Quantum Tensor Network Learning with DMRG

Gustav J L Jäger[1,2] Martin B Plenio[2] Hans-Martin Rieser[1]

1- Deutsches Zentrum für Luft- und Raumfahrt e.V. – Institut für KI-Sicherheit
Wilhelm-Runge-Straße 10, 89081 Ulm, Germany

2- Universität Ulm – Institut für Theoretische Physik and IQST,
Albert-Einstein-Allee 11, 89081 Ulm, Germany

Abstract. Tensor Networks are a relatively new machine learning approach. The architectures proposed initially are inspired by approaches from quantum many-body physics simulations. One common layout is the matrix product state (MPS) also known as a tensor train optimized with gradient descent techniques. We introduce a global normalization condition, so that the MPS represents a quantum state. We investigate two optimization methods that find the locally optimal tensors and compare them regarding their effectiveness. One is based on gradient descent and the other on an adaptation of DMRG.

1 Introduction

Tensor networks (TN) originate from the field of quantum many-body physics [1, 2]. They are used as a prominent method to simulate quantum systems [3], and approximate their properties such as ground states [4]. Their main advantage lies in their ability to approximate large quantum states with limited storage. Quantum-inspired tensor networks have found an application in classical machine learning models, see [5].

The application of tensor networks to quantum machine learning connects both fields [6]. To raise the whole potential of TN approaches developed in quantum physics it is necessary to transfer the local optimization techniques, namely Density Matrix Renormalization Group (DMRG), to the machine learning context. In this work we explore how this DMRG approach with the underlying Lanczos method to quantum machine learning. One challenge that arises, is that quantum states and quantum channels have normalization conditions due to quantum probability conservation [7].

The basic tensor network ansatz we investigate is inspired by [5]. However, we introduce a normalization condition on the tensor network, such that contracting it yields a normalized vector that may be mapped to a quantum state. We then present how the gradient descent algorithm can be modified to take into account this condition and how a modified DMRG algorithm can be applied for optimization as well. The latter is based on a post processing step taking into account the loss function of the machine learning problem. Finally, we compare how the introduction of the normalization condition impacts the accuracy and the loss, when classifying the MNIST dataset.

2 Tensor Network based Machine Learning

Our tensor network approach is based on an MPS similar to the method outlined in [5]. Additionally, we introduce a normalization condition, such that contracting the tensor network yields a normalized vector, see Fig. 1. Inference of this setup can be run directly on a quantum computer: Either checking each label $|l_i\rangle$ separately or maximally entangling system B with a readout site. The generating MPS can be applied as a quantum circuit following a translation method, e.g. the one given by [8].

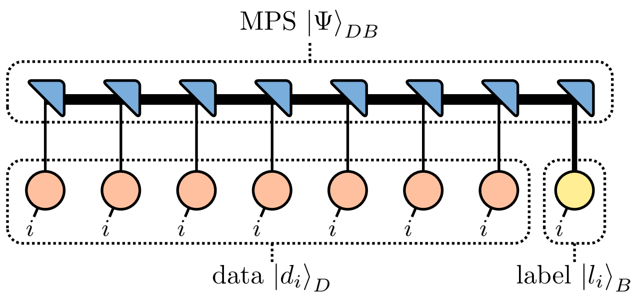

Fig. 1: Architecture of the underlying tensor network. The trainable part of the setup is the MPS, here in blue. The encoded input data of datapoint i of the dataset is represented by the orange tensors. The yellow tensor represents the single site which serves as the output of the label.

To maintain a normalized MPS, we use site canonization while sweeping, which is the process of optimizing the sites iteratively until the MPS converges [3]. This approach is found both in gradient based learning [5] and in DMRG [4]. To deal with complex quantum states, we modify the mean square error loss function from [5] to be

$$L = \frac{1}{2N} \sum_{i=1}^{N} \big(\langle f_i|_B - \langle l_i|_B \big) \big(|f_i\rangle_B - |l_i\rangle_B \big), \tag{1}$$

where

$$|f_i\rangle_B = (\langle d_i|_D \otimes \mathbb{1}_B)\, |\Psi\rangle_{DB} \,. \tag{2}$$

