Machine Learning and Many Body Problem

Chebrolu Ram Prasanth
414PH5030

Department of Physics and Astronomy
National Institute of Technology Rourkela

Machine Learning and Many Body Problem

Thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science

in

Physics

by

Chebrolu Ram Prasanth

(Roll Number: 414PH5030)

based on research carried out

under the supervision of

Prof. Sanjoy Datta

May, 2019

Department of Physics and Astronomy

National Institute of Technology Rourkela

Department of Physics and Astronomy
National Institute of Technology Rourkela

Prof. Sanjoy Datta
Asst. Professor

May 07, 2019

Supervisor's Certificate

This is to certify that the work presented in the thesis entitled *Machine Learning and Many Body Problem* submitted by *Chebrolu Ram Prasanth*, Roll Number 414PH5030, is a record of original research carried out by him under my supervision and guidance in partial fulfillment of the requirements of the degree of *Master of Science* in *Physics*. Neither this thesis nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

Sanjoy Datta

Declaration of Originality

I, *Chebrolu Ram Prasanth*, Roll Number *414PH5030* hereby declare that this thesis entitled *Machine Learning and Many Body Problem* presents my original work carried out as a M.Sc theses student of NIT Rourkela and, to the best of my knowledge, contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of NIT Rourkela or any other institution. Any contribution made to this research by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections "Reference". I have also submitted my original research records to the scrutiny committee for evaluation of my dissertation.

I am fully aware that in case of any non-compliance detected in future, the Senate of NIT Rourkela may withdraw the degree awarded to me on the basis of the present dissertation.

May 07, 2019 — *Chebrolu Ram Prasanth*

NIT Rourkela — *414PH5030*

Acknowledgment

I would like to thank Prof.Sanjoy Datta for providing me the opportunity to do my thesis in Machine Learning. I am also very thankful to Prof. Biplab Ganguli and his Research student Mr. Vinesh Vijayan for letting me work in their lab and I appreciate Mr. Sankalp Kumar, Mr. Vishal kumar Pathak, Ms. Rashi Agarwal, Ms. Disha Gupta and others for their valuable time in discussions. Lastly, I like to thank Mr. KD sumit for his valuable suggestions.

Once again, I thank my guide, Prof.Sanjoy Datta for believing me and for giving me enough freedom to explore and experiment what I am interested in and for providing valuable insight into the physical problems. —

May 07, 2019
NIT Rourkela

Chebrolu Ram Prasanth
414ph5030

Abstract

Motivated by the recent successful application of artificial neural networks to quantum many-body problems [G. Carleo and M. Troyer, Science 355, 602 (2017)]. First we learned what machine learning is and how it can be applied to solve a many-body problem. Then we verified performance of the different wave functions in converging to the ground state energy. we used Restricted Boltzmann Machine as a wave-function for the Transverse-field Ising model and Anti-Ferromagnetic Heisenberg Model for a system size of 4 spins and The feed-forward network for the Hubbard model and demonstrated that the wave function converges to the ground state energy in both the cases. We found that after a clever wave function is chosen the sampling techniques plays an important role in convergence to the ground state energy. Thus with better sampling techniques this method can become capable to calculate ground state energy for large system sizes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this study we will be focusing on the applicability of neural networks in solving many-body problem. Specifically we will focus our study on the Transverse-field Ising model, Anti-Ferromagnetic Heisenberg Model and the Hubbard Model. The appeal of this model is that it is a simplified model that can be used to provide insights in the wave function of molecular systems without explicitly computing the complex integrals required for traditional computational methods.It is here that we wish to find out if neural networks could potentially aid in the further development and improvement of quantum computational methods.

1.1 The Motivation for Machine Learning

Quantum mechanics (QM) provides a theory of matter at the atomic scale,and numerical solutions to Schrodinger's equation allow for calculation of virtually any property of a system. The major problem why aren't more problems in materials science, organic chemistry, or biochemistry solved computationally is the computational effort required, which increases so rapidly with system size that for solutions in agreement with experiment one is limited to small systems. As pointed out early last century by Paul Dirac[1], this situation necessitates approximations, trading in accuracy or generality for computational efficiency. Many such approximations were made, both on a conceptual level, such as the Born-Oppenheimer approximation, and on a numerical level, leading to a variety of approaches to solve Schrodinger's equation approximately. But the methods which can solve exactly such as Full Configuration interaction has run-time $O(N^{10})$ where N is number of particles, therefore limiting it to small system size of around 10 atoms. The other methods which as less run-time has more error in their results.[2] Some of them are listed in Table 1.1

In order to verify our theoretical results with those of experimental results we have to search for better computational techniques. Machine learning being quite popular because of its wide applicability and its success in solving many problems, provides us hope to find a better computational method to solve many-body problems.

