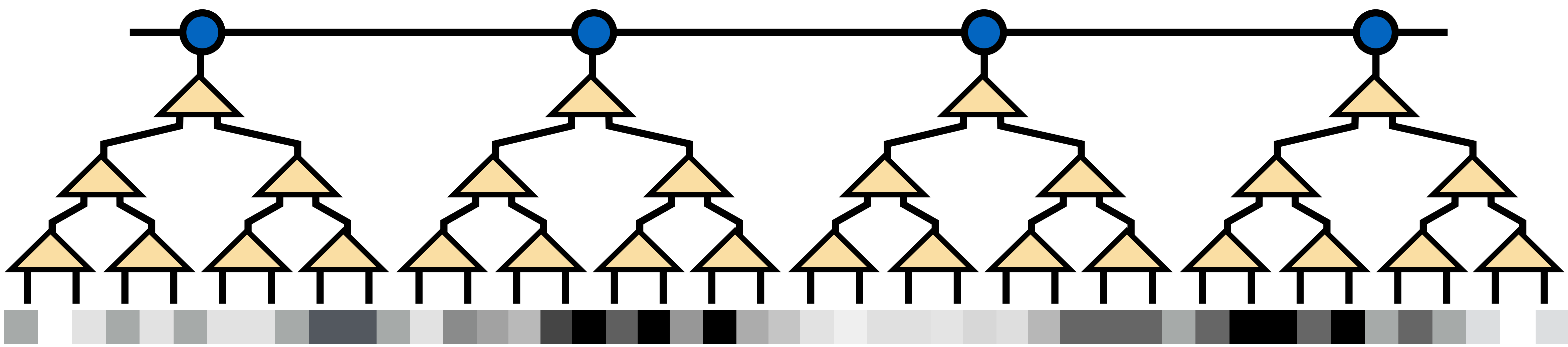
Learning Relevant Features of Data Using Multi-Scale Tensor Networks

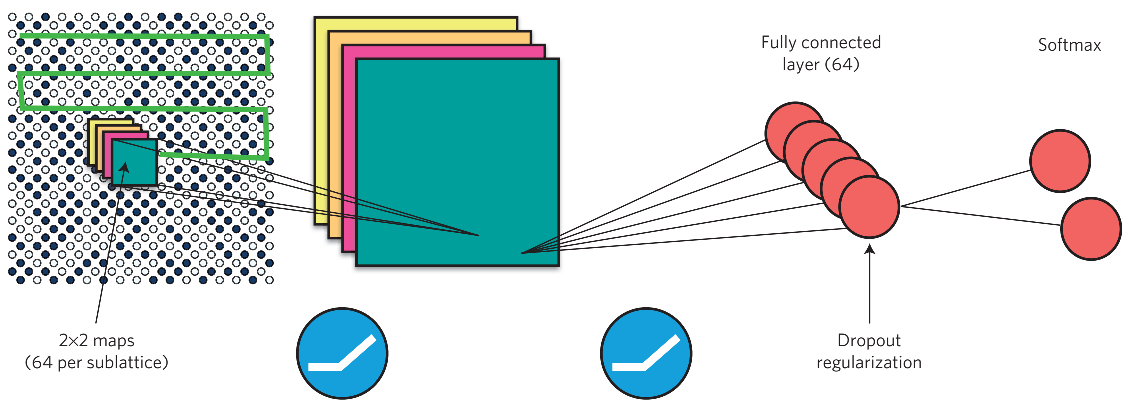


E.M. Stoudenmire

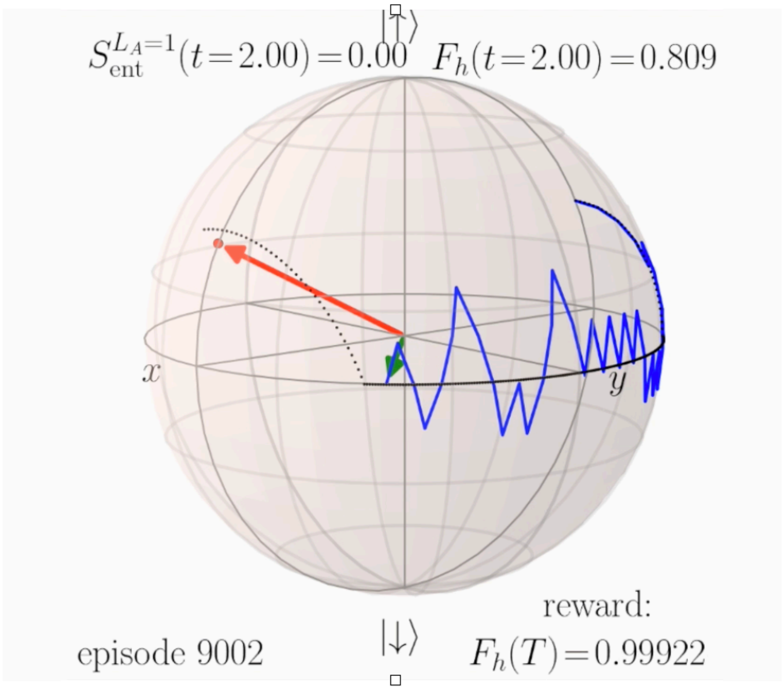
Feb 2018 - Benasque

FLATIRON INSTITUTE
SIMONS FOUNDATION

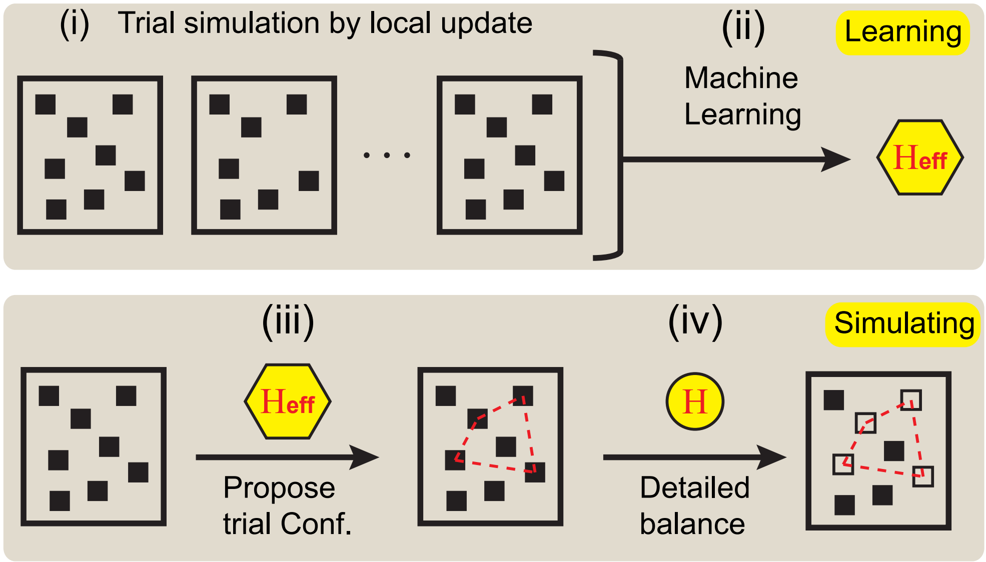
Lots of interesting idea of applying machine learning techniques to physics



Recognizing phases of matter



Control of Quantum Systems

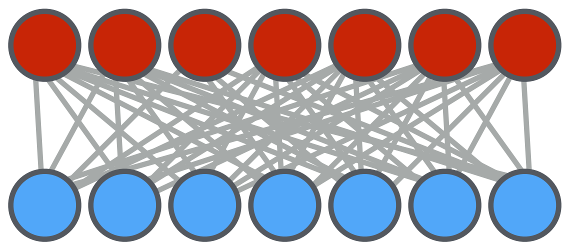


Self-Learning Monte Carlo

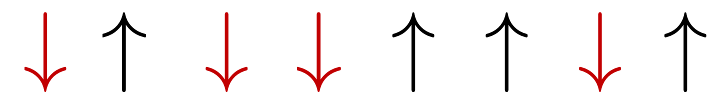
…this talk is the other way around…

physics concepts $\longrightarrow$ machine learning

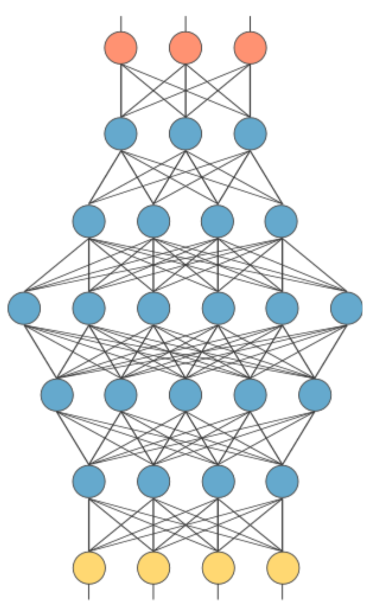
Many physics ideas appear in machine learning



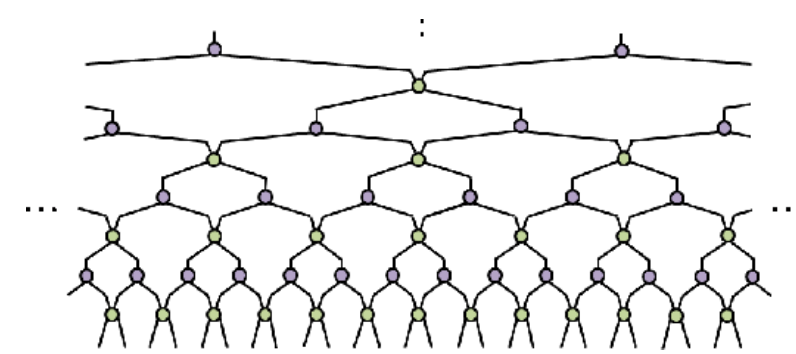
Boltzmann Machines



Disordered Ising Model



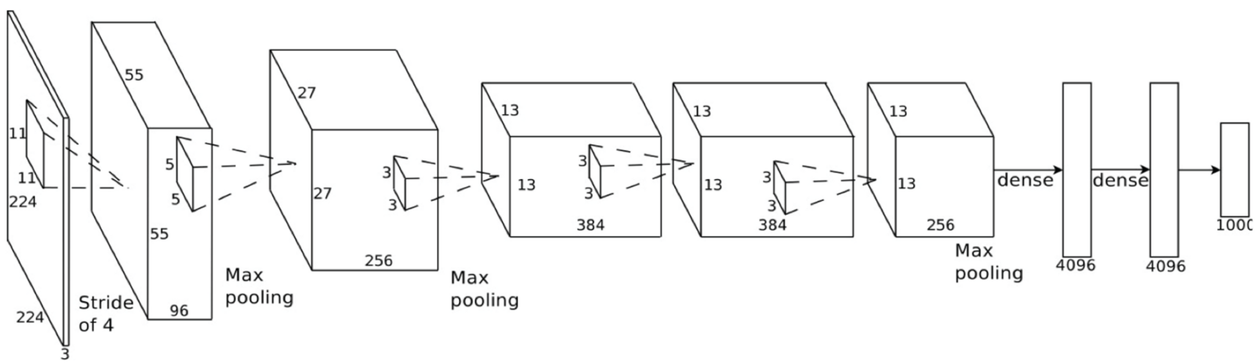
Deep Belief Networks

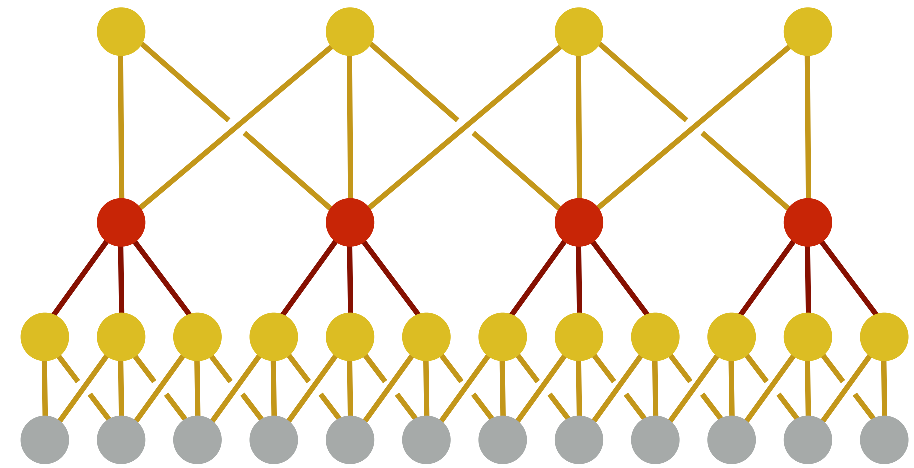


The "Renormalization Group"

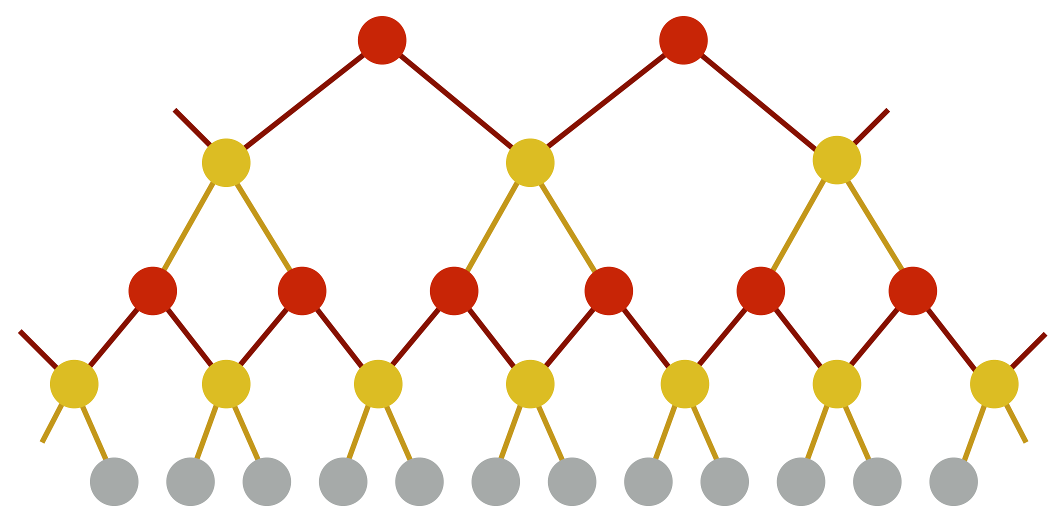
P. Mehta and D.J. Schwab, arxiv:1410.3831

Convolutional neural network

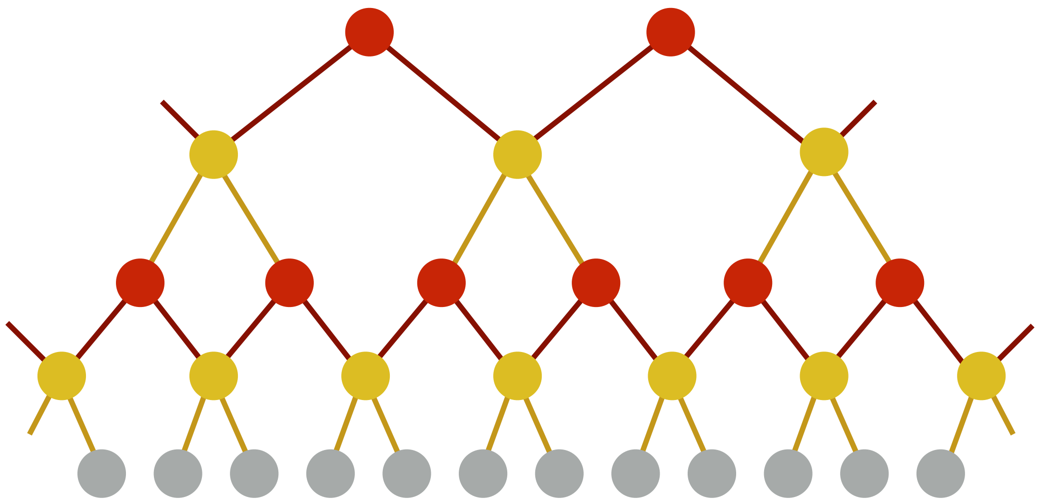




"MERA" tensor network



Are tensor networks useful for machine learning?



