Comp 576: Deep Learning
Final Project: Learning Quantum States

Shah Saad Alam and Seth Davis

December 14, 2018

Abstract

One of the major computational goals of current condensed matter physics theory is to determine the lower energy eigenvalue and eigenstate of a given physical system. Traditional methods rely on either optimizing the diagonalization of a large matrix, or variations of quantum Monte Carlo. Neural network quantum states provide a novel new way of finding the ground state and energy of certain physical systems. In our project, we implement our own code for a Restricted Boltzmann Machine neural network and demonstrate that it can solve the $(d = 1, 2, 3)$ Ising models, and 1D Heisenberg model. We further develop a method to find higher eigenstates, and discuss improvements on this research direction beyond this project.

Contents

1 Introduction

1.1 Quantum mechanics and machine learning

Quantum mechanics provides the theoretical framework for understanding physics over the smallest accessible distances, explaining the behavior of the universe from the breakdown of classical mechanics at about 10^{-8}m down to the breakdown of general-relativistic gravity at the Plank scale, 10^{-42}m. Since its storied inception just over a century ago, quantum mechanics has provided tools enabling revolutionary progress in chemistry, materials science, astronomy, metrology, and medicine. Many more technological advancements await if we are able to understand the behavior of large, complicated quantum systems - a notoriously hard problem. One of the many surprising revelations of quantum mechanics is that the universe is exponentially more complex than previously appreciated by classical physics. For a simple system of N particles with M possible quantum states for each particle, it takes a vector space of M^N basis states. Therefore, even a simple quantum simulation of the dynamics of a US penny on a classical computer would require more bits than there are particles in the known universe.

The field of many-body physics seeks to understand the qualitatively meaningful aspects of huge systems of interacting quantum-mechanical particles. It turns out that there is a massive zoology of emergent quantum phases giving rise to materials with amazing properties. These range from superconductors and exotic quantum magnets to materials with "topologically protected" spin-polarized surface currents, []. Understanding these phases from a fundamental perspective is a major contemporary research goal, as it may allow us to better engineer novel materials for applications in technology.

Extracting the key physical details from an exponentially complex quantum system requires computational algorithms capable of efficiently approximating that complexity. As such, a major goal of applied quantum mechanics over the last half-century has been the development of efficient numerical and computational schemes to approximate this physics. The most successful of these methods, quantum Monte Carlo (QMC) and density-matrix renormalization group (DMRG), have achieved continually impressive results in some areas, while falling short in others[1, 2].

More recently in computer science, breakthroughs in deep neural network machine learning have enabled a host of new achievements in artificial intelligence. Deep convolutional neural networks have achieved new records in tasks such as image classification while recurrent neural networks have caused paradigm shifts in language processing and text/image generative tasks. Other examples include Google's AI engines for games such as AlphaGo and chess.

An exciting idea on the frontier of computational quantum mechanics is the use of deep neural networks as a tool in many-body physics. As recently as 2017, an extremely basic proof-of-concept neural net for learning quantum states was introduced and was able to beat top benchmarks for quantum mechanics algorithms, [3, 4, 5]. Related ideas have since been implemented that have been able to recognize distinct types quantum phases and accomplish other notable tasks [5, 6, 7, 8, 9, 10, 11]. It is an exciting time because these ideas are only just beginning to have an impact on the field.

1.2 Project scope, goals, and summary

Our goal was to use this project as an opportunity to explore these subjects with an eye towards future research applications. To make that possible, our primary goal was to gain a working familiarity with stochastic Monte-Carlo training methods for neural networks, which are significantly different than standard neural network training algorithms.

To this end, the primary thrust of our project was to build from scratch our own original python library implementing a general restricted Boltzmann machine architecture with a Gibbs-sampling-powered quantum Monte Carlo training routine. We engineered our model in a modular style so that it could be trained on an arbitrary Hamiltonian (for a quantum spin model, a paradigm of physical model system that we chose to specialize on.) In this project specifically, we test models for the Ising model (in 1,2 and 3 dimensions) and the Heisenberg model (in 1 dimension). For both models, we slightly generalize what has been reported previously in the literature by allowing for a fully general quantum interaction with an external magnetic field.

Next, we tried our hand at finding some low-lying improvements to the model. As we will discuss, a theorem of linear algebra enforces a condition known as the zero-variance property - as our training model approaches the ground state, the variance in our energy estimates drops to zero. One thing we explore in this project is tying the learning rate of our model to the variance of this energy estimate, so that the zero-variance property naturally slows our model's learning in the vicinity of the ground state.

Finally, we worked out the theoretical basics for training a net to find higher-energy excited states. As we will explain below, we do this by first training a net to approximate the ground state. We then train a second net, but on an altered version of the energy functional that includes a Lagrange multiplier term enforcing an orthogonality constraint between the previously-trained ground state net and the currently-training higher state net.

We will discuss our results below, but all of these efforts were more-or-less successful. We were able to train our nets on all of the model Hamiltonians that we implemented, even as we pushed our system size as large as 125 lattice sites. (We note that finding an exact ground state of a 125-lattice-site model would require diagonalizing a matrix with 2^{250} elements!) We are able to witness both a plateau of energy values and a vanishing of the variance of our energy estimations as the net approaches the ground state. For all systems small enough that we are able to effect an exact diagonalization, our neural net results are in agreement with the exact answer. The strategy of tying the learning rate to the variance seems to work, although turns out to be mostly unnecessary since convergence is so efficient in the models we explore here. Finally, the Lagrange multiplier method of finding higher energy states produces promising results, though there are still obstacles to overcome to make it an effective tool.

Going forward, we hope to use our code library to be able to well-approximate eigenstates of various Hamiltonians of interest to aid in our research in many-body theory.

2 Quantum mechanical preliminaries

2.1 Schrodinger equation and energy states

The central object in quantum mechanics is the *wavefunction*, ψ, which is the solution of the famous Schrodinger equation,

$$\hat{\mathcal{H}}\psi = i\partial_t\psi, \tag{1}$$

where $\hat{\mathcal{H}}$ is the "Hamiltonian" of the system: a linear, Hermitian, differential operator encoding the physics of a given situation. The wave function of a quantum system contains all physical information such as energetics of the system, and can be used to also generate the dynamics of the system. To see this we note that a quantum state vector, denoted $|\Psi\rangle$ can be expressed in an orthornomal basis of quantum states $\{|s\rangle\}$ using the wavefunction $\psi(s)$:

$$|\Psi\rangle = \sum_{\{s\}} \psi(s)\,|s\rangle\,; \tag{2}$$

where $\psi(s)$ denotes the degree of overlap between $|s\rangle$ and $|\Psi\rangle$, denoted by their inner product $\langle s|\Psi\rangle = \psi(s)$ (note: $\langle s| = |s\rangle^\dagger$ i.e. its Hermitian conjugate).

Observables are denoted by Hermitian operators, $\hat{O}$, with their quantum expectation values given by:

$$\langle\hat{O}\rangle = \langle\Psi|\hat{O}|\Psi\rangle = \sum_{\{s,s'\}} \psi^\dagger(s)O_{ss'}\psi(s'); \tag{3}$$

where $O_{ss'}$ is the matrix element of $\hat{O}$ given by $O_{ss'} = \langle s|\hat{O}|s'\rangle$.

Since the wavefunction conatins *all* the physical information of the system, it is an exceedingly complex object with complexity that is exponential in the system size.

One usually solves the Schrodinger equation by solving the related eigenvalue equations

$$\hat{\mathcal{H}}\phi_j = E_j\phi_j. \tag{4}$$

Above, for $j \in \{0, 1, 2, ...\}$, E_j are the quantized energies of the system, and ϕ_j are the corresponding quantum eigenstates. A deep fact about quantum theory is that the operator $\hat{\mathcal{H}}$ is always *Hermitian*, which implies the following facts:

1. there are countably many eigenvalues E_j, and they have a lower bound E_0.
2. the ψ_j are all orthogonal functions,
3. the E_j are all real,
4. there is definite ordering of the eigenvalues such that $E_0 < E_1 < E_2 < ...$

If one is able to find the energies and eigenstates, then one has fully solved the system. Naturally, this is usually impossible to do analytically. While the system can be solved by writing $\hat{\mathcal{H}}$ as a matrix in a suitable basis and then diagonalizing the matrix, this can become computationally inefficient for large number of particles. For example, for a system of N quantum particles where each particle can have one of M possible states, the size of the basis scales as M^N and requires diagonalizing an $M^N \times M^N$ matrix.

