Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Boxi Li

born in Jiangsu, China

2018

Understanding and improving the neural network representation of quantum many-body ground states

This Bachelor Thesis has been carried out by Boxi Li at the
Kirchhoff-Institute for Physics in Heidelberg
under the supervision of
Prof. Thomas Gasenzer and Dr. Martin Gärttner

Abstract: For many years, neural networks have demonstrated their power in approximating different kinds of functions and modeling nonlinear processes. Recently, their abilities have been exploited for tackling the exponential complexity of representing a quantum many-body wave function: A specific type of neural networks, the Boltzmann machine, was introduced to compress the information encoded in the wave function. In this thesis, we attempt to understand the role that this neural network architecture and the network weights play in representing the quantum many-body states. We also find that the optimization method used in previous works suffer from misconvergence in many cases. To improve the convergence performance, we introduce a global search algorithm called Differential Evolution. This method can also be generalized to other neural network optimization problems and used for studying the geometry of the objective function.

Zusammenfassung: Seit vielen Jahren werden neuronale Netze genutzt um verschiedene Arten von Funktionen zu approximieren und nichtlineare Prozesse zu modellieren. Vor kurzem wurde seine Fähigkeit genutzt um die exponentielle Komplexität der Darstellung einer Quanten-Vielteilchenwellenfunktion in Angriff zu nehmen: Eine spezielle Art von neuralen Netzen, die Boltzmann-Maschine, wurde eingeführt, um die in der Wellenfunktion kodierte Information zu komprimieren. In dieser Arbeit versuchen wir zu verstehen, welche Rolle diese neuronale Netzwerkarchitektur und die Netzwerkgewichte bei der Darstellung der Quanten-Vielteilchenzustände spielen. Wir zeigen, dass die Optimierungsmethode, die in früheren Arbeiten verwendet wurde, oftmals zu Fehlkonvergenz führen kann. Um die Konvergenzverhalten zu verbessern, führen wir einen globalen Suchalgorithmus namens Differential Evolution ein. Diese Methode kann auch auf andere Optimierungsprobleme neuronaler Netze verallgemeinert und bei der Untersuchung der Geometrie der Zielfunktion verwendet werden.

Contents

1 Introduction

In quantum physics, the physical world is described by the wave function. An explicit expression of the wave function requires, in general, an exponential amount of information. For instance, the wave function describing a spin system with N sites has 2^N complex coefficients, rendering an exact simulation of a large system on a classical computer impossible. On the other hand, the wave function of a meaningful physical many-body system may only cover a small corner of the entire Hilbert space. The manifold of states that can be reached in polynomial time with a local Hamiltonian is also exponentially small[1]. Therefore, compression of the quantum state into an efficient representation with a limited number of parameters can be a useful method to solve quantum many-body problems with accessible classical resources. The key feature of those representations relies on reducing the number of parameters to a reasonable range while preserving important physical features like quantum entanglement.

The classical way of constructing a representation of correlated physical systems is starting from a non-interacting Hamiltonian, for which a well known or even analytically solvable solution is available, and then perturbatively adding mixed terms with corresponding coupling coefficients to switch on the interaction. Many of those methods will, however, fail to describe complex systems e.g. in the presence of strong correlations, because many assumptions used in simplifying the wave function are no more valid as the interactions get stronger. In this thesis, we use an efficient representation based on an artificial neural network architecture to compress quantum many-body states. Unlike the strategy mentioned above, the neural network starts from a very general ansatz with ample representation possibilities. By encoding the physical properties in the training methods, it enforces the system to evolve towards the desired representation. The flexibility of the network structure makes it possible to adapt to different requirements, without changing the main learning procedure.

The type of neural network structure we are going to use in this thesis is called the Boltzmann machine. It has recently been introduced to represent quantum many-body states[2]. Boltzmann machines can be divided into two groups, the Restricted Boltzmann machine (RBM) and the Deep Boltzmann machine (DBM), depending on the number of layers allowed for the neurons. Numerical results have shown that such a Boltzmann machine, trained with a learning algorithm, is capable of representing a lot of many-body states and tracking the time evolution[2][3]. The relationship of this neural network model with other quantum states, like Matrix Product States[4], Graph States[5][6], String-Bond States[7] and chiral topological states[7] has also been analyzed, which demonstrate the efficiency, representation power, and limitation of this model under different conditions. It has also been proven that the entanglement entropy of a short-range RBM satisfies an area law while a long-range RBM could exhibit volume-law entanglement[8]. In practical applications, the RBM is favored over the DBM since information can be extracted very efficiently. If the restriction on the number of layers is removed, the exact representation of most quantum states and the time evolution can then be achieved with the DBM structure by adding neurons and modifying the existing network parameters in a deterministic way[6][9][10]. The difference between the RBM and the DBM in their representation power is also demonstrated by the construction, proving the existence of states which only have an exact representation in the form of DBMs[6].

Despite the great success in many aspects, understanding the neural network still poses a great challenge due to its complicated structure and non-linearity. The numeric algorithm applied is also in its early stage and far from practical applications beyond benchmarking cases where exact solutions can be obtained by other methods. In this thesis, we try to gain an insight into the structure of the complex RBM by construction and numerical experiments and improve the usability of the learning algorithm. In section 2, we show the generic ansatz of the RBM representation as well as the learning algorithm: stochastic reconfiguration. The energy surface during the ground state search is studied for the Transverse-field Ising Model (TFIM) and illustrated from several different aspects. We show how the basis of the wave function and the interaction in the Hamiltonian change the RBM representation of the ground states. During the study, some weaknesses of the learning algorithm appear while finding many-body ground states, namely the misconvergence of the optimization. The situation and reasons for this misconvergence are qualitatively studied in section 3.1. The main problem turns out to be the gradient descent method used in the optimization. In section 3, the usability of the algorithm will be improved by introducing a global search algorithm, Differential Evolution (DE), and detailed tuning of hyper-parameters. DE can be adapted so that, for problems with no previous knowledge about the ground state, the desired balance between accuracy and efficiency can be achieved by tuning the hyper-parameters. The improved algorithm is also benchmarked for small and large systems with the TFIM.

2 Boltzmann machine and neural network representation

2.1 Boltzmann machine as a representation of quantum states

The Boltzmann machine is a network structure consisting of neurons and connections among them, with each neuron taking a binary value -1 or 1 and each connection a

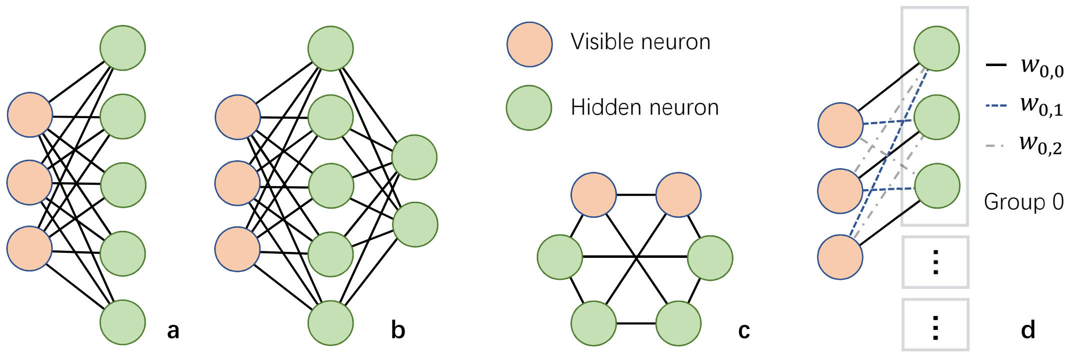


Figure 1: Examples of Boltzmann machines. The red and green circles denote the visible and hidden neurons and a weight is assigned to each connection between two neurons. In the figure, **a** is an RBM, while **b** and **c** are examples of DBMs. In general, all complex Boltzmann machines can be transformed into the form of 2-layer DBMs like in **b**[6]. **d** illustrates the simplification of network structure and weights due to the symmetry in the Hamiltonian of TFIM. The hidden neurons are grouped and the same lifestyle is used for weights with the same value.

real number (Fig. 1). Following an evaluation rule, the network will give a positive number depending on the values of neurons and connections. This structure can then be used to represent functions of binary vectors by leaving some of the neurons open as input, called visible neurons, and treating other neurons as part of the structure, called hidden neurons.

In classical Boltzmann machines, the evaluation rule is defined by

$$\Psi(\mathbf{v};\Omega) = \sum_{\{h_j\}} e^{\sum_i \theta_i s_i + \sum_{i<j} s_i w_{i,j} s_j} \tag{1}$$

where $\mathbf{s} = (\mathbf{v}, \mathbf{h})$ denotes all visible and hidden neurons and $\Omega = (\theta, w)$ are parameters associated to the connections, which are also called network weights. $\{h_j\} \in \{-1, 1\}^M$ denotes a summation over 2^M hidden neuron configurations with M the number of hidden neurons. For a given binary vector $\mathbf{v} = (v_1, v_2 \cdots v_n)$ with $v_i \in \{1, -1\}$, the evaluation of the network will give a positive number, which can be viewed as a probability up to a normalization since the outcome is always positive.

The specialty of the Boltzmann machine introduced in reference [2] is that, instead of real numbers, it uses complex numbers as weights. The network structure and the weights then serve as a representation of a many-body quantum state defined in the computational basis $|\Psi\rangle = \sum_{\mathbf{v}} \Psi(\mathbf{v}) |\mathbf{v}\rangle$ up to a normalization factor.

In general, the Boltzmann machine is divided into two types. One type contains Boltzmann machines with no connection between hidden neurons or between visible neurons, called Restricted Boltzmann machine (RBM). One example of RBMs is depicted in Fig. 1 (a). With this restriction, the evaluation rule simplifies to

$$\Psi(\mathbf{v};\mathbf{\Omega}) = \sum_{\{h_j\}} e^{\sum_i a_i v_i + \sum_j b_j h_j + \sum_{i,j} v_i w_{i,j} h_j} \tag{2}$$

If a Boltzmann machine has N visible and M hidden neurons, carrying out the sum over all possible hidden neuron configuration h_j will take 2^M summations. However, for the RBM, there are only connections between visible and hidden neurons, the sum of each h_j configuration can, therefore, be carried out separately and explicitly

$$\Psi(\mathbf{v};\mathbf{\Omega}) = e^{\sum_i a_i v_i} \prod_j \cosh\left(b_j + \sum_i v_i w_{i,j}\right) \tag{3}$$

where the constant factor 2 for each cosh function is neglected. The complexity is then the equivalent as going through all $N \cdot M$ connections. In practice, RBMs have a much broader range of applications because of their efficiency in the evaluation.