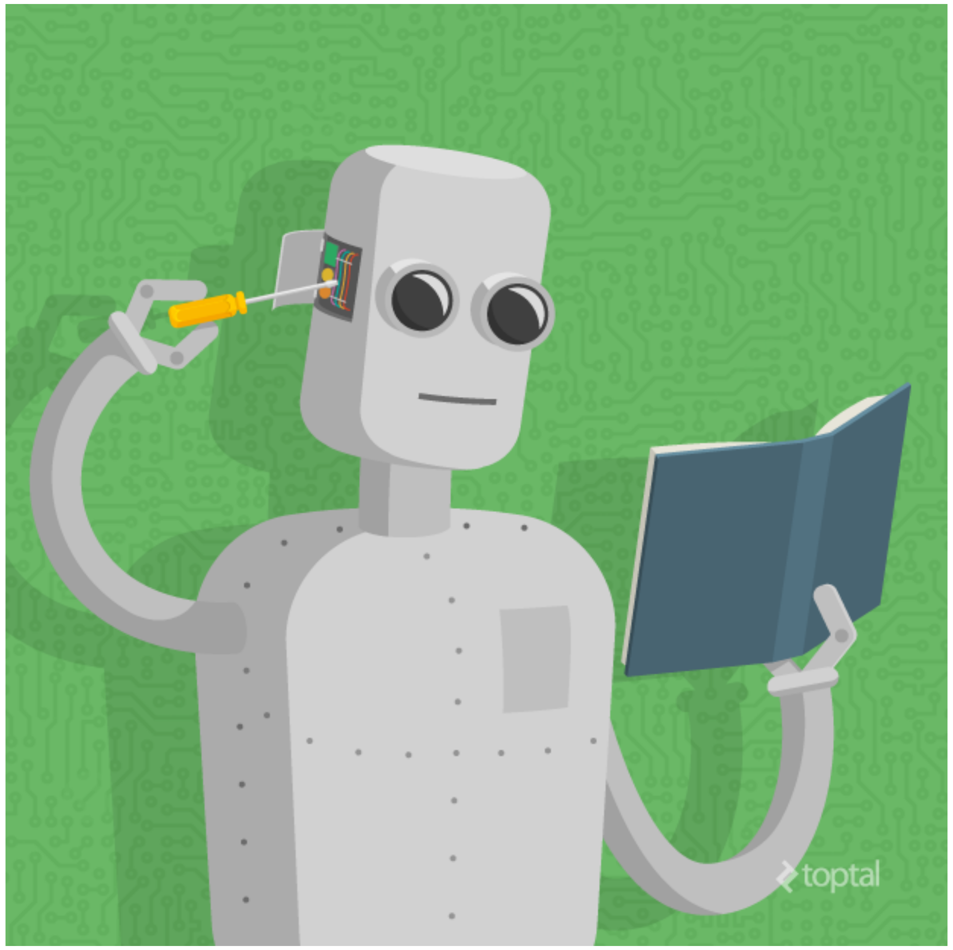
This Talk

Tensor networks can represent weights of useful and interesting machine learning models

Benefits include:

- Linear scaling
- Adaptive optimization
- Hybrid unsupervised / supervised 

What is machine learning?



Goal: train model $f(\mathbf{x})$

mapping input $\mathbf{x}$ to target

$$f(\mathbf{x}) \longrightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$ "dog"

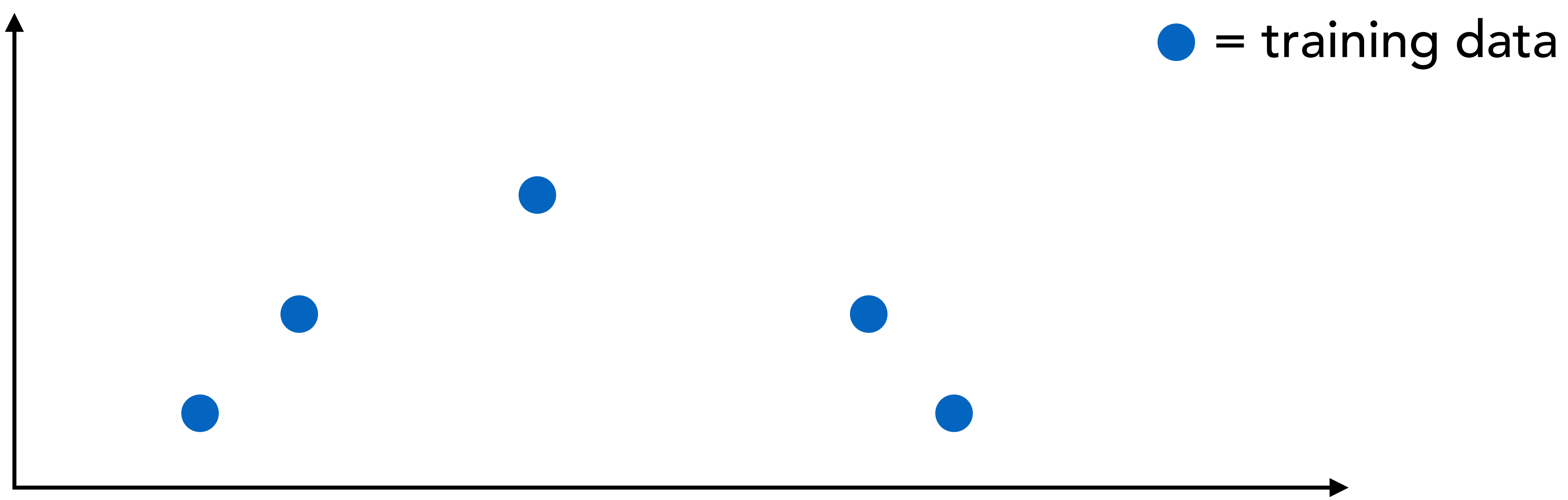
Goal: train model $f(\mathbf{x})$

mapping input $\mathbf{x}$ to target

$$f(\mathbf{x}) \longrightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ "cat"}$$

Philosophy of Machine Learning

- Map from images to labels is just a function
- Parameterize a set of very flexible functions
  (prefer convenient functions over "correct" ones)
- Prevent overfitting by regularization (prefer simple functions)

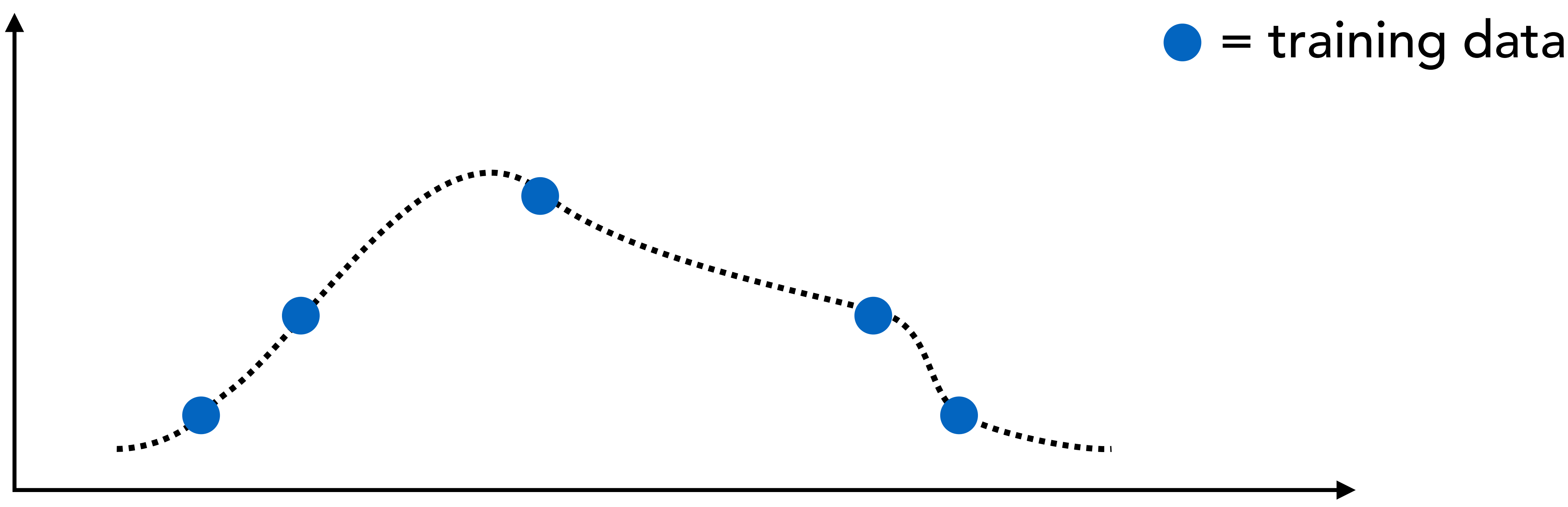


Philosophy of Machine Learning

- Map from images to labels is just a function

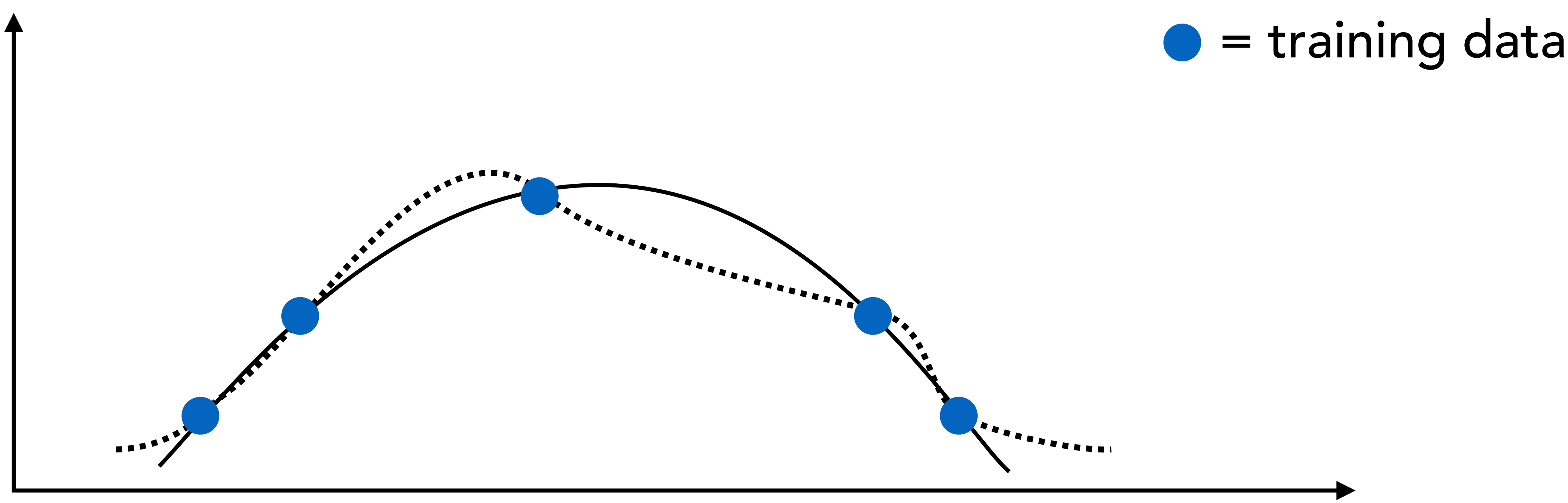
- Parameterize a set of very flexible functions
(prefer convenient functions over "correct" ones)

- Prevent overfitting by regularization (prefer simple functions)



Philosophy of Machine Learning

- Map from images to labels is just a function
- Parameterize a set of very flexible functions (prefer convenient functions over "correct" ones)
- Prevent overfitting by regularization (prefer simple functions)

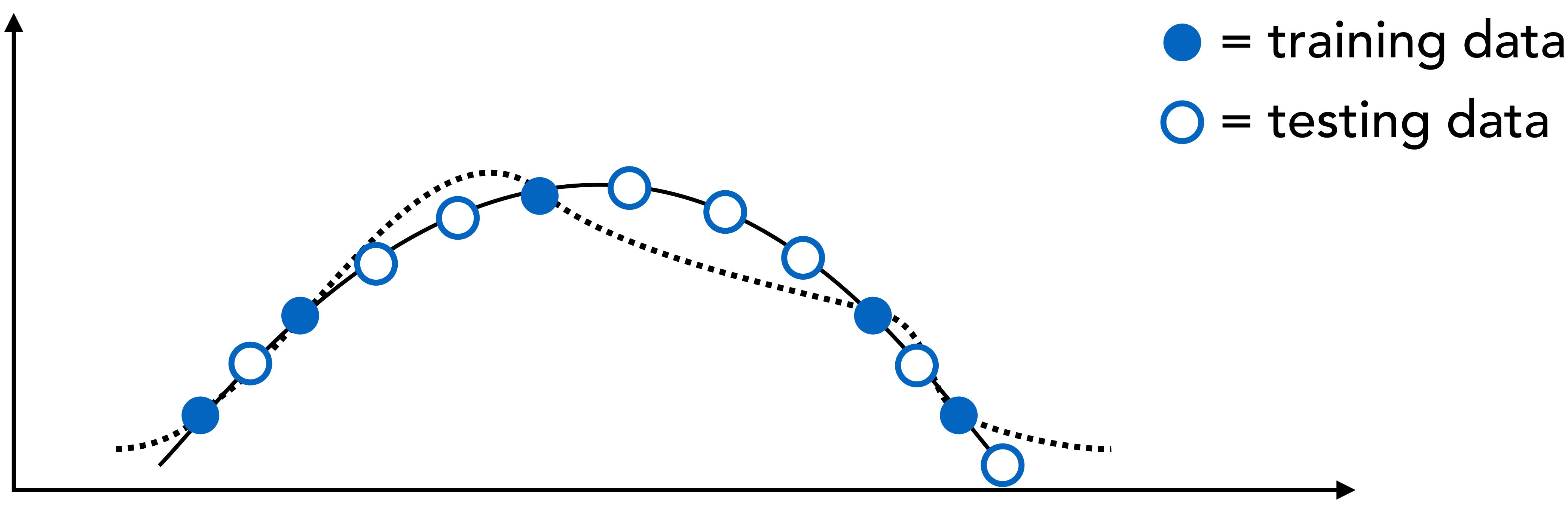


Philosophy of Machine Learning

- Map from images to labels is just a function

- Parameterize a set of very flexible functions
  (prefer convenient functions over "correct" ones)

- Prevent overfitting by regularization (prefer simple functions)



Supervised Learning

Given labeled training data (labels A and B)

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

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

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

Given training set $\{\mathbf{x}_j\}$, minimize cost function

$$C = \frac{1}{N_T} \sum_j (f(\mathbf{x}_j) - y_j)^2 \qquad y_j = \begin{cases} +1 & \mathbf{x}_j \in A \\ -1 & \mathbf{x}_j \in B \end{cases}$$

Unsupervised Learning

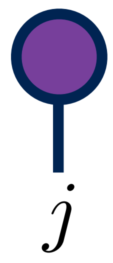
Given unlabeled training data $\{\mathbf{x}_j\}$

- Find function $f(\mathbf{x})$ such that $f(\mathbf{x}_j) \simeq p(\mathbf{x}_j)$
- Find function $f(\mathbf{x})$ such that $|f(\mathbf{x}_j)|^2 \simeq p(\mathbf{x}_j)$
- Find data clusters and which data belongs to each cluster
- Discover reduced representations of data for other learning tasks (e.g. supervised)

*Han et al., "**Unsupervised Generative Modeling Using Matrix Product States**" arxiv:1709.01662*