Fortunately, for most physical systems, a great deal can be learned from obtaining the lowest few eigenvalues and eigenstates. It is thus important to develop efficient numerical and computational schemes to approximate the lowest value eigenstates of a given Hamiltonian. As discussed in the introduction, the past half-century has seen a log of progress on this front, producing the celebrated quantum Monte Carlo (QMC) and density-matrix renormalization group (DMRG) techniques. An exciting idea on the frontier of computational quantum mechanics is the use of deep neural networks to better represent quantum eigenstates and improve searches for the lowest ('ground') eigenstate.

2.2 Variational approach to quantum mechanics

By a fundamental theorem from linear algebra on Hermitian operators, we know that every quantum state has energy at least E_0, and that a state with energy exactly E_0 must be the ground state, ψ_0. These facts can be used to construct a general method to approximate the ground state of a quantum system. In this procedure, one constructs a "variational wavefunction", a physically-motivated guess for the form of the wavefunction with as many tunable parameters as possible. One then tunes the parameters to minimize the total energy of the quantum state, finding the best approximation to the ground state possible within the given architecture. This simple framework lays a foundation for all of the algorithms mentioned and provides the theoretical grounding for our project with neural nets.

We start with a trial function $\psi(|\sigma\rangle\,; \mathbf{W})$, where $|\sigma\rangle$ is the state the wavefunction is evaluated at, and $\mathbf{W}$ are the tunable parameters. We can now define our loss function to be related to the expectation value $E \sim \psi^\dagger \mathcal{H} \psi$ of the operator $\hat{\mathcal{H}}$. We then alter the parameters $\mathcal{W}$ to directly minimize E, causing $E \to E_0, \psi \to \psi_0$. Quantum theory ensures us that as $E \to E_0$, our trial wavefunction approaches the desired ground state wavefunction, ψ_0.

We note that the variational technique gets more and more powerful the more parameters we have available to optimize. To see this, we note that if a variational wavefunction has N parameters, then it's possible configurations form an N-dimensional approximation of the true infinite-dimensional Hilbert space of the physical problem. If we want the energy minimum within our finite parameter-space to be as close as possible to the true minimum, we should make N as large as possible. (Though, we note, there is certainly an art to this. There are many ways that a wavefunction could depend on parameters.)

3 Problems in many-body physics

3.1 Statistical mechanics models

Reduction is a fundamental principle of physics. To understand a physical mechanism, physicists build a model that retains only the key aspects of the problem. This is exactly what condensed matter physicists have done to deal with the situation described in the previous section, and has lead to the formation of a staggering pantheon of seemingly simple physical "toy" models. Among this are the Hubbard, Heisenberg, Ising, t-J, sine-Gordon, XXZ, XY, percolation, etc. Most of these models can be understood in terms of relatively simple degrees of freedom living on a lattice and thus can be realized as simple, (though stupendously big), matrices. There are some special cases where we can get exact solutions, but for the most part, the models that contain insight into new physics can't be solved. This supplies the motivation for a numerical, algorithmic approach to condensed matter physics.

In this report, we focus on the "Transverse field Ising model"[12] in one, two, and three dimensions, and on the "Heisenberg model" in one dimension, two of the most famous many-body models in quantum physics. The Ising model is slightly simpler, so we will use it as our primary tool of exposition.

3.2 Transverse field Ising model

3.2.1 Definition

The Transverse Field Ising Model (TFI) considers the case of N number of spins on a lattice where each spin can interact with its neighbours, as well as interact with an external magnetic field. The Hamiltonian operator (in one dimension, with external field along the x-direction) is given by:

$$\mathcal{H} = -h\sum_{i}^{N}\hat{\sigma_i^x} - J\sum_{i}^{N}\hat{\sigma_i^z}\hat{\sigma_{i+1}^z} \tag{5}$$

where:

- the index i runs over all the locations of the spins
- h is the strength of each spin's coupling with the external magnetic field
- J is the strength of each spin's coupling with its neighbour
- the spin operators are the renowned Pauli matrices (note that ˆ is used to indicate operators; the absence of ˆ indicates scalar values):
$$\hat{\sigma^z} = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}; \qquad \hat{\sigma^x} = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \tag{6}$$

For the case where each spin can have values ± 1, the Hamiltonian becomes a $2^N \times 2^N$ matrix that needs to be diagonalized. This quickly exceeds the memory and computational power of most computers for as little as 100 spins.

3.2.2 Exact example for two particles

To illustrate some of the concepts we rely on for our discussion of neural network quantum states later, we provide an exact diagonalization example for two particles. For two particles, we choose a basis set where our basis vectors of are: $|\sigma\rangle \equiv |\sigma_1^z, \sigma_2^z\rangle \in \{|1,1\rangle, |-1,1\rangle, |1,-1\rangle, |-1,-1\rangle\}$. These basis vectors provide the possible spin configurations in the absence of interactions.

In this case, the Hamiltonian is:

$$\mathcal{H} = -\begin{pmatrix}1 & h & 0 & 0\\ h & 1 & 0 & 0\\ 0 & 0 & 1 & h\\ 0 & 0 & h & 1\end{pmatrix} - 2J\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{7}$$

By exactly diagonqlizing this Hamiltonian, we find that the energy eigenstates are:

$$E \in \{1 - \sqrt{h^2+4J^2}, 11 - \sqrt{h^2+4J^2}, 1 + \sqrt{h^2+4J^2}, 1 + \sqrt{h^2+4J^2}\} \tag{8}$$

Note the clear ordering present in the eigenvalues: the ground energy $E_0 = -h - \sqrt{h^2+4J^2}$. We have degeneracy in that we have two ground eigenstates (as well as any of their linear combinations). One such possible ground eigenstate has the wavefunction:

$$\psi_0(\sigma_1^z = 1, \sigma_2^z 1 = 1) = 0; \psi_0(1,-1) = 0; \psi_0(-1,1) = \frac{\sqrt{h^2+4J^2} - 2J}{h\mathcal{N}}; \psi_0(-1,-1) = \frac{1}{\mathcal{N}}; \tag{9}$$

where $\mathcal{N}$ is the normalization factor.

It is clearly obvious that such an approach with exact diagonalization becomes quite problematic when the Hamiltonian is of size, say, $2^{10} \times 2^{10}$. However, there are other approaches that simplify the task of finding the ground state.

3.3 Heisenberg model

3.3.1 Definition

Like the transverse field Ising model (TFI), the Heisenberg model also describes N spins on a lattice, interacting with neighbors and an external magnetic field. The Heisenberg treats the neighbor-spin interactions in a more fully quantum-mechanical way than the Ising model, and is a more realistic model of quantum magnetism. In one dimension, with a general external field, the Hamiltonian operator is given by:

$$\mathcal{H} = -\sum_{i}^{N}(h_x\hat{\sigma_i^x} + h_y\hat{\sigma_i^y} + h_z\hat{\sigma_i^z}) - \sum_{i}^{N}(J_x\hat{\sigma_i^x}\hat{\sigma_{i+1}^x} + J_y\hat{\sigma_i^y}\hat{\sigma_{i+1}^y} + J_z\hat{\sigma_i^z}\hat{\sigma_{i+1}^z}) \tag{10}$$

Above, h_x, h_y, h_z correspond to the (generally anisotropic) coupling of the spins to an external field while J_x, J_y, J_z correspond to the (generally anisotropic) spin-spin interaction. $\hat{\sigma}^z$ and $\hat{\sigma}^x$ are as given in (6), and

$$\hat{\sigma}^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}. \tag{11}$$

3.4 Ising and Heisenberg in contemporary physics

Here we choose to focus on the Ising and Heisenberg models for their simplicity and ubiquity - these are famous models than any physics student will have encountered and they are an obvious first choice for testing a new computational technique. Though both models are ”simple” - in the sense that we can easily write down their Hamiltonians - they are far from trivial.

Both models have played important historical roles. The original version of the Ising model - with no transverse field - was introduced in 1920 and was the first model used by physicists to understand finite-temperature phase transitions. (Ising solved the one-dimensional version of the model in his thesis, and dissapointedly quit physics after finding no phase transition. Lars Onsager solved the two dimensional version in 1944 and proved that the phase transition does indeed exist. The model has not been solved in more than two dimensions.)

