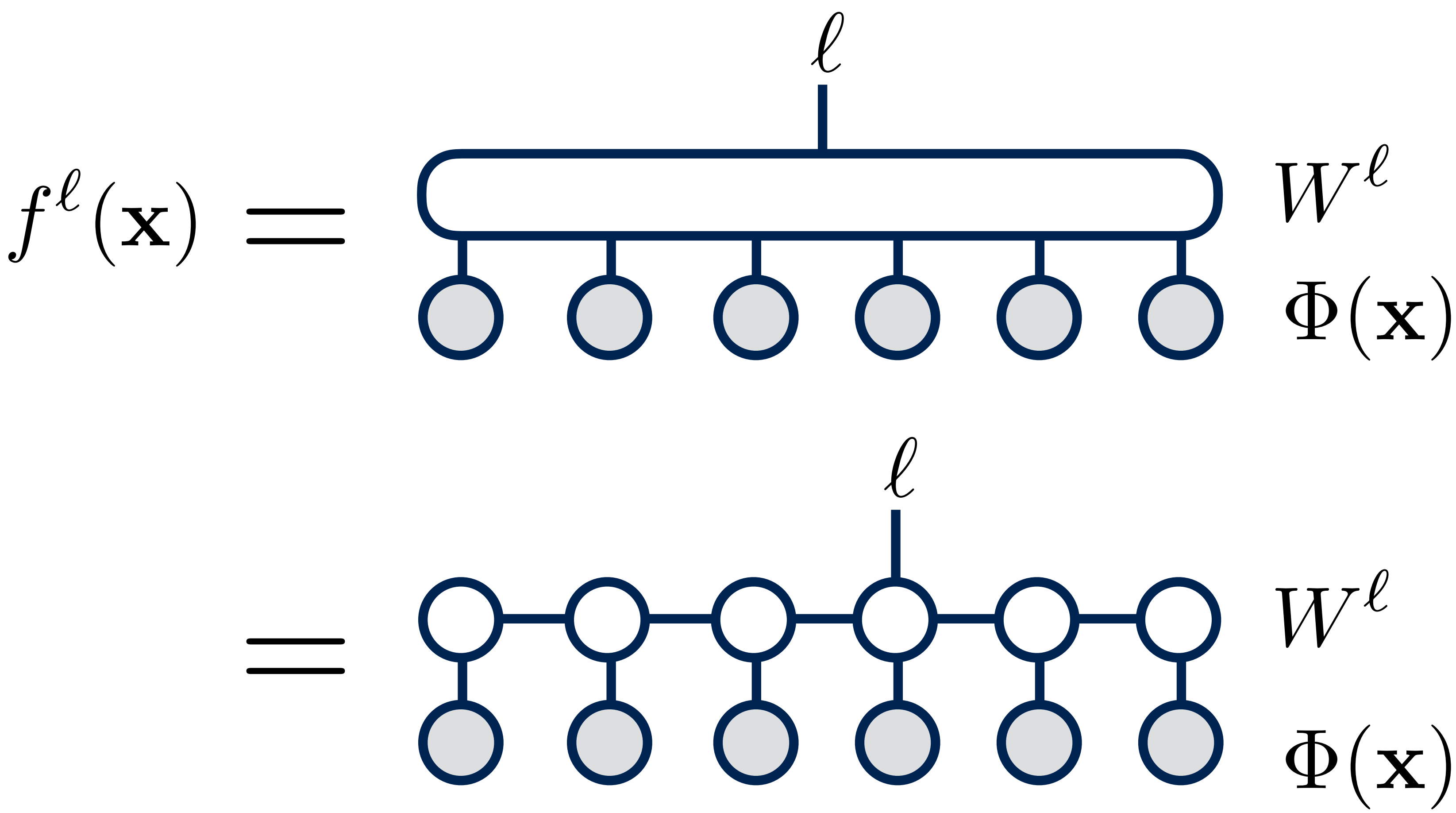


Could improve with stochastic gradient

Multi-class extension of model

Decision function $f^\ell(\mathbf{x}) = W^\ell \cdot \Phi(\mathbf{x})$

Index ℓ runs over possible labels

