Title: Machine Learning with Quantum-Inspired Tensor Networks

Date: Jun 14, 2016 03:30 PM

URL: http://pirsa.org/16060005

Abstract: Tensor networks have been very successful for approximating quantum states that would otherwise require exponentially many parameters.

I will discuss how a similar compression can be achieved in models used to machine learn data, such as sets of images, by representing the fitting parameters as a tensor network. The resulting model achieves state-of-the-art performance on standard classification tasks. I will discuss implications for machine learning research, exploring which insights from physics could be imported into this field.

Physics Motivation

Machine learning using physics ideas

- Boltzmann machines (finite T Ising model) and quantum extensions [Amin, Kulchytskyy et al.]
- Spin glass interpretation of neural networks
- Lagrangians in support vector machines

Physics Motivation

Applications to physics

- Learning ab initio material properties
- Learning density functionals [Burke et al.] and DMFT solvers [Arsenault et al.]
- Quantum computing for machine learning [Wiebe & Granade; Kapoor et al.; ...]
- Learning properties of lattice models [Carrasquilla & Melko; Lei Wang; Carleo & Troyer; Torlai & Melko]

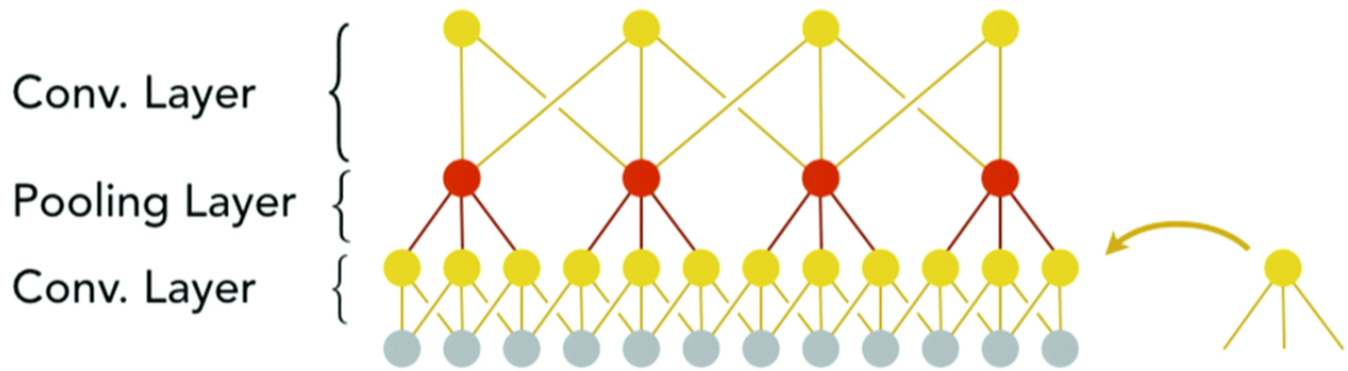
Physics Motivation

Parallels between physics / machine learning

- Hidden Markov models ⟷ MPS
- Deep neural networks ⟷ real-space RG*
- Convolutional networks ⟷ MERA

* P. Mehta, D.J. Schwab, arxiv:1410.38311

Convolutional Neural Network Layers



Similarities to MERA:

- Local mixing then rescaling
- Each layer defined by single map

Differences from MERA:

- Non-linear circuit elements
- No isometric conditions

ML Overview

Common task:
given labeled training data (supervised learning)
$\mathbf{x} \in A$ or $\mathbf{x} \in B$
find *decision function* $f(\mathbf{x})$

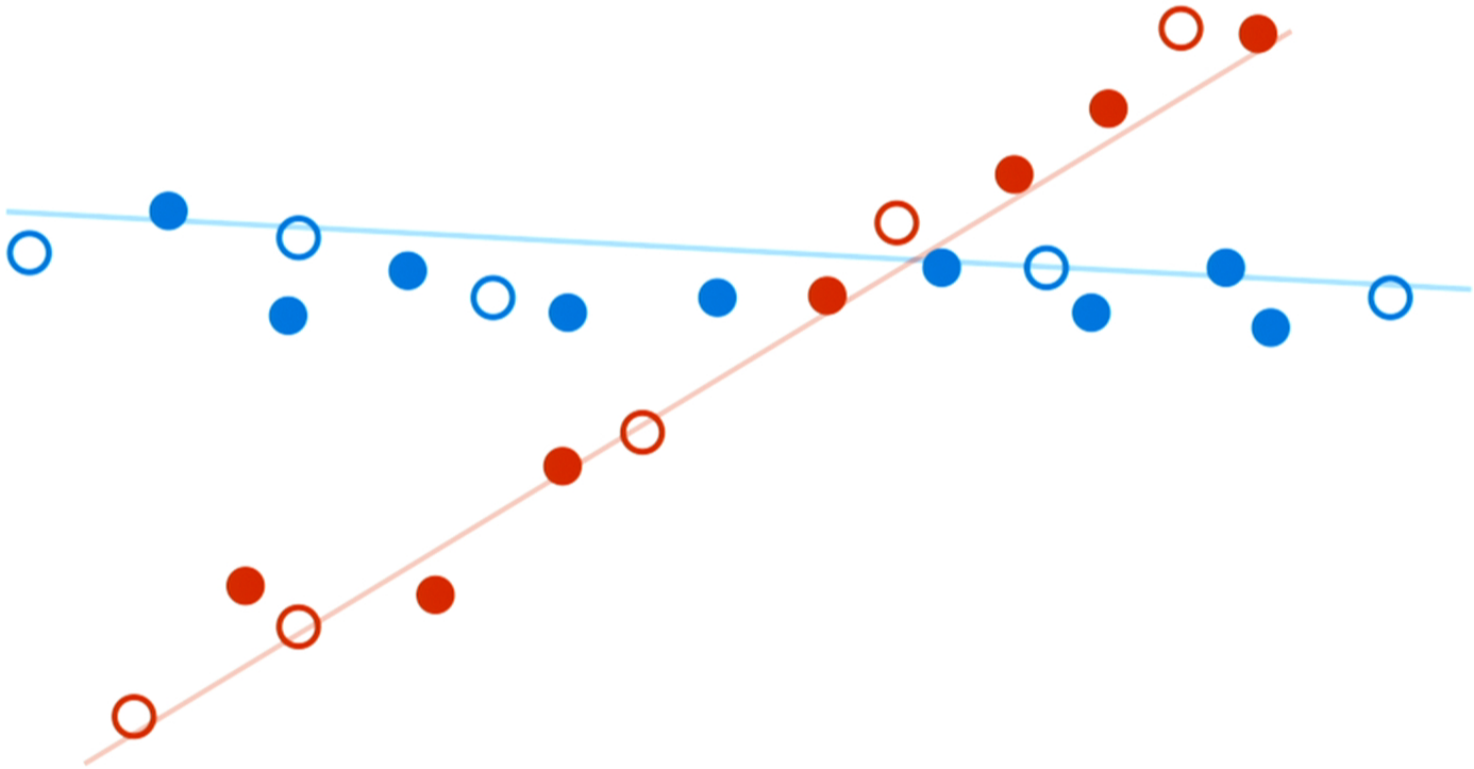
$$f(\mathbf{x}) > 0 \qquad \mathbf{x} \in A$$