Much more recently, the transverse-field Ising model has served the analogous role for the more contemporary topic of *quantum phase transitions*, phase transitions that happen at zero temperature and are due to quantum fluctuations rather than thermal fluctuations. The one-dimensional version of the model was solved exactly (in a sense) in 1970 by Pierre Pfeuty[12], using techniques developed by Lieb, Shcults, and Mattis at Bell Labs[13]. Despite the existence of this “solution”, many questions about the transverse field Ising model are of contemporary interest. For example, in 1989, Zamolodchikov[14], (an eminent string theorist) predicted that the quantum phase transition in the one-dimensional transverse field Ising model has an emergent symmetry described by E_8, the most complex of the 5 “exceptional Lie algebras”, a 248-dimensional symmetry[15]. This claim was finally (partially) experimentally verified in 2010 by Coldea[16].

The situation is similar for the Heisenberg model, which is a more realistic treatment of quantum magnetism than the Ising model. Since the Heisenberg model is more complex, it has fewer known exact solutions and symmetries, though it can be solved in one dimension with a technique called “Bethe ansat”. One can also show that the Hubbard model, which is one of the most fundamental yet-to-be-understood models in many body theory and which is expected to hold the key to high Tc superconductivity, can, (in special circumstances), be approximately mapped onto the Heisenberg model. In this sense, the Heisenberg model is still relevant to the most high-profile questions in condensed matter physics.

While we choose these models for their simplicity, we can see that they are far from trivial, and that computational techniques that can help us to understand these models (and their more complex counterparts) are highly desirable to physicists.

3.5 Computational techniques in many-body physics

There are a whole host of algorithms that are tremendously useful for carrying out quantum mechanical computations. Some famous algorithms in this field are Hartree-Fock and it's eventual descendant, ”density functional theory.” These are crucial for understanding quantum chemistry, doing exact calculations for few-molecule systems, and for calculation band structures of crystalline solids (neglecting the nuance of inter-particle interactions.)

The ”big three” algorithms currently dominating the scene in computational condensed matter physics are as follows:

1. Finite-size exact diagonalization
2. Quantum monte-carlo
3. Density matrix renormalization group.

The simplest of these is exact diagonalization, which directly calculates energies, (eigenvalues), of a finite-size version of model Hamiltonian of interest, (a big matrix). This necessarily requires truncating the size of the system to something that can be feasibly diagonalized, which is a severe limitation. Nevertheless, it is sometimes enough for a physicists purposes.

We have the density matrix renormalization group (DMRG), which is an iterative algorithm for slowly removing ”high-energy” information from a quantum system. To quote the abstract of a recent major review article,[3], The density-matrix renormalization group is a numerical algorithm for low-dimensional strongly correlated quantum systems based on a rather general decimation prescription. This algorithm has achieved unprecedented precision in the description of one-dimensional quantum systems. We note in passing that the convergence properties of DMRG are only now finally being understood formally in terms of a concept from quantum information theory, the ”matrix product state”. We won't go into the details of the algorithm here, but the idea is that it iteratively performs partial-diagonalizations on the most important subspaces of the problem, trying to maintain as much information as possible at each step. The DMRG is the most powerful method available in one dimension, but it isn't nearly as effective for solving higher-dimensional systems.

Finally, quantum Monte Carlo (QMC) methods form a broad class of approximation tools in many-body phsyics. These use an iterative procedure to approximate the ground state of a quantum system, working within the ”variational wavefunction” framework, defined above. They rely on the use of Methropolis-Hastings Monte Carlo sampling to effectively estimate large-dimensional integrals. These techniques are also central to the training methodology for the model we present in this project, and we will cover the theory in some detail in section 5.

QMC methods have achieved "chemical accuracy" - that is, they are essentially calculators for quantum chemists at this point, giving results within the needed accuracies for their purposes. On the other hand, these methods cannot yet tell if a system is a superconductor or answer similar questions in condensed matter physics, [3]. While this is an impressive achievement, it is a worthy goal to try and improve these methods even further. Using the structures developed in machine learning as a frame for the quantum Monte Carlo training technique is a promising route towards pushing this limit.

4 Neural-network quantum states

4.1 NQS as variational wavefunction

Neural networks provide a great opportunity to apply newly-developed machine learning capabilities to quantum mechanics. One can think of a neural net as a variational wavefunction exactly as described above, with the parameters $\mathcal{W}$ being represented by the weights and biases of the neural network. The universal approximation theorem for neural networks ensures us that representing a wavefunction with a neural net is a reasonable thing to try to do and modern training techniques allow for networks with far more tunable parameters than traditional variational wavefunctions.

4.2 Restricted Boltzmann machine

In this project we focus our attention on the implementation of a rather special kind of neural network, the Restricted Boltzmann Machine, as used in []. The RBM has several desirable structural properties that allow for efficient representation of the wavefunction, as well as expedited implementation and training of the neural network. Our specialization to the RBM will allow us to directly compare our results with existing literature.

4.2.1 History in machine learning

Restricted Boltzmann machines (RBM) have been present in the zoology of machine learning architectures since their introduction in the 1980s[17, 18]. They are named in an homage to Boltzmann due to their mathematical similarity to the "partition function of a system in thermodynamic equilibrium, as well discuss further on. Topologically, they resemble a single layer of a feed-forward neural network. However, the network calculates its output nonlinearly via the Boltzmann distribution. The model has been deployed for many canonical machine learning tasks, including image processing, where it has had mixed successes[18, 19]. The topology of the RBM is carefully dictated and we pay a price for this in terms of expressive power the RBM is not an extremely versatile architecture. However, there is a good reason to make this sacrifice of expressive power: the RBMs special structure allows for an extremely efficient Gibbs sampling protocol. If we interpret the output of the RBM as inducing a probability distribution on the sample space, then we can use this Gibbs sampling protocol to efficiently sample this distribution. This is the primary motivation for the consideration of the RBM in general,[17, 18, 19, 20]. With this perspective, its easy to see why an RBM is a great choice for our purposes here. The primary challenge in using a neural net architecture to learn a quantum wavefunction is that we dont have access to a "training set" of examples. The only thing we have that can act as a loss function is the energy functional itself, which we need to somehow estimate. As we will see, the RBMs efficient Gibbs sampling scheme coupled with the traditional quantum Monte Carlo framework allow us to just that.

4.2.2 Network architecture

Within the RBM neural network, the 'visible' input layer with M notes represents the input state which is the z projection of the spins: $|\sigma\rangle = |\sigma_1, \sigma_2, .., \sigma_N\rangle$, with the output of the neural network being the wavefunction $\psi(\sigma)$. There is one 'hidden' layer with M nodes, with values $h = h_1, h_2, .., h_N$, that is fully connected with the input layer. However, the RBM is "restricted" in the sense that there are no intra-layer connections. An illustration is given in Fig. 1. The weights between the layers are W_{ij} where $i \in \{1, ..., N\}, j \in \{1, ..., M\}$ with other weighting factors a_i, b_j. Then the wavefunction is given by $\psi(\sigma) = \psi(\sigma_1^z, ..., \sigma_N^z)$ such that $|\psi|^2 = \sum_{\{h\}} P(\vec{\sigma}, \vec{h})$ where the probability $P(\sigma, h)$ is given by:

$$P(\sigma, h) = \frac{1}{\mathcal{N}} \exp\left[\sum_{ij} W_{ij}\sigma_i^z h_j + \sum_j h_j b_j + \sum_i \sigma_i^z a_i\right] \tag{12}$$

where $\mathcal{N}$ represents the global normalization, and the $h_j \in -1, 1$ represent the value of the j-th hidden unit.