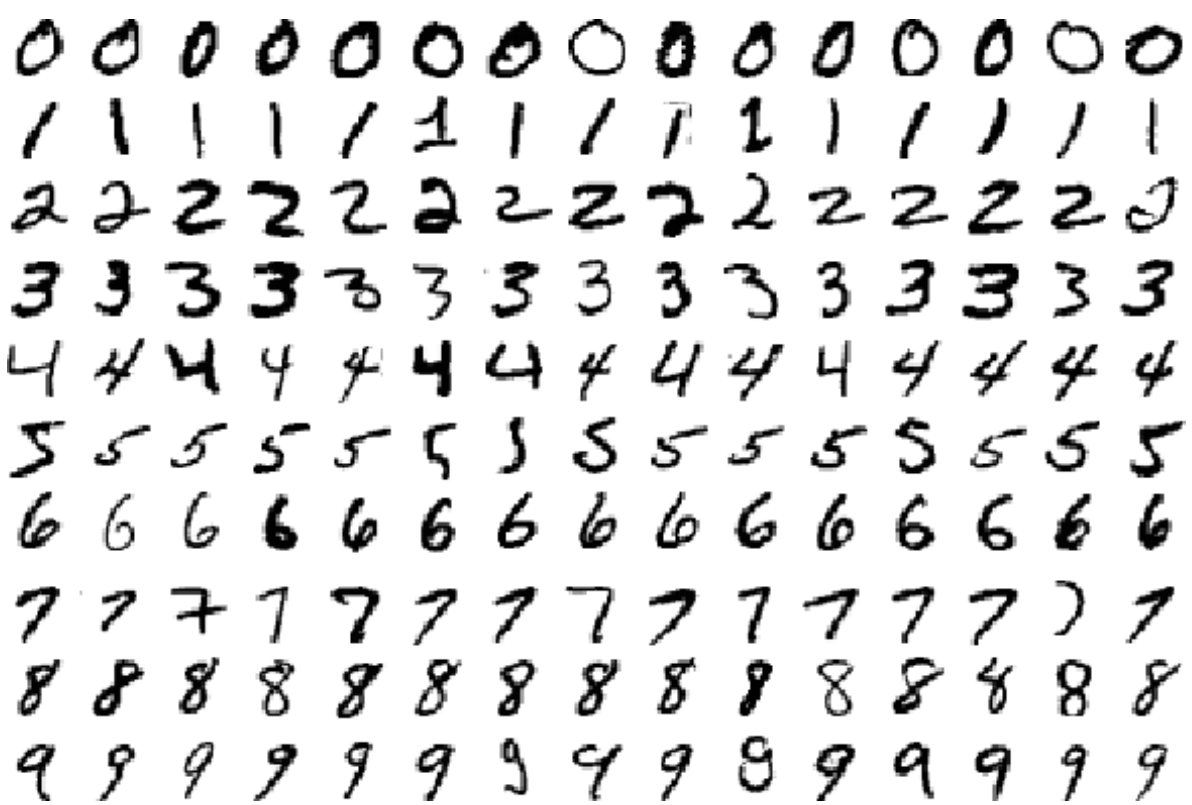


Predicted label is $\mathrm{argmax}_\ell |f^\ell(\mathbf{x})|$

MNIST Experiment

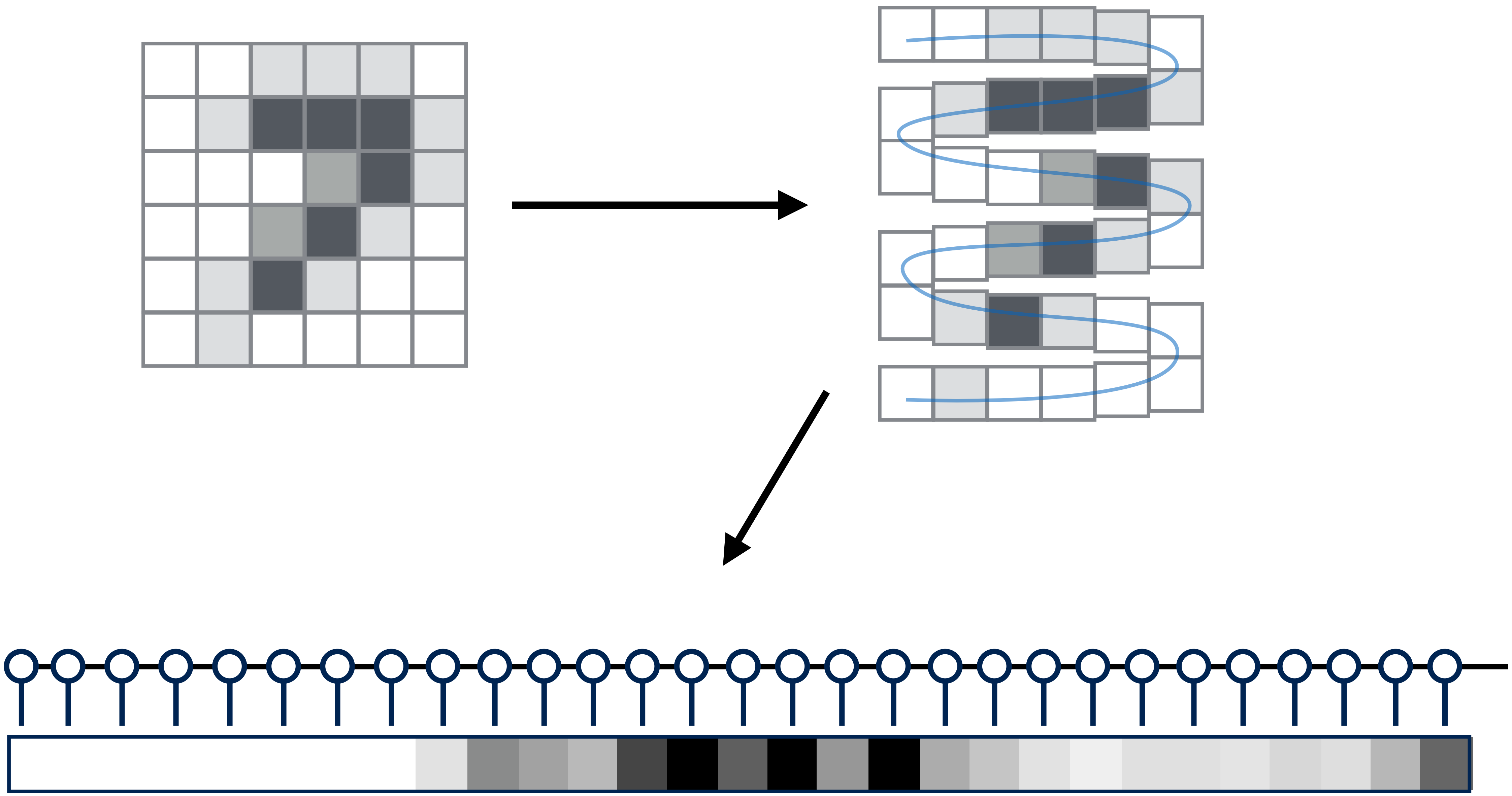
MNIST is a benchmark data set of grayscale handwritten digits (labels $\ell = 0, 1, 2, \ldots, 9$)

60,000 labeled training images
10,000 labeled test images



MNIST Experiment

One-dimensional mapping



MNIST Experiment

Results

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)