Tensor Network Machine Learning

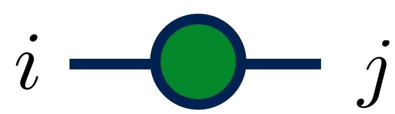
Tensor diagram notation



$$v_j$$

Tensor diagram notation

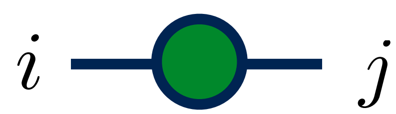
$$v_j$$



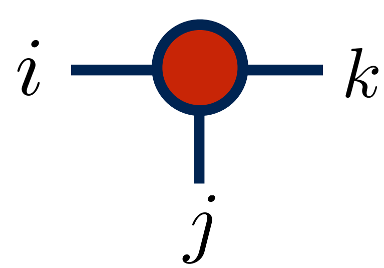
$$M_{ij}$$

Tensor diagram notation

$$v_j$$

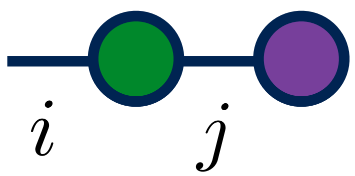


$$M_{ij}$$



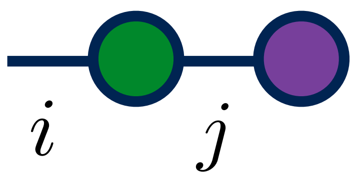
$$T_{ijk}$$

Joining lines implies contraction, can omit names

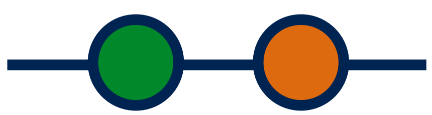


$$\longleftrightarrow \qquad \sum_j M_{ij} v_j$$

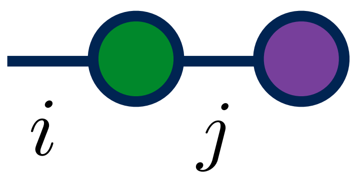
Joining lines implies contraction, can omit names



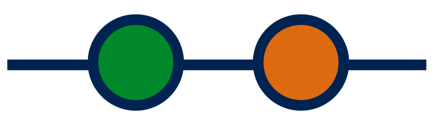
$$\longleftrightarrow \quad \sum_j M_{ij} v_j$$



Joining lines implies contraction, can omit names

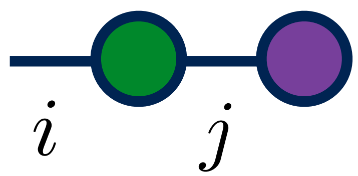


$$\longleftrightarrow \qquad \sum_j M_{ij} v_j$$

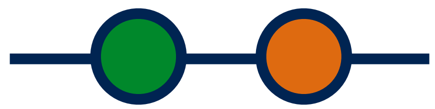


$$\longleftrightarrow \qquad A_{ij} B_{jk} \quad = AB$$

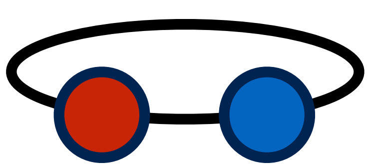
Joining lines implies contraction, can omit names



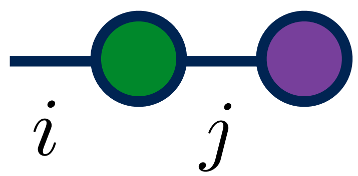
$$\longleftrightarrow \quad \sum_j M_{ij} v_j$$



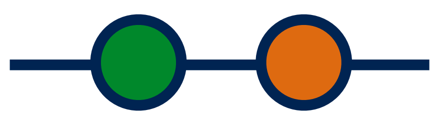
$$\longleftrightarrow \quad A_{ij} B_{jk} \quad = AB$$



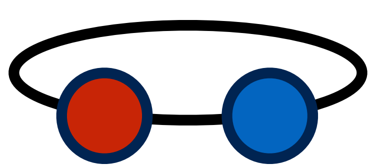
Joining lines implies contraction, can omit names



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



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



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

Raw data vectors

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

Example: grayscale images, components of $\mathbf{x}$ are pixels

$$x_j \in [0, 1]$$

Propose following model

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

$$= \sum_{\mathbf{s}} W_{s_1 s_2 s_3 \cdots s_N} \; x_1^{s_1} x_2^{s_2} x_3^{s_3} \cdots x_N^{s_N} \qquad\qquad s_j = 0, 1$$

Weights are N-index tensor
Like N-site wavefunction

Cohen et al. arxiv:1509.05009
Novikov, Trofimov, Oseledets, arxiv:1605.03795
Stoudenmire, Schwab, arxiv:1605.05775

N=3 example:

$$f(\mathbf{x}) = W \cdot \Phi(\mathbf{x}) = \sum_{\mathbf{s}} W_{s_1 s_2 s_3}\ x_1^{s_1} x_2^{s_2} x_3^{s_3}$$

$$= W_{000} + W_{100}\, x_1 + W_{010}\, x_2 + W_{001}\, x_3$$

$$+ W_{110}\, x_1 x_2 + W_{101}\, x_1 x_3 + W_{011}\, x_2 x_3$$

$$+ W_{111}\, x_1 x_2 x_3$$

Contains linear classifier, and various poly. kernels

More generally, apply local "feature maps" $\phi^{s_j}(x_j)$

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

$$= \sum_{\mathbf{s}} W_{s_1 s_2 s_3 \cdots s_N} \phi^{s_1}(x_1) \phi^{s_2}(x_2) \phi^{s_3}(x_3) \cdots \phi^{s_N}(x_N)$$

Highly expressive

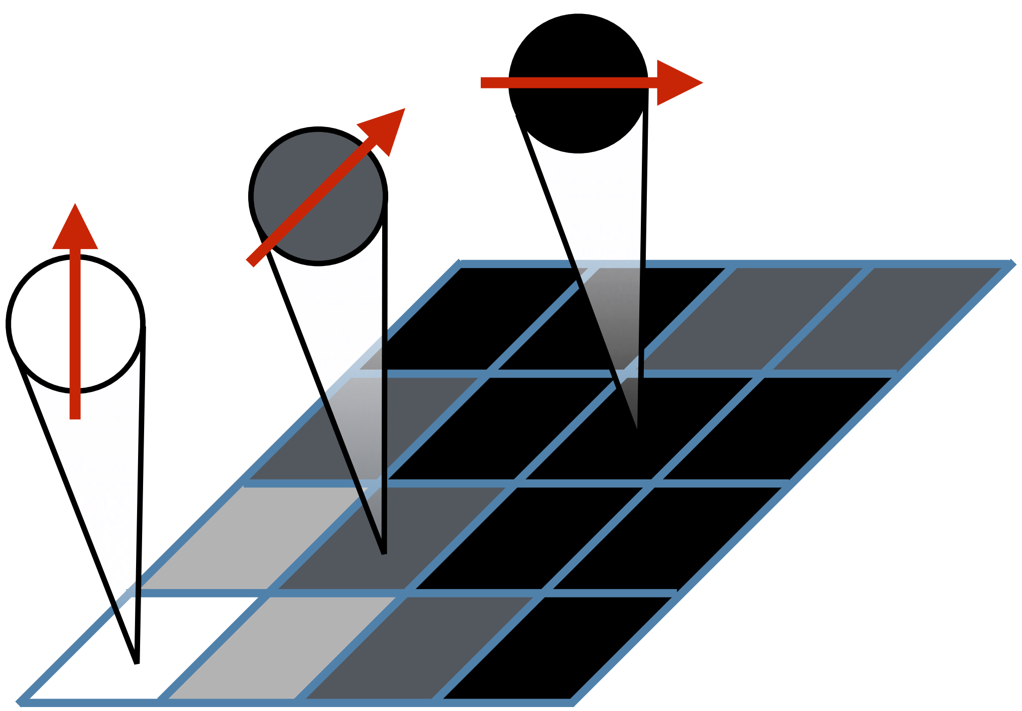
Could put additional parameters into maps ϕ

$\mathbf{x} =$ input

For example, following local feature map

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

Picturesque idea of pixels as "spins"



$\mathbf{x} =$ 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

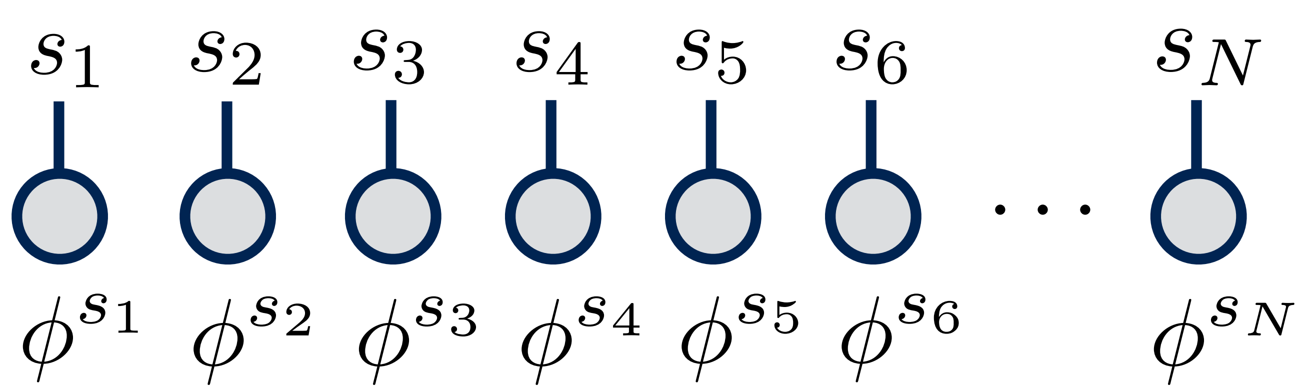
Total feature map $\Phi(\mathbf{x})$

Tensor diagram notation

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

raw inputs

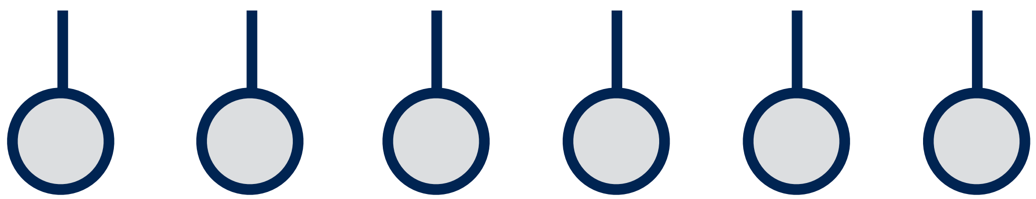
$\Phi(\mathbf{x}) =$



feature vector

Construct decision function

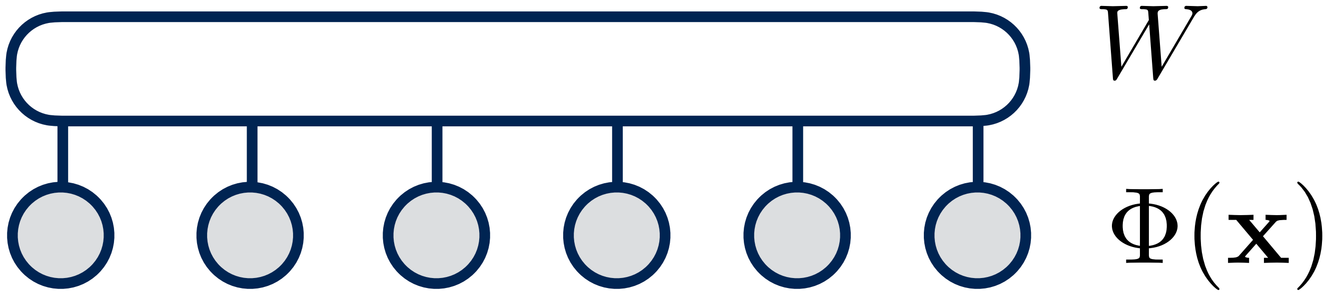
$$\boxed{f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})}$$



$\Phi(\mathbf{x})$

Construct decision function

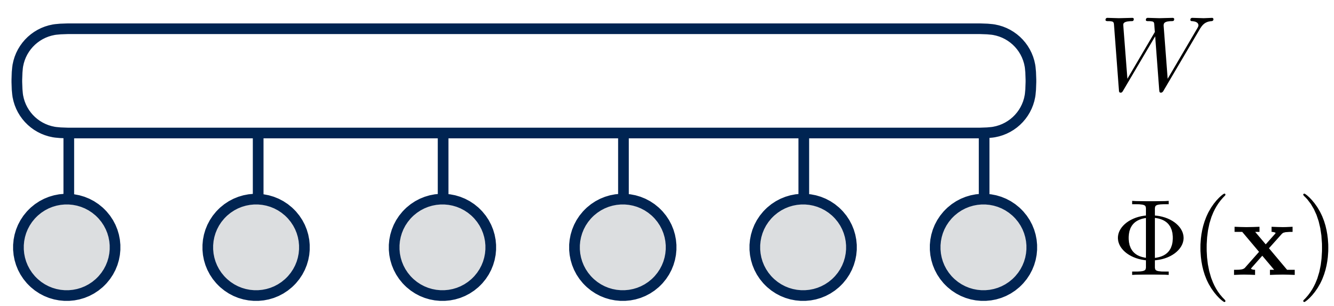
$$\boxed{f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})}$$



Construct decision function

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

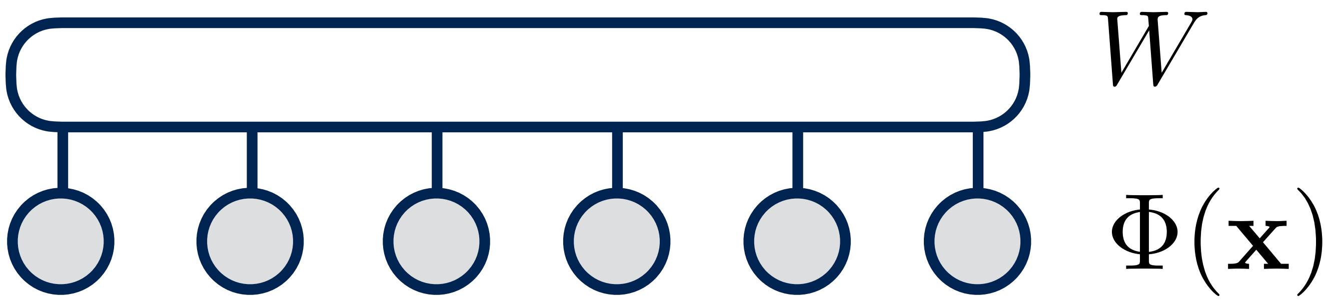
$$f(\mathbf{x}) \;=$$



Construct decision function

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

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



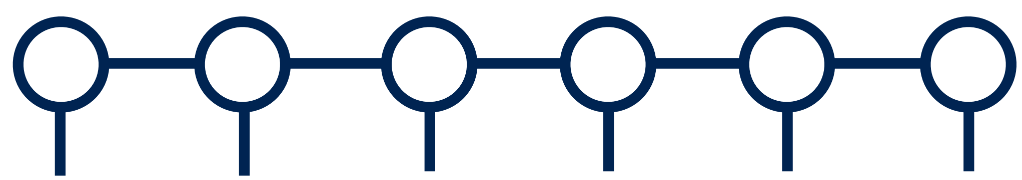
$$W =$$

Main approximation

$W =$

order-N tensor

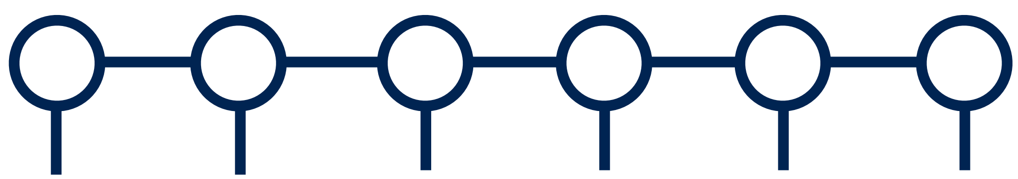
$\approx$



matrix product state (MPS)