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

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

ML Overview

Only get to use training data when building model



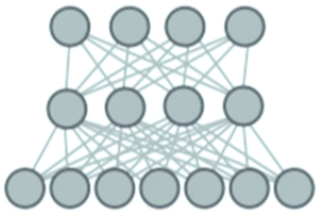
Hopefully generalize to unseen test data

ML Overview

Two popular approaches

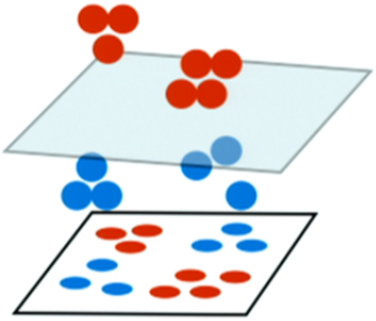
Neural Networks:

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



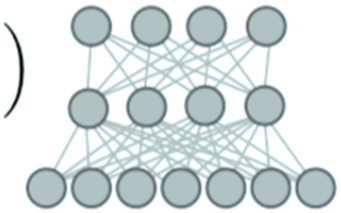
Non-Linear Kernel Learning:

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



ML Overview

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

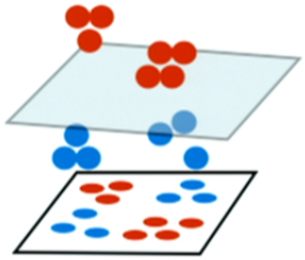


Neural Networks:

- Favored by industry, engineers; ***very successful!***
- Linear (or better) scaling with training data size
- *Limited theoretical understanding*
  - *Why are more layers helpful? How many?*
  - *How many neurons in each layer? Adaptive?*
  - *Structure/uniqueness of optimal parameters?*

ML Overview

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

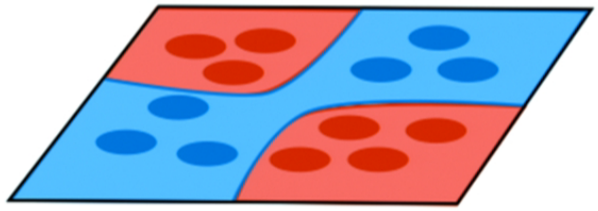


Non-linear kernel learning:

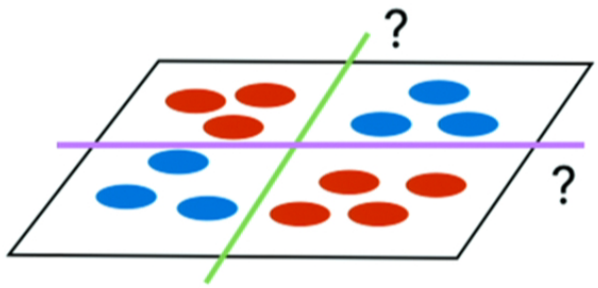
- Favored by academics
- Well understood theoretically
- *Quadratic scaling with training data set size*
- *How to choose feature map* $\Phi(\mathbf{x})$ *?*

Non-linear kernel learning

Want to fit a model that separates classes

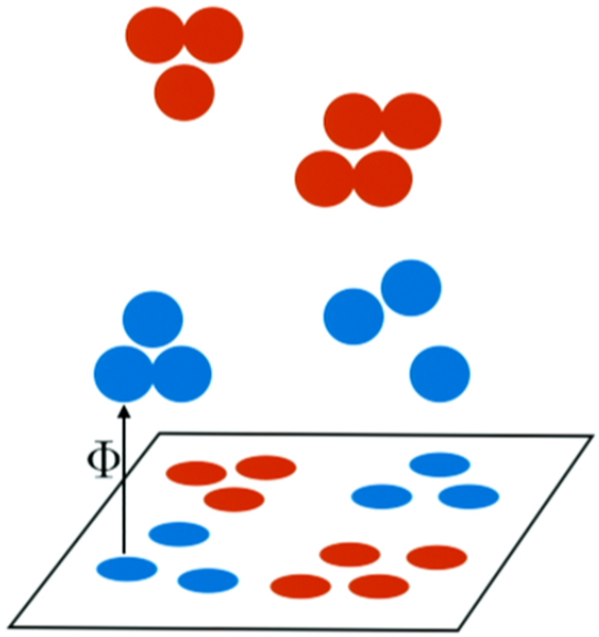


Simplest model is linear classifier $f(\mathbf{x}) = W \cdot \mathbf{x} + b$
but often data not linearly separable



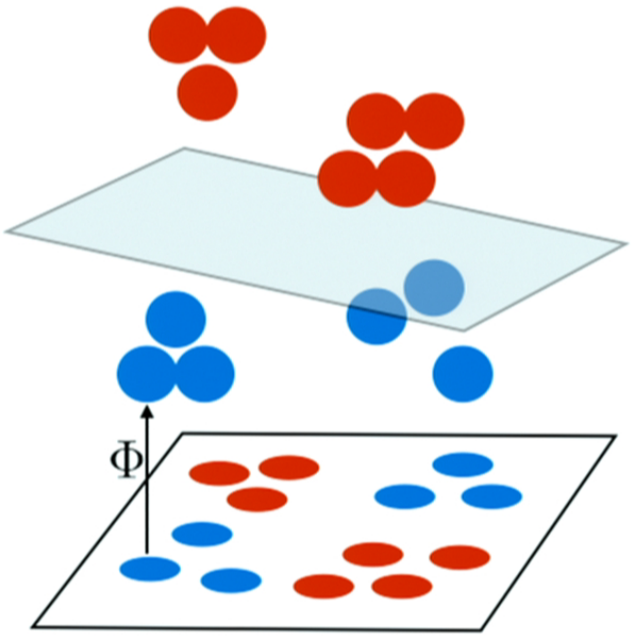
Non-linear kernel learning

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



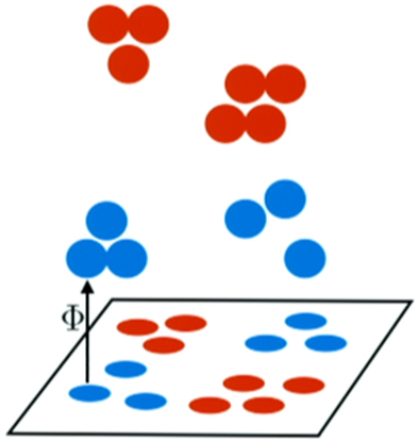
Non-linear kernel learning

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



Now linearly separable: decision function $f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})$

