Aalto University
School of Science
Department of Applied Physics

Niko Savola

Machine learning with many-body tensor networks

Submitted for approval: 2nd June 2022

Special exercise

PHYS-E0421 – Solid-State Physics

1 Introduction

Tensor networks have been originally developed for efficiently storing and manipulating high-dimensional quantum many-body states [1]. These variational families of wavefunctions emerge from low-entanglement representations of quantum states. Contemporary applications however include a wide range of fields, such as, machine learning [2], statistical mechanics [3], quantum chemistry [4], and cosmology [5]. This exercise aims to present the theory behind tensor networks and their use in machine learning. Then we present a numerical example for training a tensor network representing a quantum circuit, effectively pre-training a quantum computer for solving a problem.

2 Theory

2.1 Tensor networks

Tensor networks (TN) are a powerful tool for splitting a high-dimensional function into constituent parts of smaller tensors. Here a tensor refers to a multidimensional array, rank zero corresponding to a scalar, rank one to a vector, rank two to a matrix, and further ranks being referred to as rank n tensors. It is natural to use Tensor network notation (TNN), which can be considered a graphical generalisation of Einstein summation [6]. This notation maps tensors to nodes with n lines corresponding to the rank n of the tensor. The lines represent indices for the multidimensional array as demonstrated in Fig. 1(a).

The elegance of TNN arises from contracting tensors. Multiplication is done by linking the lines of the tensors. For example, matrix-vector multiplication consists of linking a node with one line to a node with two lines. The ensuing tensor will have one free line, resulting in a vector as expected. This logic is further demonstrated for a few examples in Fig. 1(b). Notice how multiplication of high-rank tensors is rendered graphically trivial. This is indeed useful for representing complex architectures of TN.

2.1.1 Machine learning

As popularity of deep learning has risen rapidly [8], interest in using TN as replacements or in conjunction with neural networks (NN) has been explored. Although TN are still an active research topic, some prominent TN architectures for machine learning have emerged, such as, Matrix Product States (MPS) and Tensor Renormalization Group (TRG).

TODO MPS

TODO TRG https://tensornetwork.org/trg/

One branch of research involves using a TN directly as machine learning model architecture. Another uses TNs to compress layers in neural network architectures or for other auxiliary

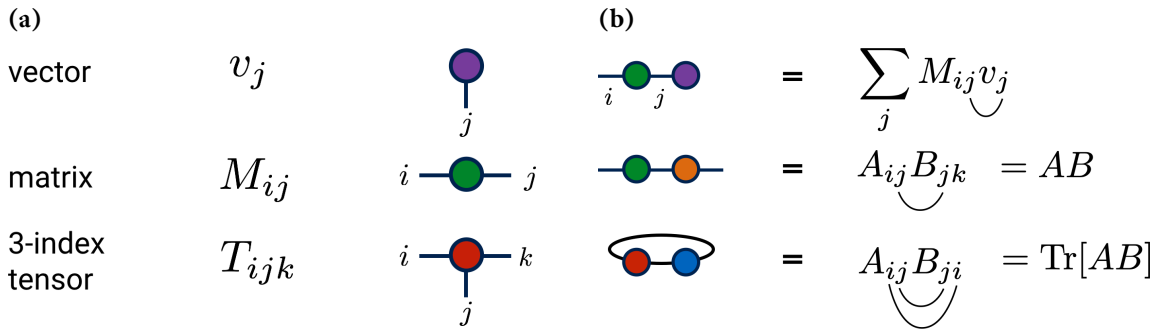

Figure 1: (a) Examples of low-rank tensors and their TNN representations. (b) Examples of tensor contractions for matrices. From top to bottom: matrix-vector product, matrix multiplication, and their trace. Note how the last example elegantly results in a scalar. Schematics from Ref. [7]

tasks.

a paradigm called differentiable programming.

In fact, it has been shown that there exists a mapping from generative neural networks referred to as Restricted Boltzmann machines to TNs [9], highlighting the link to deep learning.

2.2 Modelling many-body physics

todo work as Ansätze

3 Example: Training a quantum circuit

In this numerical example, we aim to train a quantum circuit for simulating short time evolution of the quantum transverse field Ising model [10]. This is done with unitary TN, which can easily be converted to quantum gate operations. The unitarity is ensured with local unitary tensors v obeying $v^\dagger v = vv^\dagger = I$ for identity tensor of the corresponding rank [11].