The other type, the Deep Boltzmann machine (DBM), is the complementary of the RBM (Fig. 1 (b) and Fig. 1 (c)). As more layers of hidden neurons are allowed, the simplification above loses its validity and resources needed for the evaluation of DBMs scale exponentially with the system size because of the summation over all the hidden neuron configurations $\{h_j\}$. In classical machine learning, importance sampling can be applied to reduce the computation effort. However, most of those methods are based on probability theory, which cannot be applied here because of the complex weights. Therefore, the application of the DBM in representing quantum wave functions is only restricted to very small systems.

The mathematical foundations of the ability of the classical Boltzmann machine to represent intricate probability distributions are given in [11]. It is proven that adding hidden units will always improve the approximation and RBMs are a universal approximator of discrete distributions. Although those proofs are based on probability theory, many examples have shown that the complex Boltzmann machine is capable of representing intricate quantum many-body systems[8]. Furthermore, theoretical analysis has also shown that the DBM can approximate most physical states efficiently, including many states with high entanglement, while the RBM is limited by its simplified structure in some cases[6]. The limitation of the RBM lies exactly in the reduction of its computation requirement. Those states that are proven to require an exponential amount of computation resources cannot be represented exactly by the RBM, nor can this be done by any other methods with efficient evaluation. Despite this limitation, the RBM still shows outstanding performance in many examples, therefore we focus here on the RBM and its application in solving the quantum many-body problem.

2.2 Learning rule for searching the RBM representation of many-body ground states

One of the most important tasks in quantum physics is to find the unknown ground state of a physical system. For neural networks, it means that one needs to define a training rule so that by varying the network weights, the network converges towards a good approximation of the desired state. In reference [2], a stochastic framework is presented which is capable of finding the ground state of a given Hamiltonian and solving the time-dependent Schrödinger equation with the RBM.

To find the best representation of the ground state of a given Hamiltonian $\hat{\mathcal{H}}$, we follow the variational method and define a learning rule for network optimization. One varies the network weights so that the expectation value

$$E(|\Psi\rangle) = \frac{\langle\psi|\hat{\mathcal{H}}|\psi\rangle}{\langle\psi|\psi\rangle} \tag{4}$$

is minimized. In the end, the wave function $|\psi\rangle$ represented by the neural network will be a good numeric approximation of the ground state. At each step, a stochastic estimation of the energy gradient is obtained and the next set of weights is calculated by an improved gradient descent method, called Stochastic Reconfiguration (SR).

SR is an improved gradient descent method introduced by Sorella to alleviate the sign problem in Variational Monte Carlo methods [12][13]. The idea arises from the imaginary time evolution, where the ground state Ψ_G is filtered out by iteratively applying an operator $\hat{G} = e^{-H\Delta t}$ for a short time Δt. To avoid the calculation of the exponential of Hamiltonian, one can replace it by $\hat{G} = \Lambda - \hat{\mathcal{H}}$ with a sufficiently large constant Λ. Then a projected state is defined by

$$|\Psi_P\rangle = (\Lambda - \hat{\mathcal{H}})\,|\Psi_n\rangle \tag{5}$$

which is closer to the ground state compared to the current state $|\Psi_n\rangle$. This projected state can then be approximated in the following way: One first writes down the general form of the updated wave function

$$|\Psi_{n+1}\rangle = \alpha_0\,|\Psi_n\rangle + \sum_{k=1}^{K} \delta\alpha_k \frac{\partial}{\partial\alpha_k}\,|\Psi_n\rangle \tag{6}$$

where $\delta\alpha_k$ are variations of parameters α_k of the n-th generation defined by

$$\alpha_k^{n+1} = \alpha_k^n + \gamma\delta\alpha_k^n. \tag{7}$$

with a learning rate γ. Notice that a_0 in equation 6 is not used in the numerical algorithm since it plays the same role as the learning rate. We also define the variational derivatives $\hat{\mathcal{O}}$

$$\hat{\mathcal{O}}_k \Psi_n(S) = \frac{\partial}{\partial \alpha_k} \ln \Psi_n(S) \tag{8}$$

with $\Psi_n(S)$ the coefficient for spin configuration S so that

$$|\Psi_{n+1}\rangle = \alpha_0 |\Psi_n\rangle + \sum_{k=1}^{K} \delta\alpha_k \hat{\mathcal{O}}_k |\Psi_n\rangle \tag{9}$$

To make the updated function as close as possible to the projected function, a set of mixed average correlation functions for any k is set to be equal

$$\frac{\langle \Psi_n | \hat{\mathcal{O}}_k^\dagger | \Psi_{n+1} \rangle}{\langle \Psi_n | \Psi_{n+1} \rangle} = \frac{\langle \Psi_n | \hat{\mathcal{O}}_k^\dagger | \Psi_P \rangle}{\langle \Psi_n | \Psi_P \rangle} \tag{10}$$

After plugging in equation 5 and 6, the constant Λ cancels out and one obtains an updating rule:

$$\alpha_k^{n+1} = \alpha_k^n - \gamma S^{-1} F \tag{11}$$

with

$$S_{k,l} = \langle \Psi_n | \hat{\mathcal{O}}_k^\dagger \hat{\mathcal{O}}_l | \Psi_n \rangle - \frac{\langle \Psi_n | \hat{\mathcal{O}}_k^\dagger | \Psi_n \rangle \langle \Psi_n | \hat{\mathcal{O}}_l | \Psi_n \rangle}{\langle \Psi_n | \Psi_n \rangle} \tag{12}$$

$$F_k = \langle \Psi_n | \hat{\mathcal{O}}_k^\dagger \hat{\mathcal{H}} | \Psi_n \rangle - \frac{\langle \Psi_n | \hat{\mathcal{O}}_k^\dagger | \Psi_n \rangle \langle \Psi_n | \hat{\mathcal{H}} | \Psi_n \rangle}{\langle \Psi_n | \Psi_n \rangle} \tag{13}$$

where F is called the force and S the covariance matrix. Notice that the learning rate γ is not the only parameter determining the step size of each upgrade. The vector $S^{-1}F$ itself is self-adaptive in the sense that, through the SR procedure, the step size keeps decreasing as one goes closer to the ground state.

The most fragile part of this algorithm is the inverse of S. In practice, a regularization $S_{k,l}^{reg} = S_{k,k} + \lambda(n)\delta_{k,l}S_{k,k}$ is introduced in reference [2] to avoid S being non-invertible, but the procedure could still fail, especially at the early stage where the energy surface is complicated and varies rapidly. As the range of initial weights increases, this failure happens more and more frequently. Therefore, the initial weights are suggested to be chosen close to zero to maintain the stability of this inverse.

2.3 The energy surface of the ground state search

We can now use the leaning rule introduced above to optimize the network weights to best represent the ground state by minimizing the energy expectation. The objective function, i.e. the energy surface, as a function of the RBM weights is determined by equations 3 and 4, which is a sum of 2^N energy terms with each term consisting of M multiplications of cosh functions. For large system size N, metropolis sampling is used to sample the coefficients according to their amplitude in the current network state[2]. As training the RBM refers to an optimization in a high dimensional real space, there is no direct way to imagine how this energy surface may look like. However, one can still gain some insight into this intricate network structure by looking into some special states and visualizing the energy surface on a lower dimensional subspace. In this section, the Hamiltonian of the TFIM is used as an example to analyze the energy surface. We show that the weight distribution is dependent on the basis in which the state is given, on the degeneracy of the network representation and on the interaction term in the Hamiltonian. By plotting the projection of the energy surface onto a 2-dimensional subspace, we will see that the energy surface is characterized by the state it represents and shows different features for different states.

2.3.1 The Transverse-field Ising Model

The Transverse-field Ising Model (TFIM) is one of the prototypical examples of physical models with a quantum phase transition in a discrete system. The Hamiltonian of the 1-dimensional TFIM is defined by

$$\hat{\mathcal{H}}_a = -J \sum_i \sigma_i^x \sigma_{i+1}^x - h \sum_i \sigma_i^z \tag{14}$$

where σ^x and σ^z are the Pauli matrices. The periodic boundary condition is applied so that $\sigma_{N+1} = \sigma_1$. It describes a 1-dimensional spin chain of length N with neighboring spin interaction and interaction with an external field. The first term describes the ferromagnetic interaction between spins if J is positive since the system prefers each pair of spins being parallel to each other. The second term is the interaction with an external magnetic field of strength h. Throughout this thesis, J will be set to 1 to specify the ferromagnetic case and h is the only parameter of the Hamiltonian.

The ground state energy and wave function of the Hamiltonian for the 1D-TFIM can be calculated exactly [14]. To get the exact ground state energy, one first rewrites the Hamiltonian with Jordan-Wigner operators, which satisfy the fermionic anti-commutation relations. The diagonalization of the Hamiltonian then follows by a Fourier transformation and a Bogoliubov transformation and takes the form of fermionic harmonic oscillators. The ground state ends up to have the following energy

$$E_g = \frac{1}{N} \sum_{p \in P_N} \sqrt{J^2 + h^2 - 2Jh\cos(p)} \tag{15}$$

where P_N is defined by

$$P_N = \begin{cases} \frac{2\pi}{N}\big(-\frac{N-1}{2}, -\frac{N-3}{2}, \cdots, \frac{N-1}{2}\big) \text{ if } N \text{ is even} \\ \frac{2\pi}{N}\big(-\frac{N}{2}, -\frac{N}{2}+1, \cdots, \frac{N-1}{2}-1\big) \text{ if } N \text{ is odd} \end{cases} \tag{16}$$

The difference for odd and even N arises from the Fourier transform aiming at achieving the desired sign of periodicity. This energy will be used to benchmark the accuracy of the RBM-represented ground state.