This forms our trial wavefunction, with W, a, b forming our tunable parameters. Our loss function therefore is $E = \langle\mathcal{H}\rangle = \sum_{\vec{\sigma},\vec{\sigma}'} \psi^\dagger(\vec{\sigma})\mathcal{H}\psi(\vec{\sigma})$, where the sum is over all 2^N possible spin conigurations $\vec{\sigma}$. By tweaking W, a, b, we seek to minimize our

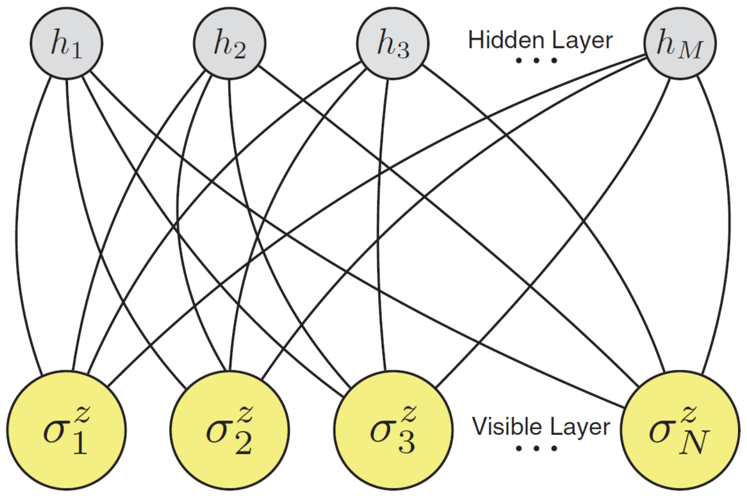


Figure 1: Illustration of the architecture, taken from []

loss function, and therefore find the ground state and energy, and use it to update our parameters (η is the learning rate):

$$a_i = a_i - \eta\, \partial_{a_i} \langle \mathcal{H} \rangle\, ; \tag{13}$$
$$b_j = b_j - \eta\, \partial_{b_j} \langle \mathcal{H} \rangle\, ; \tag{14}$$
$$W_{ij} = W_{ij} - \eta\, \partial_{W_{ij}} \langle \mathcal{H} \rangle\, ; \tag{15}$$

While this seems innocuous on surface, **it is not**. In reality, $\langle H \rangle$ is a daunting task, and calculating $\partial_p \langle \mathcal{H} \rangle$ for a given parameter p even more so. Recall that ψ here can have 2^N input states, and therefore, accurately estimating $\langle \mathcal{H} \rangle$ requires taking a 2^N size vector and calculate its inner product with a $2^N \times 2^N$ matrix. Having W, a, b, h dramatically increases the computational cost of this task, and calculating derivatives similarly proves to be another challenge.

However, this bottleneck is extremely simplified by two conceptual breakthroughs:

- utilizing the unique mathematical structure of the RBM wavefunction to carry out mathematical simplifications,
- utilitizing **Quantum Monte Carlo** methods to accurately estimate the expectation value o the Hamiltonian.

4.2.3 Inspiration and physical interpretation

The hidden units have a quite interesting role here. In physical terms, they can be thought as a fictitious gas of M spins $\vec{h}$ that couples to the N spins $\vec{\sigma}$, and indirectly mediates the interactions (otherwise encoded in the Hamitlonian) between them. Such mathematical trick where a 'bath' of fictitious spins provides a way for the spins $\vec{\sigma}$ to exchange energy and interact with each other is quite commonly seen in statistical physics systems.

5 Quantum Monte Carlo training techniques

5.1 Introduction

5.2 Expectation values and local operators

One of the key ideas is to replace the quantum expectation values of an operator $\hat{O} = \langle \hat{O} \rangle$, which involves calculating the overlap over all possible spin configurations , with a statistical average over a representative sample of N_S spin states $\{s_i\}$, where each s is a spin configuration $s = \{\sigma_1, ..., \sigma_N\}$). Then, enforcing explicit normalisation:

$$\langle \hat{O} \rangle = \frac{\langle \Psi | \hat{O} | \Psi \rangle}{\langle \Psi | \Psi \rangle} \sim \frac{\sum_{i,j=1}^{N_s} \psi(s_i)^\dagger O_{s_i s_j} \psi(s_j)}{\sum_i |\psi(s_i)|^2} = \langle \langle O \rangle \rangle \tag{16}$$

If we can find a sufficiently representative sample, we have reduced the summation from 2^N to N_S, vastly reducing computation time.

Since $\hat{O}$ may usually have non-zero off-diagonal elements, we can define a *local* operator that is by explicit construction:

$$O_l(\sigma) = \sum_{\sigma'} O_{\sigma\sigma'} \frac{\psi(\sigma')}{\psi(\sigma)} \tag{17}$$

where the sum σ' runs over all states such that $H_{\sigma\sigma'}$ is non-zero. It is key that we can calulate the local operator in polynomial time, rather than by summing over all (exponentially many) quantum states σ'. Usually in quantum mechanics, the vast majority of the matrix elements $O_{\sigma,\sigma'}$ are zero, and we can use these symmetries to compute the local energy efficiently. Then,

$$\frac{\langle\Psi|\hat{O}|\Psi\rangle}{\langle\Psi|\Psi\rangle} \sim \frac{\sum_{i,}^{N_s} |\psi(s_i)|^2 O_l(s_i}{\sum_i |\psi(s_i)|^2} = \langle\langle O_l \rangle\rangle \tag{18}$$

Since our loss function requires calculatting $\langle H\rangle$, we define the local energy:

$$E_l(\sigma) = \sum_{\sigma'} H_{\sigma\sigma'} \frac{\psi(\sigma')}{\psi(\sigma)} \tag{19}$$

Note that while this definition is general, the exact computational formula for E_l depends on the Hamiltonian H in question.

5.3 Stochastic estimates of energy and energy gradients

Assume now that the wavefunction $\psi = \psi(\sigma; p)$ where $p = \{W, a, b\}$ are our variational parameters. Then, in order to computer derivatives of the loss function, we need to calcualte:

$$\partial_{p_k} \langle H\rangle = \partial_{p_k} \langle\langle E_l \rangle\rangle \tag{20}$$

If we define a new derivative operator:

$$D_k(\sigma) = \frac{\partial_{p_k}\psi(\sigma)}{\psi(\sigma)}, \tag{21}$$

then it may be shown that

$$\partial_{p_k} \langle H\rangle \sim 2\mathrm{Re}\, \langle\langle\, [E_l - \langle\langle E_l\rangle\rangle\,]\, D_k^* \rangle\rangle\,. \tag{22}$$

Finally, we can update our parameters $p_k \in \{W, a, b\}$ with learning rate η.

$$p_k^{n+1} = p_k^n - \eta \partial_{p_k} \langle H\rangle \tag{23}$$

The only challenge remaining is to figure out how to actually *compute* our estimate of the expectation value in (18). Looking at (18), we see that we can recast the summation over states as an *expectation value* of the function $O_l(s)$ with respect to the *probability distribution* $|\psi(s)|^2$. Our strategy will then be to use the Metropolis-Hastings algorithm to generate a sequence of states s from the (known) probability distribution $|\psi(s)|^2$ to effeciently compute the expectation value of $O_l(s)$. We'll return to this later.

5.3.1 Gradient operators for RBM

We note a nice property of the restricted Boltzmann machine: the Gradient operators take a particularly simple form. For a state $s \to [s_1, ...s_N]$, we have

$$D_{a_i}(s) = \frac{1}{2} s_i, \tag{24}$$

$$D_{b_j}(s) = \frac{1}{2} \tanh\left[\theta_j(s)\right], \tag{25}$$

$$D_{W_{ij}}(s) = \frac{1}{2} s_i \tanh\left[\theta_j(s)\right], \tag{26}$$

where above we introduce "effective angles" $\theta_j(s) = b_j + \sum_i W_{ij} s_i$. With these formulas we can quickly compute a Metropolis-Hastings-generated sample's contribution to the gradient of the energy functional with respect to the parameters of our model.