Main approximation

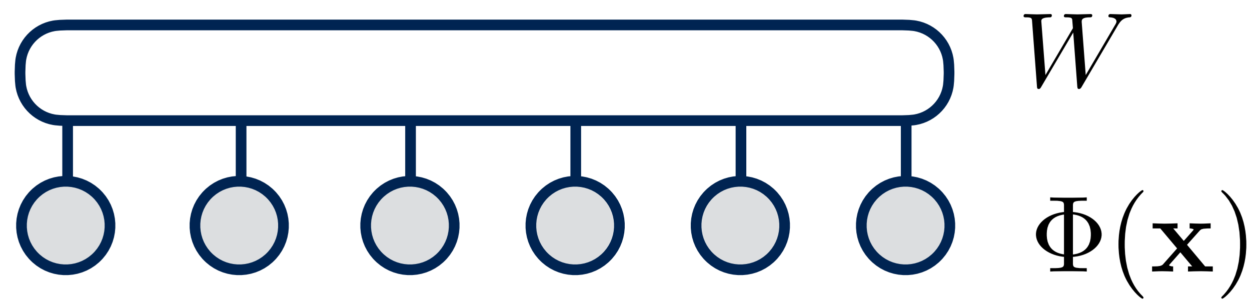
$W =$  *order-N tensor*

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

$\Big($ $\approx$  *PEPS* $\Big)$

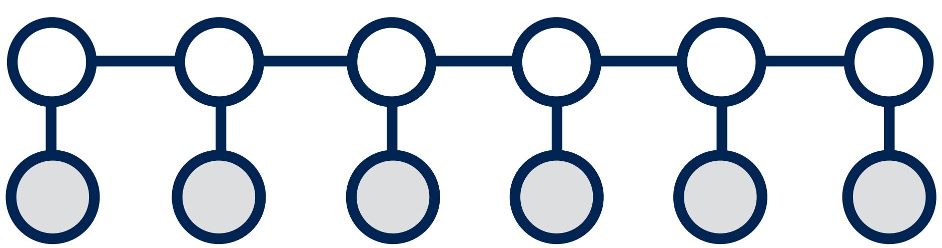
Tensor diagrams of the approach

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



$$\approx \; (M_{s_1} M_{s_2} \cdots M_{s_N}) \Phi^{s_1 s_2 \cdots s_N}(\mathbf{x})$$

$$\approx$$



Linear scaling

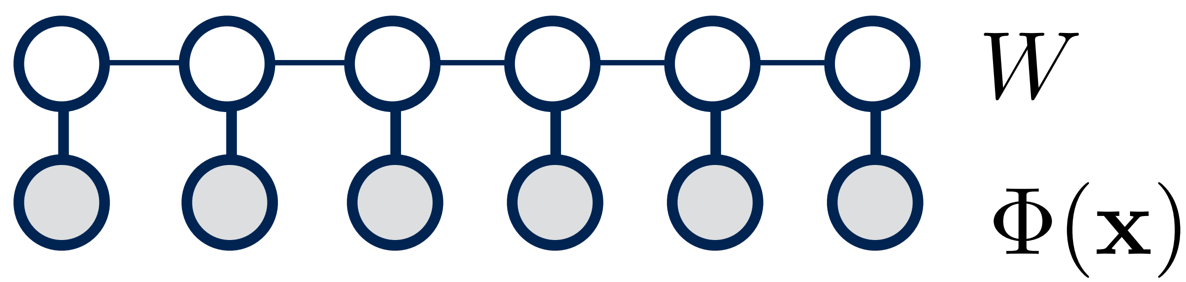
Can use algorithm similar to DMRG to optimize

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

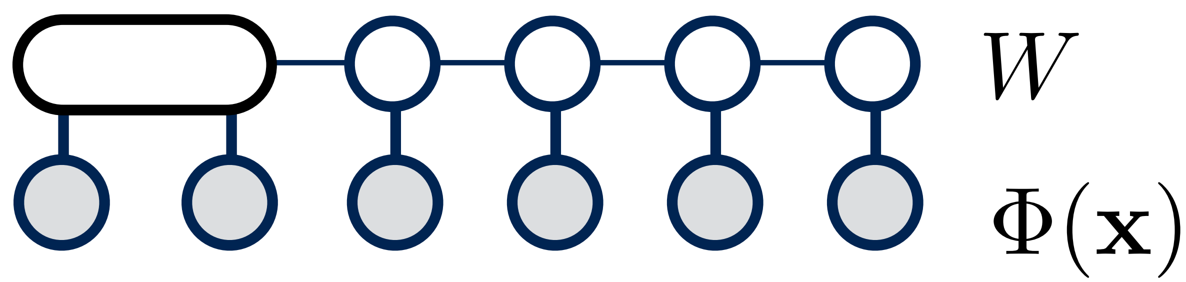
Scaling is $N \cdot N_T \cdot m^3$

$N =$ 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

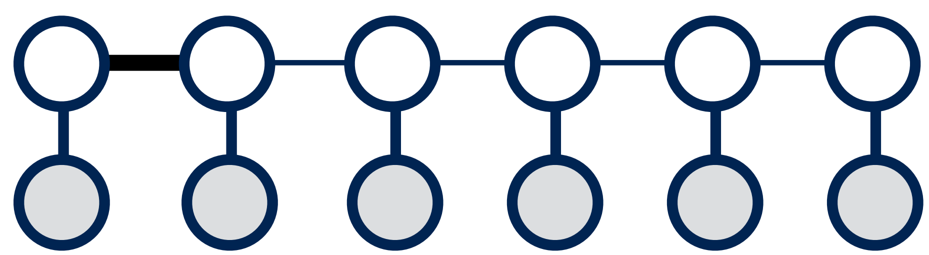
Scaling is $N \cdot N_T \cdot m^3$

$N =$ 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

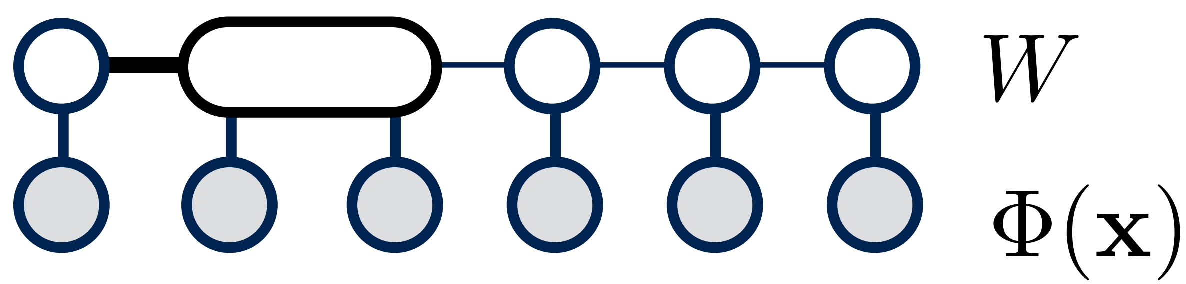
Scaling is $N \cdot N_T \cdot m^3$

$N =$ size of input

$N_T =$ size of training set

$m =$ MPS bond dimension

$f(\mathbf{x}) =$



Linear scaling

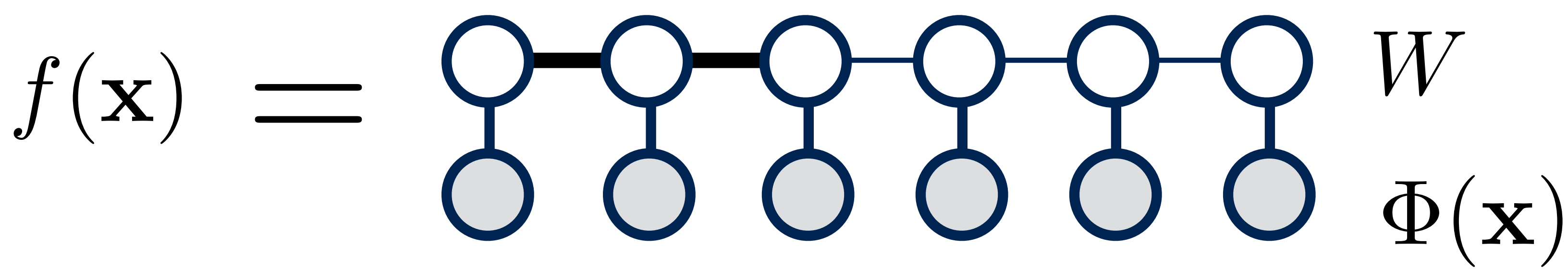
Can use algorithm similar to DMRG to optimize

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

$N =$ size of input

$N_T =$ size of training set

$m =$ MPS bond dimension



Linear scaling

Can use algorithm similar to DMRG to optimize

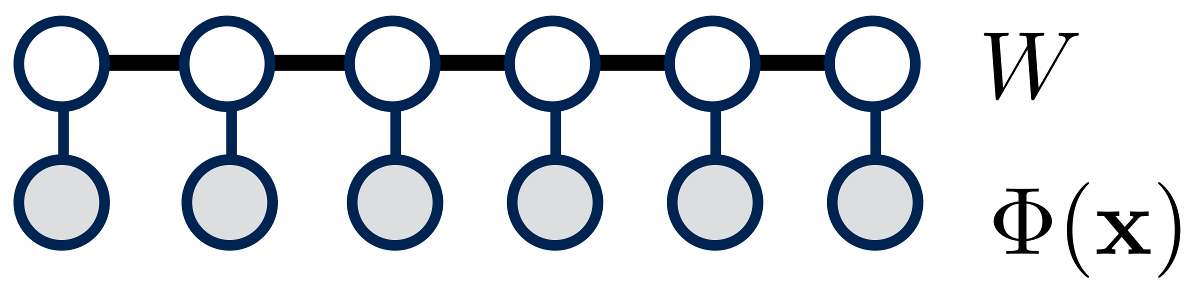
Scaling is $N \cdot N_T \cdot m^3$

$N =$ size of input

$N_T =$ size of training set

$m =$ MPS bond dimension

$f(\mathbf{x}) =$



Why should this work at all?

Linear classifier $f(\mathbf{x}) = V \cdot \mathbf{x}$ exactly m=2 MPS

$$W = \begin{bmatrix} V_0 & 1 \end{bmatrix} \begin{bmatrix} \hat{1} & 0 \\ \hat{V}_1 & \hat{1} \end{bmatrix} \begin{bmatrix} \hat{1} & 0 \\ \hat{V}_2 & \hat{1} \end{bmatrix} \begin{bmatrix} \hat{1} & 0 \\ \hat{V}_3 & \hat{1} \end{bmatrix} \cdots$$

$$\hat{1} = [1 \;\; 0]$$

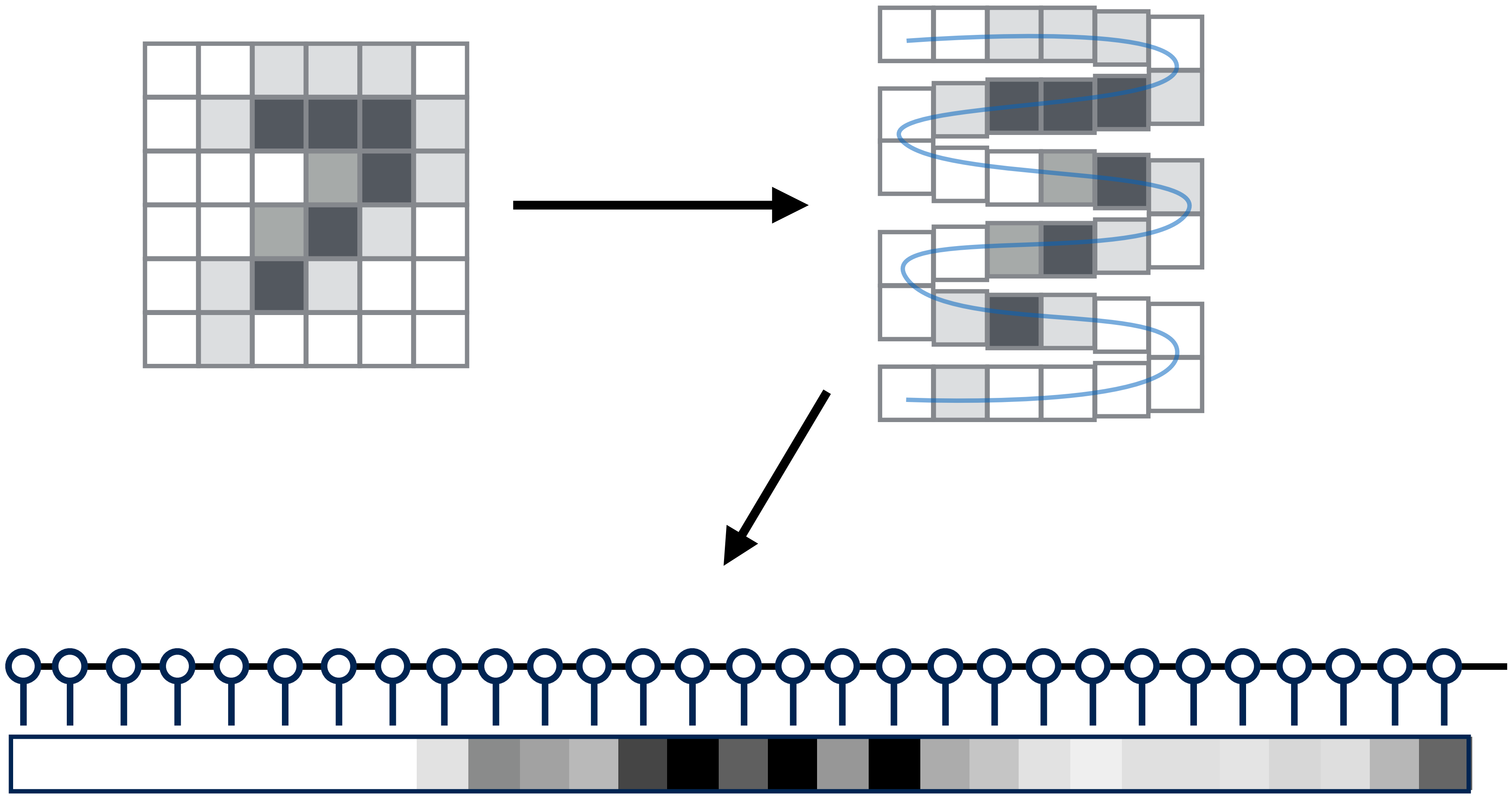
$$\hat{V}_j = [0 \;\; V_j]$$

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

$$\phi^{s_j}(x_j) = [1\,,\; x_j]$$

Novikov, Trofimov, Oseledets, arxiv:1605.03795

Experiment: handwriting classification (MNIST)



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

Papers using tensor network machine learning

Expressivity & priors of TN based models

- *Levine et al., "**Deep Learning and Quantum Entanglement: Fundamental Connections with Implications to Network Design**" arxiv:1704.01552*
- *Cohen, Shashua, "**Inductive Bias of Deep Convolutional Networks through Pooling Geometry**" arxiv:1605.06743*
- *Cohen et al., "**On the Expressive Power of Deep Learning: A Tensor Analysis**" arxiv:1509.05009*

Generative Models

- *Han et al., "**Unsupervised Generative Modeling Using Matrix Product States**" arxiv:1709.01662*
- *Sharir et al., "**Tractable Generative Convolutional Arithmetic Circuits**" arxiv:1610.04167*

Supervised Learning

- *Novikov et al., "**Expressive power of recurrent neural networks**", arxiv:1711.00811*
- *Liu et al., "**Machine Learning by Two-Dimensional Hierarchical Tensor Networks: A Quantum Information Theoretic Perspective on Deep Architectures**", arxiv:1710.04833*
- *Stoudenmire, Schwab, "**Supervised Learning with Quantum-Inspired Tensor Networks**", arxiv:1605.05775*
- *Novikov et al., "**Exponential Machines**", arxiv: 1605.03795*

Related uses of tensor networks

Compressing weights of neural nets (& other models)

Yu et al., Advances in Neural Information Processing (2017), arxiv:1711.00073

Izmailov et al., arxiv:1710.07324 (2017)

Yang et al., arxiv:1707.01786 (2017)

Garipov et al., arxiv:1611.03214 (2016)

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

Large scale linear algebra (PCA/SVD)

Lee, Cichocki, arxiv: 1410.6895 (2014)

Feature extraction & tensor completion

Bengua et al., arxiv:1606.01500, arxiv:1607.03967, arxiv:1609.04541 (2016)

Phien et al., arxiv:1601.01083 (2016)

