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

<table>
<thead>
<tr>
<th>Property</th>
<th>Variational RG</th>
<th>Variational RG</th>
</tr>
</thead>
<tbody>
<tr>
<td>How input distribution is defined</td>
<td>Hamiltonian defining $P(v)$</td>
<td>Deep Belief Networks</td>
</tr>
<tr>
<td>How interactions are defined</td>
<td>$T(v,h)$</td>
<td>$E(v,h)$</td>
</tr>
<tr>
<td>Exact transformation</td>
<td>$Tr e^{i(h)}$</td>
<td>$KL$ divergence</td>
</tr>
<tr>
<td>Approximations</td>
<td>Minimize or bound free energy differences</td>
<td>Minimize the $KL$ divergence</td>
</tr>
<tr>
<td>Method</td>
<td>Analytic (mostly)</td>
<td>Numerical</td>
</tr>
<tr>
<td>What happens under coarse-graining</td>
<td>Relevant operators grow/irrelevant shrink</td>
<td>New features emerge</td>
</tr>
</tbody>
</table>

*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

- Neural Nets
- Boltzmann Machines
- Kernel Learning
- Supervised Learning
- Unsupervised Learning

Physics

- Phase Transitions
- Topological Phases
- Quantum Monte Carlo
- Sign Problem
- Materials Science & Chemistry

(this talk)

Tensor Networks
What are Tensor Networks?
How do tensor networks arise in physics?

Quantum systems governed by Schrödinger equation:

\[ \hat{H} \Psi = E \Psi \]

It is just an eigenvalue problem.
The problem is that $\hat{H}$ is a $2^N \times 2^N$ matrix

$\mapsto$ wavefunction $\tilde{\Psi}$ has $2^N$ components

$\hat{H} \Psi = E \Psi$
Natural to view wavefunction as order-N tensor

\[ |\Psi\rangle = \sum_{\{s\}} \Psi^{s_1s_2s_3\cdots s_N} |s_1s_2s_3\cdots s_N\rangle \]
Natural to view wavefunction as order-N tensor

$$\Psi^{s_1 s_2 s_3 \cdots s_N} = s_1 \ s_2 \ s_3 \ s_4 \ \cdots \ \cdots \ \cdots \ s_N$$
Tensor components related to probabilities of
e.g. Ising model spin configurations

\[ \Psi = \begin{array}{c}
\uparrow \\
\uparrow \\
\downarrow \\
\uparrow \\
\uparrow \\
\uparrow \\
\end{array} \]
Tensor components related to probabilities of e.g. Ising model spin configurations
Must find an approximation to this exponential problem

\[ \Psi s_1 s_2 s_3 \cdots s_N = s_1 s_2 s_3 s_4 \cdots s_N \]
Simplest approximation (mean field / rank-1)
Let spins "do their own thing"

\[ \Psi^{s_1 s_2 s_3 s_4 s_5 s_6} \sim \psi^{s_1} \psi^{s_2} \psi^{s_3} \psi^{s_4} \psi^{s_5} \psi^{s_6} \]

Expected values of individual spins ok
No correlations
Restore correlations locally

\[ \Psi^{s_1 s_2 s_3 s_4 s_5 s_6} \sim \psi^{s_1} \psi^{s_2} \psi^{s_3} \psi^{s_4} \psi^{s_5} \psi^{s_6} \]
Restore correlations locally

\[ \Psi^{s_1s_2s_3s_4s_5s_6} \sim \psi^{s_1}_{i_1} \psi^{s_2}_{i_1} \psi^{s_3} \psi^{s_4} \psi^{s_5} \psi^{s_6} \]
Restore correlations locally

\[ \Psi^{s_1 s_2 s_3 s_4 s_5 s_6} \sim \psi^{s_1}_{i_1} \psi^{s_2}_{i_1 i_2} \psi^{s_3}_{i_2 i_3} \psi^{s_4}_{i_3 i_4} \psi^{s_5}_{i_4 i_5} \psi^{s_6}_{i_5} \]

matrix product state (MPS)