S.no	Method	Run-time	error (eV)
1	Coupled cluster	$O(N^7)$	$0.001eV$
2	Quantum Monte Carlo	$O(N^3) - O(N^4)$	$(0.001 - 0.01)eV$
3	Density Functional Theory	$O(N^3)$	$(0.05 - 0.1)eV$
4	Tight Binding	$O(N^2)$	$0.5 + eV$

Table 1.1: Electronic Structure Methods

1.2 Organization of Thesis

chapter 2

Here we introduced different type of machine learning methods and its application in solving various problems in physics

chapter 3

Here we discussed about principles and theories that helps in application of machine learning to many-body problems and we briefly discussed the important steps involved in solving a many-body problem.

chapter 4

Here we applied this method to Transverse-field Ising model, Anti-Ferromagnetic Heisenberg Model and the Hubbard Model to find the ground state energy and verified with exact value of the ground state energy calculated via exact diagonalization.

chapter 5

Conclusion

Appendix

Here we included the code of a program for further reference and understanding of the method.

Chapter 2

Machine Learning and its applications in Physics

The Techniques of statistical learning combined with computational power gives rise to Machine Learning. Machine Learning is about the development of algorithms and techniques that allow computers to learn. By Learning here it means finding the parameters of the model or the function that can map input and output data accurately. In traditional methods in order to write an algorithm for calculation or to recognition or to evaluation of anything, We have to code all the required conditions explicitly, But in Machine learning we just give input, output samples and the Machine itself finds out the required conditions and its performance increases with increase in size of sample data.
The learning problems can be roughly categorized into three categories.[9]

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

2.1 Supervised Learning.

It is used when both input and output data is available and needed a model to map the input to the output data. The total available data is basically divided into training data, which is used by the model to learn and test data, which is used to test the algorithm after learning. It's important that the data on which the model got trained and the data that the model going to predict should belong to the same distribution. In other words the model can only predict that data which is similar to data on which it got trained.
There are two kinds of supervised learning depends on the type of the output expected.

- Classification :- classifies given data into groups.
- Regression :- mapping each and every input to its respective output.

Few examples of its application in physics are

(1) Replacing the Kinetic energy function in the density functional theory where the machine learning one learns the mapping from potential to electron density and charge density to kinetic energy respectively[3].

(2) Solving electronic structure problems and finding its ground state energy[4].

2.2 Unsupervised Learning

It is used when we only have input data and need to understand the data or categorise the data or draw some patterns out of it.
There are two basic kinds of unsupervised learning

- Unsupervised Transformations :- It includes changing the representation of the data, reducing the dimensionality of the data, finding the principal components that make up the data.

- Clustering algorithms :- partition data into distinct groups based on similarities.

Its applied to characterise materials based on specific patterns in their electronic band structure and also to find new materials with required characteristics[5]. It's also used to compare and contrast the phase behaviour and phase transitions in several classical spin models and able to find different phases ,symmetry-breaking, can also distinguish phase transition types and locate critical points[6].

2.3 Reinforcement Learning

Reinforcement learning is similar to the supervised Learning except that we need not to provide exact input output pairs. The performance is corrected by providing the score to its performance and the algorithm is designed to try to perform with better score. Input is the initial state from which the model will start and the training will be based on inputs. There will be many possible outputs. The model will return a state and the user will give a score based on its output. The model keeps learning and the best state is one with the highest score.

It was demonstrated that its capable of describing the unitary time evolution of complex interacting quantum systems. This approach is accurately describes the equilibrium and dynamic properties of prototypical interacting spin models in both one and two dimensions, thus offering a new powerful tool to solve the Quantum many body problem which we elaborated here[7].

Chapter 3

A Variational approaches to many body problems: NQS

Many-body problem is any quantum mechanical problem involving two or more interacting particles.The particles can be electrons or ions. Experimentally we can be able to study the properties of macroscopic systems, which studies an Avogadro number of particles. But computationally we can only study up to 100 particles with desired accuracy by Quantum Monte Carlo techniques. we can study up to 10,000 particles but the accuracy is compromised. So the main challenge in understanding many-body systems is to develop techniques that help us to study large number of particles with tolerable accuracy so that we can verify our theoretical results with the experimental ones.

The problem is solving for larger systems is that with increase in particle number the parametric space or the Hilbert space increase exponentially, which requires a exponentially large computational space with high computational power. But the nature often proves herself kind and meaningful. In most of the cases we do not need the whole Hilbert space to describe the characteristics of the system. So modern approaches basically involves in finding a finite number of physically relevant configurations or efficiently compressing the quantum state down to its most essential features.

Variational approach is usually the tool of choice for tackling such difficult problems, which include many successful examples from simple mean-field approximation to more complicated methods such as those based on the matrix product states, the tensor network states, the string bond states, and more recently, the neural network states. The essence of the variational approach is to find an efficient representation of the relevant quantum many-body states. With an efficient representation, one can then optimize those variational parameters by optimization techniques, such as the gradient descent method.

The many-body wave function is a mapping of the degrees of freedom to the complex number which fully specify the amplitude(energy) and phase of the quantum state.