5.4 Metropolis-Hastings algorithm and Gibbs sampling

5.4.1 General method

Metropolis-Hastings is a general algorithm for efficiently sampling a known probability distribution. It is a Markov chain Monte Carlo method, in that it uses only the current sample to generate the next sample. We must construct some transition rate, $\tau(x \to x')$ that we *are* able to generate samples from. One can then show that the stationary distribution of the Markov chain is given by the probability density $p(x)$ if and only if the following "principle of detailed balance" holds

$$p(x)\tau(x \to x') = p(x')\tau(x' \to x) \tag{27}$$

for all x, x' in the sample space. This is usually enforced in Metropolis-Hastings via a "accept-reject" policy. A sample is generated via τ, and then it is either accepted or rejected according to an "acceptance probability", $A(x \to x')$ that satisfies

$$\frac{A(x \to x')}{A(x' \to x)} = \frac{p(x')}{p(x)} \frac{\tau(x' \to x)}{\tau(x \to x')} \tag{28}$$

The above condition is sufficient to enforce the principle of detailed balance in (27). Finally, we close the general discussion by nothing that given $p(x)$ and $\tau(x \to x')$, we can construct an acceptance probability via

$$A(x \to x') = \min\left[1, \frac{p(x')}{p(x)} \frac{\tau(x' \to x)}{\tau(x \to x')}\right] \tag{29}$$

5.4.2 Efficient sampling for RBM

We can apply the Metropolis-Hastings algorithm efficiently by using the following selection rules. At each step, we generate both a new visible state and a new hidden state. The transition rates are given by:

$$T_\sigma\left[(s,h) \to (s',h)\right] = \frac{P(s',h)}{N_s(h)} = P(s|h) \tag{30}$$

$$T_h\left[(s,h) \to (s,h')\right] = \frac{P(s,h')}{N_h(s)} = P(h|s), \tag{31}$$

where

$$P(\sigma,h) = \exp\left[\sum_{ij} W_{ij} s_i h_j + \sum_j h_j b_j + \sum_i s_i a_i\right] \tag{32}$$

and the normalization factors are

$$N_s(h) = \sum_s P(s,h) \tag{33}$$

$$N_h(s) = \sum_h P(s,h) \tag{34}$$

The normalization factors are actually not feasible to compute, but it turns out that normalization factors aren't necessary for Markov chain Monte Carlo sampling - we only care about relative frequencies.

(As an aside, we note that *because* Markov chain sampling doesn't require the normalizations, and thus is a viable method, we can actually use Markov chain sampling to then go back and efficiently estimate the normalizations, if we wanted to.)

We note that this technique is highly efficient. It allows us to generate an entirely new pair of samples (visible and hidden) at each iteration such that our acceptance probability is 1 - all samples generated are useful! To see that the acceptance probability is 1, we invoke the Metropolis-Hastings acceptance rule discussed above in terms of the conditional probabilities for s, h:

$$A((s,h) \to (s',h)) = \min\left[a, \frac{P(s',h)}{P(s,h)} \frac{P(s|h)}{P(s'|h)}\right] = 1, \tag{35}$$

Where we above used the facts that ratios of probabilities and conditional probabilities are equal.

Plugging the form of the transition rules into the above protocol, we find the following simple algorithm for generating a new state (s',h') from a state (s,h):

1. From s and h we generate γ and θ, where

$$\theta_j = b_j + \sum_i W_{ij} s_i \tag{36}$$

$$\gamma_i = a_i + \sum_j W_{ij} h_j \tag{37}$$

$$\tag{38}$$

2. For each spin in the physical sector (s_i), choose a random number $0 < r < 1$ and set $s_i = \mathrm{sgn}[L(2\gamma_i) - r]$

3. For each spin in the hidden sector (h_j), choose a random number $0 < r < 1$ and set $h_j = \mathrm{sgn}[L(2\theta_j) - r]$

   Above, $L(x)$ is the logistic function,

$$L(x) = \frac{1}{1+e^{-x}} \tag{39}$$

5.5 Zero-variance property

A rather interesting property of E_l is the zero-variance property. It can be shown that

$$var(E_l) = \langle E_l^2 \rangle - \langle E_l \rangle^2 = \langle H^2 \rangle - \langle H \rangle^2 \tag{40}$$

But H is a Hermitian observable. Thus if we pick ψ such that it is an eigenstate of H (as the ground state will be), then $\langle H^2 \rangle = \langle H \rangle^2$ i.e.

$$var(E_l) = 0; \tag{41}$$

This is an important property: it implies that as our trial wavefunction approaches ψ_0, the ground (lowest energy) eigenstate, the fluctuations in our estimate of the local energy approach zero, aiding convergence. One of our proposals for trying to improve on existing implementations of this architecture is to try and tie the learning rate of our model to the variance of our energy estimates, so that our model's training naturally slows down as the model nears the ground state.

6 Implementation and Results

6.1 Code:

When first starting on this project, we studied the C++ code provided in the supplementary materials of Carleo,Troyer(2015) [3], as well as the (similar concept, quite different implementation) provided in [21]. We eventually wrote our own Python code library almost from scratch, which is attached with this report.

We define a general class `NeuralQuantumState` that implements a lot of the general methods, such as calculating γ,θ,ψ etc. For specific Hamiltonian, we define separate training methods and methods for calculating the local energy, since the local energy is the only operator that encodes the physics of a specific Hamiltonian.

Our code libraries are all included in `IsingNQS.py`. Our import statements look like the following:

```python
import numpy as np
import matplotlib as mp
%matplotlib inline
import pandas as pd
import IsingNQS as INQS
import matplotlib.pyplot as plt
import matplotlib.cm as cm
```

with the matplotlib and assorted imports mostly for plotting purposes.

In order to run most our code, we need to provide the values of $J, \vec{h} = (h_x, h_y, h_z)$. Training is separated into multiple epochs of number `EpochNum`, each with size `EpochSize`. We also implemented regularization, which can be turned on or off to improve training.

To improve our statistical averaging, we implemented 'burnout' when generating sample states using our Markov Chain. Usually, it can take a few iterations before the Markov Chain converges to samples that accurately reflect the desired probability distribution. Therefore, when generating samples, we discard the first few samples (indicated by the variable `NumBurn`) and then use the next N_S samples for our statistical averages.

Finally, we tied the number of hidden units to those of the visible units by letting the user define the ratio

$$\alpha = \frac{\text{No. of hidden units}}{\text{No. of input/visible units}} \tag{42}$$

6.1.1 1D Ising Model:

For the 1D model, we need to first define it's local energy operator.

$$E_l(\sigma) = \sum_{\sigma'} H_{\sigma\sigma'} \frac{\psi(\sigma')}{\psi(\sigma)} \tag{43}$$

However, the summation can be vastly simplfied, since the summation only connects states σ' to σ such that either $\sigma' = \sigma$, or σ' differs from σ by at most one spin flip. We also use periodic boundary conditions.

To run the Ising1D, we use code similar to the sample provided below:

```python
J = 4;
hx = 0.1;
hy = 0
hz = 0.0;
N = 10;
alpha = 4;
eps = 0.1#weight initializer
EpochNum = 100#
```

```
EpochSize = 1000#
NumBurn = 5#
Skip = 1#divisibility by skip
eta = 0.01#learning rate
reg = 0.#multiplier for regularization

IsingNQS, Energy, EnergyVariance = INQS.TrainIsingNQS(J,hx,hy,hz,N,alpha,eps,EpochNum,EpochSize,NumBurn,
    Skip,eta,reg);
```

Here, we define the coupling between adjacent spins as J, with the external field vector denoted $\vec{h} = (h_x, h_y, h_z)$. Since we are working with a 1D model, the number of spins is just indicated with N.

The method `TrainIsingNQS` returns the `NeuralQuantumState` class for the 1D Ising problem, as well as a list of the local Energy adn its variance for each iteration. Note that we expect that the variance should approach zero, and therefore, the size of the variance as training progress provides a check on whether training is converging on a true eigenstate.

6.1.2 2D Ising Model:

Similar to the 1D model, we now define the 2D local energy, except this time the summation over σ' includes states with one spin flip but in nearest neighbour interactions.

For Ising model in 2 dimensions, we use sample code such as the one given below:

```
Na = 3
Nb = 3
IsingNQS2D, Energy2D, EnergyVariance2D=INQS.Train2DIsingNQS(J,hx,hy,hz,Na,Nb,alpha,eps,EpochNum,EpochSize,
    NumBurn,Skip,eta,reg)
```

One difference compared to 1D, is that we now specify the size of the system as $Na \times Nb$.

6.1.3 3D Ising Model:

Similar to the 1D and 2D model, we now define the 2D local energy, except this time the summation over σ' includes states with one spin flip but in nearest neighbour interactions over the three dimensional lattice.