In the TFIM, the quantum phase transition occurs at $h_c = \pm|J|$ where the spin correlation length is at its maximum. One can define the magnetization as an order parameter in the spin system. For $|h| > 1$, the expectation of this order parameter will be 0. In this case, the system has a $\mathbb{Z}_2$ symmetry: the order parameter is invariant if all the spins are flipped. As one goes below $|h| = 1$, the magnetization will acquire a non-zero value. This breaking of the $\mathbb{Z}_2$ symmetry is called the spontaneous symmetry breaking.

In the following, we will use $|\uparrow\rangle$ and $|\downarrow\rangle$ to denote the eigenstates of σ^z with eigenvalues 1 and -1 and $|+\rangle$, $|-\rangle$ for eigenstates of σ^x respectively. In addition, if all the spins are in the same state, we will shorten the notation, e.g. $|\uparrow\rangle$ for $|\uparrow\uparrow\cdots\uparrow\rangle$

2.3.2 The dependence of network weights on the basis

Equation 14 is not the unique way to define the Hamiltonian as an objective function. Another physically equivalent definition of the Hamiltonian $\hat{\mathcal{H}}_a$ is

$$\hat{\mathcal{H}}_b = -J\sum_i \sigma_i^z\sigma_{i+1}^z - h\sum_i \sigma_i^x \tag{17}$$

which can be easily obtained through a basis transformation. Although those two Hamiltonians are physically equivalent, the energy surface of the RBM and the performance of the ground state search is different. Because the state is always represented in z basis by the RBM, the same quantum states on different bases have also different RBM-representations. Therefore, basis in which the state is given will also influence the performance of algorithm. Here we can see the difference between the wave function and the network representation. This difference is also suggested by the following two examples:

- For negative h, finding the ground state is much harder for $\hat{\mathcal{H}}_b$ than for $\hat{\mathcal{H}}_a$, because finding $|-\rangle$ is harder with the applied learning strategy, which will be discussed later in this work.

- If $h = 0$, where the ground state is degenerated, $\hat{\mathcal{H}}_a$ will select the $|+\rangle$ state because of our initial choice of weights while $\hat{\mathcal{H}}_b$ will end up in a superposition of $|\uparrow\rangle$ and $|\downarrow\rangle$. The spontaneous symmetry breaking and an equal probability of obtaining one of the pure states cannot be observed in the network. If there is degeneracy in the Hamiltonian, which state the network will choose merely depends on which representation is easier to reach.

Since the Hamiltonian is the only information we give to the training process, the resulting state will also only contain properties of this Hamiltonian, regardless of other physical properties. The network may also add its own bias if the difficulty of reaching the degenerated states is not equal.

2.3.3 Degeneracy in the Hamiltonian and network representation

In the last section, we see that our learning rule does not treat the degenerated state in an equal way. In the presence of degeneracy, the difficulty of reaching the state determines which state the RBM will choose. The state found by our learning rule does not fully reproduce the freedom in the degenerated states. Furthermore, even for small $|h|$ it is difficult to distinguish between the ground state and the second lowest state. The resulting energy is very close to the ground state energy, but the state it represents is usually still a mixture of the two states.

Apart from this degeneracy, there is also a translation invariance if the periodic boundary condition is applied: The shift of all the spins $s_i \to s_{i+k}$ should leave the state intact. This results in a large amount of redundancy in our representation. To reduce this redundancy, one can simplify the RBM structure in the following way[2]: First, we only allow the ratio between the number of hidden and visible neurons to be an integer $\alpha = M/N$. The hidden neurons can then be divided into α groups and the translation of spins is absorbed by the translation of hidden neurons. To guarantee the invariance, the weights w_{ij} are set to be equal if the distance between the visible and hidden neurons connected by them $d = |i-j| \in \{1, 2, \cdots, N\}$ are the same. The ansatz then simplifies to

$$\Psi(\mathbf{v}; \mathbf{\Omega}) = e^{a\sum_i v_i} \prod_g \prod_d \cosh\left(b_g + \sum_i v_i w_{g,i+d}\right) \tag{18}$$

where $w_{g,i+d} = w_{g,i+d-N}$ if $i+d > N$ denotes the weights in group g. This simplification is illustrated in Fig. 1 (d). It reduces the number of weights w from $N \cdot M$ to M, same as the number of hidden neurons and there are only α different weights b and only one weight a. However, notice that even if the degeneracy in the Hamiltonian is eliminated, the network representation is still not unique since the state remains the same by translation of weights within one group $w_{g,d} \to w_{g,d+1}$. This is the consequence of translation invariance of the periodic boundary condition.

In general, degeneracy exists at two levels, in the Hamiltonian and in the network weights representation. A lot of computation effort can be saved by simplifying the RBM structure according to the symmetry in the Hamiltonian and using an appropriate network representation of the wave function.

2.3.4 Construction of specific states

Once the objective function is specified by the Hamiltonian and an appropriate mapping between network weights and wave function is defined, finding the desired RBM representation of a quantum state is basically varying the weights so that in the end a balance between each cosh term in equation 3 is achieved, where each spin contributes to an appropriate amount of amplitude and phase determined by the real and imaginary part of the network weights. For a given state, there could be many different ways of achieving this balance in the RBM, which make the understanding of the RBM representation very difficult. In this section, we start with some simple examples and construct some special states to understand the RBM structure and how the state reacts to the change of network weights. However, one should keep in mind that the constructed weights are usually not the only way to achieve the desired states. In many cases, the constructed weights are not those found by the variational method because the simplicity in the constructed weights does not necessarily have advantage in the numerical procedure and some weights can be very sensitive to small perturbations. The construction should be viewed as an attempt to understand the network weights and does not always characterize the weights of the ground state found by the RBM.

- **$|\uparrow\rangle$ and $|\downarrow\rangle$**

  Among all states, $|\uparrow\rangle$ and $|\downarrow\rangle$ are the easiest ones to construct. This construction involves only the weight a. The weight a describes how much the amplitude and the phase will change if one spin is flipped independently of the network structure and other weights. The only thing that matters is the total number of up-spins or down-spins, i.e. the magnetization. This gives us a way of generating $|\uparrow\rangle$ and $|\downarrow\rangle$ states by simply amplifying this selection. If all the weights w and b are set to be zero, expression 3 becomes

  $$\Psi = e^{a\sum_i v_i} \tag{19}$$

  The ratio of $|\uparrow\rangle$ and the state with only one spin down is $e^{\sum_i 2a}/N$, which can be used to characterize how good is the $|\uparrow\rangle$ state approximated. The division of N comes from the fact that states with only one spin down have N configurations. The weight b has a similar function but works only within one group and its influence also depends on all other weights w in the same group.

If this construction was the weights that the variational method would pursue, it would have nothing to do with the number of hidden neurons, but achieving the exact representation would then require a to be infinite. For a large system, one should be careful if one of those bias weights starts to explode.

- **$|+\rangle$ and $|-\rangle$**

  $|+\rangle$ and $|-\rangle$ are quite an opposite in terms of network representation. The $|+\rangle$ state is easily achieved by setting all the weights to zero. This construction is also favored by the variational method since the weights are always initialized near 0 in our algorithm to maintain the stability of SR. It is observed that while the network approaches the $|+\rangle$ state, all the weights are getting smaller and smaller.

  To get the $|-\rangle$ state, we need an expression in which the flip of one spin gives us exactly a phase -1. Thus we need the help of w and b. Since all spin configurations have the same amplitude, the real part of all the weights can be set to 0. The cosh function then becomes a cos function. Like in weights a, the imaginary part of b and w also controls the relative phase, but in a much more complex manner. We can give $b_{g=0}$ an imaginary value of $\pi/2$ to transfer cos to sin and choose all weights w to be zero except one $w_{g=0,d=0} = wi$ with $w \in \mathbb{R}$ in group 0. Since the weights in all other groups are 0, each v only appears once and the expression 3 then becomes

$$\Psi = \prod_g \prod_d \cosh(b_g + v_d w_{g,0}) = \prod_d \cosh(\frac{\pi}{2}i + v_d wi) = (-1)^d \prod_d \sin(v_d w) \quad (20)$$

  This construction is usually not the one chosen by the variational method since $\pi/2$ is already a very large value. But we can see that to achieve $|-\rangle$ starting from $|+\rangle$ is not an easy task because of the alternating sign. This is actually the main obstacle while applying our method on the TFIM and will be discussed later in this thesis.

- **Two non-zero weights**

  If some of the weights w are very large, the other weights can be seen as zero. Only one very large real weight in each group is not enough to make a big difference because the expression will simply become

$$\Psi = \prod_g \prod_d \cosh(v_d w_{g,0}) \quad (21)$$

  which is still the $|+\rangle$ state only with a larger normalization constant.

If two weights in one group are very large compared to the others we can simplify the ansatz (with $a, b = 0$) to be

$$\Psi = \prod_g \prod_d \cosh(v_d w_{g,0} + v_{d+l} w_{g,l}) \tag{22}$$

If both weights are positive and the interaction term is in the z basis, they simulate the ferromagnetic interaction between two spins. We will see this more clearly if we generalize the interaction term of the TFIM to long-distance interaction:

$$\hat{\mathcal{H}} = -J \sum_i \sigma_i^x \sigma_{i+l}^x - h \sum_i \sigma_i^z \tag{23}$$

For example, if $l = 1$, the expression 22 selects states with parallel neighboring spins. which will be reflected in the real part of the weight distribution as a single peak. The result is then a superposed state of $|\uparrow\rangle$ and $|\downarrow\rangle$, provided that the two weights are large enough to suppress other states.