3.1 The Variational Principle

Given any Normalized function Ψ (that satisfies the appropriate boundary conditions),then the expectation value of the Hamiltonian represents an upper bound to the exact ground state energy

$$E_g \leq \langle\Psi|H|\Psi\rangle \tag{3.1}$$

Any normalized function Ψwhatsoever,which is presumably in correct state is certain to overestimate the ground state.

The technique apply only to find the upper bound of ground state, which is sometimes all you need. One limitation of this method is you never know for sure how close you are to the exact value of the ground state energy, all you are certain of is that you got an upper bound. The closer our wave function to the exact ground state, the closer our expectation value to the exact ground state. A clever choice of the wave function should represent the ground state interactions that are reflected in the Hamiltonian. So our Approximate wave function should be different for different Hamiltonian. Any wave function that represent a many-body interactions is multi-dimensional complex function of its parameters.

3.2 Neural-network Quantum states

An Artificial neural network is nothing but a high-dimensional function, composition of simple one-dimensional functions, and depending on internal parameters, which can approximate any high-dimensional function.

Kolmogorov - Arnold representation theorem

The Kolmogorov - Arnold representation theorem[10] considers a generic continuous function of n variables $F(x_1, x_2, ..., x_n)$, and demonstrates that any high dimensional function can be approximated using only a finite number of one-variable functions.

$$F(x_1, x_2, ..., x_n = \sum_{q=0}^{2n} \phi(\sum_{p=1}^{n} \Lambda_p \phi(x_p + \eta q) + q))) \tag{3.2}$$

This theorem is particularly important because, in a sense, demystifies the complexity of high-dimensional functions and gives us hope that a many-body wave function can be approximated by a possibly easier one-dimensional functions there by provides us the Mathematical support to solve a Many Body problem using Neural Networks.

Any wave function which is represented or approximated my a neural network is called Neural-network Quantum State(NQS). The point of view that was taken here is to interpret the wave-function as a computational black box which,given an input many-body configuration S,returns a phase and amplitude according to $\Psi(S)$. The goal is to approximate this computational black box with a neural network, trained to best represent $\Psi(S)$. Different possible choices for the artificial neural network architectures have been proposed to solve specific tasks, the best architecture to describe a many-body quantum system may from one case to another.

3.3 The Machine Learning Method

The method involved in solving a many-body problem using machine learning has the following general steps:-[11, 12]

(1) The Network we choose to represent our Wave function

(2) Energy Calculation with the network

(3) Optimization of parameters

(4) Sampling

3.3.1 The Wave Function

The wave function should contain similar type of symmetries and entanglements as of the state. It should take our basis state as input, so it also depends on the type of space in with our basis state is defined. Even though the wave function is problem specific, they all have same common structure, which is inspired form the biology of the brain. The basic elements comprising the neural network wave function are artificial neurons, which are the mathematical abstractions of the biological neurons. After the input have reached the neuron, they are added together according to their weights, and then the value is compared with the bias b of the neuron to determine whether the neuron is activated or deactivated.This activation function is generally nonlinear. Examples of activation functions are logistic/sigmoidal function:

$$\phi(x) = \frac{1}{1 + exp(-x)} \tag{3.3}$$

Hyperbolic tangent function:

$$\phi(x) = \tanh x \tag{3.4}$$

The output of a network is in a form:

$$y_j = \phi(\sum_i w_{ij}x_i + b_j) \tag{3.5}$$

The network is generally contains mainly layers, the output of the first layer becomes the input of the next layer(hidden layers):

$$z_k = \phi(\sum_j w'_{jk} y_j + b'_k) \tag{3.6}$$

where ϕ is activation function.
and this will be continued until we reach the final layer. The representation power of the network increases with increase in number of hidden layers. Thus the function can be seen as a nested composition of non-linear vector functions $\Psi = F(\mathbf{x}; w, b)$ representing some arbitrary physical state.

3.3.2 The Energy

Once the wave function is represented by a network this network is used to find the energy as
For diagonal elements of H:

$$E[\Psi] = \frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\rangle\Psi} = \frac{\sum_x |\Psi(x)|^2 E_x}{\sum_x |\Psi(x)|^2} = \langle\langle H\rangle\rangle \tag{3.7}$$

For off-diagonal elements of H:

$$E[\Psi] = \frac{\sum_x |\Psi(x)|^2 E_{loc_x}}{\sum_x |\Psi(x)|^2} = \langle\langle H_{loc}\rangle\rangle \tag{3.8}$$

where E_{loc_x} is:

$$E_{loc_x} = \sum_{x'} H_{xx'} \frac{\Psi(x')}{\Psi(x)} \tag{3.9}$$

where x is the basis and H is the Hamiltonian of the system.
The energy calculated will be greater than the actual ground state energy according to variational principle $E[\Psi] \geq E_g$.

3.3.3 Optimization

The weights in the wave function are optimized using stochastic gradient method. The parameters are updated according to:

$$\mathbf{p^{(s+1)}} = \mathbf{p^{(s)}} - \eta G(\mathbf{p^{(s)}}) \tag{3.10}$$

where s indicates step, $G(\mathbf{p^{(s)}})$ is some approximation of gradient of the energy function which is to be optimized, and η is a parameter known as learning rate.