Non-linear kernel learning



Feature space (space of vectors $\Phi(\mathbf{x})$) can be exponentially large or infinite

For this reason, more common to work with scalar-valued "kernel"

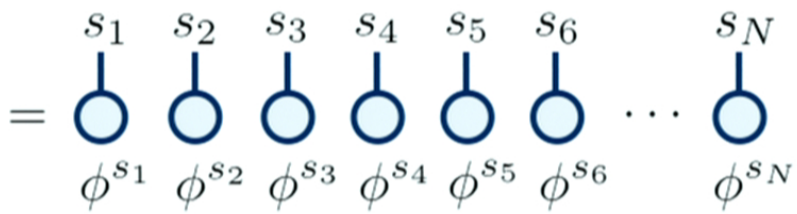
$$\kappa(\mathbf{x}', \mathbf{x}) = \Phi(\mathbf{x}')^\dagger \Phi(\mathbf{x})$$

Representer theorem states that $W = \sum_j c_j \Phi(\mathbf{x}_j)^\dagger$

So that $f(\mathbf{x}) = \left(\sum_j c_j \Phi(\mathbf{x}_j)^\dagger \right) \Phi(\mathbf{x}) = \sum_j c_j \kappa(\mathbf{x}_j, \mathbf{x})$

To make feature space resemble quantum Hilbert space, use *tensor-product feature map* *

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

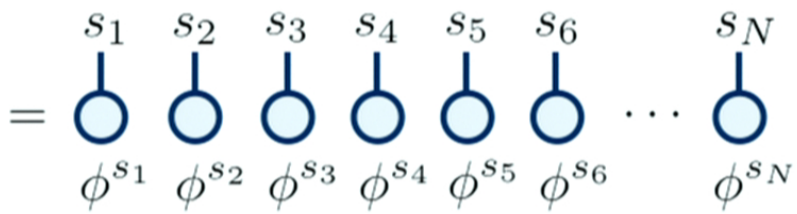


- Each local map $\phi^{s_j}(x_j)$ normalized, d-component vector
- $\Phi(\mathbf{x})$ maps N-dimensional to d^N-dimensional space
- Resembles product state wavefunction

* Vapnik, 2000, p. 193

To make feature space resemble quantum Hilbert space, use *tensor-product feature map* *

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

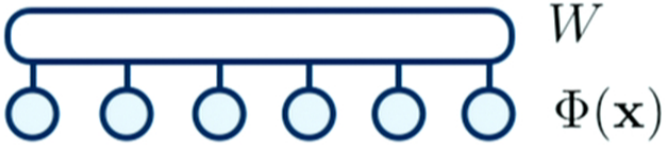


- Each local map $\phi^{s_j}(x_j)$ normalized, d-component vector
- $\Phi(\mathbf{x})$ maps N-dimensional to d^N-dimensional space
- Resembles product state wavefunction

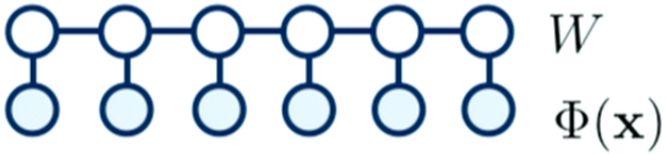
* Vapnik, 2000, p. 193

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

Weights W live in same space as $\Phi(\mathbf{x})$

$f(\mathbf{x}) =$ 

Proposal: decompose W as a tensor network (e.g. MPS)

$f(\mathbf{x}) =$ 

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

Approach:

(1) Shrink 28x28 images to 14x14, more manageable

(2) Order 2d pixels in a "zig-zag" pattern for MPS decomposition of W^{ℓ}

(3) Optimize with DMRG-like algorithm but with core step gradient descent

MNIST Experiment

Approach:

(3) Optimize with DMRG-like algorithm
but with core step gradient descent

Used quadratic cost function

$$C = \frac{1}{2} \sum_{n=1}^{N_T} \sum_{\ell} (f^{\ell}(\mathbf{x}_n) - \delta^{\ell}_{L_n})^2$$

Wants $f^{L_n}(x_n) = 1$, other $f^{\ell}(x_n) = 0$

MNIST Experiment

Approach:

(3) Optimize with DMRG-like algorithm
but with core step gradient descent

Training time scales like $m^3 N N_L N_T$

At most linear in number of training points

Could be improved with stochastic gradient

Key is direct access to weight tensor W

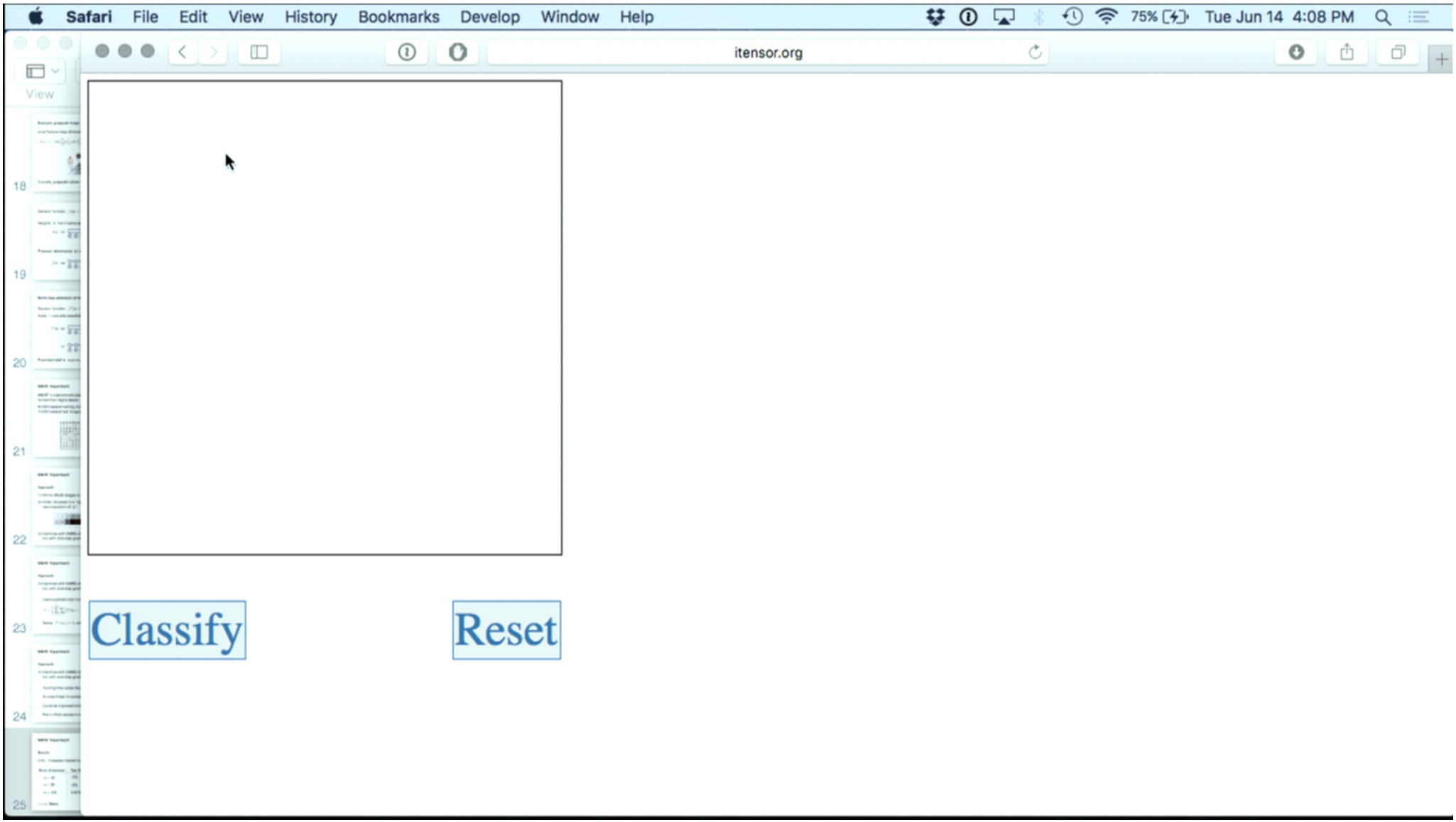
MNIST Experiment

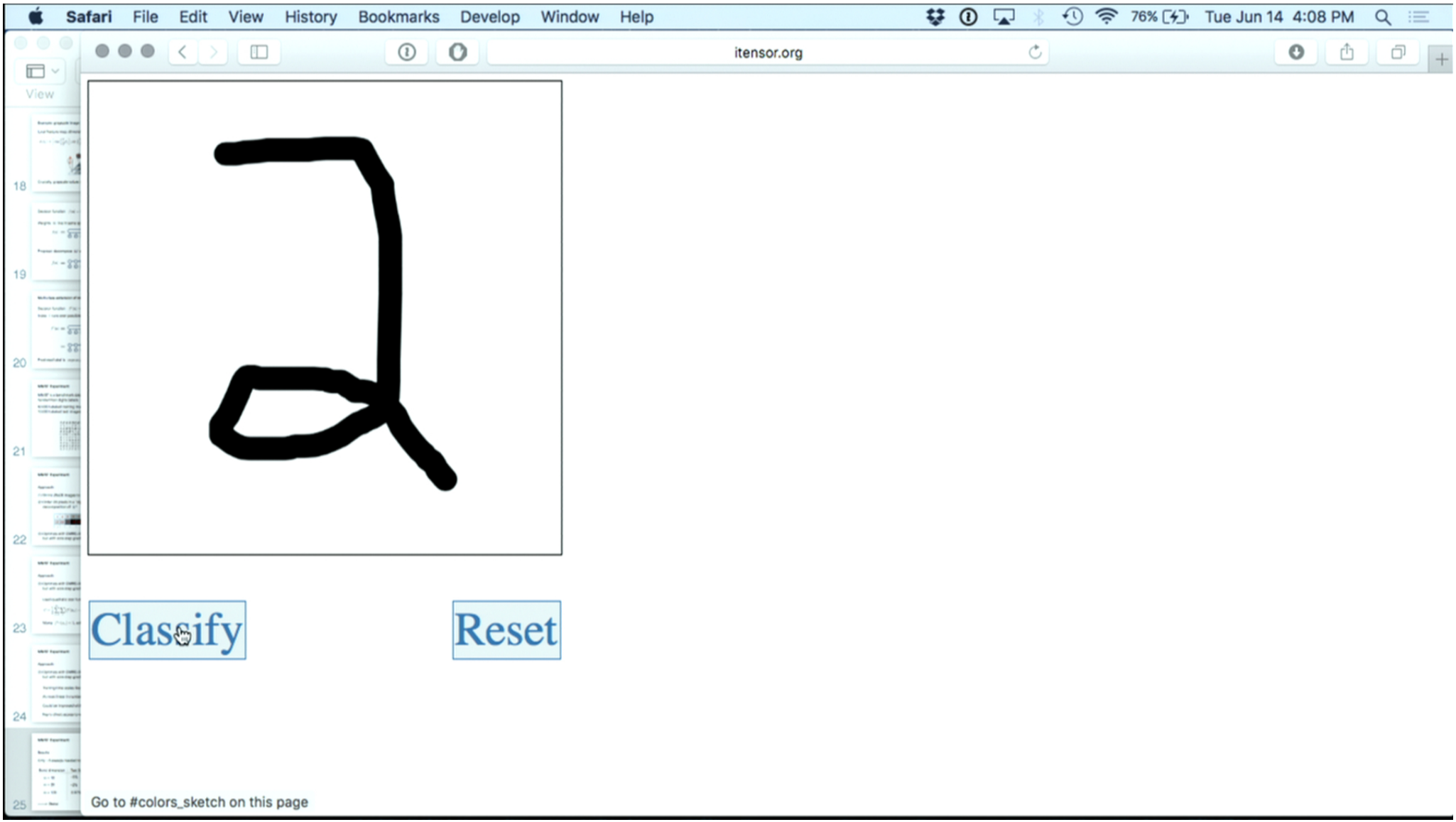
Results

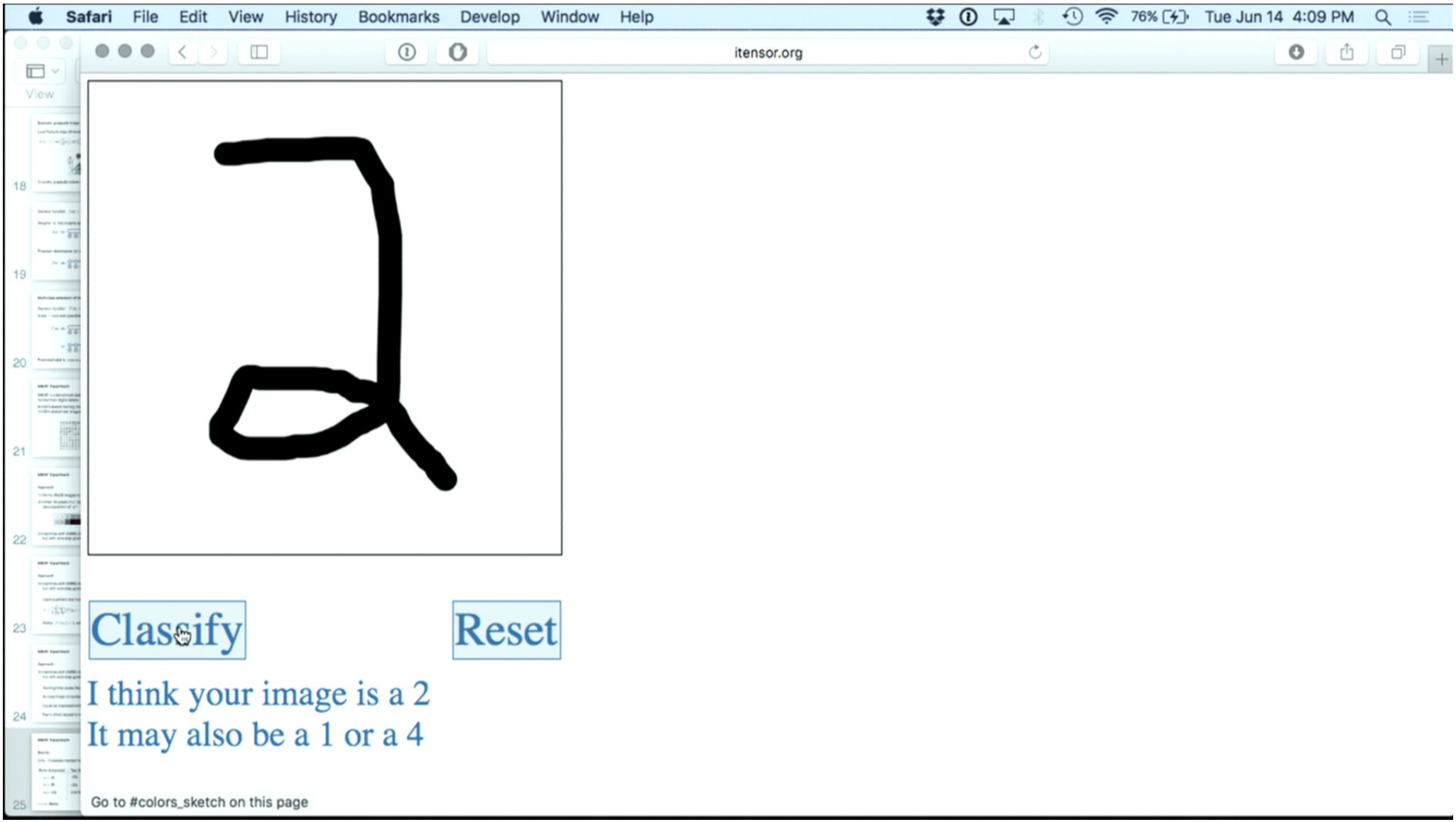
Only ~3 sweeps needed to converge

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)

→ ***Demo***


Safari File Edit View History Bookmarks Develop Window Help
75% Tue Jun 14 4:08 PM
itensor.org
View
Classify
Reset


Safari File Edit View History Bookmarks Develop Window Help
76% Tue Jun 14 4:08 PM
itensor.org
View
Classify
Reset
Go to #colors_sketch on this page


Safari File Edit View History Bookmarks Develop Window Help
76% Tue Jun 14 4:09 PM
itensor.org
View
Classify
Reset
I think your image is a 2
It may also be a 1 or a 4
Go to #colors_sketch on this page

Advantages of this model:

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