2.3.5 Weight distribution of the ground state

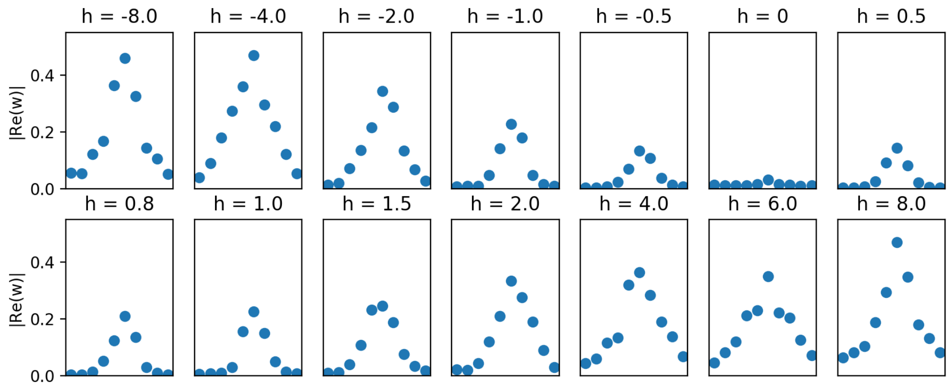


Figure 2: The distribution of the real part of the network weights for different h with Hamiltonian $\mathcal{H}_a$ of a 10-site system is depicted in the figure. The Hamiltonian has interaction between neighboring spins in the x basis. For each h, the average value of $|\text{Re}[w_{g,i}]|$ from 10 samplings is calculated. The weights within one group are shifted so that the largest one is always in the middle to compare the broadness, but the relative position of the weights remains the same. Because of the symmetry of the cosh function, the absolute value is taken to avoid cancellation. There is no significant change in the shape of the distribution despite the height.

In the previous section, we showed that the real part of two dominant weights resembles the interaction in z basis. An interesting question arising from this construction is, whether the network weights of the ground state of the TFIM show a similar correlation as in the spin system. This is, unfortunately, not true. The

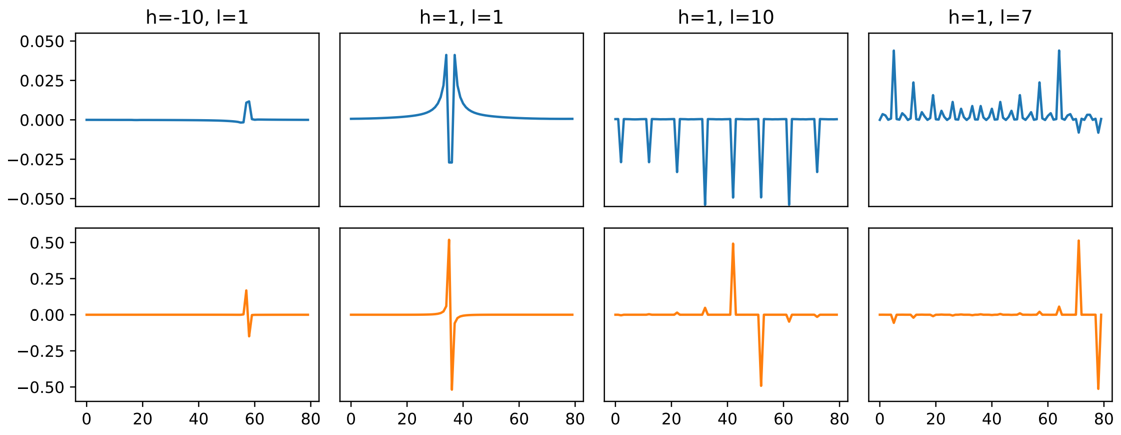


Figure 3: The distribution of the network weights of an 80-site spin chain is depicted for different h and interaction distance l. The number of visible neurons is equal to the number of hidden neurons ($\alpha = 1$). The blue and yellow lines are the real and imaginary part of the weight. The interaction occurs on z basis. For neighboring interaction, there is only one position where the weights deviate far from 0. For other l we can see periodic peaks selecting spin configurations with the desired interaction. If l is not a divisor of N, other small peaks arise to remedy this asymmetry. Since the ground state wave function of the TFIM is real, the imaginary part does not provide much information. The non-zero imaginary part simply comes from the real part during the SR optimization. They compensate for each other during the evaluation of the wave function and only leave a global phase in the end.

distribution of weights is characterized mainly by the interaction between the spins. To clarify this idea, we first recall that the RBM representation depends strongly on the basis. A basis transformation from z to x in the Hamiltonian will totally change the amplitude in the computational basis as shown in the previous construction. In the following, we will illustrate the distribution of weights in the TFIM with two examples concerning interaction terms $\sigma^x\sigma^x$ and $\sigma^z\sigma^z$.

We first look at the Hamiltonian with interaction term $\sigma^x\sigma^x$ in a small system. Fig 2 depicts the weight distribution of the ground state of $\mathcal{H}_a$ for different N, where a ferromagnetic interaction $\sigma^x\sigma^x$ exists between neighboring spins in a spin chain of length 10. If the weights we get from the variational methods (at least the real part) simulate the interaction, we should observe a very high and narrow peak for large field strength h and broadening of the weight distribution as the system approaches the critical point $h = 1$, where the correlation length increases. In Fig 2, although the distribution of $|\mathrm{Re}[w_{g,i}]|$ shows a certain pattern, the shape of the distribution is almost the same for all the ground states of the TFIM as h is varied across the phase transition points. There is no significant change in the broadness of the distribution but only the height. The absolute value of weights always has one peak within a group and the shape barely changes. The balance between the weights, in this case, is achieved not through the relative difference of the absolute value, but the relative sign.

Now we switch to the Hamiltonian with interaction term $\sigma^z\sigma^z$ and also generalize the Hamiltonian for different interaction lengths as defined in equation 23. Fig. 3 shows how the interaction distance influences the weight distribution of the ground state in a spin chain of length 80. For nearest neighbor interaction, the weights only deviate from 0 significantly around one position. An interaction term with distance l result in a periodic behavior of the weights. This kind of distribution can be observed as long as one stays away from the physical degeneracy at $h = 0$.

In a word, the shape of the weight distribution is characterized by the interaction between spins and the basis in which the interaction term is given. It does not, in general, resemble the spin correlation in the physical system.

2.3.6 2D weights diagram

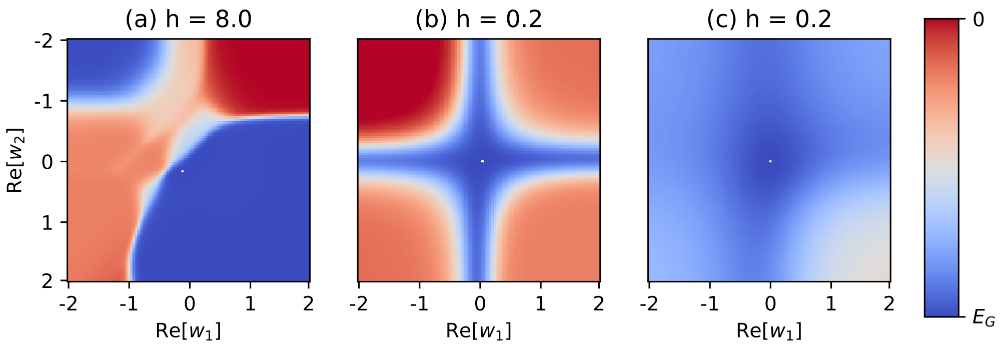


Figure 4: The energy distribution as a function of two randomly chosen weights in a small system with 10 spins and 20 hidden neurons. The white point denotes the minimum. In (a) and (b), the weights are chosen from the same group, while in (c) they are chosen from two different groups. (a) is the typical energy surface for ground states near the $|\uparrow\rangle$ and $|\downarrow\rangle$ states and (b) is a typical one for ground states near the $|+\rangle$ state. The energy difference in (c) is small because one dominant weight will give us $|+\rangle$ state, which is very close to the desired ground state. The figure illustrates several characteristic features which are typically found, like a large plateau and a narrow valley.

We have seen in the previous section that, although some quantum states can be represented with only a few weights, these representations are usually not what the variational method converges to. In this section, we turn to the numerical aspect and try to visualize the energy surface as a function of the network weights by plotting its projection onto a two-dimensional subspace, where only two degrees of freedom are left open. The most interesting areas of the energy surface are those around the ground state, thus we use the variational method to first find a good approximation of the ground state and then plot the energy dependence on two random weights around it. The Hamiltonian used here is $\hat{\mathcal{H}}_a$.

$$\hat{\mathcal{H}}_a = -\sum_i \sigma_i^x \sigma_{i+1}^x - h \sum_i \sigma_i^z \tag{24}$$

Fig. 4 shows three examples of those energy distribution maps, where the color shows the energy and the white point is the minimum. The distribution is characterized by the state it represents. Because of the translation invariance we discussed in section 2.3.3, the groups from which the weights are chosen will also affect the energy surface. Fig. 4 (a) is a typical energy distribution for states near $|\uparrow\rangle$ or $|\downarrow\rangle$. One can see that a large plateau of low energy appears with energy $E = -|h|$ (energy for $|\uparrow\rangle$ or $|\downarrow\rangle$ state) when both weights are very large and have the same sign. The shallow minimum lies in a small area close to the edge of the plateau. This is because two large weights in a same group with the same sign will select the state with aligned nearest neighbor spins. For small $|h|$, the state is selected to be near the $|+\rangle$ state. In Fig. 4 (b), the weights are chosen from the same group and the energy increases significantly if both weights are far away from the minimum. If two weights are chosen from different groups, varying them shows a relatively weak effect on the energy since one dominant weight in a group reproduces the $|+\rangle$ state, as shown in Fig. 4 (c). The balance between all the spin configurations are achieved mainly by the weights within one group, weights from different groups work independently.

3 Improvement of the optimization algorithm

In the previous section, the RBM structure and a learning algorithm are introduced for the ground state search problem and we have grasped the first insight of the complicated energy surface we are dealing with. For a set of network weights, the energy is calculated from 2^N wave function coefficients, each determined by a complex number calculated from a product of M cosh functions. With such a complicated objective function, one would expect the presence of local minima. On the other hand, the SR method we used for the optimization is an improved gradient-descent (GD) based algorithm which offers relatively fast convergence by moving towards the direction of the ground state. Although this method makes use of the first-order differential, it is short-sighted and easily gets itself trapped in local minima since the next step is only determined by the current position. If we have no previous knowledge of the ground state energy, it is difficult to find out whether the ground state found by SR is a good approximation or some local minima that lie far beyond it. For the TFIM, SR works well for positive h, but the problem of local minima is encountered while looking for ground states at large negative h which are close to $|-\rangle$.