For Ising model in 3 dimensions, we use sample code such as the one given below:

```
Na = 3;
Nb = 3;
Nc = 3;
IsingNQS3D, Energy3D, EnergyVariance3D = INQS.Train3DIsingNQS(J,hx,hy,hz,Na,Nb,Nc,alpha,eps,EpochNum,
    EpochSize,NumBurn,Skip,eta,reg)
```

6.1.4 1D Heisenberg Model:

For Heisenerg, we have to now include $\vec{J}$ couplings in each Cartesian axis:

```
Jx = 3
Jy = 3
Jz = 3

hx = 1
hy = 0
hz = 1

N = 8

HeisenbergNQS, HeisenbergEnergy, HeisenbergEnergyVariance = INQS.TrainHeisenbergNQS(Jx,Jy,Jz,hx,hy,hz,N,
    alpha,eps,EpochNum,EpochSize,NumBurn,Skip,eta,reg)
```

6.2 Results

6.3 Benchmarking the 1D Ising model

: We first run the 1D Ising model as a diagnostic, comparing it to the known exact diagonalization result. We ran the model with the following parameters, but for different system sizes N:

```
J = 1;
hx = 4;
hy = 0
hz = 0.0;
N = 3
alpha = 2
eps = 0.1#weight initializer
```

```
EpochNum = 100#
EpochSize = 1000#
NumBurn = 5#
Skip = 1#divisibility by skip
eta = 0.05#learning rate
reg = 0.#
```

The resulting plot compared with the exact result is given in Fig. 2. As we can see, the system rapidly converges on the

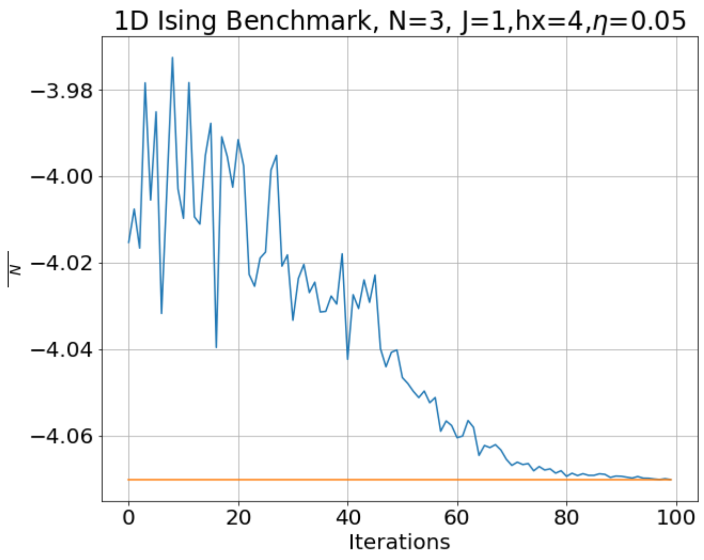


Figure 2: Comparison of Ising1D with exact diagonalization result

exact diagonalization result.

6.3.1 Ising 1D for different N

We now explore the Ising 1D model for different system sizes. We choose to plot E_{local}/N, since thermodynamically, this value also converges as N increases. In other words, the different N should approach the same value of E/N for larger N. We also plot the Var(E), since we expect this to rapidly approach zero as we approach an eigenstate, and therefore provides a good physical check that the Neural Network is indeed finding the right state. We ran the code with the following parameters:

```
J = 4;
hx = 0.1;
hy = 0
hz = 0.0;
N = 3
alpha = 3
eps = 0.1#weight initializer
EpochNum = 100#
EpochSize = 1000#
NumBurn = 5#
Skip = 1#divisibility by skip
eta = 0.01#learning rate
reg = 0.#

IsingNQS, Energy, EnergyVariance = INQS.TrainIsingNQS(J,hx,hy,hz,N,alpha,eps,EpochNum,EpochSize,NumBurn,
    Skip,eta,reg);
```

The results are given in Fig. **??**

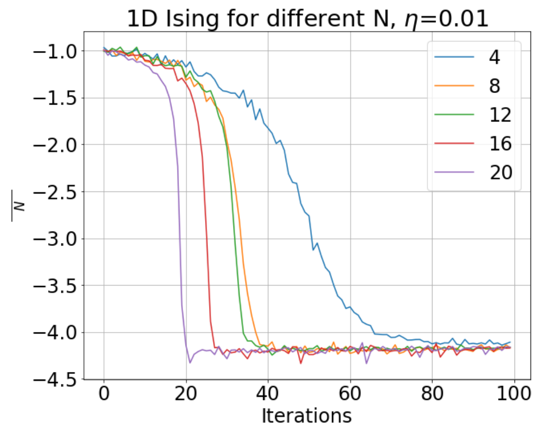
We see that convergence is more rapid for higher N, and that in all cases, the the Var(E) does converge to zero as desired.

6.3.2 Exploring size of hidden layer

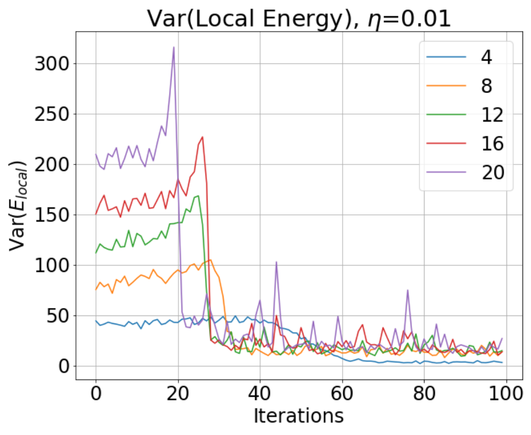
We then further test our code by varying α which determines the ratio of the size of the hidden layer to that of the input layer. We ran the code with the same parameters are the previous test.

Our results are shown in Fig **??**. We note that:

- The convergence of the energy as well as the decay of the fluctuations happens significantly faster with larger α,

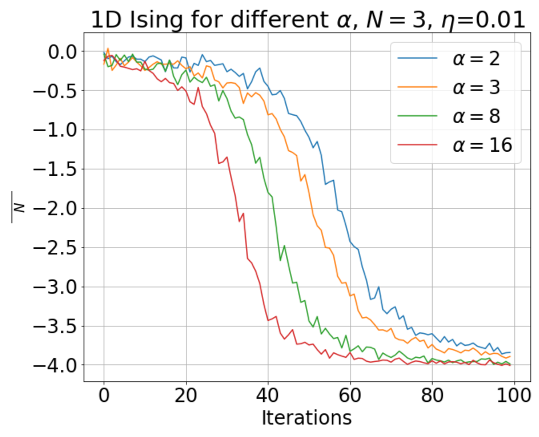


(a) 1D Ising Energy/N values for different N

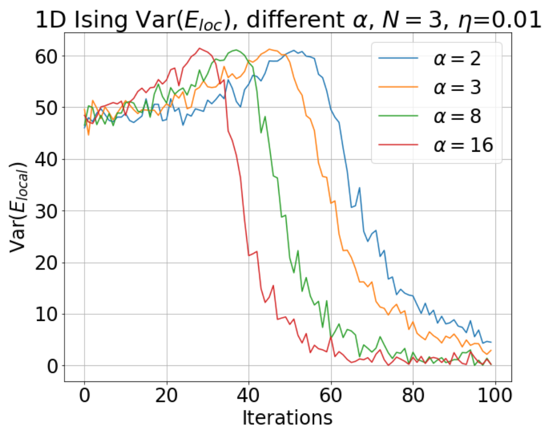


(b) 1D Ising Energy fluctuations for different N

- We can still see significant improvement with large α; even $\alpha = 16$ provides significant improvement in convergence,
- The code computes remarkably fast even with a large hidden layer size (on the order of 10s).



