Towards Quantum Machine Learning with Tensor Networks

William Huggins, Piyush Patil , K. Birgitta Whalley, M. Stoudenmire

Novosibirsk State University

Presented by: **Raphael Blankson**

April 30, 2020

Overview

Introduction

Tensor networks have been a useful tool for physicists for many years, but more recently they have been applied to a wide spread of problems in machine learning:

- Classification
- Analyzing Representation power of Neural networks
- Model Compression etc

Proposed Method

The authors:

- Propose quantum algorithm that implement both discriminative and generative machine learning tasks
- Used circuits equivalent to tensor networks specifically
  - Tree tensor Networks
  - Matrix Product States (MPS)
- Approach is conceptually related to **quantum variational eigensolver**

Motivation

Is there a subset of quantum circuits which are especially natural or advantageous for machine learning tasks?

Motivation

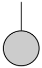
Tensor network circuits might provide a compelling answer for three main reasons:

- Implemented on **small near-term quantum devices** for input and output dimensions greater than the number of physical qubits.
- **Gradual crossover** from classical tensor network circuits to circuits that require quantum computers.
- Rich **theoretical understanding** of the properties of tensor networks.

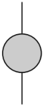
Tensor Networks allow us to approximate a high order tensor using a collection of lower order tensors and a prescription for contracting them. Rather than writing out a summation over a collection of different

variables and indices, we use graphical notation.

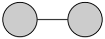
Graphical Notation of Tensor Networks



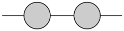
(a) A vector



(b) A matrix



(c) Dot product



(d) Matrix product

Figure: Notation for tensor networks

Learning with Tensor Networks Quantum Circuits

- Tree and MPS tensors can always be realized by a quantum circuit.

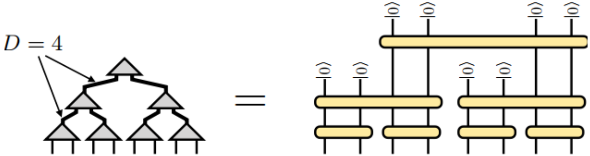


Fig 2: quantum state of **N** qubits of tree tensor network(left) = quantum circuit acting on N qubits(right) $\equiv D =$ bond dimension, $V =$ no. of qubits $|D = 2^V$

- Quantum tensor networks are carefully designed with classical computers in mind.
- Tree and MPS tensor can capture wide range of states to produce powerful machine learning models.

A. Discriminative Algorithm

Goal: Given some pieces of data $\vec{x} \in \mathbb{R}^n$ and their associated labels $l \in \{1 \ldots\ k\}$, learn a function that maps from the data to the labels:

$$f : \mathbb{R}^n \to \{1 \ldots\ k\}$$

We could use a linear classifier for this task but they are not flexible enough.

A. Discriminative Algorithm

The input vector $\vec{x} = (x1, x2, \ldots\ x_N)$, be normalized s.t. $x_i \in [0, 1]$. Map vector $x \in \mathbb{R}^N$ to a product state by the feature map:

$$g(\vec{x}) := \begin{bmatrix} sin(\frac{\pi}{2}x_0) \\ cos(\frac{\pi}{2}x_0) \\ 1 \end{bmatrix} \otimes \begin{bmatrix} sin(\frac{\pi}{2}x_1) \\ cos(\frac{\pi}{2}x_1) \\ 1 \end{bmatrix} \otimes \ldots \otimes \begin{bmatrix} sin(\frac{\pi}{2}x_{n-1}) \\ cos(\frac{\pi}{2}x_{n-1}) \\ 1 \end{bmatrix}$$

The state can be prepared by starting from computational basis $|0\rangle^{\otimes N}$, then apply a single qubit unitary to each qubit $n = 1, 2, \ldots\ N$.

A. Discriminative Algorithm

The model proposed:

- Is an iterative procedure that parameterizes a **CPTP** - completely positive trace preserving map from N-qubit input space to a small no of output qubits that encodes different possible class labels.

A. Discriminative Algorithm

- The circuit takes the form of a tree, with V qubit lines connecting each subtree to the rest of the circuit.
- A larger V can capture a larger set of functions, just as a tensor network with large bond dimension can parametrize any N-index tensor.