3.1 Implementation

We use an ansatz circuit consisting of quantum U gates and Controlled-Z gates (CZ). The U-gates represent all possible single-qubit operations and are of the form

$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}. \tag{1}$$

The CZ gates are two-qubit gates flipping the phase if one of the qubits is in the $|1\rangle$ state. It is a symmetric operation and is represented as

$$CZ(q_1, q_0) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes \sigma_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \tag{2}$$

where σ_z is a Pauli-Z gate. Our ansatz is implemented numerically in Python using the *quimb* library [12] as

```python
import quimb as qu
import quimb.tensor as qtn

n = 5
depth = 4

circ = qtn.Circuit(n)

for d in range(depth):
    for i in range(circ.N):
        params = qu.randn(3, dist='uniform') # initialize with random parameters
        circ.apply_gate('U3', *params, i, gate_round=d, parametrize=True)

    for i in (reversed(regs) if (d % 2 == 0) else regs):
        circ.apply_gate('CZ', i, i + 1, gate_round=d)

# final single qubit layer
for i in range(circ.N):
    params = qu.randn(3, dist='uniform') # initialize with random parameters
    circ.apply_gate('U3', *params, i, gate_round=r, parametrize=True)
```

The resulting ansatz consists of exclusively tensors. Thus, a graph complying with TNN presented in Fig. 1 can be generated and are shown in Fig. 2 with colouring displaying the quantum gates and qubit indices in (a) and (b), respectively. The free lines of the TN are consequently

(a) **(b)**

Figure 2: TNN graph of our ansatz quantum circuit with colouring for (a) quantum gates (b) qubit indices. The open ends of the TN, labelled with k and b, are consequently linked to a target problem.

connected to a unitary matrix representing the short time evolution of the transverse field Ising model $U(t) = \mathrm{e}^{-itH}$. This is effectively the target unitary matrix we are trying to replicate with the ansatz. We implement this in quimb for $t = 2$ as follows:

```python
H = qu.ham_ising(n, cyclic=False)
U_dense = qu.expm(-1j * (t := 2) * H)
U = qtn.Tensor(
    data=U_dense.reshape([2] * (2 * n)),
    inds=[f'k{i}' for i in range(n)] + [f'b{i}' for i in range(n)],
    tags={'U_TARGET'}
)
```

Our loss function to minimise is $L(H_{\text{ansatz}}) = 1 - \left|(H_{\text{ansatz}})_\gamma U^\gamma\right|$, where γ denotes indices for tensor contraction with *tensor index notation* [11]. We accelerate the training with JAX [13], a framework for automatic differentiation and a just-in-time compiler for Graphics processing units (GPU). Limited-memory BFGS (L-BFGS) is chosen as the optimization algorithm [14]. Additionally, basin-hopping is used to avoid local minima by adding random perturbations found minima.

3.2 Results

The training was performed using an NVIDIA RTX 3070 GPU for the ansatz circuit with five qubits and four layers of U-gate rotations per qubit as shown in Fig. 2(a).

As the system size was not large, it was possible to solve the exact time evolution of the state $U(t)$ and compare it to the evolution provided by the trained ansatz. This was done for a randomly-initialised state $|\psi\rangle$ by estimating the *fidelity* of the ansatz. As our states are pure, we can define the fidelity to be $\langle U^*\psi|H_{\text{ansatz}}\psi\rangle$. For this value, we attained 98.94% from several random trials. Thus, the ansatz time evolution appears competent.

Ultimately, the trained TN was converted back to unitary matrices for quantum circuits. This final circuit is depicted graphically in Fig. 3. As quantum computers with five or more qubits have already been realised [15], this circuit could be run on an actual quantum computer. Although it would typically first require transpilation to a native gateset from the general U-gates in the noisy intermediate-scale quantum era (NISQ) [16], [17].

Figure 3: Result of the trained quantum ansatz circuit generated with Qiskit [18]. The purple blocks represent U-gates with the given θ, ϕ, and λ parameters while the blue lines with dots are CZ-gates.