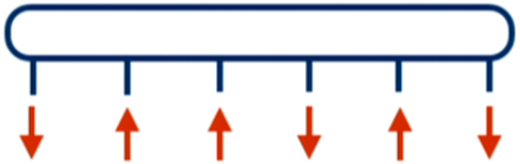
- Working directly with W allows optimization to be ***linear (or better)*** in number of training samples
- Adaptive: number of parameters ***discovered***, not pre-selected
- Tensor-product feature map sensitive to high-order ***correlations***
  [vs. order 2 polynomial $\Phi(\mathbf{x}) \sim (1, x_1^2, x_2^2, x_1 x_2)$]
- Import ***insights*** from physics (analysis of MPS; could use PEPS or MERA instead)

Tensor networks are linear.
Is resulting model linear?

Consider pure black & white image:

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

$$=$$



$$= W^{211212}$$

Computes arbitrary function on these images

For grayscale, smooth interpolation between b&w

Tensor network form also ***regularizes*** model

- MPS form reduces number of parameters, no explicit regularization was needed for MNIST
- Local dimension ***d*** also regularizes, consider N=2 toy model

$$f^{\ell}(x_1, x_2) = \sum_{s_1=1}^{d} \sum_{s_2=1}^{d} W^{\ell}_{s_1 s_2} \phi^{s_1}(x_1) \phi^{s_2}(x_2)$$