Before we turn to the details, it is worth to spend some time to first clarify the meaning of local minima and global minima. Mathematically, the difference between those two is obvious. But working with a numerical approximation, especially with such an intricate objective function, it is neither possible nor necessary to dig out the real global minimum in the search space. Thus, in our case, the word global minimum would be used as a synonym of a minimum with high enough quality. For different precision requirements, the meaning of the global minimum may also vary.

In the first part of this section, we analyze the local minima that appear in the ground state search of the TFIM and illustrate them by plotting the 2D weight distribution. Furthermore, we show that the convergence behavior of SR is dependent on the hyper-parameters, which are parameters that control the learning algorithm. Repeating SR with different initial weights is not enough to solve the problem of the local minima because of this dependence. In the second part, a global optimization method called Differential Evolution (DE) is introduced to guide the optimization and locate the global minimum. As one always has to make a compromise between accuracy and efficiency, a hybrid algorithm is built and benchmarked with the Hamiltonian of the TFIM. The stability of the learning procedure is greatly improved and a representation of the ground state with good quality can be found with very high probability.

3.1 Misconvergence and local minima

In general, gradient-based optimization methods often have problems with saddle-points and local minima. As an improved method, SR gets rid of some of the drawbacks of the common GD. For example, since SR does not strictly follow the deepest gradient, it does not have problems with a narrow curved valley in the energy surface. Furthermore, the step size is also self-adaptive as the difference between projected and current state gets smaller and smaller as we get closer to the ground state. But it still cannot avoid the problem of local minima, which will be described in detail in this section.

3.1.1 Difficulty in finding $|-\rangle$

For the TFIM, the misconvergence happens most frequently while using $\hat{\mathcal{H}}_b$

$$\hat{\mathcal{H}}_b = -\sum_i \sigma_i^z \sigma_{i+1}^z - h \sum_i \sigma_i^x \tag{25}$$

with negative h. As h goes to negative infinity, the ground state approaches the $|-\rangle$ state, which requires a balanced amplitude for all spin configurations and an alternating sign.

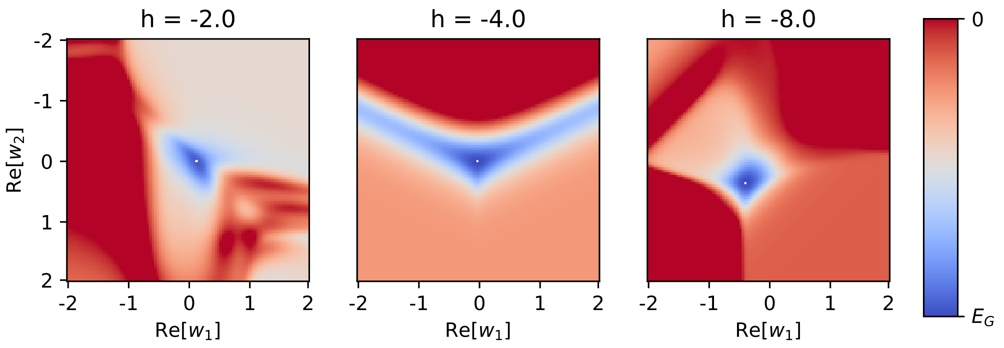


Figure 5: The energy surface of $\hat{\mathcal{H}}_b$ with negative h, for which the ground state is close to $|-\rangle$. The RBM has the same structure as in Fig. 4. The ground state energy is relatively sharp and occasionally surrounded by high barriers. If the initial weights lie far away in the high dimensional search space, it is quite likely that following the current gradient does not lead one to this minimum. It is worth to mention that although all these minima lie close to zero and seem to be well behaved around it, it is only a 2-dimensional subspace with all other weights in their optimal values. If all weights deviate from this value, the landscape can be much more complex.

Let's first have a look at the energy surface around the variational ground state for large negative h (approximately the $|-\rangle$ state). As shown in Fig. 5, the minimum is sharper than those shown in Fig. 4 and surrounded by a high plateau. Despite this difference, the minimum is still well behaved. However, as SR cannot climb over high energy barriers easily, its convergence performance relies more on the choice of a good starting point.

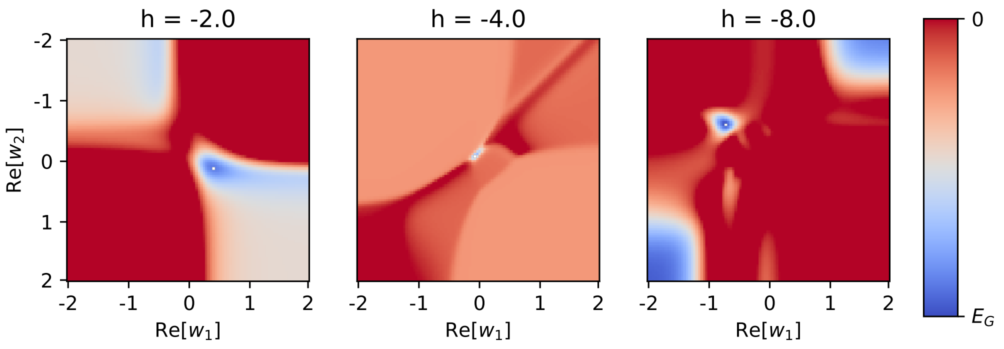


Figure 6: The energy distribution around local minima with the same RBM setup as in Fig. 5. The local minima are usually located among very complex landscape.

It is also interesting to have a look at how the energy surface looks like around the local minima. Those local minima the RBM encounters are usually even sharper and narrower than the global one (Fig. 6). This is also validated by several other evidence, e.g. convergence into a local minimum (starting to fluctuate and stagnate instead of decreasing) usually takes only several hundred steps, while converging into a high-quality minimum requires ten times of that or even more steps. It is not clear how those sharp local minima arise from an analytical function. The multiplication of many cosh functions may result in some network weights being very sensitive at a certain point and forming the local minima. From this point of view, the number of local minima could increase as we go to larger systems.

With those sharp traps in our picture, it is not difficult to gain some insight into why it is so hard to find a good set of network weights representing the $|-\rangle$ state with our algorithm. For the stability of SR, one usually starts from a set of initial weights close to 0, the initial state is then very close to $|+\rangle$ and the RBM has to wander a long way down to find the $|-\rangle$ state. Numerical evidence shows that starting randomly from those initial weights, the optimization could fall into many different local minima. Optimization with SR is like walking downhill from the peak ($|+\rangle$) with many local traps all along the way. If it can only see a small region around itself, the longer it has to go, the higher is the probability that it will fall into a local minimum.

In general, the difficulty of finding a state $|f\rangle$ from an initial state $|i\rangle$ is characterized by the distance and the landscape of the energy surface between them. Similar difficulties are observed when one starts from $|-\rangle$ as described in equation 20 and searching for the $|+\rangle$ state. This distance is not solely determined by the energy difference, but the distance in the search space. It can also happen that although they are far from each other, the energy surface between them is smooth and only a few traps exist. Example for the latter one is that this difficulty is only observed between $|+\rangle$ and $|-\rangle$ but not between $|\uparrow\rangle$ and $|\downarrow\rangle$.

3.1.2 Dilemma in the choice of learning rate and initial range

As described in section 2.2, two hyper-parameters, the learning rate γ and the range of initial weights, can be adjusted to control the convergence behavior. In practice, however, the adjustment of those hyper-parameters faces a dilemma. It is difficult to keep the algorithm stable and avoid local minima at the same time. Finding appropriate values of these hyper-parameters may need a lot of test runs.

For the stability of SR, the learning rate γ should not be too large. Although the RBM may take longer to reach the minimum, starting with a relatively small γ seems to be a safe choice if we don't know what kind of landscape we landed in. However, since many local minima exist in the form of sharp holes on the energy surface, a small learning rate, i.e. a small step size, makes it easily fall into the local minimum, as can be seen in Fig. 7, where the minimum energy and corresponding relative energy difference found by SR is plotted for two different

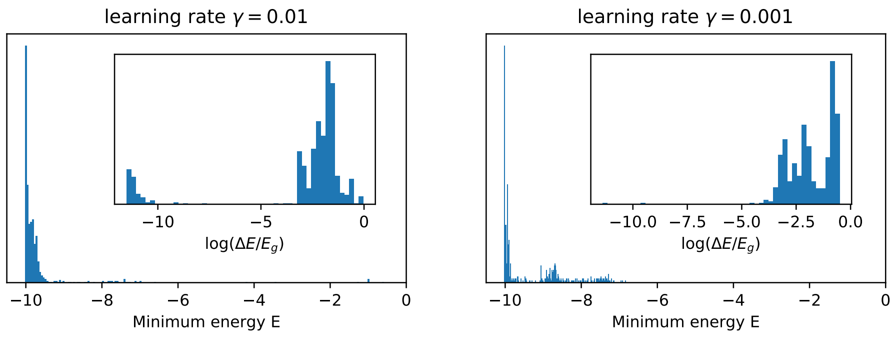


Figure 7: The histogram of the ground state energy found by the RBM with $h = -10$, two learning rates and initial range 0.0001. The RBM has 10 visible and 20 hidden neurons. The exact ground state energy is at about -10.025. For each learning rate, several hundred minima found by SR are recorded and both the minimal energy and the relative difference is shown in the figure. For a large learning rate, a bunch of very accurate minima can be found, which disappear after the learning rate is reduced to 0.001.

learning rates. After reducing the learning rate, the few minima found with high accuracy disappear.

The adjustment of initial weights faces the same situation. For an unknown problem, the safest way is to sample with random initial weights, but in which range should the weights be generated? On the one hand, a too small range could make the sampling meaningless. On the other hand, large initial weights will reduce the stability of SR and the probability of finding a good representation with SR also drops as the search range grows rapidly with the initial range (Fig. 8). The high-quality minima are clearly not spread uniformly around the search space.