State of the art is < 1% test set error

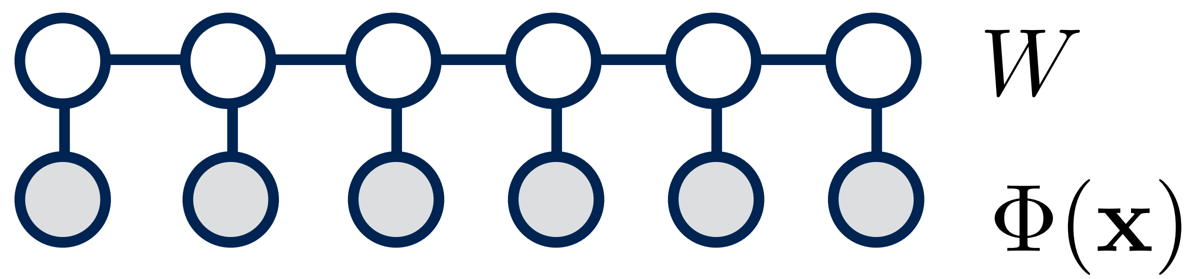
MNIST Experiment

→ *Demo*

Link: *http://itensor.org/miles/digit/index.html*

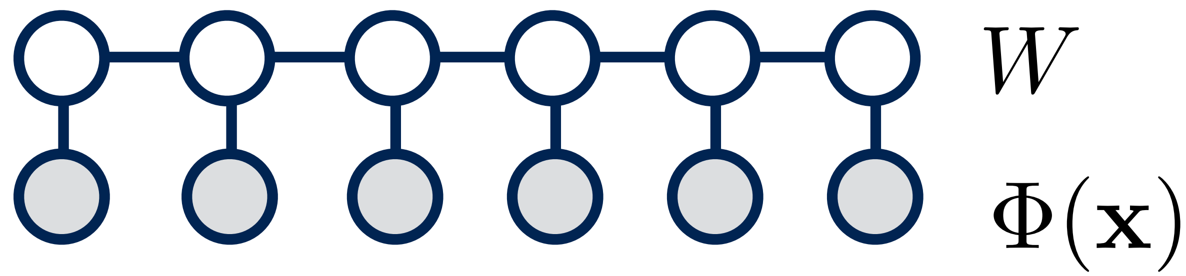
Understanding Tensor Network Models

$f(\mathbf{x}) =$



Again assume W is an MPS

$f(\mathbf{x}) =$



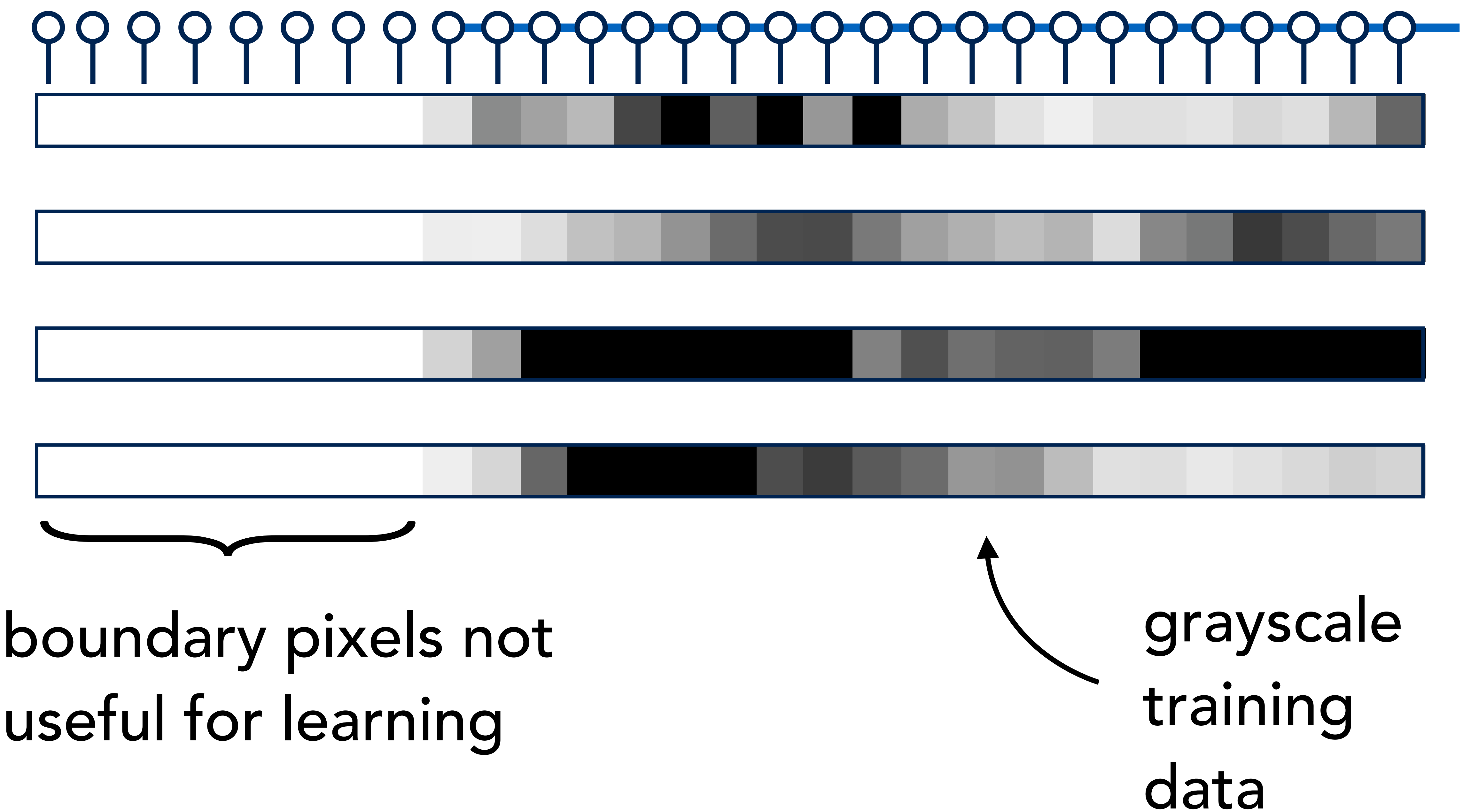
Many interesting benefits

Two are:

1. Adaptive

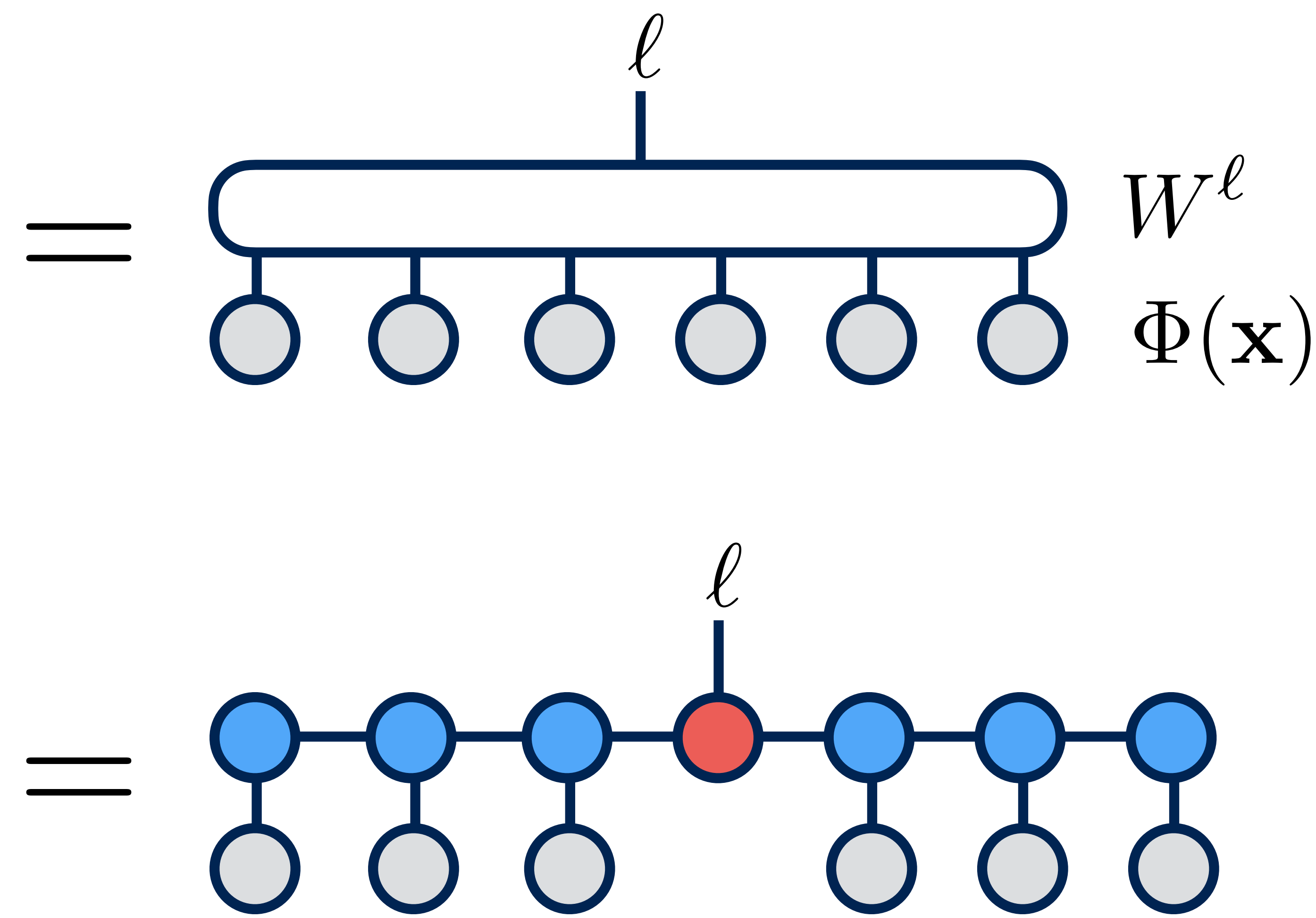
2. Feature sharing

1. Tensor networks are adaptive



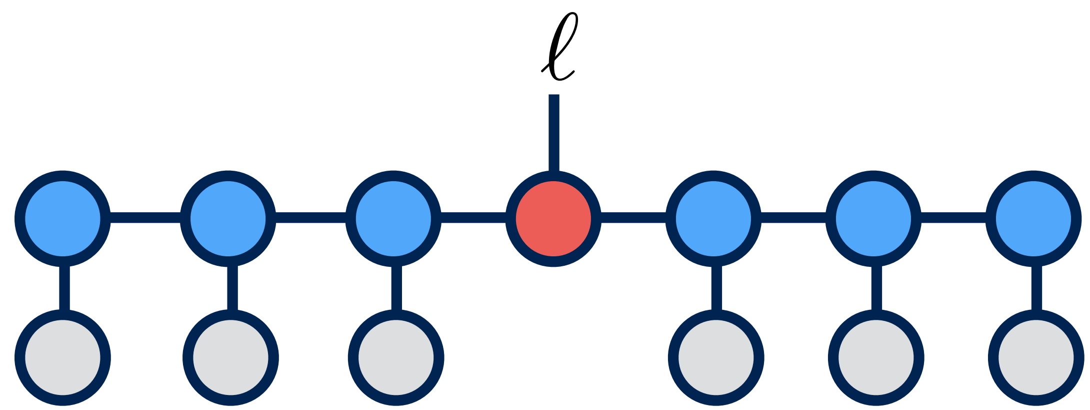
2. Feature sharing

$$f^{\ell}(\mathbf{x}) = $$

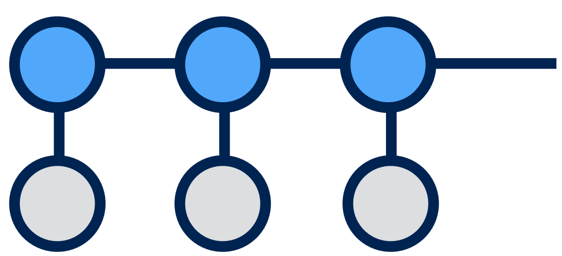


- Different central tensors
- "Wings" shared between models
- Regularizes models

2. Feature sharing

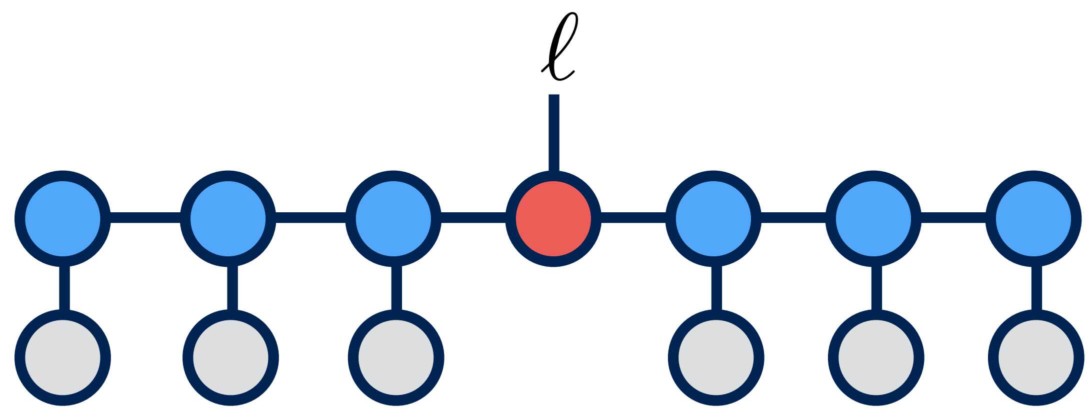
$f^{\ell}(\mathbf{x}) =$ 

Progressively learn shared features

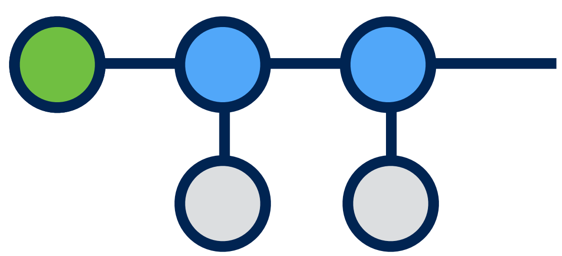