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

Learning Relevant Features of Data

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

Given training data $\{\mathbf{x}_j\}$

Can show optimal W is of the form

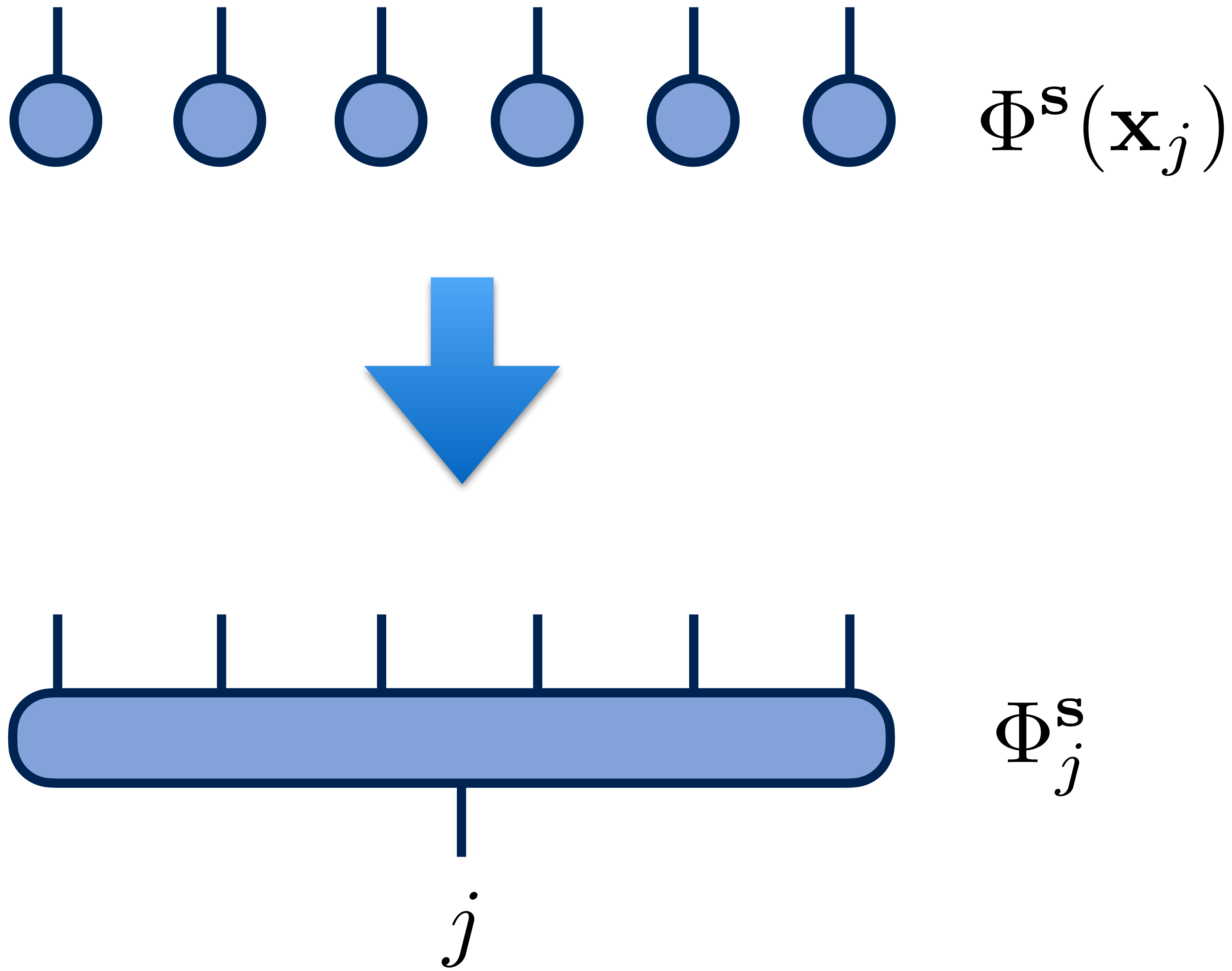
$$W = \sum_j \alpha_j \, \Phi(\mathbf{x}_j)$$

Holds for wide variety of cost functions / tasks

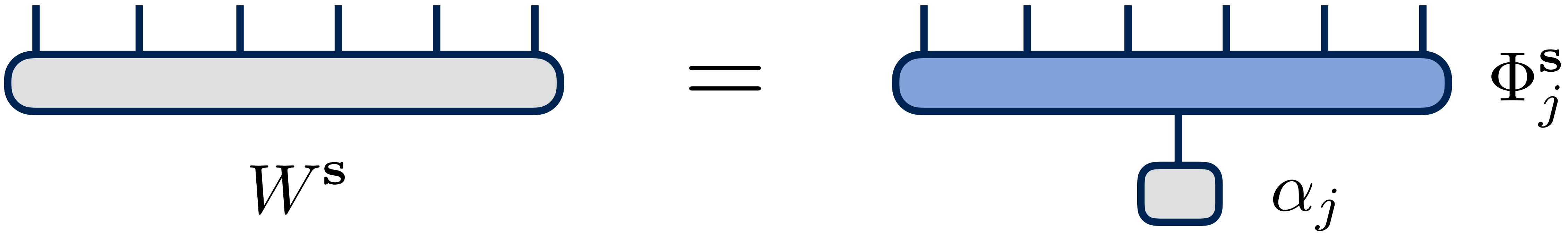
"representer theorem"

Schölkopf, Smola, Müller, Neural Comp. 10, 1299 (1998)

View $\Phi^{\mathbf{s}}(\mathbf{x}_j) = \Phi^{\mathbf{s}}_j$ as a tensor

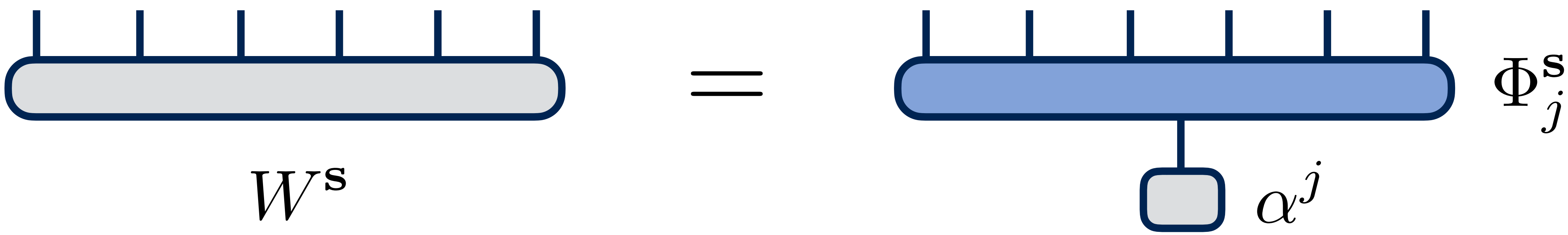


Representer theorem says

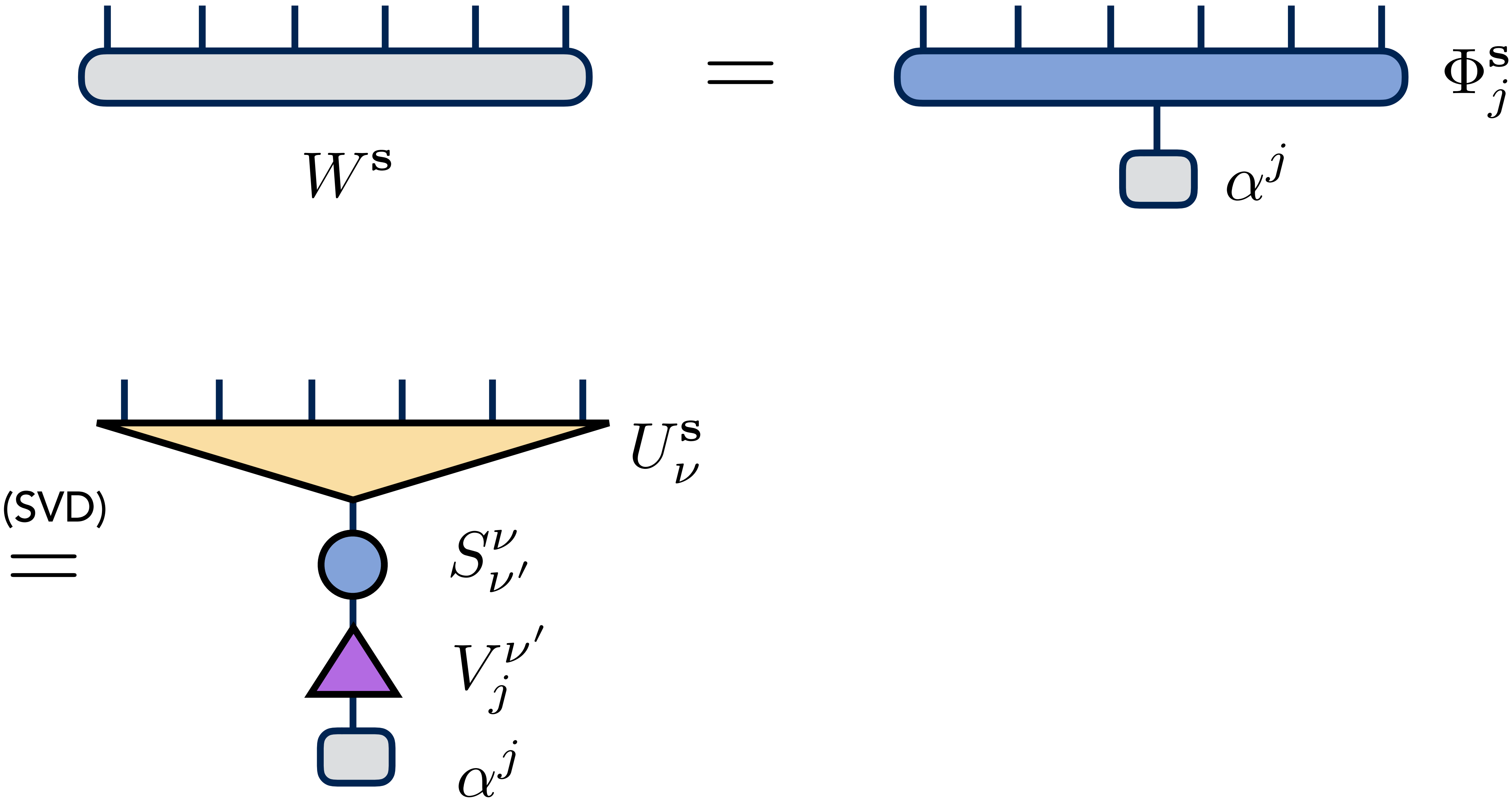


Really just says weights in the span of $\{\Phi_j^{\mathbf{s}}\}$

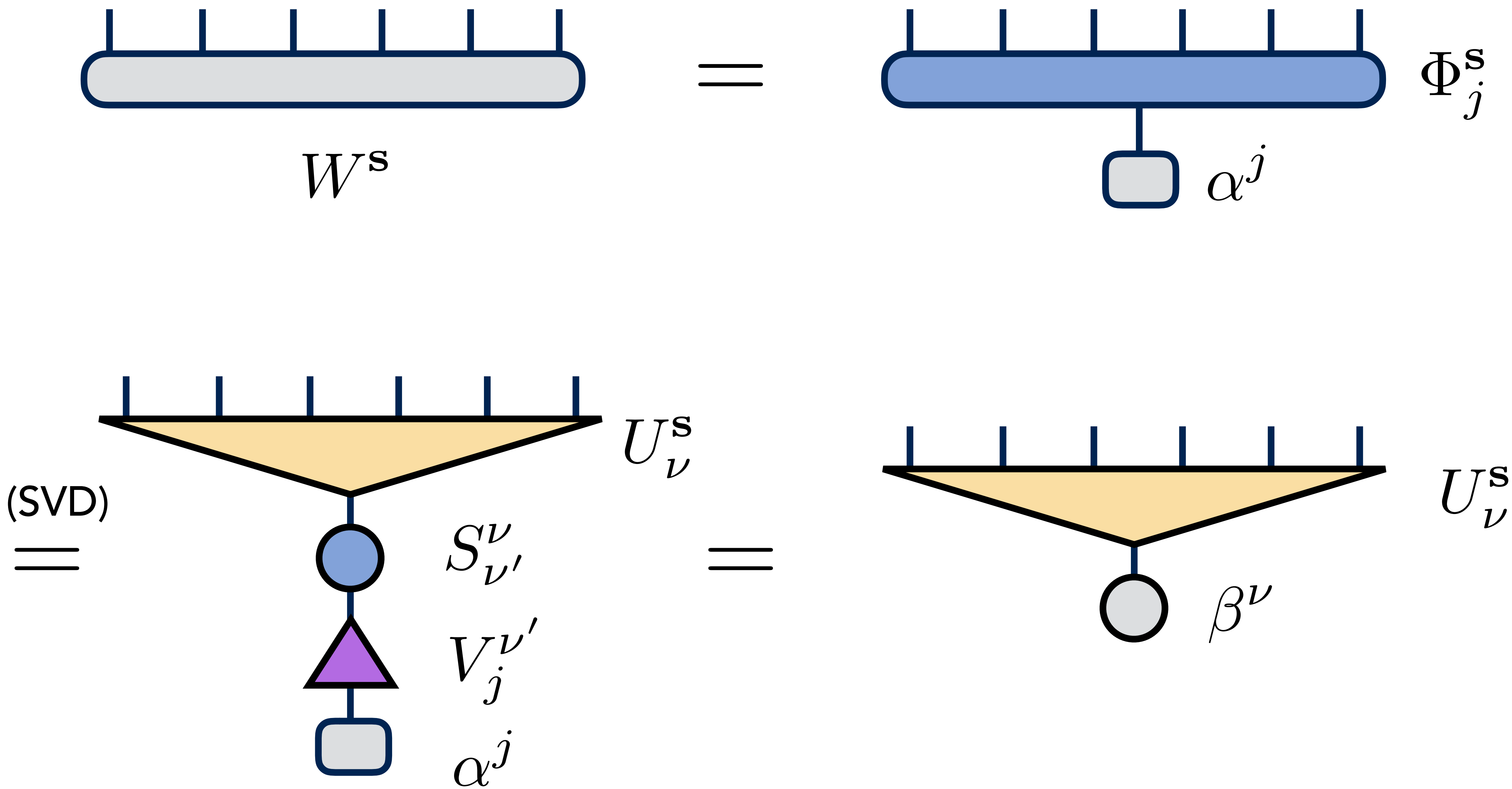
Can choose any basis for span of $\{\Phi_j^{\mathbf{s}}\}$



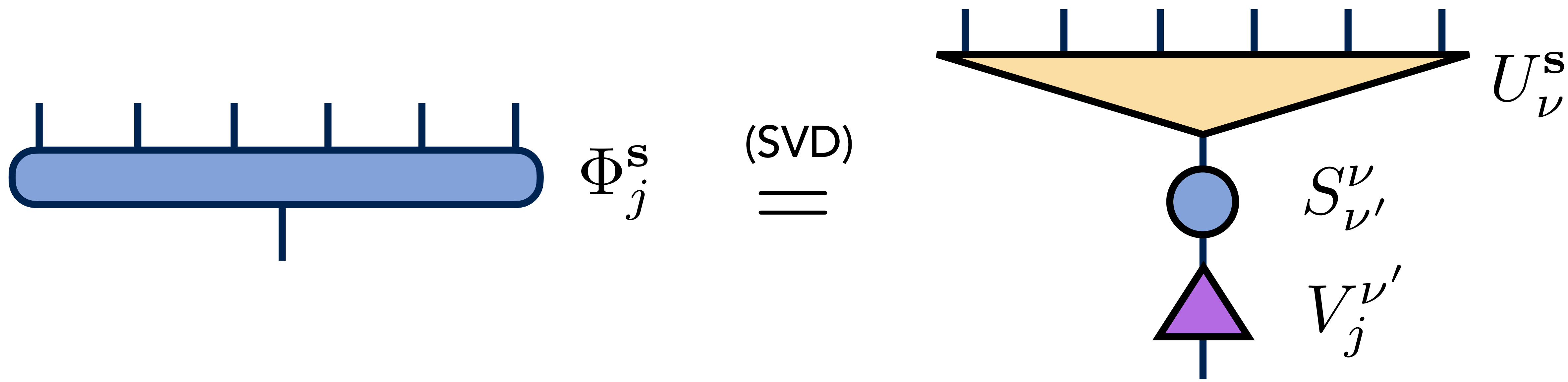
Can choose any basis for span of $\{\Phi_j^{\mathbf{s}}\}$



Can choose any basis for span of $\{\Phi^{\mathbf{s}}_j\}$



Why switch to $U^{\mathbf{s}}_{\nu}$ basis?