In some situations, a good starting range and learning rate may exist, but they are not only dependent on each other but also dependent on the size of the system and the Hamiltonian. A wrong choice of those hyper-parameters in SR will not only reduce the stability but also affect the performance of the ground state search. As SR itself hardly offers any information about the quality of the minimum, we need to turn to other methods to improve the stability and quality of the found minima.

3.2 Differential evolution

In supervised neural network machine learning, like feed-forward networks, the problem of local minima does not attract much attention as long as the network is confident in its prediction. It has also been shown that, under certain assumptions, most local minima in large-size networks have high quality and yield similar performance[15], which is very similar to what we get in Fig. 6. However, for our use, the local minima will directly limit our accuracy. The simplest way of tackling this problem is trying to repeat SR with different initial values. But for a larger

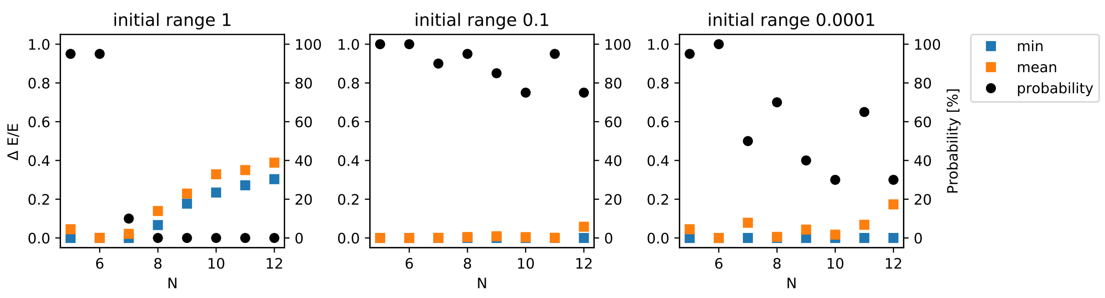


Figure 8: The convergence performance of SR as a function of N for different initial range, where $\hat{\mathcal{H}}_b$ is used at $h = -10$. The blue and orange points denote the minimum and average ground state energy by repeating SR optimization 20 times with random initial weights and the black points show the probability of convergence with good quality ($\Delta E/E < 10^{-4}$). Since for large initial range, the matrix inverse fails frequently at the beginning, a preselection is used so that the weights are ditched if SR fails within the first hundreds of steps. As can be seen in the figures, different choices of initial range lead to different convergence behaviors. The convergence also shows a trend of getting worse as the system size N increases.

system or a more complex Hamiltonian the probability of SR converging to a minimum with good quality might decrease and one always has to tune the learning rate and initial weights to adapt it to the current problem. To mitigate these problems, we introduce a global optimization algorithm to guide the system towards a high-quality minimum. In this section, a global search algorithm called Differential Evolution (DE) is introduced to improve the convergence performance.

Compared to a gradient-based algorithm, DE has several advantages: First, it is much more robust than SR, since no matrix inverse is needed in the evolution and it is much less likely falling into a local minimum. The reason will be discussed later in this section. Second, DE is a population-based search algorithm and, at each step, the update of each trial vector is independent and can be implemented parallelly. The free choice of the population size also offers the possibility to balance between efficiency and accuracy. In consideration of the efficiency, the goal of applying DE is not to find the true global minimum but to filter out the low-quality minima. As one always has to trade the accuracy for efficiency, DE offers the flexibility to achieve this balance.

3.2.1 Evolutionary algorithm

The term evolutionary algorithm (EA) refers to a class of population-based stochastic search algorithms inspired by the mechanism of natural evolution. It resembles the natural evolution by simulating the process of reproduction and selection. EA is particularly suitable for solving complex optimization problems with many local optima and can deal with problems with no gradient information. These features greatly increase the robustness and the scope of applications of EA. The most well known EA for optimization problems includes the Genetic Algorithm (GA), the Evolutionary Strategy (ES) and the Differential Evolution (DE)[16].

For more than two decades, EA has been applied to training neural networks at three different levels: network weights, architectures, and learning rules [17]. Since the structure of RBMs is already quite clear and accompanied by a well-defined learning rule, the application of EA in our case focuses on optimizing the network weights.

The first step in building any EA is always to find an appropriate genetic representation of the problem. For network weights optimization, the representation is simply a vector of real numbers with the length twice of the number of the weights. Since it is an optimization in continuous space, DE is a suitable choice.

3.2.2 Basic concept and implementation of DE

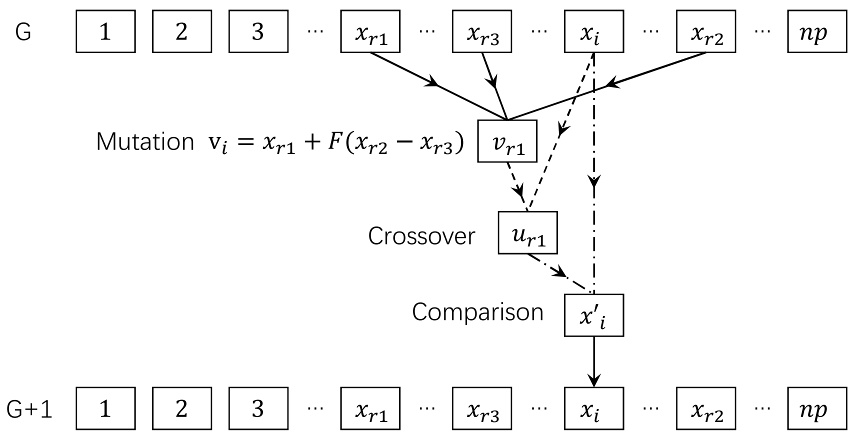


Figure 9: Illustration of the Differential Evolution strategy.

DE was first introduced by Price and Storn in 1995[18][19]. In the language of evolutionary algorithms, one potential solution to the problem is called one individual. The individual in DE is represented by a vector of real numbers. The length of this vector is the dimension of the search space. Like other search algorithms in the family of EA, DE explores the search space to find the individual minimizing the objective function. DE is characterized by its unique way of generating new individuals: It perturbs the target vector with a scaled difference between two randomly selected vectors. Not like ES, where the new generation is created around an existing individual with a predefined distribution, the step size of DE is self-adaptive. The search algorithm of DE has three steps: mutation, crossover and selection. In the beginning, the system generates an initial population consisting of *np* vectors within the search range. In each iteration, a trial vector will be generated by the mutation and crossover operation and is compared with a current vector in the population. The vector with a lower value in the objective function, i.e a lower energy, will survive to the next generation so that the number of vectors in the population remains constant. The detailed implementation of the original DE algorithm is described as follows[19]:

- Initialization

  To establish a starting point, an initial population P of np vectors x_i is generated randomly within the initial range. For low dimensional searches, a population of np is suggested to be 5 to 10 times the dimension of the search space D[19]. For high dimensions, it can be reduced to save computation resources.

  In classical DE, the initial population should cover the search area. For optimizing neural networks, however, the weights don't have a well-defined boundary. If we naively generate random numbers within a real boundary [-R, R], most of the vectors will be created far from 0 where a good solution rarely appears and the convergence will be extremely slow. In high dimensions, DE may have to stagnate for a long time until one of the offspring occasionally hits a lower area to lead the convergence. Instead, one can generate the initial population near the origin and set a relatively large boundary of the search space. In this way, we allow some weights to be large while reducing the exploration of the unnecessary area. The population will first expand and spread out until a rough area of a minimum is located (usually not far from 0) and then start to converge. The only thing one should notice is that enough diversity of the population should be guaranteed by the mutation and crossover operation to avoid premature convergence, where DE converges to a local minimum before it spreads out.

  In practice, the initial range of about 0.1 is sufficient and the boundary of vector components can be set to 3 or π, depending on whether it comes from the real or the imaginary part of the network weights. These small initial weights speed up the search for the TFIM, but whether DE can find the minimum is not sensitive to it (as long as 0 itself is not a strong local minimum). The population will spread out and then converge to the area with low energy. How these values need to be adapted for other models remains to be analyzed further.

- Mutation

  For each target vector x_i, where i goes from 1 to np, a mutated vector v_i is constructed by

$$v_i = x_{r_1} + F(x_{r_2} - x_{r_3})$$

where r_1, r_2, r_3 are random numbers from 1 to np and $r_1 \neq r_2 \neq r_3 \neq i$ and F is called the mutation scale factor. It is suggested empirically that F should lie between 0 and 1[19]. F is one of the most important factors to guarantee the diversity of the population. Zaharie has demonstrated that a scale factor too low could cause premature convergence while the selection pressure is still absent[20]. Randomizing the scale factor, on the opposite, will be useful in high dimensional search with a relatively small population. Since for small population, the combinations between present individuals are limited, random F can increase the diversity of the population and prevent stagnation, when no better solution can be found by combining existing vectors in DE [19].

In practice, it is observed, while applying DE to RBMs with 10 visible and 20 hidden neurons, that F=0.2 could cause premature convergence. A uniformly distributed F between 0 to 1 or a Gauss distribution will help DE to detect minima close to 0 and hence shorten the convergence time dramatically while still leave DE enough possibility of exploiting relatively far away areas. The possibility of generating small F can be regarded as enhancing local exploitation ability.

Another variant of DE mutation is to replace x_r by x_{best} where the latter one denotes the best individual in the population. This choice will speed up the convergence, but it should not be used if the population or initial range is small, since searching only around the best individual is too greedy and may lead to convergence into a local minimum.