✓ Local expected values accurate
✓ Correlations decay with spatial distance
"Matrix product state" because

retrieving an element $=\text{product of matrices}$
"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

\[ \sum_{j} M_{ij} v_{j} \]

\[ A_{ij} B_{jk} = AB \]

\[ A_{ij} B_{ji} = \text{Tr}[AB] \]
MPS approximation controlled by bond dimension "m" (like SVD rank)

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

$m \sim 2^{N/2}$ can represent any tensor
Friendly neighborhood of "quantum state space"
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)
Supervised Kernel Learning
Supervised Learning

Very common task:

Labeled training data (= supervised)

Find decision function \( f(x) \)

\[
\begin{align*}
    f(x) &> 0 & x &\in A \\
    f(x) &< 0 & x &\in B
\end{align*}
\]

Input vector \( x \) 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

\[ f(x) = \Phi_2 \left( M_2 \Phi_1 (M_1 x) \right) \]

Non-Linear Kernel Learning

\[ f(x) = W \cdot \Phi(x) \]
Non-linear kernel learning

Want $f(x)$ to separate classes

Linear classifier often insufficient

$$f(x) = W \cdot x$$
Non-linear kernel learning

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

Apply non-linear "feature map" \( x \rightarrow \Phi(x) \)

Decision function \( f(x) = W \cdot \Phi(x) \)
Non-linear kernel learning

Decision function

$$ f(x) = W \cdot \Phi(x) $$

Linear classifier in feature space
Non-linear kernel learning

Example of feature map

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

\[ \Phi(\mathbf{x}) = (1, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3) \]

\( \mathbf{x} \) is "lifted" to feature space
Proposal for Learning
Grayscale image data
Map pixels to "spins"
Map pixels to "spins"
Map pixels to "spins"
Local feature map, dimension d=2

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

Crucially, grayscale values not orthogonal
Total feature map $\Phi(x)$

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

- Tensor product of local feature maps / vectors
- Just like product state wavefunction of spins
- Vector in $2^N$ dimensional space
Total feature map \( \Phi(x) \)

More detailed notation

\[
x = [x_1, x_2, x_3, \ldots, x_N]
\]

\[
\Phi(x) = \left[ \begin{array}{c} \phi_1(x_1) \\ \phi_2(x_1) \\ \phi_1(x_2) \\ \phi_2(x_2) \\ \phi_1(x_3) \\ \phi_2(x_3) \\ \vdots \\ \phi_1(x_N) \\ \phi_2(x_N) \end{array} \right]
\]

\( x \) = input
\( \phi \) = local feature map

raw inputs

feature vector
Total feature map $\Phi(x)$

Tensor diagram notation

\[ x = [x_1, x_2, x_3, \ldots, x_N] \]

\[ \Phi(x) = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & \cdots & s_N \end{bmatrix} \]

\[
\begin{array}{cccccccc}
\phi^{s_1} & \phi^{s_2} & \phi^{s_3} & \phi^{s_4} & \phi^{s_5} & \phi^{s_6} & \cdots & \phi^{s_N}
\end{array}
\]

$x = \text{input}$

$\phi = \text{local feature map}$
Construct decision function

\[ f(x) = W \cdot \Phi(x) \]
Construct decision function

\[ f(x) = W \cdot \Phi(x) \]
Construct decision function

\[ f(x) = W \cdot \Phi(x) \]
Construct decision function

\[ f(x) = W \cdot \Phi(x) \]

\[
\begin{align*}
  f(x) &= W \\
  \Phi(x) &= W
\end{align*}
\]
Main approximation

\[ W = \] order-N tensor

\[ \approx \] matrix

product

state (MPS)
MPS form of decision function

\[ f(x) = W \Phi(x) \]
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

\( f(x) = W \Phi(x) \)
Linear scaling

Can use algorithm similar to DMRG to optimize

Scaling is \( N \cdot N_T \cdot m^3 \)

\( N = \text{size of input} \)
\( N_T = \text{size of training set} \)
\( m = \text{MPS bond dimension} \)

\[ f(x) = W \Phi(x) \]
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

\[ f(x) = \Phi(x) \cdot W \]
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

\[ f(x) = W \Phi(x) \]
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

$f(x) = W \Phi(x)$

Could improve with stochastic gradient
Multi-class extension of model

Decision function \( f^\ell(x) = W^\ell \cdot \Phi(x) \)