A. Discriminative Algorithm

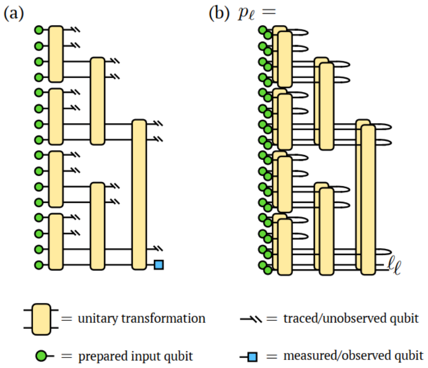


Fig 3: Discriminative tree tensor network model architecture for V = 2 qubits connect different subtrees (a) quantum circuit (b) tensor network diagram

A. Disciminative Algorithm

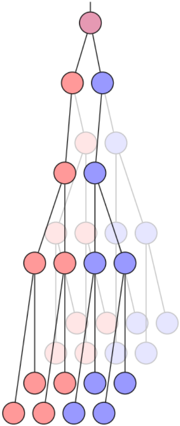


Fig 4: The connectivity of nodes of the tree network model if it was applied to 4x4 image

A. Discriminative Algorithm

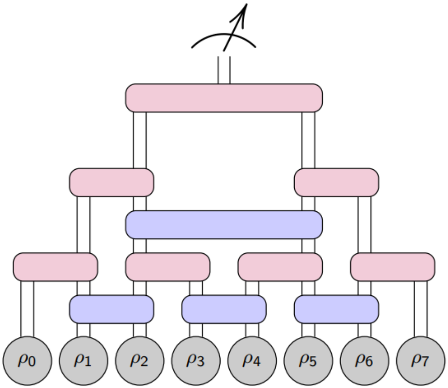


Fig 5: Using quantum we could add additional unitary element to the circuit to address the shortcomings of correlation

A. Discriminative Algorithm

- An MPS model can be viewed as maximally unbalanced tree.
- The difference is for each unitary operation on 2V inputs, only one set of V qubits are passed to the next node

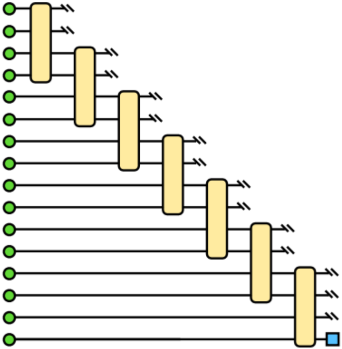


Fig 6: Discriminative tensor network model for MPS architecture with V = 2

B. Generative Algorithm

The generative algorithm proposed:

- Is nearly the reverse of the discriminative algorithm.
- produces random samples by first measuring it in the computational basis.
  - puts them in a family of the "**Born Machines**"

B. Generative Algorithm

The goal of generative is to generate samples from a probability distribution inferred from the data set.

- Begins by preparing 2V qubits in basis state $\langle 0|^{\bigotimes 2V}$ and entangles them by unitary operations
- Another set of 2V qubits are prepared and half are entangled with the first V entangled qubits and half with the second V entangled qubits.
- Two more unitary operations are applied to each new grouping of 2V qubits.
- The output are split into four groups and the process repeats for each group.
- The process ends when the total number of outputs = size of output one want to generate.

B. Generative Algorithm

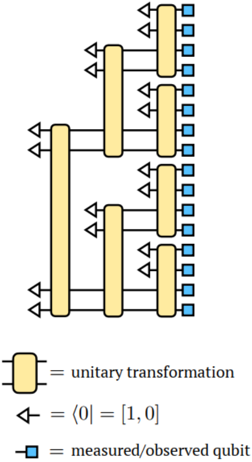


Fig 7: Generative tree tensor network model architecture for V = 2 connecting each subtree

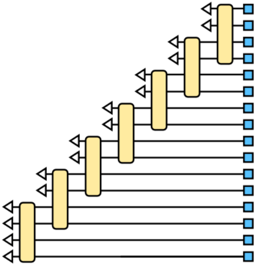


Fig 7: Generative MPS tensor network model architecture for V = 2 connecting each unitary

Model Architecture

For illustration: the model shown is with 16 inputs and 4 layers but actual experiment, model had 64 inputs and 6 layers

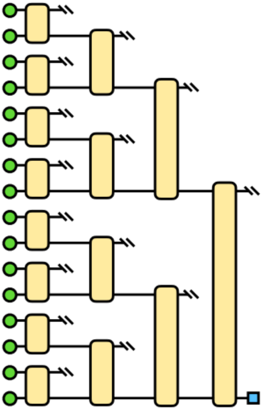


Fig 8 Model architecture used in experiment - special case of discriminative tree tensor