$=$

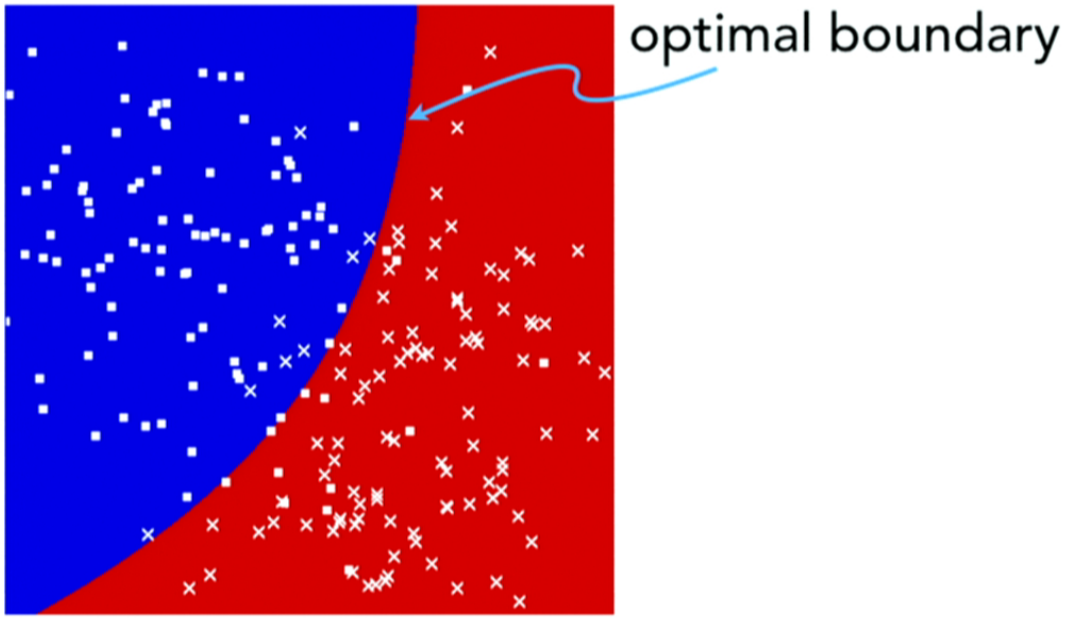
W^{ℓ}

$\Phi(\mathbf{x})$

Generate overlapping (non-separable) data sets

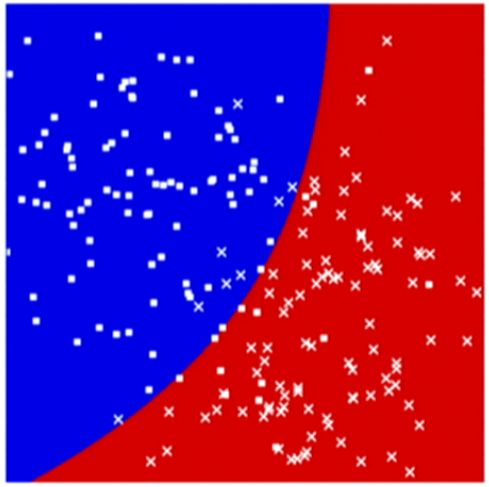
$$P_A(x_1, x_2) \propto \exp\left(ax_1^2 + bx_1x_2 + cx_2^2\right)$$