The gradient of the energy function is:

$$G(\mathbf{p^{(k)}}) = \partial_{p_k}\langle H\rangle = \partial_{p_k}\frac{\sum_{x,x'}\Psi^*(x)H_{xx'}\Psi(x')}{\sum_x |\Psi(x)|^2} \tag{3.11}$$

3.3.4 Sampling

Sampling is a very important step in any statistical method. Here,basically sampling was done to pick out states, which form a ground state from the complete basis set. Few sampling techniques are:[11]

- Markov chain Monte Carlo (MCMC) sampling
- Gibbs sampling
- Metropolis-Hastings sampling

3.4 Zero-Variance Property

The energy and the energy gradient estimators have the so-called zero-variance property. Their statistical fluctuations are exactly zero when sampling from the exact ground-state wave-function.

$$var(E_{loc}) = \langle H^2\rangle - \langle H\rangle^2) \tag{3.12}$$

If Ψ is the eigenstate of H then $\langle H^2\rangle = \langle H\rangle^2) = E_0^2$, that is the statistical fluctuations completely vanish. Thus the closer we get to the ground state the less fluctuations we have. Thus confirms us reaching the ground state energy.

Chapter 4

Application to Different Models

we are here going to discuss about some of the successful application of this technique in solving the following models in many-body theory:

- The Transverse-field Ising(TFI) model.
- Anti-ferromagnetic Heisenberg(AFH) model.
- Hubbard model.

4.1 The Transverse-field Ising model

The Hamiltonian

The Hamiltonian of 1D transverse field Ising model is:

$$H_{TFI} = -h \sum_i \sigma_i^x - \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z \tag{4.1}$$

with the Pauli matrices,

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \quad \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

where h is the applied magnetic field.

The Trail Wave Function

Our trail wave function is:

$$\Psi_M(\mathcal{S}) = e^{\sum_i^N a_j \sigma_j^z \prod_j^M 2\cosh(\theta_j)} \tag{4.2}$$

where,

$$\theta_j(\mathcal{S}) = b_j \sum_i^N W_{ij} h_j \sigma_i^2 \tag{4.3}$$

where $\{W_{ij}, a_j, b_j\}$ are parameters to be learned. In the graphical form the network looks, as shown in the figure 4.1.

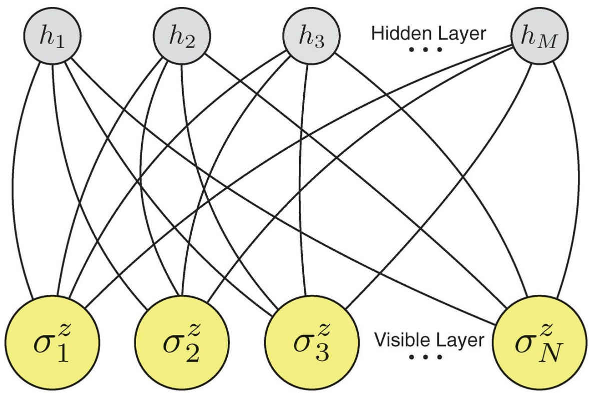


Figure 4.1: Restricted Boltzmann Machine

Energy

The local energy is calculated as

$$E_{\text{loc}} = \frac{\langle \mathcal{S} | H | \Psi_M \rangle}{\Psi_M(\mathcal{S})} \tag{4.4}$$

using

$$|\Psi_M\rangle = \sum_{s'} \Psi_M(s') |s'\rangle \quad \langle s, s' \rangle = \delta_{ss'} \tag{4.5}$$

The final expression of energy after the Hamiltonian operated on the state is

$$E_{TFI} = -h \sum_i \frac{\Psi_M(\mathcal{S}')}{\Psi_M(\mathcal{S})} - \sum_{\langle i,j \rangle} s_i s_j \tag{4.6}$$

where s_i, s_j are the spin of the particle at position i, j respectively. s will take either $+1$ or -1.

Gradient's for Optimization

The Gradient's of energy, for optimization and updating the weights $\mathcal{W} = (a, b, W)$ is calculated from energy function as

$$\nabla_{\mathcal{W}} \langle E \rangle = \nabla_{\mathcal{W}} \sum_s E_{\text{loc}} P(s) = \sum_s P(s) \nabla_{\mathcal{W}} E_{\text{loc}} + E_{\text{loc}} \nabla_{\mathcal{W}} P(s) \tag{4.7}$$

Here we are only keeping the track of first term The Gradient's of energy using different weights are

$$\partial a_j E_{loc}(\mathcal{S}) = -h \frac{\Psi_M(\mathcal{S}')}{\Psi_M(\mathcal{S})} \left[2 s_j^z \right] \tag{4.8}$$

$$\partial b_j E_{loc}(\mathcal{S}) = -h \sum_i \frac{\Psi_M(\mathcal{S}')}{\Psi_M(\mathcal{S})} \Big[\tanh\theta_j(\mathcal{S}') - \tanh\theta_j(\mathcal{S}) \Big] \tag{4.9}$$

$$\partial W_{ij} E_{loc}(\mathcal{S}) = -h \sum_i \frac{\Psi_M(\mathcal{S}')}{\Psi_M(\mathcal{S})} \Big[\theta_i^z \tanh\theta_j(\mathcal{S}') - \theta_i^z \tanh\theta_j(\mathcal{S}) \Big] \tag{4.10}$$