2. Feature sharing

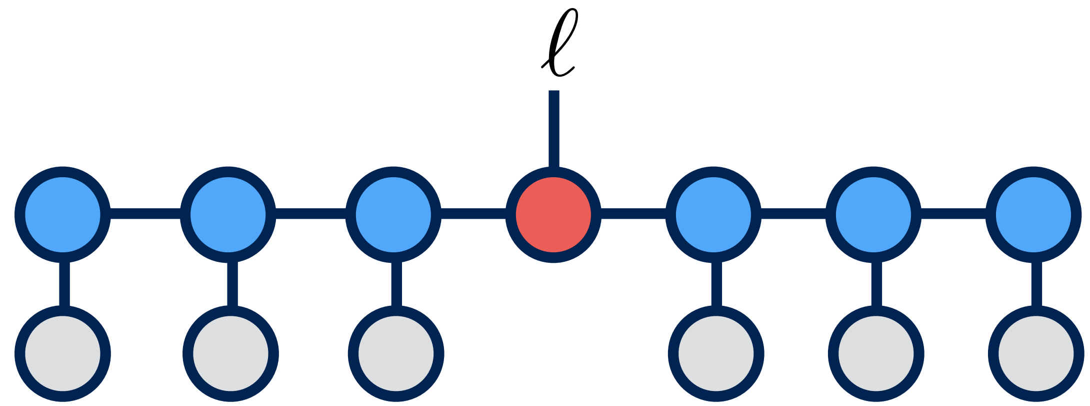
$$f^{\ell}(\mathbf{x}) =$$



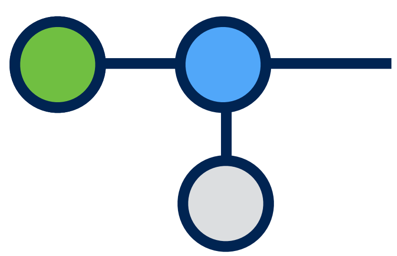
Progressively learn shared features



2. Feature sharing

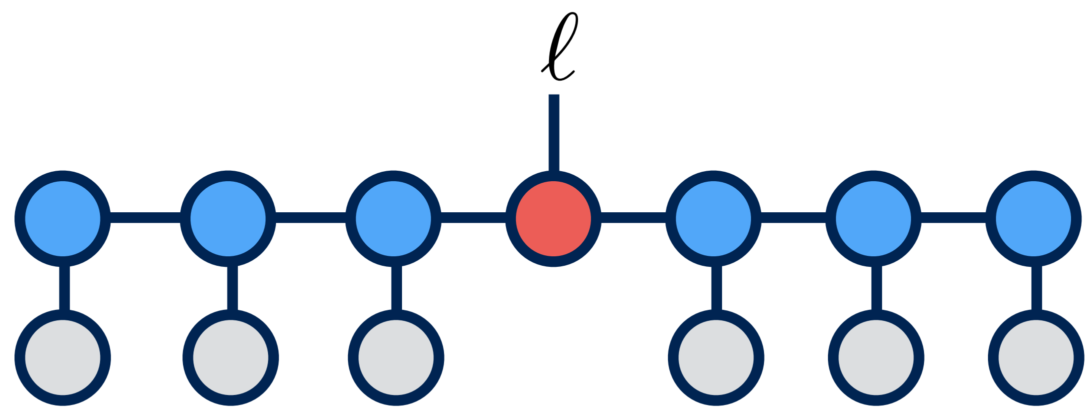
$f^{\ell}(\mathbf{x}) =$ 

Progressively learn shared features

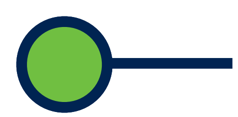


2. Feature sharing

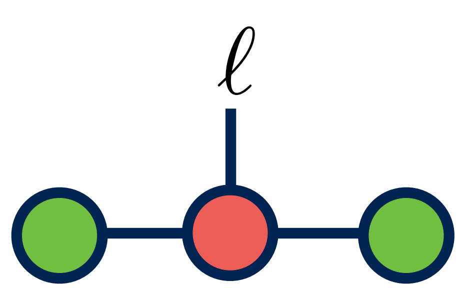
$$f^{\ell}(\mathbf{x}) =$$



Progressively learn shared features



Deliver to central tensor



Nature of Weight Tensor

Representer theorem says exact $W = \sum_j \alpha_j \Phi(x_j)$

Density plots of trained W^ℓ for each label $\ell = 0, 1, \ldots, 9$

Nature of Weight Tensor

Representer theorem says exact $W = \sum_j \alpha_j \Phi(x_j)$

Tensor network approx. can violate this condition

$W_{\mathrm{MPS}} \neq \sum_j \alpha_j \Phi(x_j)$ for any $\{\alpha_j\}$

- Tensor network learning not interpolation
- Interesting consequences for generalization?