Loss Function

Let:

- $\wedge$ be the model parameters
- **d** be an element of the training dataset
- $p_\iota(\wedge, x)$ be probability of model to output ι
- ι_x be the correct label for input x
- $$p_{\text{largest false}}(\mathbf{\Lambda}, \mathbf{x}) = \max_{\ell \neq \ell_{\mathbf{x}}} \left[p_\ell(\mathbf{\Lambda}, \mathbf{x}) \right] \qquad (2)$$ be the probability of incorrect output

Loss Function

the loss function for a single input $\mathbf{x}$ to be

$$L(\mathbf{\Lambda}, \mathbf{x}) = \max(p_{\text{largest false}}(\mathbf{\Lambda}, \mathbf{x}) - p_{\ell_{\mathbf{x}}}(\mathbf{\Lambda}, \mathbf{x}) + \lambda, 0)^{\eta}, \tag{3}$$

and the total loss function to be

$$L(\mathbf{\Lambda}) = \frac{1}{|\text{data}|} \sum_{\mathbf{x} \in \text{data}} L(\mathbf{\Lambda}, \mathbf{x}). \tag{4}$$

Optimization

We want our model to generalize well to unobserved inputs,

- we optimize the loss function over subset of the training data.
  - we use a stochastic estimate of the true training loss given by:
  - and use a variant of the simulataneous perturbation stochastic approximation (SPSA)

$$\tilde{L}(\mathbf{\Lambda}) = \frac{1}{|\text{mini-batch}|} \sum_{\mathbf{x} \in \text{mini-batch}} L(\mathbf{\Lambda}, \mathbf{x}) \qquad (5)$$

Optimization Algorithm

1. Initialize the model parameters $\mathbf{\Lambda}$ randomly, and set $\mathbf{v}$ to zero.
2. Choose appropriate values for the constants, $a, b, A, s, t, \gamma, n, M$ that define the optimization procedure.
3. For each $k \in \{0, 1, 2, ..., M\}$, set $\alpha_k = \frac{a}{(k+1+A)^s}$ and $\beta_k = \frac{b}{(k+1)^t}$, and randomly partition the training data into mini-batches of n images. Perform the following steps using each mini-batch:
   (a) Generate random perturbation $\mathbf{\Delta}$ in parameter space.
   (b) Evaluate $g = \frac{\tilde{L}(\mathbf{\Lambda}_{old}+\alpha_k\mathbf{\Delta})-\tilde{L}(\mathbf{\Lambda}_{old}-\alpha_k\mathbf{\Delta})}{2\alpha_k}$, with $\tilde{L}(\mathbf{x})$ defined as in Eq. 5.
   (c) Set $\mathbf{v}_{new} = \gamma\mathbf{v}_{old} - g\beta_k\mathbf{\Delta}$
   (d) Set $\mathbf{\Lambda}_{new} = \mathbf{\Lambda}_{old} + \mathbf{v}_{new}$

λ

Results

Circuit was trained single output qubit to recognize grayscale images of size 8x8. The images were obtained from the MNIST dataset.

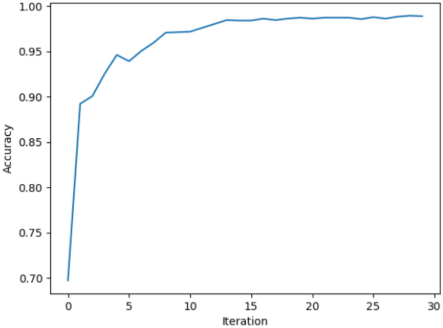


Fig 9: Test accuracy as a function of the number of epochs for 0's and 7's

Results

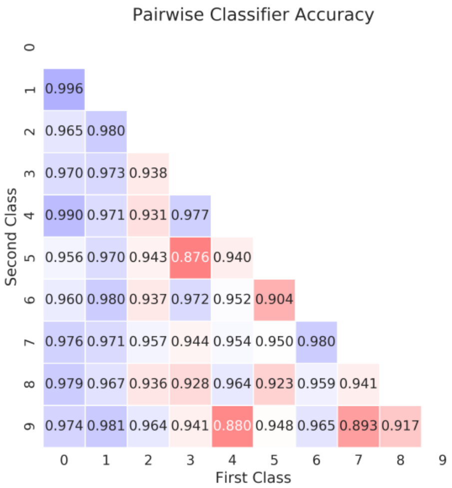


Fig 10. Test accuracy for each of the pairwise classifiers

Results

The networks was trained with the choices $\lambda = .234$, $\eta = 5.59$, $a = 28.0$, $A = 74.1$, $s = 4.13$,$t = .658$,$\gamma = 0.882$, $n = 222$ and achieved accuracy above 95%

It was observed that different choices of the hyper-parameters could significantly affect which pairs were classified accurately.