Sampling

The probability of acceptance of new configuration is:

$$P = \frac{\Psi(\mathcal{S}')\Psi^*(\mathcal{S}')}{\Psi(\mathcal{S})\Psi^*(\mathcal{S})} \tag{4.11}$$

if P is grater that some non-zero value the new state $\mathcal{S}'$ is accepted.

4.1.1 Result

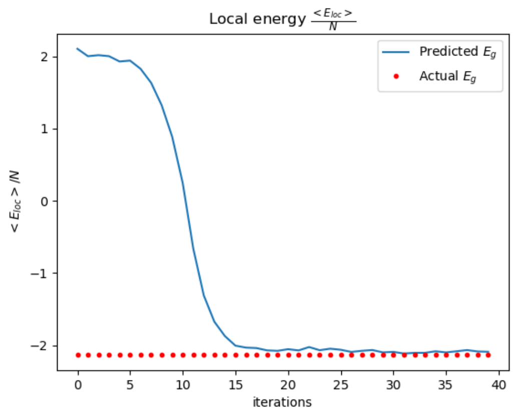


Figure 4.2: Energy Convergence to the Ground state energy(TFI) for a system of 4 spins($N = 4$) and in external magnetic field ($h = 2$). Actual ground state energy value which is used to compare, is calculated using exact diagonalization.

4.2 Anti-Ferromagnetic Heisenberg Model

The Hamiltonian

The Hamiltonian of 1D anti-ferromagnetic Heisenberg model is:

$$H_{AFH} = \sum_{\langle i,j \rangle} \sigma_i^x \sigma_j^x + \sigma_i^y \sigma_j^y + \sigma_i^z \sigma_j^z \tag{4.12}$$

with the Pauli matrices,

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \quad \sigma^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

The Trail Wave Function

Our trail wave function is same as in the Transverse-field Ising model:

$$\Psi_M(\mathcal{S}) = e^{\sum_i^N a_j \sigma_j^z \prod_j^M 2\cosh(\theta_j)} \tag{4.13}$$

where,

$$\theta_j(\mathcal{S}) = b_j \sum_i^N W_{ij} h_j \sigma_i^2 \tag{4.14}$$

where W_{ij}, a_j, b_j are parameters to be learned.

Energy

The local energy is calculated as

$$E_{\text{loc}} = \frac{\langle \mathcal{S} | H | \Psi_M \rangle}{\Psi_M(\mathcal{S})} \tag{4.15}$$

using

$$|\Psi_M\rangle = \sum_{s'} \Psi_M(s') |s'\rangle \quad \langle s, s' \rangle = \delta_{ss'} \tag{4.16}$$

same as in the above model. The final expression of energy after the Hamiltonian operated on the state is

$$E_{AFH} = \sum_{\langle i,j \rangle} s_i s_j + (1 - s_i s_j) \frac{\Psi_M(\mathcal{S}')}{\Psi_M(\mathcal{S})} \tag{4.17}$$

where s_i, s_j are the spin of the particle at position i, j respectively. s will take either $+1$ or -1.

Gradient's for Optimization

The Gradient's for optimization of energy by updating the weights $\mathcal{W} = (a, b, W)$ is calculated from energy function as

$$\nabla_{\mathcal{W}} \langle E \rangle = \nabla_{\mathcal{W}} \sum_s E_{\text{loc}} P(s) = \sum_s P(s) \nabla_{\mathcal{W}} E_{\text{loc}} + E_{\text{loc}} \nabla_{\mathcal{W}} P(s) \tag{4.18}$$

again here we are only keeping the track of first term.
The Gradient's of energy using different weights are

$$\partial a_j E_{loc}(\mathcal{S}) = \frac{\Psi_M(\mathcal{S}')}{\Psi_M(\mathcal{S})}\Big[-2s^z_j(1-s_is_j)\Big] \tag{4.19}$$

$$\partial b_j E_{loc}(\mathcal{S}) = \sum_i \frac{\Psi_M(\mathcal{S}')}{\Psi_M(\mathcal{S})}\Big[\big(\tanh\theta_j(\mathcal{S}') - \tanh\theta_j(\mathcal{S})\big)\big(1-s_is_j\big)\Big] \tag{4.20}$$

$$\partial W_{ij} E_{loc}(\mathcal{S}) = \sum_i \frac{\Psi_M(\mathcal{S}')}{\Psi_M(\mathcal{S})}\Big[\big(\theta^z_i\tanh\theta_j(\mathcal{S}') - \theta^z_i\tanh\theta_j(\mathcal{S})\big)\big(1-s_is_j\big)\Big] \tag{4.21}$$

Sampling

The probability of acceptance of new configuration is:

$$P = \frac{\Psi(\mathcal{S}')\Psi^*(\mathcal{S}')}{\Psi(\mathcal{S})\Psi^*(\mathcal{S})} \tag{4.22}$$

if P is grater that some non-zero value the new state $\mathcal{S}'$ is accepted.