Here, $|l_i\rangle_B$, the MPS $|\Psi\rangle_{DB}$, andthe data representation $|d_i\rangle_D$ are normalized, but the ML output $|f_i\rangle_B$ is not. Given Eq. (2), we obtain the loss

$$L = \frac{1}{2} + \frac{1}{2N} \sum_{i=1}^{N} \Big[\langle\Psi|\, (|d_i\rangle\langle d_i|_D \otimes \mathbb{1}_B)\, |\Psi\rangle - 2\mathrm{Re}\big[\, \langle\Psi|_{DB}\, (|d_i\rangle_D \otimes |l_i\rangle_B)\big] \Big].$$

As abbrevations, we define the Hermitian A and the vector b

$$A = \sum_{i=1}^{N} |d_i\rangle\langle d_i|_D \otimes \mathbb{1}_B \quad \text{and} \quad b = \sum_{i=1}^{N} |d_i\rangle_D \otimes |l_i\rangle_B \,.$$

For local optimization, one needs locally effective variants A_s and b_s on the site s. They are obtained by removing the tensor of the respective site in the MPS and contracting it with A or b, [5, 4]. Both A and A_s are positive semidefinite. $\langle k|b\rangle = 0$ if $|k\rangle$ is in the kernel of A, otherwise a negative loss could be found, which contradicts Eq. (1). Thus, from now on the kernel of A is removed from A, when calculating $A^{-1}|b\rangle$. We use the same initialization procedure as [5] for the normalized MPS to avoid vanishing gradients. Furthermore, a lookup table of major building blocks of A_s and b_s is maintained while sweeping. This ensures that the sweep only depends linearly on the number of sites, but at the cost of storage space.

3 Gradient descent with Normalization

The local optimization problem we tackle is given by

$$|\Psi_s\rangle = \underset{|\Psi\rangle\,\forall\,\langle\Psi|\Psi\rangle=1}{\arg\min}\ \langle\Psi|\,A_s\,|\Psi\rangle - 2\mathrm{Re}(\langle\Psi|b_s\rangle). \tag{3}$$

If $|\Psi_s\rangle$ is not normalized, the optimum is $A^{-1}|b\rangle$, see [9, 10]. Normalizing this optimal solution directly does not yield the optimal solution for the problem with the normalization constraint. This can be seen from the following example:

$$A_s = \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix} \text{ and } |b_s\rangle = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \text{ with } |\Psi_s\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ but } A_s^{-1}|b_s\rangle \approx \begin{pmatrix} 0.14 \\ 0.71 \end{pmatrix}$$

A simple approach to solve the constraint problem is to use the gradient $|g\rangle = A_s|\Psi\rangle - |b_s\rangle$ to give a small update with step size α on $|\Psi\rangle$ and then normalize afterwards. This is known as projected gradient descent [11]. Thus, we obtain

$$|\Psi'\rangle = \frac{|n\rangle}{\sqrt{\langle n|n\rangle}} = \frac{|\Psi\rangle - \alpha\,|g\rangle}{\sqrt{(\langle\Psi| - \alpha\,\langle g|)(|\Psi\rangle - \alpha\,|g\rangle)}}.$$

Thus, $|\Psi\rangle$ is updated along the negative gradient on the surface of the unit sphere of $|\Psi\rangle$'s Hilbert space. One simple improvement of the approach is optimizing the step size α. Setting the derivative of the loss w.r.t. α equal to zero yields the condition

$$0 = \mathrm{Re}(\langle g'|n\rangle)\mathrm{Re}(\langle n|g\rangle) - \mathrm{Re}(\langle g'|g\rangle)\,\langle n|n\rangle \tag{4}$$

with the gradient $|g'\rangle$ of $|\Psi'\rangle$ obtained by the next iterative step. We disregard the solution for $\langle n|n\rangle \to \infty$ as this is associated with an infinite step size α. Calculating $|g'\rangle$ requires one additional matrix-vector multiplication $A_s|g\rangle$ that can be reused to calculate $|\Psi'\rangle$ after the optimal α is found.

We obtain the derivative of Eq. (4) by using the product rule and the derivatives:

$$\frac{d}{d\alpha}|g\rangle = 0,\ \frac{d}{d\alpha}|n\rangle = -|g\rangle\,,\text{ and } \frac{d}{d\alpha}|g'\rangle = A_s\left(\frac{|n\rangle\,\mathrm{Re}(\langle n|g\rangle)}{\langle n|n\rangle^{\frac{3}{2}}} - \frac{|g\rangle}{\sqrt{\langle n|n\rangle}}\right).$$

We then solve for α with Newton iterations. A local maximum $\alpha \neq \pm\infty$ may exist. To avoid it, $\alpha = 0$ is chosen as the starting point of the Newton iteration.