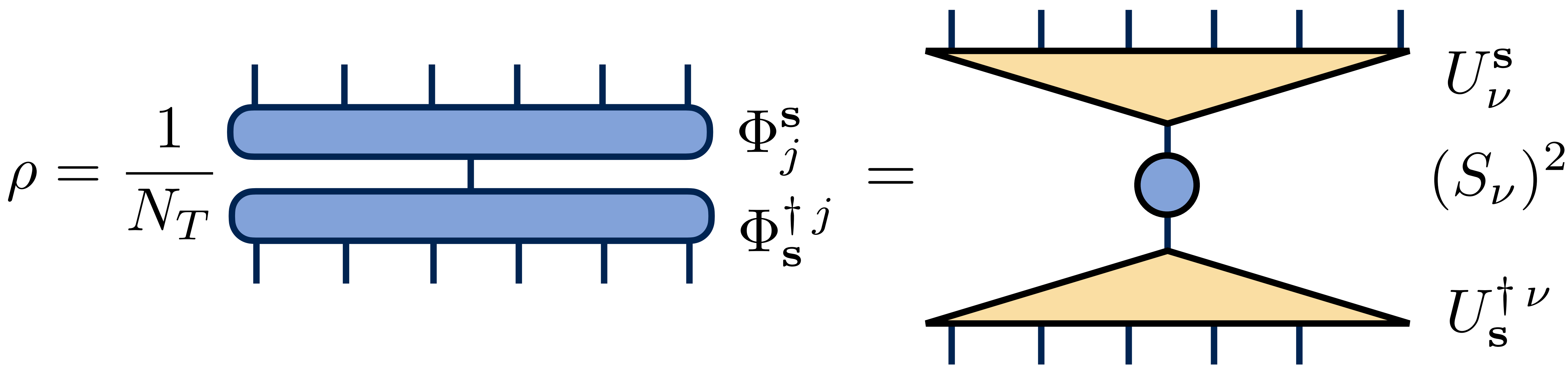
Orthonormal basis

Can discard basis vectors corresponding to small s. vals.

Can compute $U^{\mathbf{s}}_{\nu}$ fully or partially using *tensor networks*

Computing $U^{\mathbf{s}}_{\nu}$ efficiently

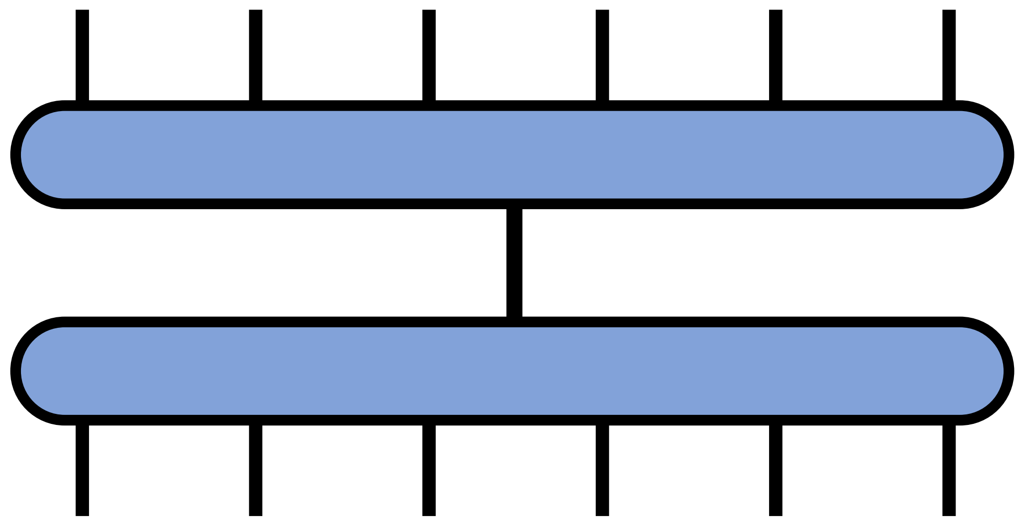
Define *feature space covariance matrix* (similar to density matrix)



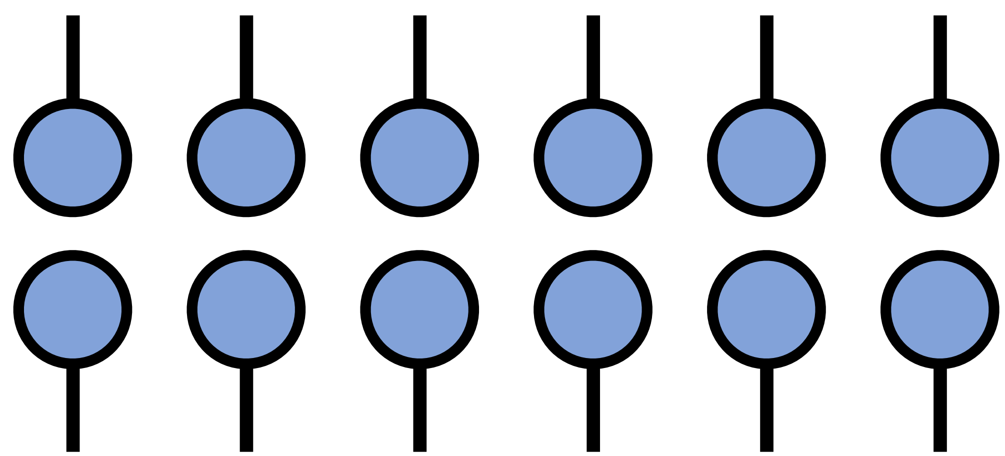
Strategy: compute $U^{\mathbf{s}}_{\nu}$ iteratively as a layered (tree) tensor network

For efficiency, exploit product structure of Φ

$$\rho = \Phi\Phi^\dagger = \frac{1}{N_T}$$



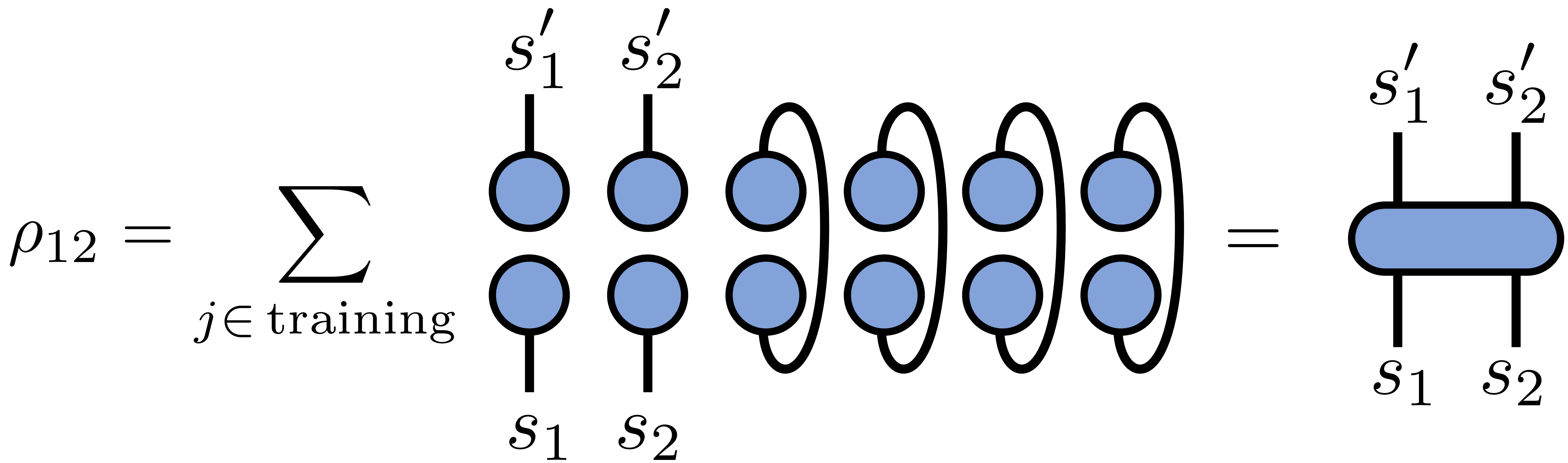
$$= \frac{1}{N_T}\sum_{j=1}^{N_T}$$

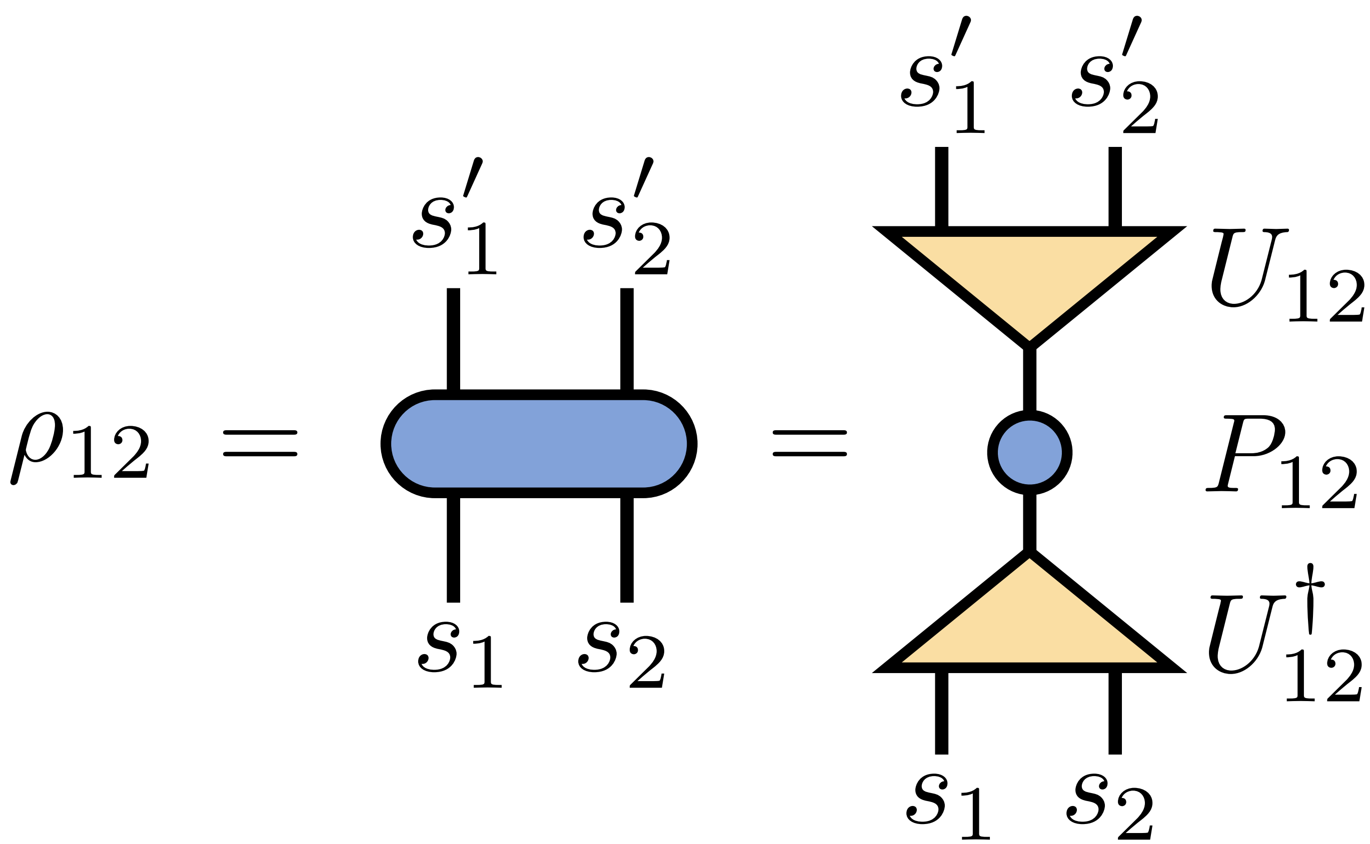


$\Phi(\mathbf{x}_j)$

$\Phi^\dagger(\mathbf{x}_j)$

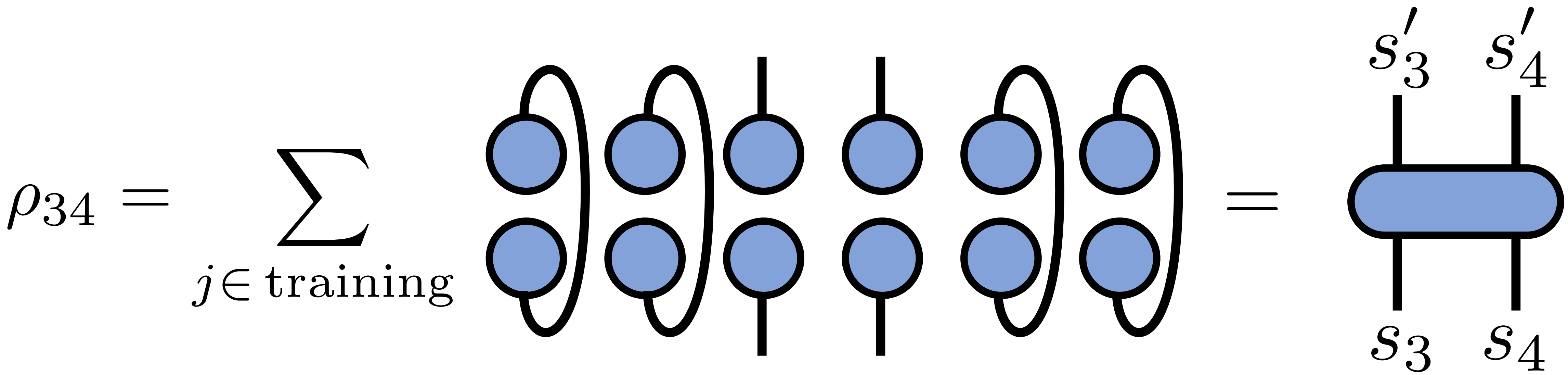
Compute tree tensors from reduced matrices