4.2.1 Result

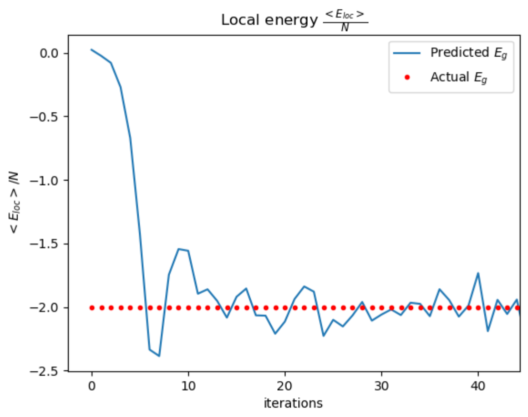


Figure 4.3: Energy Convergence to the Ground state energy(AFH)for a system of 4 spins($N = 4$). Actual ground state energy value which is used to compare, is calculated using exact diagonalization

4.3 Hubbard Model

The Hamiltonian for Hubbard Model is:

$$H = -t\sum_{<ij>,\sigma} a^\dagger_{i,\sigma}a_{j,\sigma} + \frac{U}{2}\sum_{i,\sigma} n_{i,\sigma}n_{i,-\sigma} \tag{4.23}$$

The Trail Wave Function

Our trail wave function is:

$$\Psi_M(\mathbf{n}) = \exp(u_1^{(2)}(\mathbf{n}) + iu_2^{(2)}(\mathbf{n})) \tag{4.24}$$

where,

$$u_m^2 = \sum_{k=1}^{N_H} W_{mk}^{(2)} \tanh u_k^1(\mathbf{n}) + h_m^{(2)} \tag{4.25}$$

$$u_k^1(\mathbf{n}) = \sum_{j=1}^{M} W_{kj}^{(1)} n_j + h_k^{(1)} \tag{4.26}$$

where N_H is the number of units in the hidden layer, and $m = 1, 2$. The weights $W_{mk}^{(2)}$ and $W_{kj}^{(1)}$ and the biases $h_k^{(1)}$ and $h_m^{(2)}$are real. n_j is number of particles at a sight. $n_j \in \{0, 1, 2\}$.

Energy

$$E_{\text{loc}} = \frac{\langle \mathbf{n} | H | \Psi_M \rangle}{\Psi_M(\mathbf{n})} \tag{4.27}$$

using

$$|\Psi_M\rangle = \sum_{n'} \Psi_M(n') |n'\rangle \quad \langle n, n' \rangle = \delta_{nn'} \tag{4.28}$$

The final expression of energy after the Hamiltonian operated on the state is

$$E_{od} = -t \frac{\Psi_M(\mathbf{n'})}{\Psi_M(\mathbf{n})} \tag{4.29}$$

$$E_d = \frac{U}{2} \tag{4.30}$$

where E_{od} for off-diagonal elements and E_d for diagonal elements The analytical expression for the ground state is:

$$E_g = \frac{U - \sqrt{U^2 + 16t^2}}{2} \tag{4.31}$$

Gradient's for Optimization

The Gradient's for optimization of energy by updating the weights $\mathcal{W} = \{h, W\}$ is calculated from energy function as

$$\nabla_{\mathcal{W}} \langle E \rangle = \nabla_{\mathcal{W}} \sum_s E_{\text{loc}} P(s) = \sum_s P(s) \nabla_{\mathcal{W}} E_{\text{loc}} + E_{\text{loc}} \nabla_{\mathcal{W}} P(s) \tag{4.32}$$

Here we are only keeping the track of first term.
The Gradient's of energy using different weights are

$$\partial h_m^{(2)} E_{loc}(\mathbf{n}) = 0 \tag{4.33}$$

$$\partial h_k^{(1)} E_{loc}(\mathbf{n}) = -t\frac{\Psi_M(\mathbf{n}')}{\Psi_M(\mathbf{n})}\left((\tanh^2 u_k(\mathbf{n}') - \tanh^2 u_k(\mathbf{n}))(W_{1k}^{(2)} + iW_{2k}^{(2)})\right) \tag{4.34}$$

$$\partial W_{1k}^{(2)} E_{loc}(\mathbf{n}) = -t\frac{\Psi_M(\mathbf{n}')}{\Psi_M(\mathbf{n})}\left(\tanh u_k^{(1)}(\mathbf{n}) - \tanh u_k^{(1)}(\mathbf{n}')\right) \tag{4.35}$$

$$\partial W_{2k}^{(2)} E_{loc}(\mathbf{n}) = -it\frac{\Psi_M(\mathbf{n}')}{\Psi_M(\mathbf{n})}\left(\tanh u_k^{(1)}(\mathbf{n}) - \tanh u_k^{(1)}(\mathbf{n}')\right) \tag{4.36}$$

$$\partial W_{kj}^{(1)} E_{loc}(\mathbf{n}) = t\frac{\Psi_M(\mathbf{n}')}{\Psi_M(\mathbf{n})}\Big(\big((1-\tanh^2 u_k(\mathbf{n}'))n_j' - (1-\tanh^2 u_k(\mathbf{n}))n_j\big)(W_{1k}^{(2)} + iW_{2k}^{(2)})\Big) \tag{4.37}$$