4 A Modified DMRG Algorithm

Optimization techniques based on Krylov subspaces have shown to be far more effective than simple gradient descent [10]. Exemplary methods are conjugate gradient descent algorithm, and the Lanczos algorithm used in DMRG. However, DMRG with the underlying Lanczos algorithm does not involve the second term in Eq. (3), which appears in the machine learning problem. Thus, to modify DMRG to fit this problem, we define a post-processing step after the Lanczos algorithm. First, we compress A_s into $A'_s = VTV^\dagger$ with the unitary V and tridiagonal and real T obtained from the Lanczos algorithm. With the fast diagonalization of T [10], we obtain the eigendecomposition of $A'_s = \sum_{i=1}^m e_i \left|\lambda_i\right\rangle \left\langle\lambda_i\right|$. The starting vector of the Lanczos iteration is chosen to be $|b_s\rangle$ so that $|b_s\rangle$ is fully represented in the basis of the Lanczos vectors and thus the eigenbasis of A'_s.[1] After compressing, our local effective loss function results in the gradient

$$x\left|\Psi_s\right\rangle \overset{!}{=} \left|\mathrm{g}\right\rangle := A'_s\left|\Psi_s\right\rangle - \left|b_s\right\rangle . \tag{5}$$

For the optimal normalized $|\Psi_s\rangle$ the gradient is not 0 but equal to $x\left|\Psi_s\right\rangle$, with a real scaling factor $x = \pm\sqrt{\langle \mathrm{g}|\mathrm{g}\rangle}$. This ensures the constraint of the local optimum to the unit sphere on which $|\Psi_s\rangle$ lives, i.e., the gradient is perpendicular to the unit sphere. Eq. (5) can also be derived by Lagrangian optimization with the normalization constraint on $|\Psi\rangle$. Assuming $(A'_s - x\mathbb{1})$ is invertible, $|\Psi_s\rangle$ is given by

$$\left|\Psi_s\right\rangle = (A'_s - x\mathbb{1})^{-1}\left|b_s\right\rangle .$$

Given that $|b_s\rangle$ is completely decomposable into the $|\lambda_i\rangle$, we obtain

$$\left|\Psi_s\right\rangle = (\sum_{i=1}^{N} \frac{1}{e_i - x}\left|\lambda_i\rangle\langle\lambda_i\right|)\left|b_s\right\rangle .$$

We define $b_{si} = \langle b_s|\lambda_i\rangle\,\langle\lambda_i|b_s\rangle$, the overlap of the vector $|b_s\rangle$ and the respective eigenvector. With the normalization condition $\langle\Psi_s|\Psi_s\rangle = 1$ we obtain

$$1 = \sum_{i=1}^{m} \frac{b_{si}}{(e_i - x)^2}. \tag{6}$$

We recall that A_s and subsequently A'_s are positive semidefinite. Thus, their smallest eigenvalue is non-negative $0 \le e_{\min}$. Eq. (6) suggests multiple solutions x_s, yet there is only one $x_s < e_{\min}$ because the right-hand side of Eq. (6) is strictly monotone for $x < e_{\min}$, i.e. $x_1 < x_2 \Leftrightarrow \mathrm{RHS}(x_1) < \mathrm{RHS}(x_2)$.

Using the compressed A' in the loss function Eq. (3), we obtain the loss

$$L(x) = \sum_{i=1}^{m} \frac{e_i b_{si}}{(e_i - x)^2} - 2\frac{b_{si}}{(e_i - x)}$$

[1] Similar lines of reasoning can be found in randomized numerical linear algebra, see [12].

with $0 < e_i, b_{si}$. For $L(x)$ we find $L(-|x|) \leq L(|x|)$. Thus, the global minimum can only be obtained by selecting $x_s < 0$. As established above, there can be only one solution $x_s < 0$, which thus is the global minimum. In the gradient descent algorithm, we expect the gradient to converge on the same $|\Psi_s\rangle$. The solution given by $x_s < 0$ is obtained because the resulting gradient update is positive and thus amplifies the direction $|\Psi_s\rangle$ over other directions. Thus, the gradient descent algorithm converges on the global minimum.

Without an analytical solution to Eq. (6), we find this $x_s < 0$ by using a Newton iteration. Due to the asymptotical nature of Eq. (6), it only converges if the iteration is initialized with $x_s < x_{\text{init}} < 0$. We find x_{init} by checking whether the right-hand side of Eq. (6) is larger than 1. As a side note, during the Lanczos iteration, to avoid numerical instability, we substract $a_i\, |v_k\rangle\langle v_k|$ and $\eta_i(|v_{k-1}\rangle\langle v_k| + |v_k\rangle\langle v_{k-1}|)$ from A, with entries of the tridiagonal matrix a_i and η_i, and the Lanczos vectors $|v_k\rangle$.