Index \( \ell \) runs over possible labels

Predicted label is \( \text{argmax}_\ell |f^\ell(x)| \)
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

<table>
<thead>
<tr>
<th>Bond dimension</th>
<th>Test Set Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m = 10$</td>
<td>~5% (500/10,000 incorrect)</td>
</tr>
<tr>
<td>$m = 20$</td>
<td>~2% (200/10,000 incorrect)</td>
</tr>
<tr>
<td>$m = 120$</td>
<td>0.97% (97/10,000 incorrect)</td>
</tr>
</tbody>
</table>

State of the art is < 1% test set error
MNIST Experiment

Demo

Link: http://itensor.org/miles/digit/index.html
Understanding Tensor Network Models

\[ f(x) = W \Phi(x) \]
Again assume $W$ is an MPS

$$f(x) = \Phi(x)$$

Many interesting benefits

Two are:

1. Adaptive
2. Feature sharing
1. Tensor networks are **adaptive**

boundary pixels not useful for learning

grayscale training data
2. Feature sharing

\[ f^\ell(x) = \Phi(x) \]

- Different central tensors
- "Wings" shared between models
- Regularizes models
2. Feature sharing

\[ f^\ell(x) \]
2. Feature sharing

\[ f^\ell(x) = \]

Progressively learn shared features
2. Feature sharing

\[ f^\ell(x) = \]

Progressively learn shared features
2. Feature sharing

\[ f^\ell(x) = \]

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 \( W^\ell \) 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

\[ W_{\text{MPS}} \neq \sum_j \alpha_j \Phi(x_j) \quad \text{for any} \quad \{\alpha_j\} \]

• 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 $|W \cdot \Phi(x)|^2$ 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

$$\Phi(x) = \left[ \begin{array}{c} \phi_1(x_1) \\ \phi_2(x_1) \end{array} \right] \otimes \left[ \begin{array}{c} \phi_1(x_2) \\ \phi_2(x_2) \end{array} \right] \otimes \left[ \begin{array}{c} \phi_1(x_3) \\ \phi_2(x_3) \end{array} \right] \otimes \cdots \otimes \left[ \begin{array}{c} \phi_1(x_N) \\ \phi_2(x_N) \end{array} \right]$$

*Novikov, et al., arxiv:1605.03795
N=2 case

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

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

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

$$(W_{11}, W_{21}, W_{12}, W_{22})$$

$$f(x) = W \cdot \Phi(x) = \cdot (1, x_1, x_2, x_1 x_2)$$

$$= W_{11} + W_{21} x_1 + W_{12} x_2 + W_{22} x_1 x_2$$
N=3 case

\[ \phi(x) = (1, x) \]

\[ \Phi(x) = \left[ \begin{array}{c} 1 \\ x_1 \end{array} \right] \otimes \left[ \begin{array}{c} 1 \\ x_2 \end{array} \right] \otimes \left[ \begin{array}{c} 1 \\ x_3 \end{array} \right] \]

\[ = (1, x_1, x_2, x_3, x_1 x_2, x_1 x_3, x_2 x_3, x_1 x_2 x_3) \]

\[ f(x) = W \cdot \Phi(x) \]

\[ = W_{111} + W_{211} x_1 + W_{121} x_2 + W_{112} x_3 \]
\[ + W_{221} x_1 x_2 + W_{212} x_1 x_3 + W_{122} x_1 x_3 \]
\[ + W_{222} x_1 x_2 x_3 \]
General N case

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

\[
= W_{111\ldots1} + W_{211\ldots1} x_1 + W_{121\ldots1} x_2 + W_{112\ldots1} x_3 + \ldots \\
+ W_{221\ldots1} x_1 x_2 + W_{212\ldots1} x_1 x_3 + \ldots \\
+ W_{222\ldots1} x_1 x_2 x_3 + \ldots \\
+ \ldots \\
+ W_{222\ldots2} x_1 x_2 x_3 \cdots x_N
\]

Model has exponentially many formal parameters

\[
\mathbf{x} \in \mathbb{R}^N
\]
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

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)