Implementation on Near-Term Quantum Devices

- Advantage of using tree or matrix product tensor network is they could be implemented using a very small number of physical qubits.
- The key requirement is the hardware must allow measurement of individual qubits without further disturbing the state of the others
  - This is also key for certain approaches to quantum error correction.

A. Qubit Efficient Tree Network Models

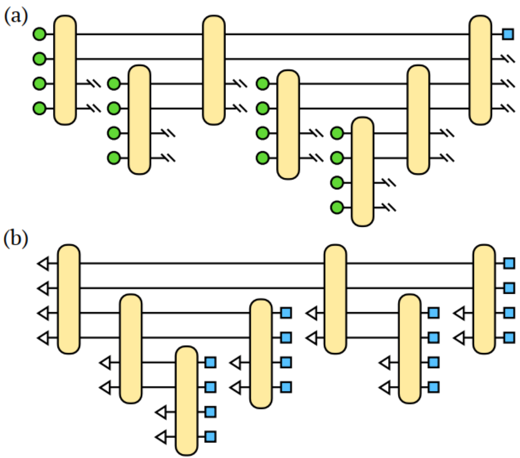


Fig 11: Qubit efficient scheme for (a) discriminative (b) generative tree models with V = 2 and N = 16 input or output

A. Qubit-Efficient Tree Network Models

- In the discriminative case (a) number of physical qubits needed is can be significantly reduced to $Q(N, V) = V \log(2N/V)$ for what would have required N physical qubits.
- For the generative case, generating (b) it is the same as the discriminative case.

Note

- The expressivity of tensor network model is measured by the bond dimension $D = 2^V$.
- The model used scales the bond dimension to $Q(N, D)\ \log(D) \log(N)$
- The SOTA for classical tensor network is $D = 2^{15}$ or 30000
- So for only $V = 16$, we could quickly exceed the power of any classical tensor network.

B. Qubit-Efficient Matrix Product Models

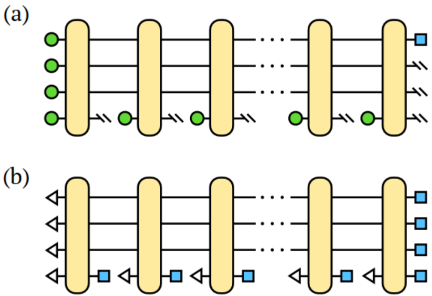


Fig 12: Qubit efficient scheme for evaluating (a) discriminative and (b) generative MPS for V = 3 qubits connecting the nodes of the network

B. Qubit-Efficient Matrix Product Models

- For an arbitrary number of qubits, you will require $V + 1$ physical qubits when using MPS in discriminative or generative case.

C. Noise Resilience

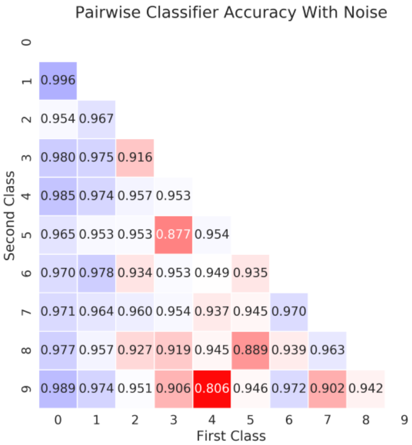


Fig 13: Test Accuracy for each pairwise classifier under noise

C. Noise Resilience

- we chose different hyper-parameters for training under noise
- the accuracy is comparable to the training without noise - only slight reduction

Conclusion and Discussion

- Most of the features that make tensor networks appealing for classical algorithms also make them a promising framework for quantum computing.
- Tensor networks strike a careful balance between expressive power and computational efficiency, thus can be useful for quantum circuits
- Based on rich theoretical understanding of their properties and powerful algorithms for optimizing them, they can provide interesting avenues for quantum machine learning research.

The End