(a) 1D Ising Energy/N values for different α



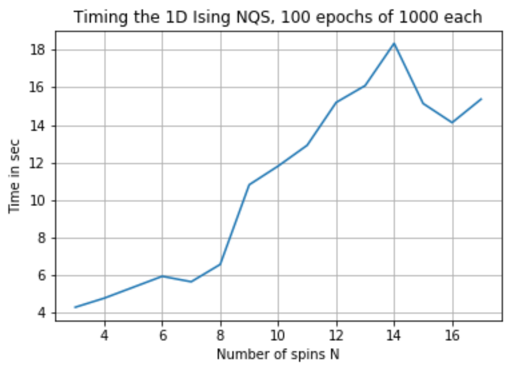
(b) 1D Ising Energy fluctuations for different α

6.3.3 Benchmarking Clock Time using 1D Ising

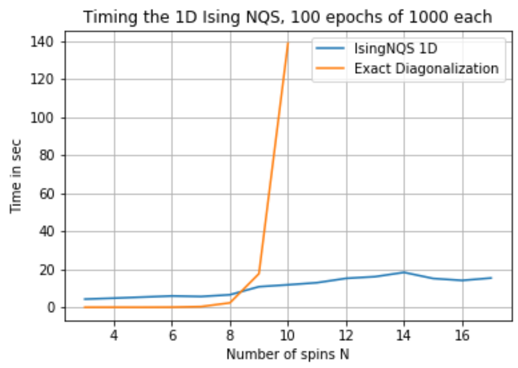
To provide more concrete justification for why the Neural Quantum State method proves to be a viable source of solving for the ground state, we investigated the clock time of our model for different system sizes using the following code to average over 5 computations.

```
import time;
T=15;
time_list=np.zeros(T);

J = 4;
hx = 0.1;
hy = 0
hz = 0.0;
alpha = 3
eps = 0.1#weight initializer
EpochNum = 100#
EpochSize = 1000#
NumBurn = 5#
Skip = 1#divisibility by skip
eta = 0.01#learning rate
reg = 0.#

for i in range(0,T):
    start=time.time();
    N=i+3;
    for thru in range(5):
        INQS.TrainIsingNQS(J,hx,hy,hz,N,alpha,eps,EpochNum,EpochSize,NumBurn,Skip,eta,reg);
    end=time.time();
    time_list[i]=(end-start)/5.0;
```

We then benchmarked this versus the time taken for exact diagonalization. Our results our shown in Fig. **??**.



(a) Scaling of computation time for different system sizes.



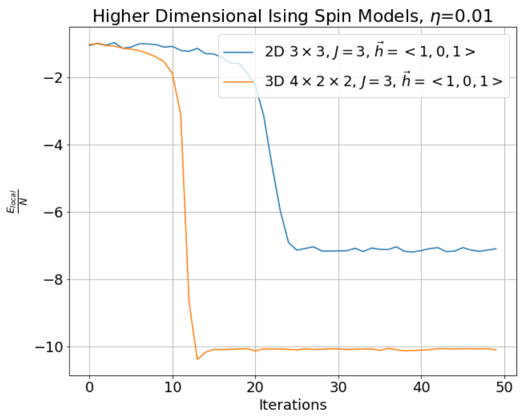
(b) Comparison of clock time vs exact diagonalization

6.4 Higher dimensional Ising Models

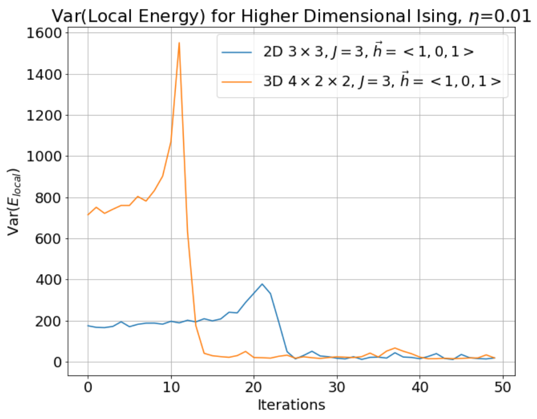
We then test our code for higher dimensional Ising Models. For the 3D case, we have been able to run our for upto $5 * 5 * 5$ sites, a feat which would otherwise require diagonalizing a $2^{125} \times 2^{125}$ matrix, a daunting task on most computers.

Our results are shown in Fig. **??**. Key takeaways are:

- Converegence of the energy and the variance seems to occur faster in higher dimensions, even when the 3D system has an overall larger size than the 2D one.
- Physical insight for this might be that the system has more nearest neighbours to couple with for higher dimensions.



(a) Ising Energy/N values for higher dimensions.



(b) Ising Energy fluctuations for higher dimensions.

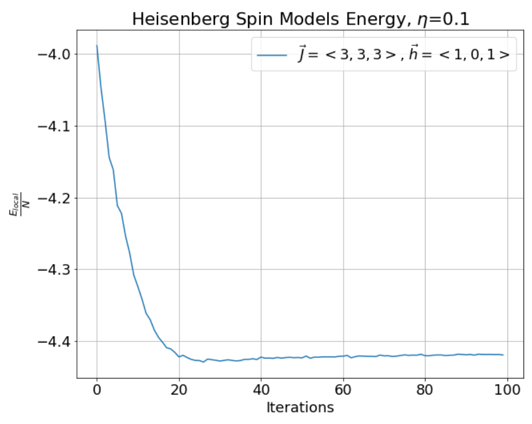
6.5 1D Heisenberg Model:

We now explored the 1D Heisenberg model, since it encodes different physics i.e. both the spin-spin couplings as well as the couplings of the spins to the external field are now vectors: $\vec{h}, \vec{J}$.

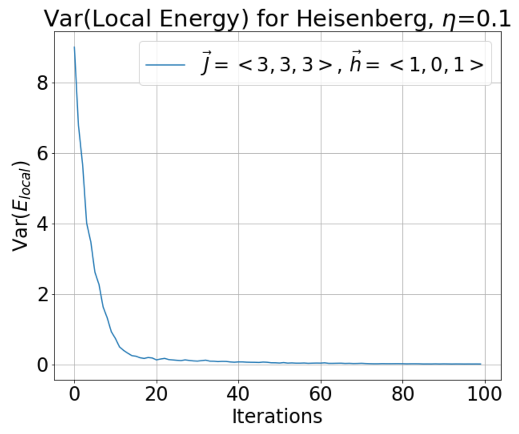
Running the code with the settings $\vec{J} = \langle 3,3,3 \rangle, \vec{h} = \langle 1,0,1 \rangle$, we obtain the results in Fig, **??**

7 Different Physical Phases

Having investigated that the neural network quantum state does indeed predict the correct energies, we now turn our tattention to the wavefunction. One of the reasons the Ising model and Heisenberg models are so powerful within condensed



(a) 1D Heisenberg Energy/N.



(b) 1D Heisenberg Energy fluctuations for higher dimensions.

matter physics is that by tweaking the coupling strengths and external field, we can demonstrate the behaviour of different physical states by looking at the nature of the ground wavefunction.

7.1 Zero temperature behaviour of the 1D Ising model

An example of this within the 1D Ising model is by simply varying J. If $J > 0$, the spin's can reduce their overall energy by being aligned with each other, and therefore minimizing $-J\sum_i \sigma_i\sigma_{i+1}$. In the absence of transverse external field $h_x, h_y = 0$, this would cause all the spins to be pointed in the same direction: a *ferromagnetic* state. The exact direction of the alignment can be picked by applying a small bias field h_z, since that would encourage the spins to point in the direction of h_z. For $h_z > 0, J > 0$, we therefore expect all the spins to be aligned in the spin-up direction i.e. $\forall i, \sigma_i = 1$.

Similarly, having $J < 0$ means spins can minimize their overall energy by being in a state where neighbouring spins are anti-aligned to each other: an *anti-ferromagnetic* state. Therefore, we would expect the system to be in a state of the form $|\sigma\rangle = |..., 1, -1, 1, -1, ..\rangle$ and so on.

We test the capability of our neural network to pick out by plotting the Restricted Boltzmann Machine probability $P(\sigma) = |\Psi(\sigma)|^2$ where Ψ is the wavefunction, and σ is the input spin configuration. We run the 1D Ising code for the settings:

```
J = 5;
hx = 0.0;
hy = 0
hz = 1.0;
N = 4
alpha = 3
eps = 0.1#weight initializer
EpochNum = 100#
EpochSize = 1000#
NumBurn = 5#
Skip = 1#divisibility by skip
eta = 0.01#learning rate
reg = 0.#

IsingNQSf, Energy, EnergyVariance = INQS.TrainIsingNQS(J,hx,hy,hz,N,alpha,eps,EpochNum,EpochSize,NumBurn,
    Skip,eta,reg);

J=-5;
IsingNQSa, Energy, EnergyVariance = INQS.TrainIsingNQS(J,hx,hy,hz,N,alpha,eps,EpochNum,EpochSize,NumBurn,
    Skip,eta,reg);
```

Plotting the results in Fig. 8, we find that that for $h_z > 0, J > 0$, almost all the probability is in the $|1,1,1,1\rangle$ state, with all spins aligned and pointing up, as expected. Therefore, the system is in the *Ferromagnetic phase*.