Some Future Directions

- Apply to 1D data sets (audio, time series)

- Other tensor networks: TTN, PEPS, MERA

- Useful to interpret $|W \cdot \Phi(\mathbf{x})|^2$ as probability? Could import even more physics insights.

- Features extracted by elements of tensor network?

What functions realized for arbitrary W ?

Instead of "spin" local feature map, use*

$$\phi(x) = (1, x)$$

Recall total feature map is

$$\Phi(\mathbf{x}) = \begin{bmatrix} \phi_1(x_1) \\ \\ \phi_2(x_1) \end{bmatrix} \otimes \begin{bmatrix} \phi_1(x_2) \\ \\ \phi_2(x_2) \end{bmatrix} \otimes \begin{bmatrix} \phi_1(x_3) \\ \\ \phi_2(x_3) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \phi_1(x_N) \\ \\ \phi_2(x_N) \end{bmatrix}$$

*Novikov, et al., arxiv:1605.03795

N=2 case

$$\phi(x) = (1, x)$$

$$\Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ x_2 \end{bmatrix}$$

$$= (1,\ x_1,\ x_2,\ x_1 x_2)$$

$$f(\mathbf{x}) = W \cdot \Phi(\mathbf{x}) \qquad = \qquad \begin{array}{l} (W_{11}, W_{21}, W_{12},\ W_{22}) \\ \cdot\, (\ 1,\quad x_1,\quad x_2,\ \ x_1 x_2) \end{array}$$

$$= W_{11} + W_{21}\, x_1 + W_{12}\, x_2 + W_{22}\, x_1 x_2$$

N=3 case

$$\phi(x) = (1, x)$$

$$\Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ x_2 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ x_3 \end{bmatrix}$$

$$= (1,\ x_1,\ x_2,\ x_3,\ x_1x_2,\ x_1x_3,\ x_2x_3,\ x_1x_2x_3)$$

$$f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})$$

$$= W_{111} + W_{211}\, x_1 + W_{121}\, x_2 + W_{112}\, x_3$$

$$+ W_{221}\, x_1x_2 + W_{212}\, x_1x_3 + W_{122}\, x_1x_3$$

$$+ W_{222}\, x_1x_2x_3$$

General N case

$$\mathbf{x} \in \mathbb{R}^N$$

$$
\begin{aligned}
f(\mathbf{x}) = W \cdot \Phi(\mathbf{x}) & \\
= \quad & W_{111\cdots1} && \textit{constant} \\
& + W_{211\cdots1}\, x_1 + W_{121\cdots1}\, x_2 + W_{112\cdots1}\, x_3 + \ldots && \textit{singles} \\
& + W_{221\cdots1}\, x_1 x_2 + W_{212\cdots1}\, x_1 x_3 + \ldots && \textit{doubles} \\
& + W_{222\cdots1}\, x_1 x_2 x_3 + \ldots && \textit{triples} \\
& + \ldots \\
& + W_{222\cdots2}\, x_1 x_2 x_3 \cdots x_N && \textit{N-tuple}
\end{aligned}
$$

Model has exponentially many formal parameters

Novikov, Trofimov, Oseledets, arxiv:1605.03795 (2016)

Related Work

Novikov, Trofimov, Oseledets *(1605.03795)*

- matrix product states + kernel learning
- stochastic gradient descent

Cohen, Sharir, Shashua *(1410.0781, 1506.03059, 1603.00162, 1610.04167)*

- tree tensor networks
- expressivity of tensor network models
- correlations of data (analogue of entanglement entropy)
- generative proposal

Other MPS related work (= "tensor trains")

Markov random field models

Novikov et al., Proceedings of 31st ICML (2014)

Large scale PCA

Lee, Cichocki, arxiv: 1410.6895 (2014)

Feature extraction of tensor data

Bengua et al., IEEE Congress on Big Data (2015)

Compressing weights of neural nets

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