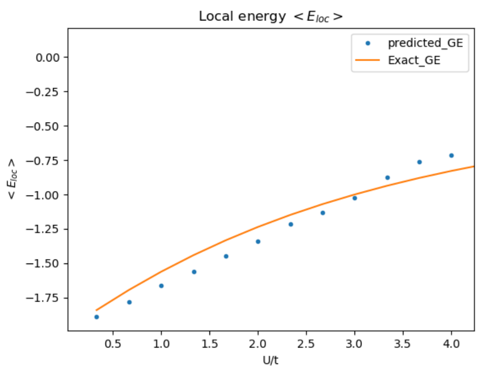
Sampling

The probability of acceptance of new configuration is:

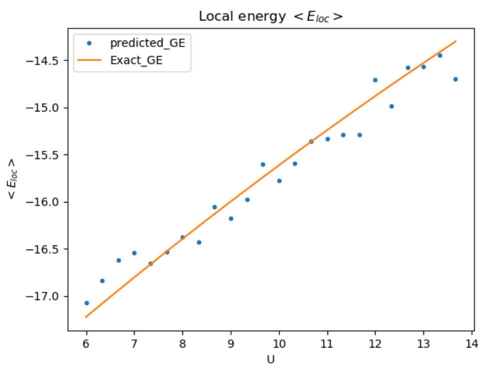
$$P = \frac{\Psi(n')\Psi^*(n')}{\Psi(n)\Psi^*(n)} \tag{4.38}$$

if P is grater that some non-zero value the new state n' is accepted.

4.3.1 Result



(a) Predicted and actual Ground state energy for different values of $\frac{U}{t}$ for $t = 1$.



(b) Predicted and actual Ground state energy for different values of U around $t = 10$.

Figure 4.4: Ground state Energy's of Hubbard Model for different values of t, the hopping parameter for system of 2 particles $(N = 2)$. Actual ground state energy value which is used to compare, is calculated analytically.

From the above results(fig.4.4) we have seen that the program we have written predicting the energy values with acceptable error only for comparable values of U and t. That is for

$\frac{U}{t} \leq 4$. If U and t values are not comparable then the predicted value deviated largely from the actual ground state energy values of the system. This may be because of the limitations

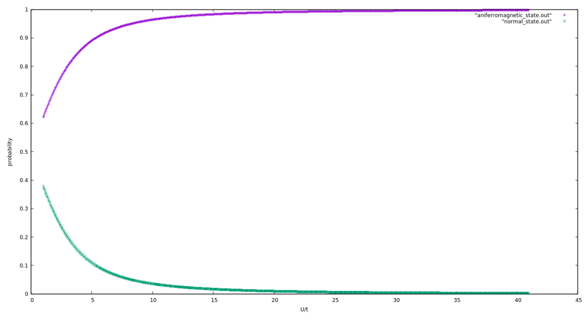


Figure 4.5: Probability of states

in sampling technique that we are using here, which is working well only when all the basis states has same probability.This happens when U and t are comparable (see.fig. 4.5). with improved sampling techniques this method can become capable of predicting ground state energy in the whole parametric space.

Chapter 5

Conclusion

we have shown that the approximate ground state energy can be obtained by a simple optimization of the network parameters. The results for one-dimensional systems are in good agreement with those obtained by exact diagonalization, even for small networks, which implies that the information of many-body quantum state features are efficiently stored in the artificial neural networks.

we found that even after a clever wave function is chosen the sampling techniques plays an important role in convergence to the ground state energy.

Scope for Further Research

Thus with better sampling techniques this method can become capable to calculate ground state energy for large system sizes and has the scope of application of this method to the most challenging questions like interacting fermion models by using more advanced deep neural network architectures.