- Crossover

  After a mutated vector v_i is generated, it is mixed with the vector in the current population x_i to yield a trial vector u_i according to

$$u_{ij} = \begin{cases} v_{ij}, & \text{rand}(j) \leq Cr \text{ or } j = \text{rand}\, np \\ x_{ij}, & \text{else} \end{cases}$$

  where $\text{rand}(j) \in [0, 1]$ is a uniform random number for the j-th component. The random integer $\text{rand}(np) \in \{1, 2, \cdots, np\}$ choses a random element from the trial vector and forces it to take the value from the mutated vector to prevent u_i from being identical to x_i. Cr is the crossover probability and describes how many parameters are inherited from the mutated vector. For decomposable objective functions, Cr can be chosen around 0.2 so that each time only one or two elements are altered[18]. In case of dependent parameters, like in the neural network, the most effective choice of Cr proves to be about 0.9, which will provide extra diversity to the pool of possible solutions[18]. However, $Cr = 1$ should also be avoided since it may result in an extremely slow convergence on some occasions[19].

- Selection

  The classical DE, not like GA, sets no selection pressure while choosing parents. All vectors in the current population have equal probability to contribute to the next generation. Therefore, a survival selection must be adopted to push the evolution towards better solutions. DE adopts a tournament selection, which can be described by

$$x_i^{G+1} = \begin{cases} u_i^G, \ f(u_i^G) < f(x_i^G) \\ x_i^G, \ f(u_i^G) \geq (x_i^G) \end{cases}$$

  with the number of generation G and f the objective function. Under this survival selection scheme, the best individual will always remain in the next generation. The population is evolving towards better solutions while the randomness in the choice of pairs to be compared (consequence from the random number r_1 in the mutation stage) prevents the population from converging too greedily towards a local minimum.

The general search process of DE can be described, intuitively, in three steps: First, all individuals are generated around 0 and spread out in several dozen generations until a large area of low energy is covered. Second, DE starts to search in this area by reproduction according to the rules introduced above. The population tends to wander towards the low-energy region and if more than one regions are detected, competition takes place. Third, all the individuals converge into one minimum and DE starts to refine the accuracy.

3.2.3 Advantages and disadvantages of DE

Compared to many other global search algorithms, the implementation of DE is very straightforward and there are only three hyper-parameters, np, F, and Cr. As a general global optimization method, DE outstands many other methods in several aspects. For our purpose, the most important of them are robustness and flexibility. These advantages also come with some weakness, e.g. slow convergence and weak local exploitation.

In terms of optimizing neural networks, the most important advantage of DE is the robustness. An abrupt change on the energy surface does not affect the stability of the algorithm and it is also much less affected by the presence of sharp, deep local minima. As the mutated individual is generated by adding the scaled difference between two other random individuals, the probability of accidentally falling into a sharp local minimum is relatively low. Even if one vector occasionally hit a local minimum, as long as the population is not yet converged, the competition between detected minima can come to the rescue of the lost individuals.

The flexibility of DE lies in the fact that we are able to choose a balance between efficiency and accuracy. We expect DE to offer a better result compared to SR based on the fact that if more than one minimum is found, the competition will decide which one shall survive. It is also clear that the more area DE can cover and the more time it has, the better the result. This balance is maintained mainly by the number of individuals *np*. The convergence of a large population requires more time and they will also find more candidates for global minima. With a sparse population, DE will help us filter out local minima with low quality. Since the energy difference is large, this can be done fairly efficiently. If an intensive search is required, a dense population will explore the search area more thoroughly and a minimum with high resolution can be achieved. The latter, of course, takes much more computing resources since a lot of effort will be spent on searching the area between minima before the convergence in one minimum is good enough to outperform the others.

Furthermore, as a population-based search algorithm, DE can be used to analyze the geometry of the energy surface. The broadness of a minimum can be estimated by comparing the convergence of minimal energy and the average distance among the individuals d defined by

$$d = \frac{\text{distance of all pairs}}{\text{number of pairs}} = \frac{\sum_{i<j}^{np} \|v_i - v_j\|}{\frac{(np-1)np}{2}} \tag{26}$$

where $\mathbf{v_i}$ and $\mathbf{v_j}$ are two vectors from the current population. If there are two minima competing with each other, two rough peaks should be observed if the distances between all the pairs are plotted in a histogram. If the population converges, this distance, together with the standard deviation of the energy should become very small. This offers a way to monitor the convergence status.

Although DE has many advantages and is considered to be one of the most powerful stochastic real-parameter optimization methods, it also has some shortcomings. The most significant drawback of DE is a common one for any global optimization method: slow convergence. Since in each iteration, the trial vectors are constructed from all members in the current population, the local exploitation ability of DE is regarded weak[21]. The convergence is slow even if the minimum is already well located and the system is exploring the nearby region to refine the result. Furthermore, the advantage that DE can deal with objective functions without gradient information is not an advantage in our case since the gradient information is available. In the review of Das and Suganthan[21], many variants of DE have been discussed and some of them can be helpful in terms of optimizing network weights. In the next section, we are going to adapt DE to our special case of optimizing RBMs and to build a hybrid method from DE and SR.

3.2.4 Hybrid optimization with DE and SR

The ability to optimize the system without the gradient information is one of the advantages of DE. In our case, however, it becomes a drawback compared to SR, which makes use of the gradient in an implicit way. In many occasions, a combination of those two methods will bring us a lot of benefits. This combination can be very flexible because an SR process can always be initialized with one of the current individuals during DE. In this section, we mention some of these combinations useful in different situations.

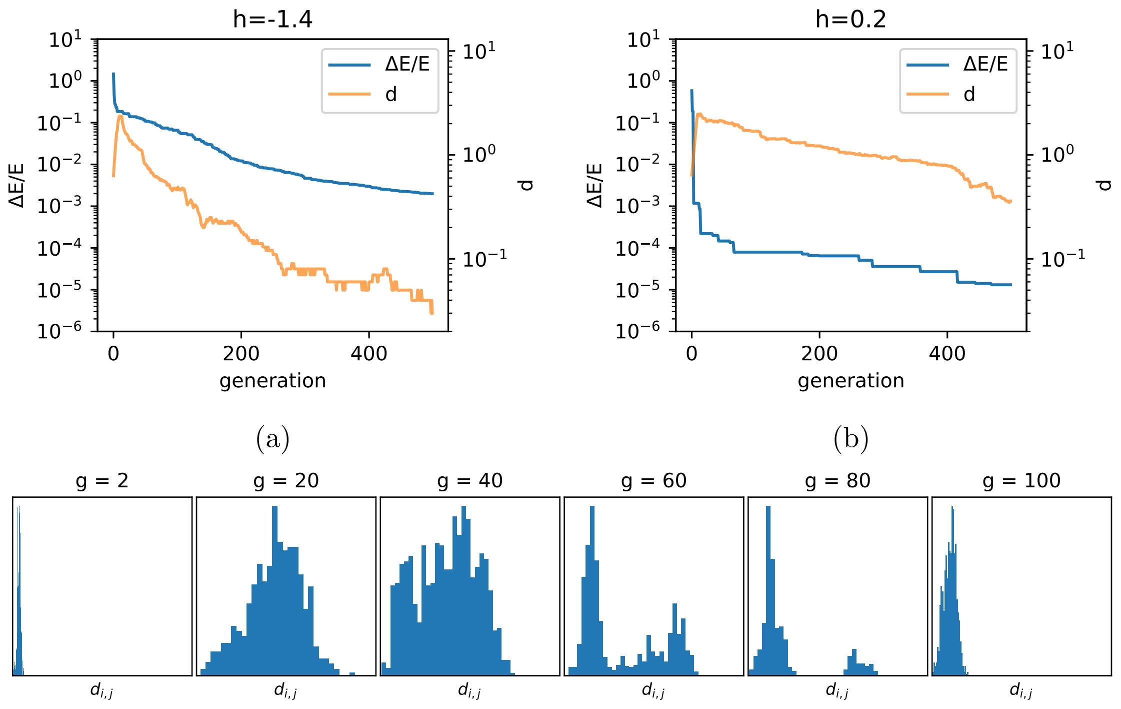
Usually, the convergence of DE is slow especially in the last step, where the minimum is located and only the refinement after several digits is needed. In this case, it is clear that we no longer need DE and SR can take over. In this sense, SR can be used to improve the accuracy of the minimum with a small step size and large sampling number in the gradient determination.

Apart from that, SR can also be used during the DE process. The convergence of DE to a minimum in a plateau can be difficult since locating the minimum is based on pure luck. This could cause stagnation of DE. Therefore, if the best individual exists for a long time without being replaced, an SR with several iterations can be applied to lead the direction of the convergence.

A more advanced improvement can be applied in the case that high accuracy needs to be guaranteed. As mentioned before, if a few minima with good quality are already well located and competing with each other, a lot of computing effort will be wasted searching the area between them. If those clusters can be identified, one can adjust the scale factor F according to which cluster the source vectors come from so that DE only searches within those clusters or jumps between them. Several SR-branches initialized in each minimum will also be very efficient. These are not yet implemented in the current algorithm.