4 Summary

Todo but the motivation is that the architecture generalises to unknown problem

References

[1] S. R. White, 'Density matrix formulation for quantum renormalization groups,' *Phys. Rev. Lett.*, vol. 69, no. 19, pp. 2863–2866, November 1992, DOI: 10.1103/PhysRevLett.69.2863.

[2] C. Roberts, A. Milsted, M. Ganahl *et al.*, *TensorNetwork: A Library for Physics and Machine Learning*, 2019, arXiv: 1905.01330.

[3] M. Levin and C. P. Nave, 'Tensor Renormalization Group Approach to Two-Dimensional Classical Lattice Models,' *Physical Review Letters*, vol. 99, no. 12, p. 120 601, Sep. 2007, DOI: 10.1103/PhysRevLett.99.120601.

[4] S. R. White and R. L. Martin, 'Ab initio quantum chemistry using the density matrix renormalization group,' *The Journal of Chemical Physics*, vol. 110, no. 9, pp. 4127–4130, March 1999, DOI: 10.1063/1.478295.

[5] N. Bao, C. Cao, S. M. Carroll and A. Chatwin-Davies, 'de Sitter space as a tensor network: Cosmic no-hair, complementarity, and complexity,' *Physical Review D*, vol. 96, no. 12, p. 123 536, December 2017, DOI: 10.1103/PhysRevD.96.123536.

[6] J. C. Bridgeman and C. T. Chubb, 'Hand-waving and interpretive dance: an introductory course on tensor networks,' *Journal of Physics A: Mathematical and Theoretical*, vol. 50, no. 22, p. 223 001, May 2017, DOI: 10.1088/1751-8121/aa6dc3.

[7] E. M. Stoudenmire, *Tensor Diagram Notation*, April 2021, [Online]. Available: https://tensornetwork.org/diagrams (visited on 30th May 2022).

[8] M. M. Yapıcı, A. Tekerek and N. Topaloğlu, 'Literature Review of Deep Learning Research Areas,' *Gazi Journal of Engineering Sciences*, vol. 5, no. 3, pp. 188–215, December 2019, DOI: 10.30855/gmbd.2019.03.01.

[9] J. Chen, S. Cheng, H. Xie, L. Wang and T. Xiang, 'Equivalence of restricted Boltzmann machines and tensor network states,' *Physical Review B*, vol. 97, no. 8, p. 085 104, February 2018, DOI: 10.1103/PhysRevB.97.085104.

[10] A. Cervera-Lierta, 'Exact Ising model simulation on a quantum computer,' *Quantum*, vol. 2, p. 114, December 2018, DOI: 10.22331/q-2018-12-21-114.

[11] R. Haghshenas, 'Optimization schemes for unitary tensor-network circuit,' *Physical Review Research*, vol. 3, no. 2, p. 023 148, May 2021, DOI: 10.1103/PhysRevResearch.3.023148.

[12] J. Gray, 'quimb: A python package for quantum information and many-body calculations,' *Journal of Open Source Software*, vol. 3, no. 29, p. 819, 2018, DOI: 10.21105/joss.00819.

[13] J. Bradbury, R. Frostig, P. Hawkins *et al.*, *JAX: composable transformations of Python+NumPy programs*, 2018, [Online]. Available: http://github.com/google/jax.

[14] C. Zhu, R. H. Byrd, P. Lu and J. Nocedal, 'Algorithm 778: L-BFGS-B,' *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, December 1997, DOI: 10.1145/279232.279236.

[15] P. Pursula and H. Majumdar, *Finland's first 5-qubit quantum computer is | VTT News*, Espoo: VTT, IQM, November 2021, [Online]. Available: https://www.vttresearch.com/en/news-and-ideas/finlands-first-5-qubit-quantum-computer-now-operational (visited on 2nd Jun. 2022).

[16] E. Wilson, S. Singh and F. Mueller, 'Just-in-time Quantum Circuit Transpilation Reduces Noise,' May 2020, arXiv: 2005.12820.

[17] A. Li, S. Stein, S. Krishnamoorthy and J. Ang, 'QASMBench: A Low-level QASM Benchmark Suite for NISQ Evaluation and Simulation,' May 2020, arXiv: 2005.13018.

[18] M. S. ANIS, Abby-Mitchell, H. Abraham *et al.*, *Qiskit: An open-source framework for quantum computing*, 2021, DOI: 10.5281/zenodo.2573505.