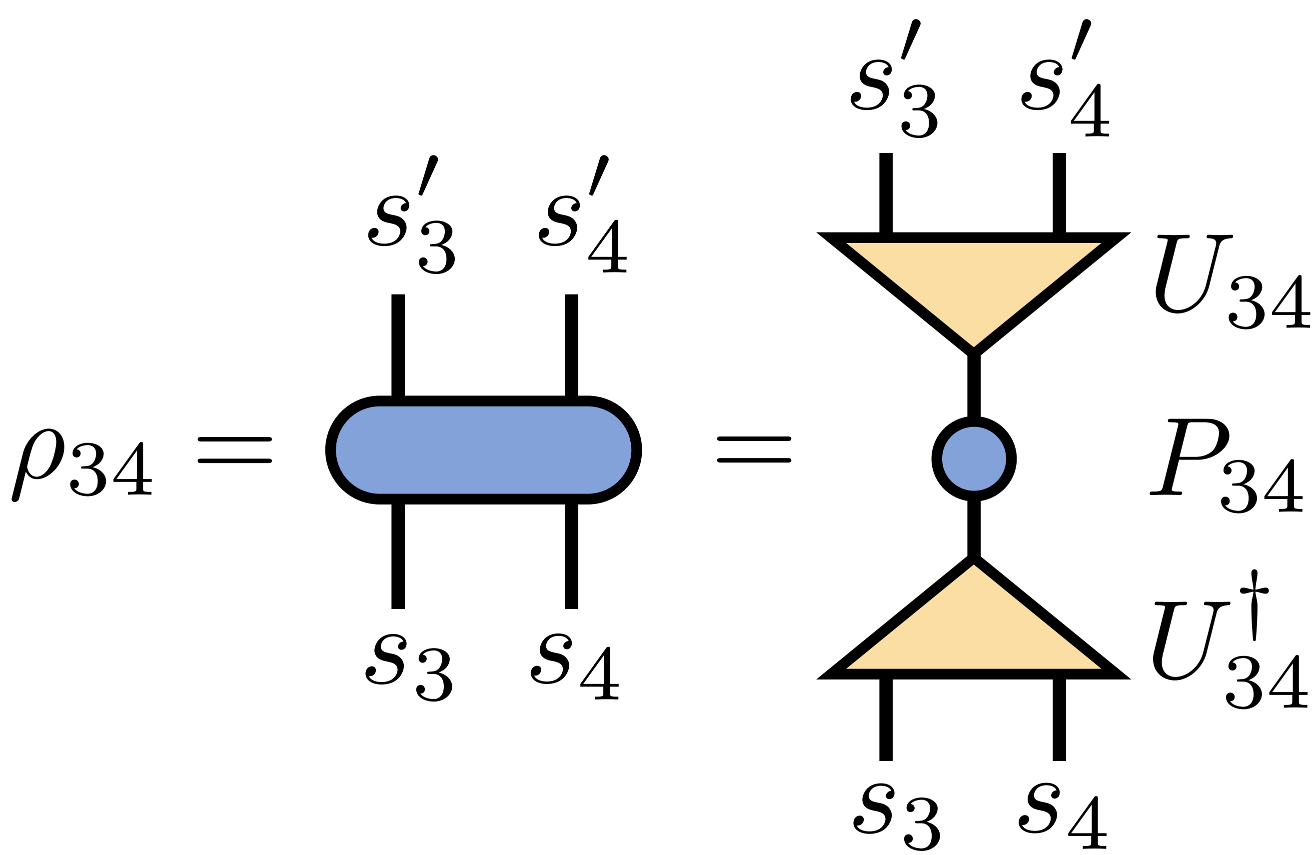




Truncate small eigenvalues

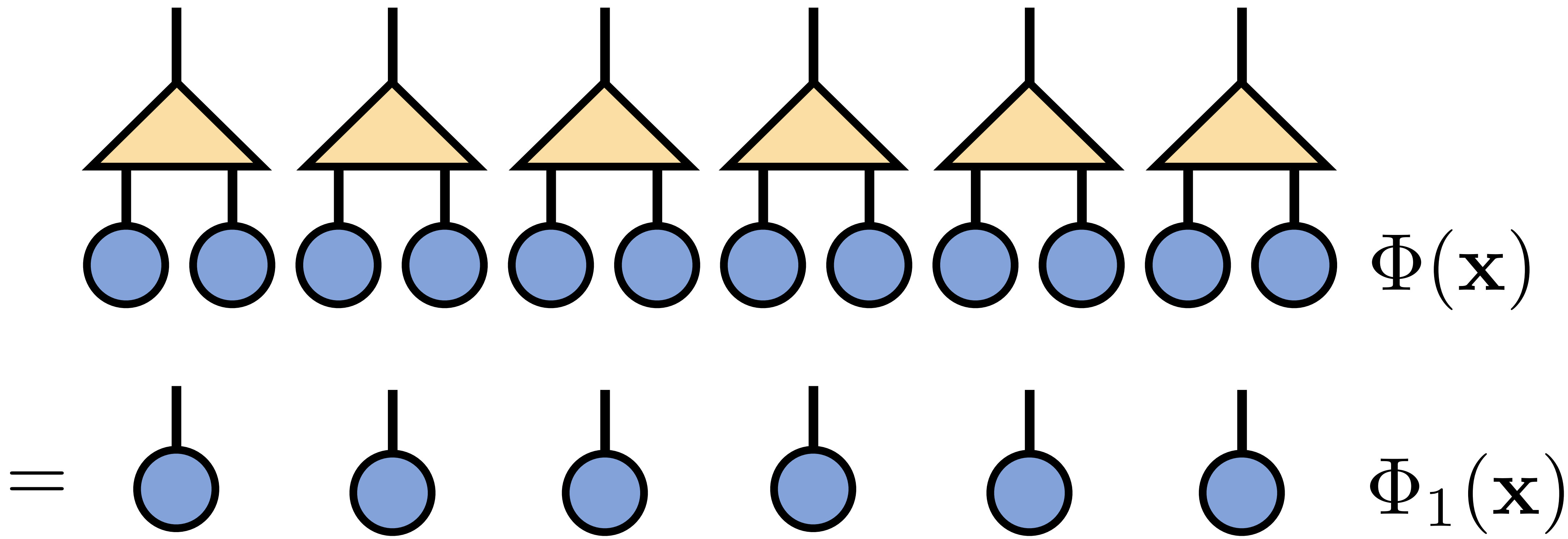
Compute tree tensors from reduced matrices





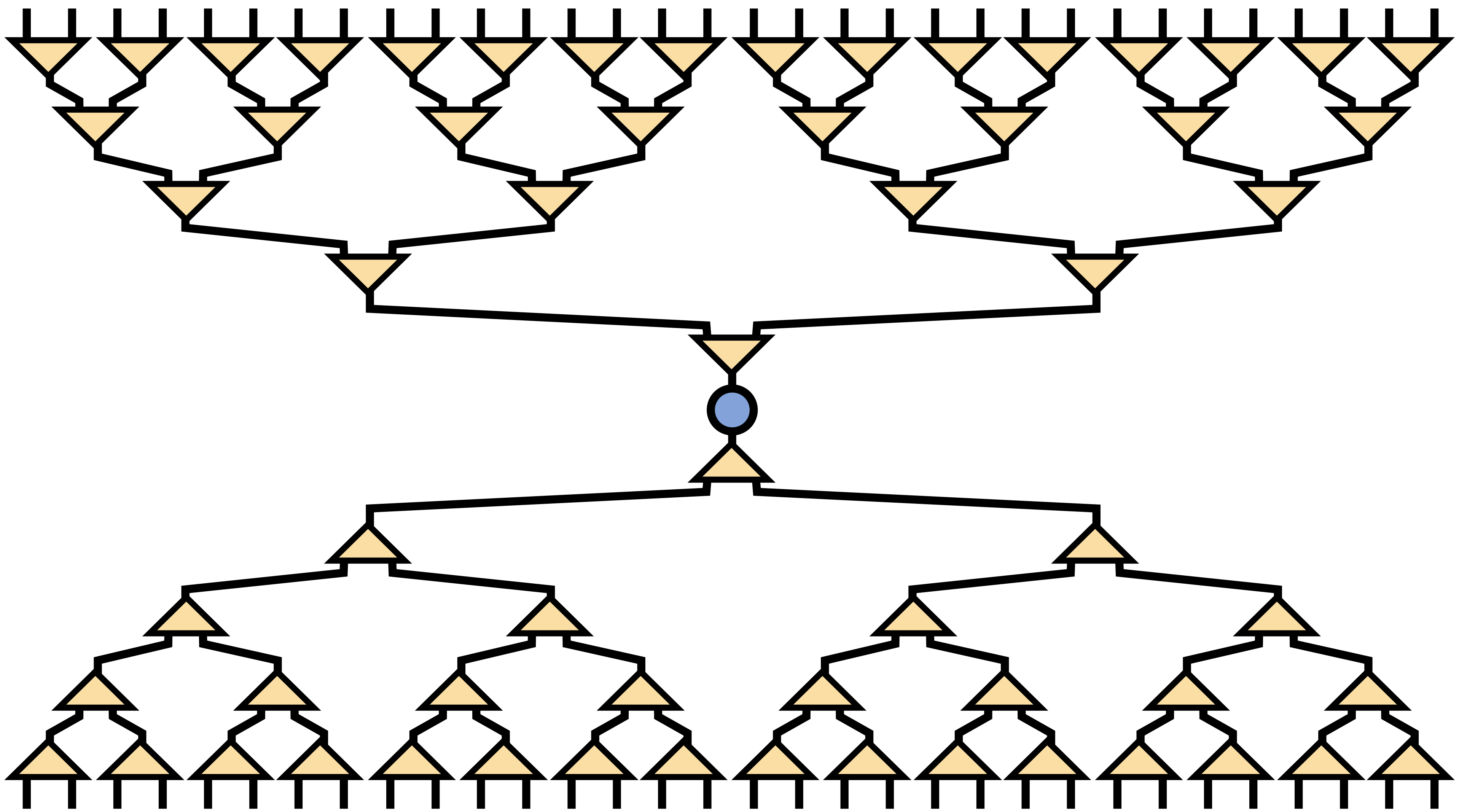
Truncate small eigenvalues

Having computed a tree layer, rescale data



With all layers, have approximately diagonalized ρ

$\rho \simeq$

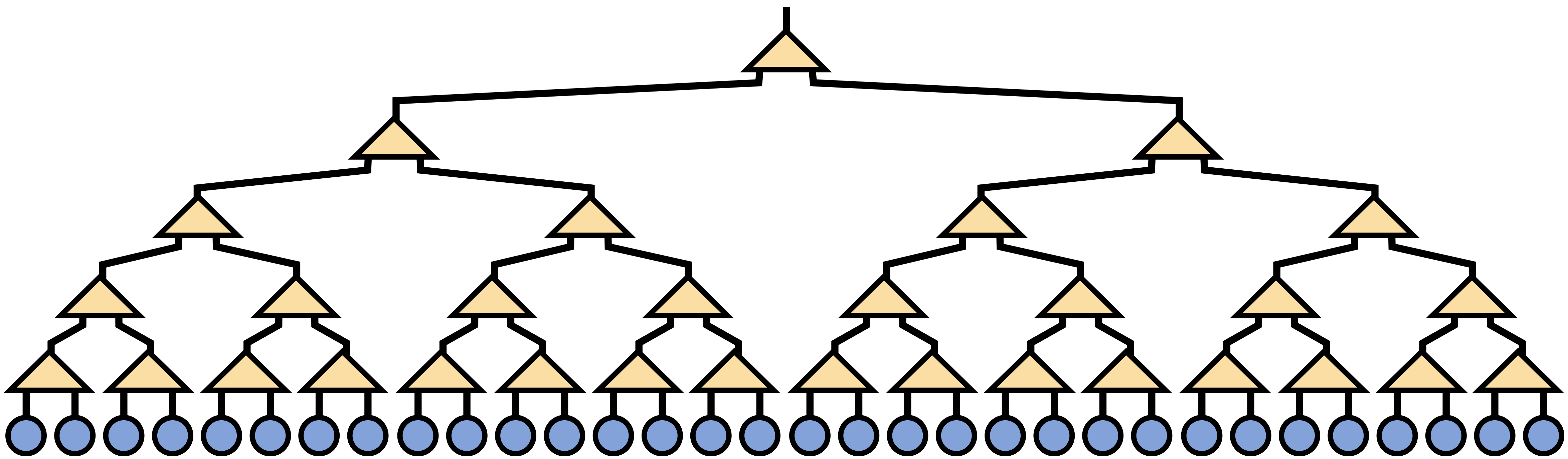


U

$U^\dagger$

Equivalent to *kernel PCA,*
but linear scaling with size of data set

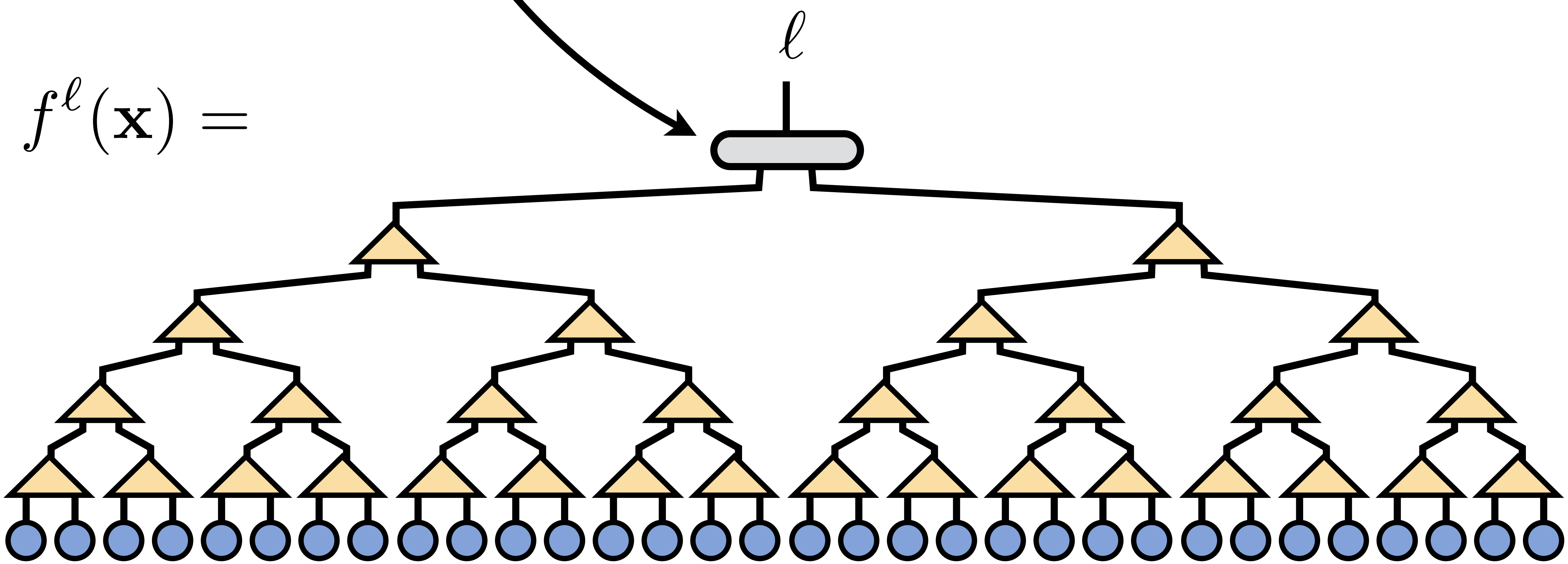
Can view as *unsupervised learning* of representation of training data



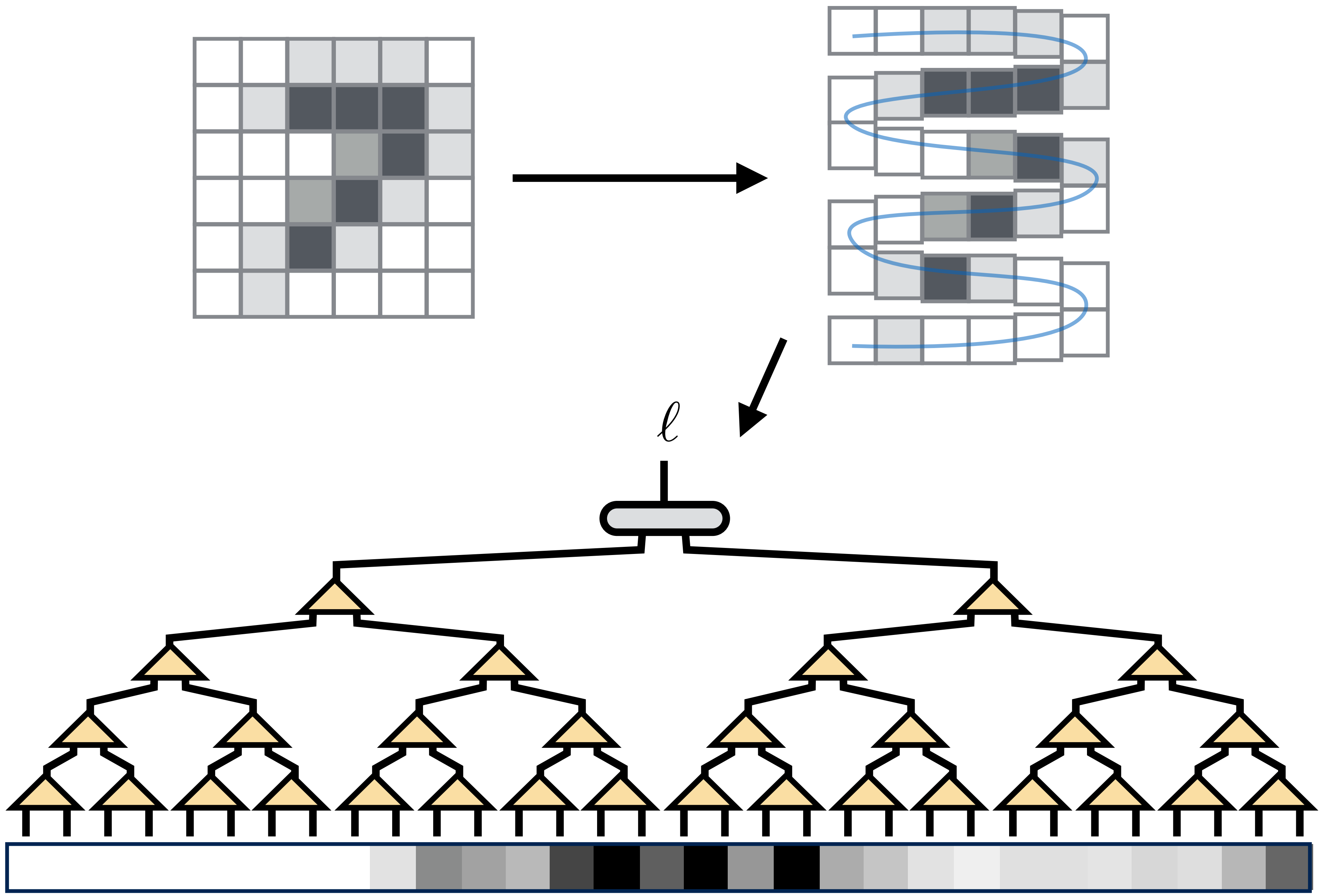
Use as starting point for supervised learning