$$P_B(x_1, x_2) \propto \exp\left(wx_1^2 + tx_1x_2 + vx_2^2\right)$$

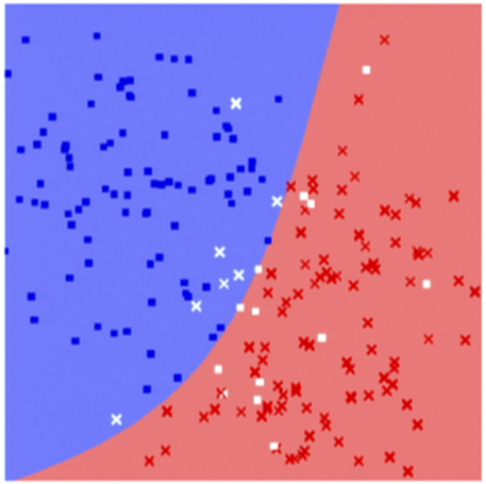


Learn back decision boundary from data

optimal boundary

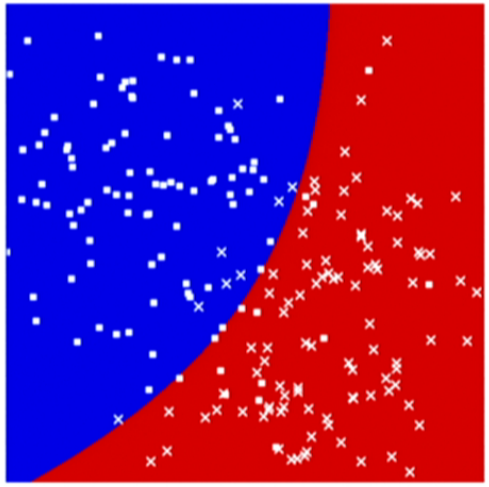


d=2 model

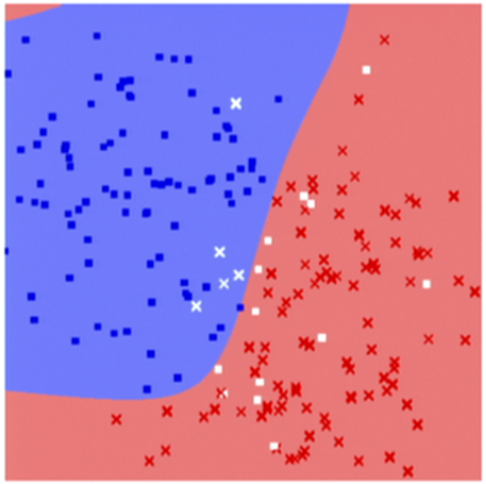


Learn back decision boundary from data

optimal boundary

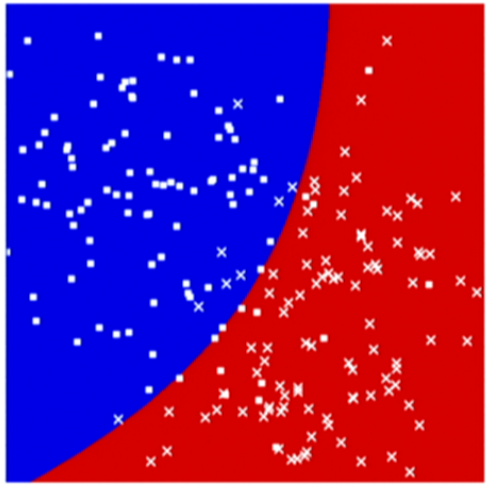


d=3 model

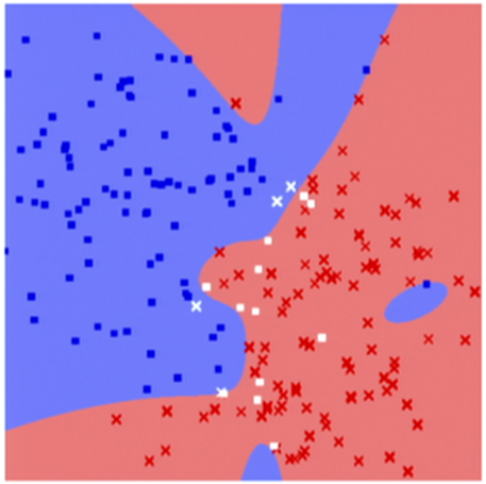


Learn back decision boundary from data

optimal boundary

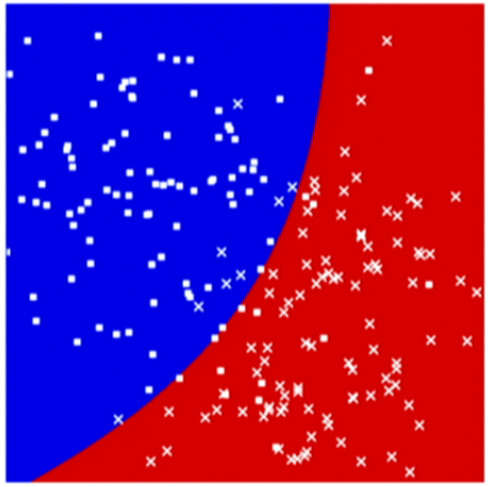


d=6 model

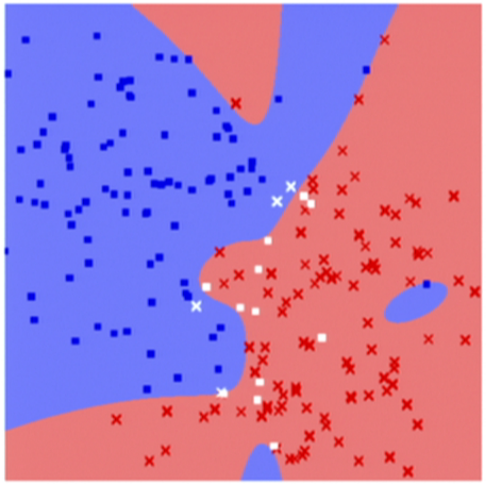


Learn back decision boundary from data

optimal boundary

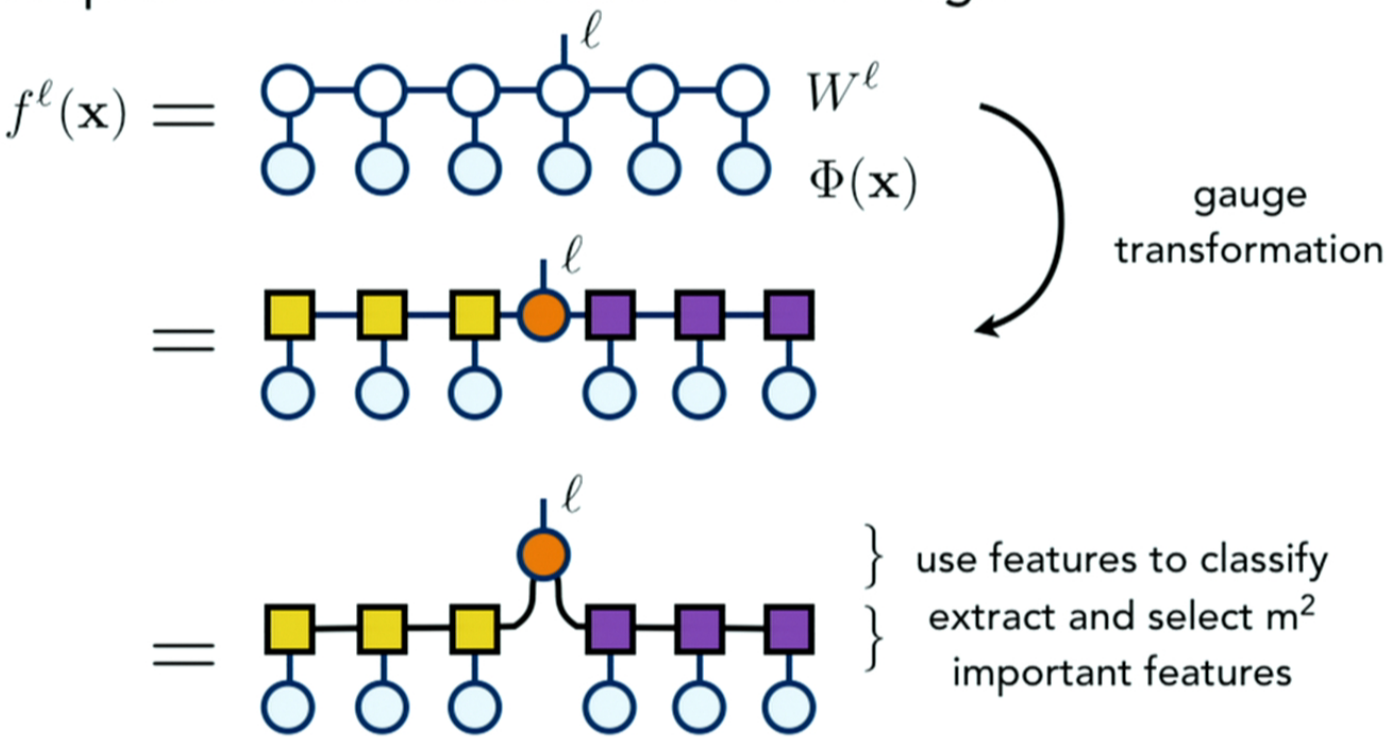


d=6 model



Multi-Stage Learning

Helpful to view classification in two stages



Why is weight tensor W "lightly" entangled?

MPS approximation not useful if optimal tensor W^ℓ highly entangled

Low-entanglement approx sufficient, why?

Possible explanations:

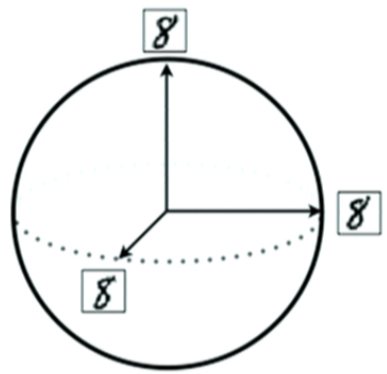
1. Structure of data + feature map
2. Supervised classification problem

Why is weight tensor W "lightly" entangled?

If every feature vector $\Phi(\mathbf{x}_j)$ was orthogonal to every other $\Phi(\mathbf{x}_i)$, then optimal weight vector would be

$$W^\ell = \sum_{j=1}^{N_T} \delta^\ell_{L_j} \Phi(\mathbf{x}_j)^\dagger \left(= \sum_j \langle L_j| \, \langle \Phi(\mathbf{x}_j)| \right)$$

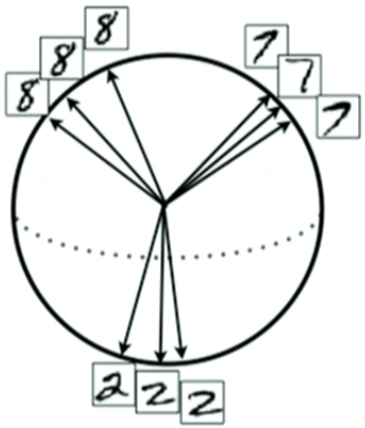
Equal-weight sum of N_T orthogonal vectors, could be highly entangled



Why is weight tensor W "lightly" entangled?

Instead, feature map $\Phi(\mathbf{x})$ maps similar inputs $\mathbf{x}$ to nearly parallel vectors

Weight vector needs much fewer components, one component for each cluster + components capturing small deviations



Why is weight tensor W "lightly" entangled?

Another explantion could be that very little information needed to assign correct labels (just a few choice features of data)

More work needed to understand!

Nature of Weight Tensor

Representer theorem says $W \sim \sum_j \alpha_j \Phi_j$

Physics intuition: W is a superposition of training data

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

Nature of Weight Tensor

Is W^ℓ a wavefunction? Tempting to interpret $|W^\ell_{s_1 s_2 \ldots s_N}|^2$ as a probability...