5 Experimental validation

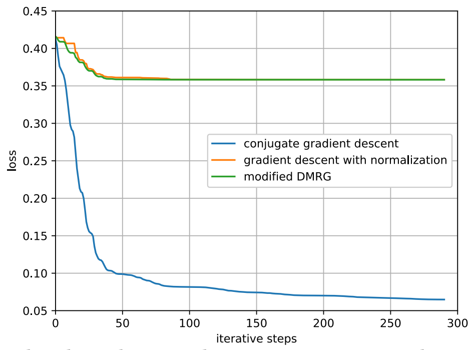

Fig. 2: The loss is displayed over the iterative steps that are given by three sweeps. The modified DMRG method, and the gradient descent with normalization are compared with conjugate gradient descent.

To validate our approach and to compare to standard methods, we apply the standard conjugate gradient descent and our two outlined algorithms to the same subset of 5000 MNIST images that are rescaled to 7x7. 4000 images are used in training and 1000 for testing because the matrix A scales with the squar of the number of images. The three approaches were tested for three full two-site sweeps with a maximum bond dimension of 20. Thus, we have $(49 \times 2 - 1) \times 3 = 291$ total local iteration steps, where each iterative step solves our local optimization problem. Fig. 2 displays the loss over the number of local iteration steps during the sweep. The results for loss and accuracy are given in Tab. 1. We observe that introducing the normalization condition significantly impacts both loss and accuracy. However, the norm of the MPS obtained with conjugate gradient descent is $3.9 \cdot 10^6$. Normalizing this MPS, gives only vanishing overlaps in the loss function Eq. (1), which results in a trivial loss of ≈ 0.5.

method	training loss	test loss	training acc.	testing acc.
initialization	0.43646	0.43984	67.725%	63.300%
conjugate grad. desc.	0.06487	0.08680	96.950%	94.700%
norm. grad. desc.	0.35820	0.36256	75.750%	73.000%
modified Lanczos	0.35820	0.36257	75.775%	73.100%

Table 1: Overview of accuracy and loss of the different methods.

6 Conclusion and Outlook

In this work, we take the step to adapt tensor network methods to approach quantum machine learning more efficiently. Implementing the normalization condition is necessary to make the algorithm fit for deployment on quantum computers. However, the results of the validation show that the performance needs to be increased to be competitive with other contemporary algorithms. Clearly, further improvements are needed to achieve parity with classical tensor network methods. To this end, more complex normalization conditions should be studied, like those in quantum channels and improvements in the evaluation of the methods need to be developed. Overall, major research still needs to be done to bring machine learning efficiently to quantum computers.

References

[1] R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. Annals of Physics, **349**:117–158, 2013.

[2] J. C. Bridgeman and C. T. Chubb. Hand-waving and interpretive dance: an introductory course on tensor networks. J. Phys. A: Math. Theor., **50**:223001, 2017.

[3] S.-J. Ran, E. Tirrito, C. Peng, L. Tagliacozzo X. Chen, G. Su, and M. Lewenstein. Tensor Network Contractions. Springer Cham, 2020.

[4] U. Schollwöck. The density-matrix renormalization group. Rev. Mod. Phys., **77**:259, 2005.

[5] M. E. Stoudenmire and D. J. Schwab. Supervised learning with tensor networks. NeurIPS, **29**:4799, 2016.

[6] H.-M. Rieser, F. Köster, and A. P. Raulf. Tensor networks for quantum machine learning. Proc. R. Soc. A., **479**:20230218, 2023.

[7] M. M. Wilde. Quantum Information Theory. Cambridge University Press, 2 edition, 2017.

[8] S.-J. Ran. Encoding of matrix product states into quantum circuits of one- and two-qubit gates. Phys. Rev. A, **101**:032310, 2020.

[9] J. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[10] Gérard Meurant. The Lanczos and Conjugate Gradient Algorithms. Society for Industrial and Applied Mathematics, 2006.

[11] S. Bubeck. Convex optimization: Algorithms and complexity. Foundations and Trends® in Machine Learning, **8**(3-4):231–357, 2015.

[12] P-G. Martinsson and J. A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. Acta Numerica, **29**:403–572, 2020.