3.2.5 Performance of the algorithm

In this section, the performance of DE and the hybrid method is benchmarked by the task of searching the ground state of the TFIM. To get the accuracy of 10^{-2}, i.e. filtering the low-quality minima, the computing effort required for DE is usually comparable to that of 5 to 20 SR runs. The increase of the computing resources against the system size N for SR and DE are the same if the population of DE

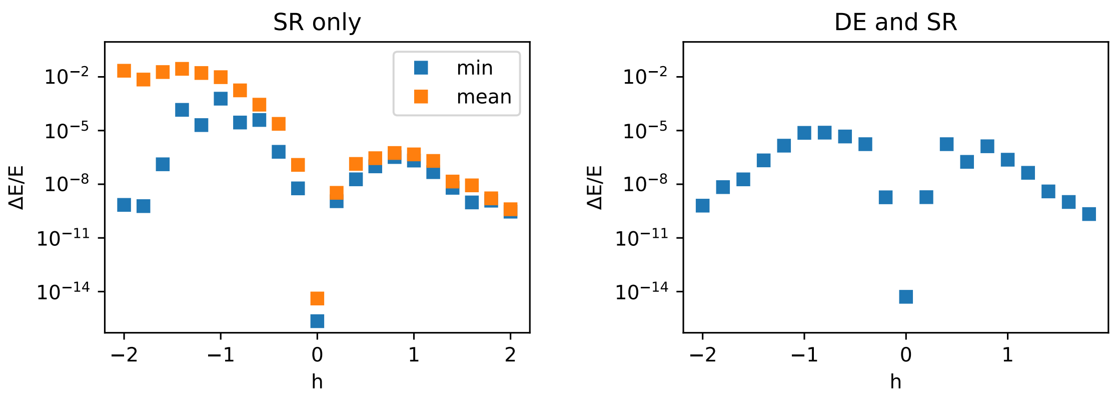


(c)

Figure 10: The convergence process of DE for a spin chain of length 10 is shown in the plots with $D = np$, $Cr = 0.9$. The RBM has 20 hidden neurons. In (a) and (b), the lowest energy of the current population is recorded at each generation and converted into the difference relative to the exact ground state. $d_{i,j}$ is the average distance between all pairs of individuals defined by equation 26. In (a), the accuracy of energy is limited but the minimum is narrow so that the average distance shrinks rapidly. (b) is just the opposite, the minimum found by DE has high quality but the convergence is slow. It is because the minimum is very shallow or several different minima with similar quality are located at the same time. (c) shows the competition between minima. There are six histograms of the distance between all pairs of individuals and g denotes the number of generations computed. In the beginning, all the individuals are generated around zero. The population spreads out and detects more than one minimum. During the convergence, they form two groups, showing as two peaks in the plot, one for the distance within the group and one for the distance between them. In the end, individuals in the low-quality peak disappear and the algorithm converges.

is kept constant. So far, in the 1D-TFIM model, we have not find a convincing situation where the efficiency of the hybrid method can outperform that of SR. Since we set the population of DE to be equal to the dimension of the search space, the hybrid method will be more efficient only if the probability of getting a good convergence of SR drops faster than $1/N$. The dependence of the performance of SR on hyper-parameters makes a systematical analysis of its convergence behavior more complicated.

For small systems with 10 visible and 20 hidden neurons, the convergence process of DE is illustrated in Fig. 10 for 500 generations. In Fig 10a and 10b, The



(a) (b)

Figure 11: The convergence performance for SR and hybrid DE. On the left panel, SR is repeated 20 times for each h and the average and minimal energy found by the RBM is depicted. On the right panel, the hybrid method is used for the same RBM structure. In many cases, SR will eventually find a good representation of the ground state, but most of them are trapped in a local minimum. In this number of population ($D = np$) DE cannot guarantee that it will always find a better representation than SR. Especially for $|h| \leq 0.5$, where the ground state is close to the $|\uparrow\rangle$ and $|\downarrow\rangle$ states and the shallow minimum is surrounded by a plateau.

relative energy difference $\Delta E/E$ and the average distance between each pair of the individuals d are plotted against the generation number. One can see the difference between a shallow minimum and a sharp minimum from the convergence behavior of average distance and energy. The competition between the two minima is also observed and shown in Fig. 10c.

The convergence performance of SR and the hybrid method of DE with SR is depicted in Fig. 11. For the case with only SR method, it is repeated 20 times and the minimal and the average ground state energy is recorded. In many cases, SR can eventually find a good representation of the ground state, but most of them are trapped in a local minimum. As for the hybrid method, DE is followed by SR initialized with the first 3 individuals with the lowest energy from DE. The convergence of DE also shows the symmetry of the accuracy in representing $|+\rangle$ and $|-\rangle$ for positive and negative h. This is also an evidence that the misconvergence is mainly due to the distance between the initial and the final state, not because of the difficulty in representing the state by the RBM.

Since for small systems, an exact diagonalization is still feasible, one can also calculate the fidelity given by $f = |\langle\Psi|\Psi_G\rangle|^2$, where $|\psi_G\rangle$ is the exact ground state. As the Hamiltonian has degenerated ground states at h and the closing energy gap between the ground state and the second lowest state for $h \to 0$ will significantly influence the performance of the search, we add a field strength in z direction to lift the degeneracy. The Hamiltonian then reads

$$\hat{\mathcal{H}}_{b'} = -\sum_i \sigma_i^z \sigma_{i+1}^z - h \sum_i \sigma_i^x - B \sum_i \sigma_i^z \tag{27}$$

As the fidelity is very close to 1, we show in Fig. 12 the infidelity in logarithmic scale.

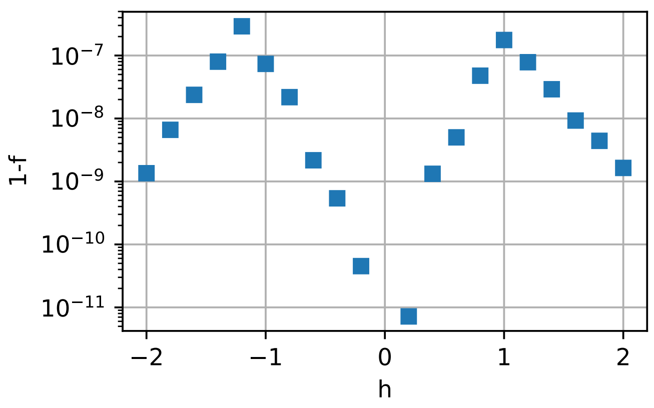


Figure 12: The infidelity of the ground states found by the RBM with hybrid DE and SR for the spin chain of 10. The Hamiltonian is modified with a field $B = 0.1$ in z direction to lift the degeneracy. The accuracy is comparatively worse for $|h| = 1$ where the wave function is more complicated because the interaction term and the field term are equally important.

4 Summary

The success of using neural networks to represent complex quantum many-body states has promoted the applications of neural networks in quantum physics. However, the understanding of how the network works and what else one can learn from its architecture is hindered by the complexity and non-linearity of the ansatz. In this thesis, we presented some relationships between the RBM weights and the quantum many-body states it represents. As a representation of the ground state, the RBM does not always keep all the properties of the state, but only those contained in the Hamiltonian and favors the state that is easy to represent in presence of degeneracy. We showed that the representation is strongly dependent on the choice of basis and the weight distribution is mainly characterized by the interaction term in the Hamiltonian. The performance of the learning algorithm depends eventually on the energy surface in the continuous weight space rather than the discrete spin configuration.

In addition to these attempts to understand the neural network, we also improved the optimization method by adopting a global search algorithm DE. The application of DE is not restricted to the problem of finding the many-body ground state but is also suitable for other optimization tasks of neural networks. For a problem where the ground state is not known a priori, one can first make an estimation of the difference between the initial state and the target state by, e.g. the energy difference, and decide if a hybrid method with DE is necessary. Compared to SR, DE requires, in general, more computation resources, but it can deal with complicated landscape and bypass low-quality local minima. With high stability and robustness, its performance relies more on the efficiency and accuracy of the energy estimation. Apart from the optimization, DE, as a population-based algorithm, also offers a way to analyze the geometry of the energy surface if a large population is available to explore the search area. This method could be used for the further analysis of the neural network. Still, the performance of the RBM representation and the learning rule need more numerical investigation since the model we used so far for the benchmarking, the 1D-TFIM, is still relatively simple and also mainly restricted to small system size. Whether the efficiency of the hybrid method can really outperform SR in a larger system or a more complicated Hamiltonian also needs further analysis.

References

[1] D. Poulin, A. Qarry, R. Somma, and F. Verstraete, “Quantum simulation of time-dependent hamiltonians and the convenient illusion of hilbert space,” *Physical review letters*, vol. 106, no. 17, p. 170501, 2011.

[2] G. Carleo and M. Troyer, “Solving the quantum many-body problem with artificial neural networks,” *Science*, vol. 355, no. 6325, pp. 602–606, 2017.

[3] S. Czischek, M. Gärttner, and T. Gasenzer, “Quenches near ising quantum criticality as a challenge for artificial neural networks,” *arXiv preprint arXiv:1803.08321*, 2018.

[4] 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. 085104, 2018.

[5] D.-L. Deng, X. Li, and S. D. Sarma, “Machine learning topological states,” *Physical Review B*, vol. 96, no. 19, p. 195145, 2017.

[6] X. Gao and L.-M. Duan, “Efficient representation of quantum many-body states with deep neural networks,” *Nature communications*, vol. 8, no. 1, p. 662, 2017.

[7] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac, “Neural-network quantum states, string-bond states, and chiral topological states,” *Physical Review X*, vol. 8, no. 1, p. 011006, 2018.

[8] D.-L. Deng, X. Li, and S. D. Sarma, “Quantum entanglement in neural network states,” *Physical Review X*, vol. 7, no. 2, p. 021021, 2017.

[9] N. Freitas, G. Morigi, and V. Dunjko, “Neural network operations and susuki-trotter evolution of neural network states,” *arXiv preprint arXiv:1803.02118*, 2018.

[10] G. Carleo, Y. Nomura, and M. Imada, “Constructing exact representations of quantum many-body systems with deep neural networks,” *arXiv preprint arXiv:1802.09558*, 2018.

[11] N. Le Roux and Y. Bengio, “Representational power of restricted boltzmann machines and deep belief networks,” *Neural computation*, vol. 20, no. 6, pp. 1631–1649, 2008.

[12] S. Sorella and L. Capriotti, “Green function monte carlo with stochastic reconfiguration: An effective remedy for the sign problem,” *Physical Review B*, vol. 61, no. 4, p. 2599, 2000.

[13] S. Sorella, “Generalized lanczos algorithm for variational quantum monte carlo,” *Physical Review B*, vol. 64, no. 2, p. 024512, 2001.

[14] H. Cakir, “Dynamics of the Transverse Field Ising Chain after a Sudden Quench,” 2015. (Master thesis in Heidelberg University).

[15] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, “The loss surfaces of multilayer networks,” in *Artificial Intelligence and Statistics*, pp. 192–204, 2015.

[16] A. E. Eiben, J. E. Smith, *et al.*, *Introduction to evolutionary computing*, vol. 53. Springer, 2003.

[17] X. Yao, “Evolving artificial neural networks,” *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[18] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[19] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[20] D. Zaharie, “Critical values for the control parameters of differential evolution algorithms,” in *Proceedings of MENDEL*, vol. 2, p. 6267, 2002.

[21] S. Das and P. N. Suganthan, “Differential evolution: a survey of the state-of-the-art,” *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2011.

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 16.07.2018