Correspondingly, view feature map as unitary change of basis $\Phi^{s_1 s_2 \cdots}(\mathbf{x}) \sim U^{s_1 s_2 \cdots}_{\mathbf{x}}$

Then $|f^\ell(\mathbf{x})|^2 = |W^\ell \cdot \Phi(\mathbf{x})|^2$

would be probability distribution over ℓ, $\mathbf{x}$

Such probabilistic or "generative" models very useful e.g. unsupervised learning

Generative Interpretation

Bare minimum requirement for prob. interpretation

$$\sum_{\ell} \int_{\mathbf{x}} |f^{\ell}(\mathbf{x})|^2 d\mu(x) = 1$$

for some integration measure $d\mu(x)$

Should hold for arbitrary W^{ℓ} as long as

$$\sum_{\ell} \sum_{s_1, s_2, \ldots, s_N} \bar{W}^{\ell}_{s_1 s_2 \cdots s_N} W^{\ell}_{s_1 s_2 \cdots s_N} = 1$$

Generative Interpretation

Normalization of $|f^{\ell}(\mathbf{x})|^2$ follows automatically if

$$\int_x \bar{\phi}^s(x)\phi^{s'}(x)\, d\mu(x) = \delta_{ss'}$$

Continue to require

$$\sum_s |\phi^s(x)|^2 = 1$$

Generative Interpretation

Unfortunately, simple feature map

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

does not fulfill condition

$$\int_x \bar{\phi}^s(x)\phi^{s'}(x)\, d\mu(x) = \delta_{ss'}$$

Generative Interpretation

Here is a d=2 feature map which does work

$$\phi(x) = \left[e^{i(3\pi/2)x} \cos\left(\frac{\pi}{2}x\right),\ e^{-i(3\pi/2)x} \sin\left(\frac{\pi}{2}x\right) \right] \qquad x \in [0,1]$$

Not totally trivial to generalize to d > 2 however!

Difficulty is normalization condition

$$\sum_s |\phi^s(x)|^2 = 1$$

Generative Interpretation

Test of generative interpretation

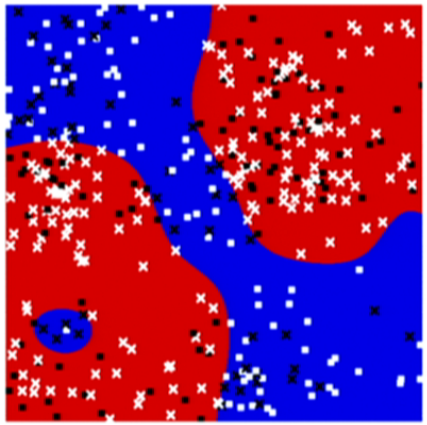
Start with random, but fixed N=2 model

$$f^{\ell}(x_1, x_2) = W^{\ell}_{s_1 s_2} \phi^{s_1}(x_1) \phi^{s_2}(x_2) \qquad \ell = 1, 2$$

Sample N_s points from distribution

$$p(\ell, x_1, x_2) = |f^{\ell}(x_1, x_2)|^2$$

N_s = 400 example:



Generative Interpretation

Test of generative interpretation

From sampled points, train a new model by optimizing "log likelihood" cost function

$$C = -\sum_{n=1}^{N_s} \log |f^{L_n}(\mathbf{x}_n)|^2$$

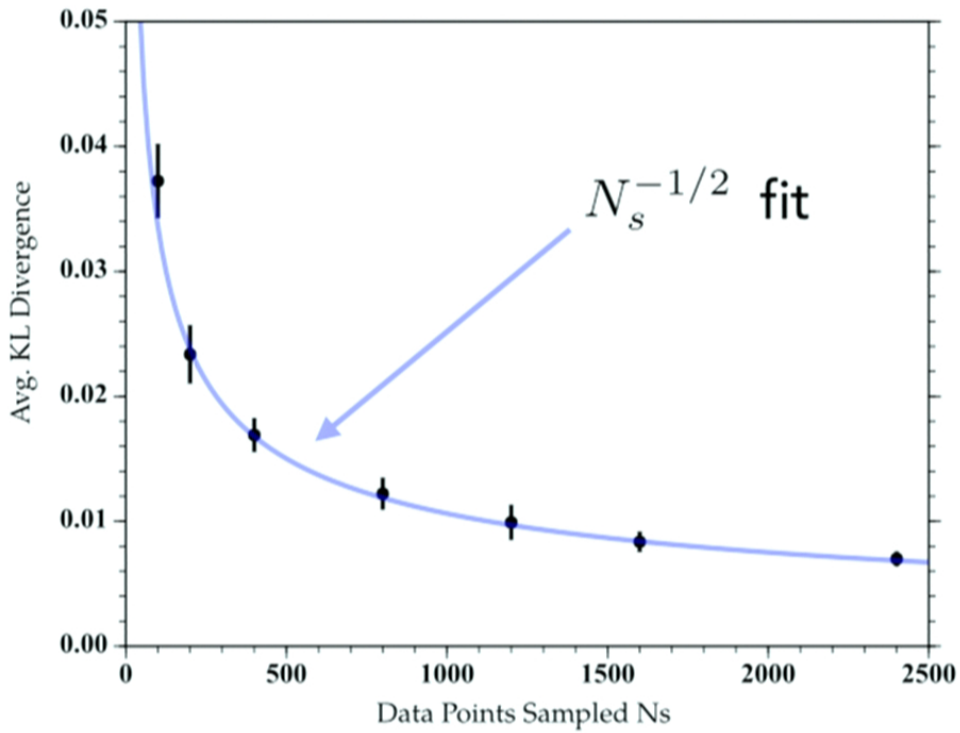
Measure typical distance (KL divergence) of learned model from original model

$$D_{\mathrm{KL}} = \sum_{\ell} \int_{\mathbf{x}} p(\ell, \mathbf{x}) \log\left(\frac{p(\ell, \mathbf{x})}{\tilde{p}(\ell, \mathbf{x})}\right)$$

Repeat many times for each N_s and average

Generative Interpretation

Test of generative interpretation



Wrap Up

- Tensor network model successful at learning challenging data set (MNIST handwritten digits)
- Type of kernel-learning model but with ***linear*** worst-case dependence on # training points
- Other networks (PEPS, MERA) + better training techniques (stoc. gradient) could greatly improve performance
- Tensor network structure + possible generative interpretation means many physics insights could be imported to field of machine learning

$\phi_j = \left[\cos\left(\frac{\pi}{2}x_j\right), \sin\left(\frac{\pi}{2}x_j\right)\right]$
• Tensor network structure + pos
interpretation means many phys
be imported to field of machine