Similarly, for $J < 0, |J| > |h_z|$, we find that the spins are indeed anti-aligned and in the state $|1,-1,1,-1\rangle$, demonstrating the *anti-ferromagnetic phase*.

7.2 Implications

While we explored this only for 1D Ising model, our aim beyond this proect is to similarly use our code to explore richer variety of spin phases using the more complicated Heisenberg and higher dimensional Ising models. That will not require

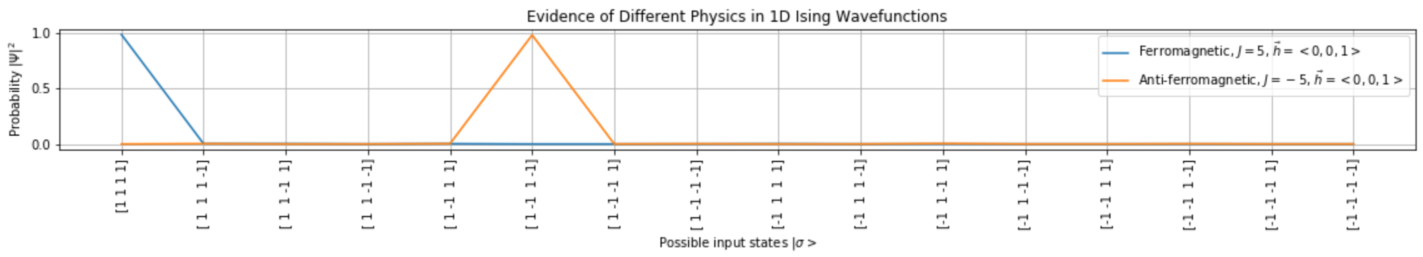


Figure 8: Neural Network picks out different (anti-)ferromagnetic states correctly.

any news changes to the code: what we've demonstrated here is that our code is already capable of correctly determining the ground eigenstate for the system, which can then be used to explore the phase diagram for a given Hamiltonian.

8 Higher energy state calculation

We would like to be able to adapt the above methodology to the computation of higher energy states. We will attempt to do this by exploiting the following fact of Hermitian operators: their eigenstates are *orthogonal.* Thus, if we know the ground state, and we minimize energy within the subspace of the physical Hilbert space that is orthogonal to the ground state, we will arrive at the firsrt excited state.

8.1 Lagrange multiplier technique

We will implement the above idea by adding a term to our energy functional that implements a Lagrange multiplier enforcing the constraint of orthogonality. Let $\langle\Psi_0\rangle$ be an already-learned approximation to the ground state, and let $\langle\Psi\rangle$ be a new NQS we hope to train. We define the *overlap* of two states as

$$\langle\Psi_0|\Psi\rangle = \sum_s \psi_0^\dagger(s)\psi(s). \tag{44}$$

We then alter our energy function by adding a new term designed to minimize the overlap:

$$\frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle} + \lambda\left|\frac{\langle\Psi_0|\Psi\rangle}{\langle\Psi|\Psi\rangle}\right|^2, \tag{45}$$

where λ is a Lagrange multiplier. We worked out this theory along similar lines to that discussed in section 5. We want to sample the distribution corresponding to $|psi(s)|^2$ to estimate the summations. It turns out that to take the Lagrange multiplier into account, we need to augment our gradients by a new term, given by

$$\lambda 2\mathrm{Re}\left[\mathcal{E}[F]^\dagger\left(\mathcal{E}[D_k F] - 2\mathcal{E}[F]\mathrm{Re}[\mathcal{E}[D]]\right)\right], \tag{46}$$

where $\mathcal{E}[\cdot]$ indicates an expectation value over the Metropolis-Hastings sampling of $|\psi(s)|^2$ and we have defined a new function

$$F(s) = \frac{\psi_0(s)}{\psi(s)^\dagger} \tag{47}$$

This is implemented in our Python library as a part of our submission.

8.2 Results and challenges

We were able to implement the above framework in our library. The code runs and the results seem promising, though they aren't a fully functional tool yet. There are some major challenges to handle here in the hyperparameter engineering regime. The biggest challenge is that - though the energy and the overlap are implemented correctly - the magnitude of the gradient of the overlap with respect to the parameters varies greatly. It can be orders of magnitude more powerful than the gradient due to just the energy. This can lead to instability in the gradient descent unless a careful balance is struck between the learning rate and Lagrange multiplier hyperparameters - the Lagrange multiplier will almost certainly need to be a dynamic quantity that updates so that the gradients due to energy and overlap minimization are on the same order of magnitude. We note that the RBM depends exponentially on it's parameters, so if the gradient descent is unstable we could (and do) quickly run into overflow issues.

9 Summary and future work

We've demonstrated the effectiveness and computational advantage of using Neural Quantum States(NQS) using the Restricted Boltzmann Machine architecture for a variety of traditional spin Hamiltonians. We've demonstrated that we can improve the accuracy by increasing the size of the hidden layer, and that for higher dimensions, the NQS tends to converge faster. There are several avenues of research beyond this project that we are currently implement:

- Exploring different training schemes: we have implemented our verison of Adam Optimizer, and intend to explore its effectiveness
- Exploring other Hamiltonians, as well as more novel physical spin phases for some of the current Hamiltonians,
- A more indepth hyperparameter search: we have developed a method to anneal the learning rate based on the variance of the local energy, and intend to explore it,
- We finally intend to use our method of Lagrange multipliers to bootstrap the code into finding higher energy eigenstates

Altogether, there are quite a few possible exciting research directions for NQS alone, and we can also extend our search to other neural architectures.

References

[1] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal. Quantum monte carlo simulations of solids. *Rev. Mod. Phys.*, 73:33–83, Jan 2001.

[2] U. Schollwöck. The density-matrix renormalization group. *Rev. Mod. Phys.*, 77:259–315, Apr 2005.

[3] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[4] Michael R. Hush. Machine learning for quantum physics. *Science*, 355(6325):580–580, 2017.

[5] Broecker et al. Machine learning quantum phases of matter beyond the fermion sign problem. *Nature Scientifc Reports volume 7, Article number: 8823*, 2017.

[6] Juan Carrasquilla and Roger G. Melko. Machine learning quantum phases of matter beyond the fermion sign problem. *Nature Physics*, 13:431–434, 2017.

[7] Giacomo Torlai and Roger G. Melko. Learning thermodynamics with boltzmann machines. *Phys. Rev. B*, 94:165134, Oct 2016.

[8] Kelvin Ch'ng, Juan Carrasquilla, Roger G. Melko, and Ehsan Khatami. Machine learning phases of strongly correlated fermions. *Phys. Rev. X*, 7:031038, Aug 2017.

[9] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. Machine learning topological states. *Phys. Rev. B*, 96:195145, Nov 2017.

[10] Hiroki Saito. Solving the bosehubbard model with machine learning. *Journal of the Physical Society of Japan*, 86(9):093001, 2017.

[11] Biamonte et al. Quantum machine learning. *Nature*, 549:195–202, 2017.

[12] Pierre Pfeuty. The one-dimensional ising model with a transverse field. *Annals of Physics*, 57(1):79 – 90, 1970.

[13] Elliott H. Lieb, Theodore Schultz, and Daniel Mattis. Two soluble models of an antiferromagnetic chain. *Annals Phys.*, 16:407–466, 1961.

[14] A. B. ZAMOLODCHIKOV. Integrals of motion and s-matrix of the (scaled) t = tc ising model with magnetic field. *International Journal of Modern Physics A*, 04(16):4235–4248, 1989.

[15] Alexander Altland and Ben D. Simons. *Condensed Matter Field Theory.* Cambridge University Press, 2 edition, 2010.

[16] R. Coldea, D. A. Tennant, E. M. Wheeler, E. Wawrzynska, D. Prabhakaran, M. Telling, K. Habicht, P. Smeibidl, and K. Kiefer. Quantum criticality in an ising chain: Experimental evidence for emergent e8 symmetry. *Science*, 327(5962):177–180, 2010.

[17] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147 – 169, 1985.

[18] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[19] Asja Fischer and Christian Igel. An introduction to restricted boltzmann machines. In Luis Alvarez, Marta Mejail, Luis Gomez, and Julio Jacobo, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 14–36, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[20] G. Carleo. Neural-network quantum states.

[21] Bart Olsthoorn. Nqs code.