Only train top tensor for supervised task

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



Experiment: handwriting classification (MNIST)



Cutoff 6x10^{-4} gave top indices sizes 328 and 444

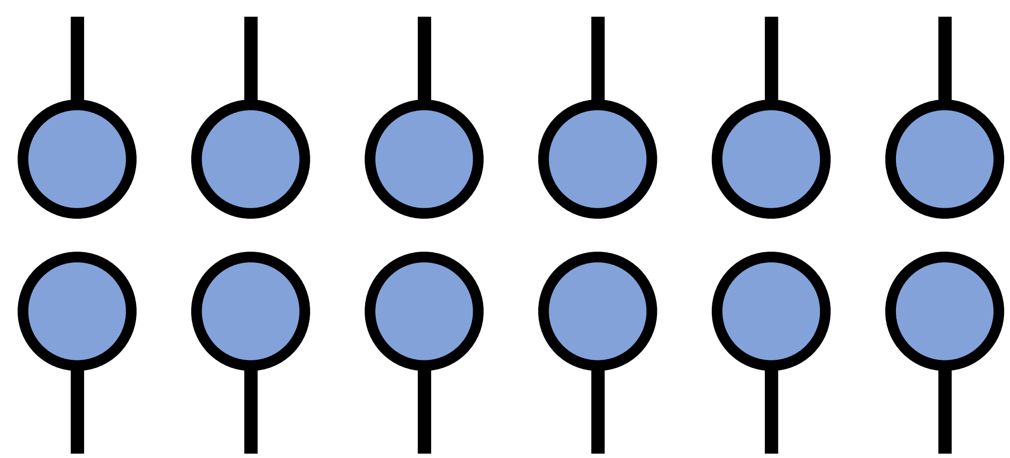
Training acc: 99.68%   **Test acc:** 98.08%

Refinements and Extensions

No reason we must base tree around ρ

Could reweight based on importance of samples

$$\tilde{\rho} = \frac{1}{N_T} \sum_{j=1}^{N_T} {\color{red} w_j}$$



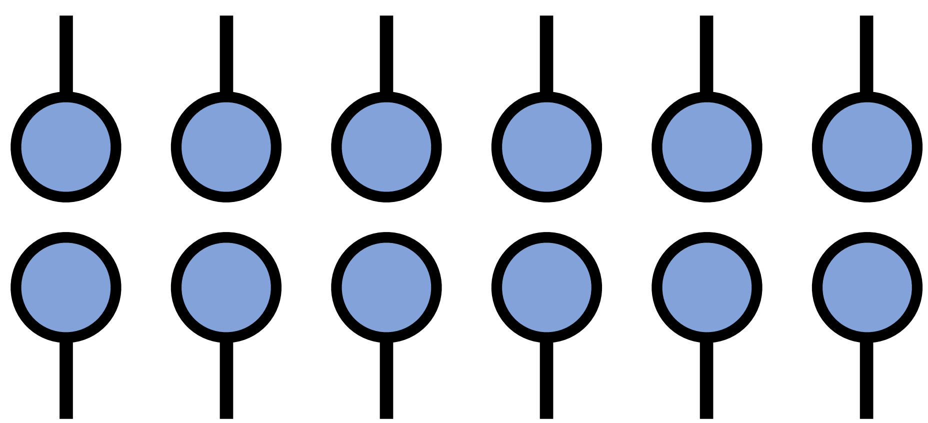
$\Phi(\mathbf{x}_j)$

$\Phi^\dagger(\mathbf{x}_j)$

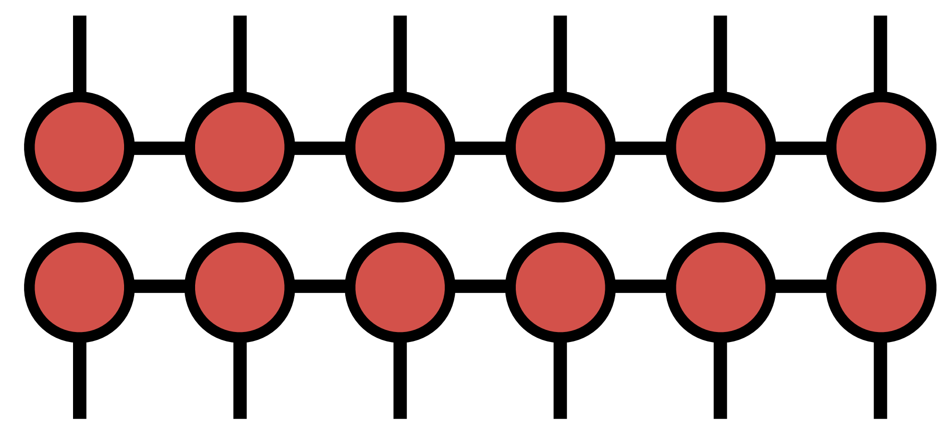
Another idea is to mix in a "lower level" model trained on a given task (e.g. supervised learning)

$$\rho^{\mu} =$$

$$(1-\mu)\sum_{j}$$



$$+\ \mu$$



If $\mu = 1$, tree provides basis for provided weights

If $0 < \mu < 1$, tree is "enriched" by data set

Experiment: mixed correlation matrix for MNIST

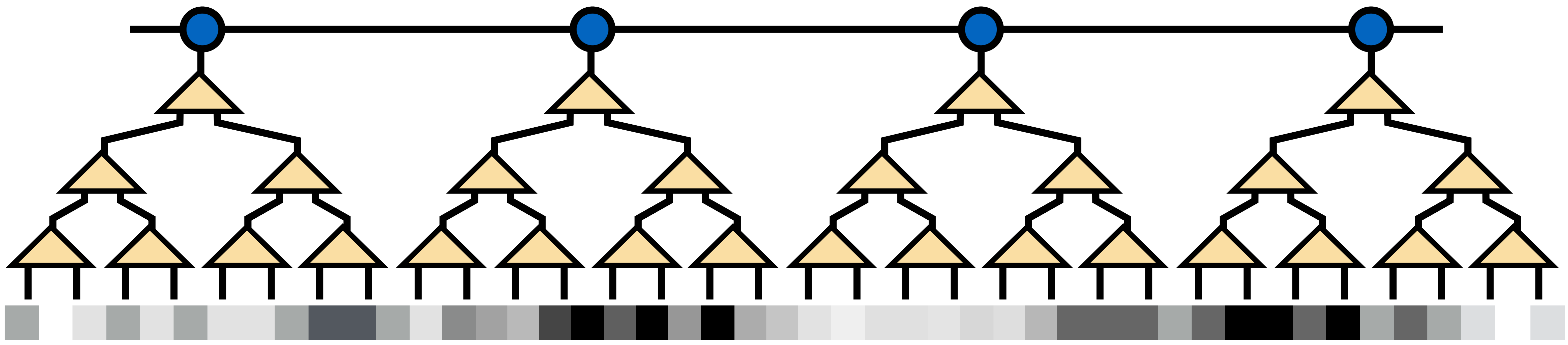
Using $$\rho^{\mu} = (1-\mu)\rho + \mu \sum_{\ell} |W^{\ell}\rangle\langle W^{\ell}|$$

with trial weights trained from a linear classifier and $\mu = 0.5$

Train acc: 99.798% **Test acc:** 98.110%
Top indices of size 279 and 393.

Comparable performance to unmixed case with top index sizes 328 and 444

Also no reason to build entire tree



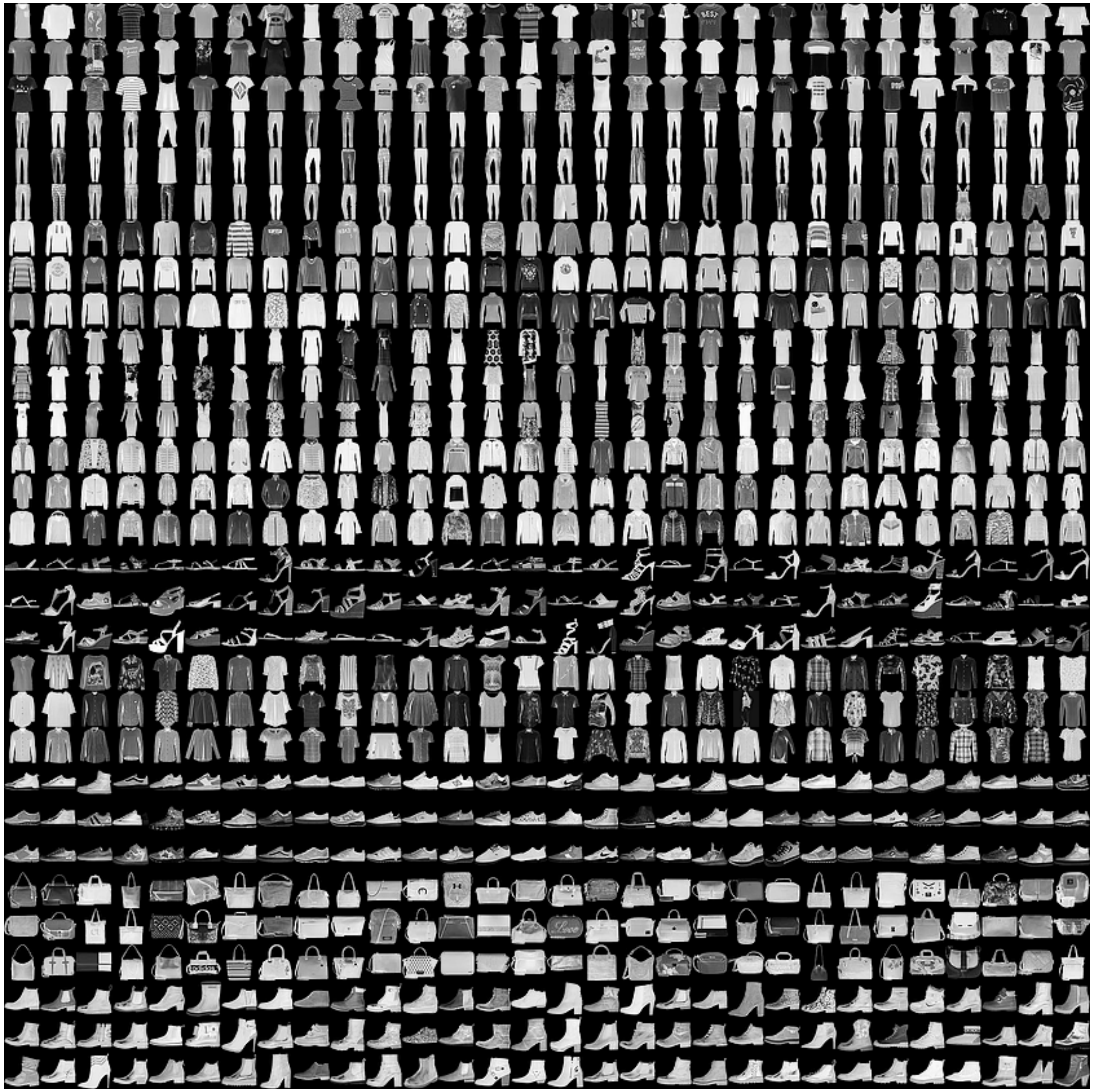
Approximate top tensor by MPS

Experiment: "fashion MNIST" dataset

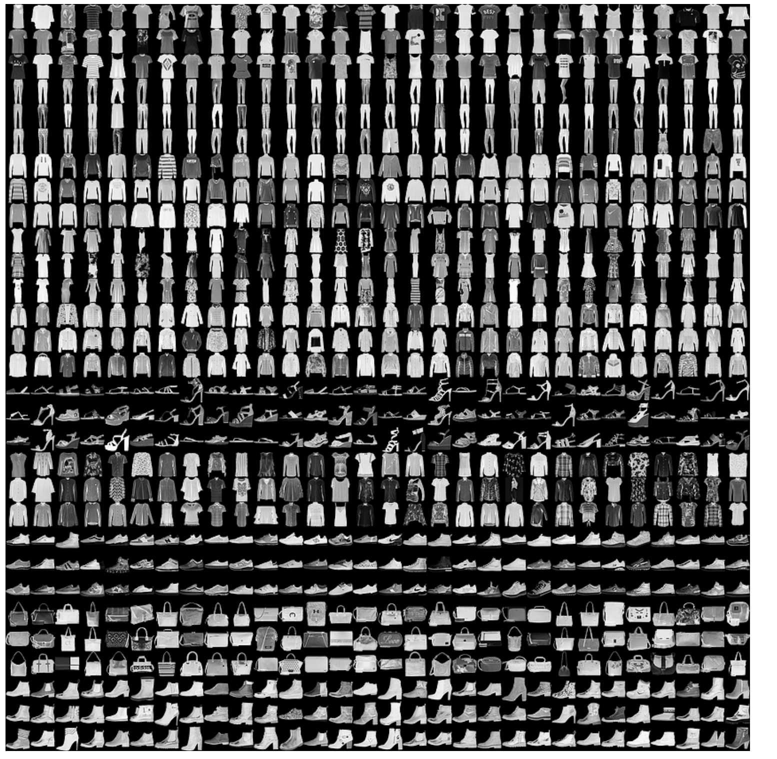
28x28 grayscale

60,000 training images

10,000 testing images



Experiment: "fashion MNIST" dataset



- Used 4 tree tensor layers
- Dimension of top "site" indices ranged from 11 to 30
- Top MPS bond dimension of 300 and 30 sweeps

Train acc: 95.38% **Test acc:** **88.97%**

Comparable to XGBoost (**89.8%**), AlexNet (**89.9%**), Keras Conv Net (**87.6%**)

Best (w/o preprocessing) is GoogLeNet at **93.7%**

Much Room for Improvement

- Use MERA instead of tree layers
- Optimize all layers, not just top, for specific task
- Iterate mixed approach: feed trained network into new covariance/density matrix
- Stochastic gradient based training

Implications for near-term quantum computing

- Tensor networks are equivalent to low-depth quantum circuits

- Kim & Swingle recently showed layered tensor network (MERA) inherently robust to noise*

- Prepare and optimize tensor networks on quantum computer for classical data?

Robust entanglement renormalization on a noisy quantum computer

Isaac H. Kim[1,2] and Brian Swingle[3,4]

*arxiv:1711.07500

arXiv.org > quant-ph > arXiv:1802.06002

Quantum Physics

Classification with Quantum Neural Networks on Near Term Processors

Edward Farhi, Hartmut Neven

(Submitted on 16 Feb 2018)

Recap & Future Directions

- Trained layered tensor network on real-world data in unsupervised fashion

- Specializing top layer gives very good results on challenging supervised image recognition tasks

- Linear tensor network approach gives enormous flexibility. Progress toward interpretability.