Chapter A

Code

```
#random initialization of parameters
import numpy as np

def random_complex(size):
    a = (np.random.random(size) - .5) * 10e-2
    b = (np.random.random(size) - .5) * 10e-2
    return a + 1j*b

N = 4
alpha = 2
M = alpha * N
hfield = 2
a = random_complex(N)
b = random_complex(M)
W = random_complex((N,M))
state = np.random.randint(2, size=N)
state[state == 0] = -1
state_i = list(range(N))

# defining wave function

def effective_angles(state):
    return b+np.inner(np.transpose(W),state)

def Psi_M(state,a,b,W):
    return np.exp(np.inner(a,state)) * np.prod(2*np.cosh(effective_angles(state)))

# Calculation of local energy
```

```
def E_loc(state):
    E = 0
    # \sigma^z part
    for i in state_i:
        if i == N-1:
            E-=(state[i]*state[0])
        else:
            E-=(state[i]*state[i+1])

    # \sigma^x part
    Psi_M_s = Psi_M(state,a,b,W)
    for i in state_i:
        state[i] *= -1
        E -= -hfield*Psi_M(state,a,b,W)/Psi_M_s
        state[i] *= -1 # flip back

    return E/N

# Sampling
def step():
    # Choose random sites to be flipped
    sites = np.random.choice(state_i,1)
    Psi_M_before = Psi_M(state,a,b,W)
    for i in sites:
        state[i] *= -1 # flip
    Psi_M_after = Psi_M(state,a,b,W)
    acceptance = np.real(Psi_M_after*np.conj(Psi_M_after)/(Psi_M_before*np.conj(Ps

    if acceptance < np.random.uniform():
        for i in sites:
            state[i] *= -1 # flip back
        return 1 # return 1 to count # of rejections
    else:
        return 0

# Main Program
# Optimization and upgrading of parameters
block_E = []
for block_i in range(40):
```

```
state = np.random.randint(2, size=N)
state[state == 0] = -1
for k in range(10000):
    step()
iterations = 20000
rejected = 0
array_E_loc = []

array_a_d = []
array_b_d = []
array_w_d = []

for k in range(iterations):
    rejected += step()

    if k % 100 == 0:
        Psi_M_s = Psi_M(state,a,b,W)

        # Derivative a
        a_deriv = np.zeros(N, dtype=np.complex_)
        for i in range(N):
            state[i] *= -1 # flip
            a_deriv[i] = -hfield*Psi_M(state,a,b,W)/Psi_M_s*2.*state[i]
            state[i] *= -1 # flip back

        # Derivative W
        dW = np.zeros((N,M),dtype=np.complex_)
        for w_i in range(N):
            for w_j in range(M):
                dw_sum = 0
                before_flip = np.tanh(effective_angles(state))
                for i in range(N):
                    state[i] *= -1 # flip
                    dw_sum += Psi_M(state,a,b,W)/Psi_M_s*(
                        -state[i]*np.tanh(effective_angles(state)[w_j])-state[
                    state[i] *= -1 # flip back
                dw_sum *= hfield
                dW[w_i,w_j] = dw_sum
            # Derivative b
            b_deriv = np.zeros(M, dtype=np.complex_)
            for b_j in range(M):
                tanh_before_flip = np.tanh(effective_angles(state))
                db_sum = 0
                for i in range(N):
                    state[i] *= -1 # flip
                    db_sum += Psi_M(state,a,b,W)/Psi_M_s*(
                        np.tanh(effective_angles(state)[b_j])-tanh_before_flip[b_j
                    state[i] *= -1 # flip back
                b_deriv[b_j] = -hfield * db_sum



            array_a_d.append(a_deriv)
            array_b_d.append(b_deriv)
            array_w_d.append(dW)
            array_E_loc.append(np.real(E_loc(state)))

    print('%d. E_loc=%.4f std=%.4f (%.1f %% moves rejected)' % (block_i+1,
        np.mean(array_E_loc),np.std(array_E_loc)/(np.sqrt(len(array_E_loc))), 100.
    print(state)
    block_E.append(np.mean(array_E_loc))
    mean_da = np.mean(np.array(array_a_d),axis=0)
    mean_db = np.mean(np.array(array_b_d),axis=0)
    mean_dw = np.mean(np.array(array_w_d),axis=0)
    #print(mean_da,mean_db,mean_dw)
    a = a - .05 * mean_da
    b = b - .05 * mean_db
    W = W - .05 * mean_dw
    #print(a,b,W)

#Exact Diagonalization
from itertools import product

basis = list(product([-1,1],repeat=N))

print('Generated %d basis functions' % (len(basis)))
#print(len(basis_functions))
```

```
#list(permutations([0,1,0,0]))
H = np.zeros((2**N,2**N))
for H_i in range(2**N):
    for H_j in range(2**N):
        H_sum = 0
        for i in range(N):
            if H_i == H_j:
                if i == N-1:
                    H_sum -= basis[H_j][i]*basis[H_j][0]
                else:
                    H_sum -= basis[H_j][i]*basis[H_j][i+1]

            sj = list(basis[H_j])
            sj[i] *= -1
            if H_i == basis.index(tuple(sj)):
                H_sum -= hfield

        H[H_i,H_j] = H_sum

print('Ground state energy:', np.min(np.linalg.eigvals(H))/N)
G_E = np.ones(40)
G_E = G_E*np.min(np.linalg.eigvals(H))/N

#Plots
import matplotlib.pyplot as plt
plt.plot(block_E,label = 'Predicted $E_{g}$')
#plt.legend()
plt.title(r'Local energy $\frac{<E_{loc}>}{N}$ ')
plt.ylabel(r'$<E_{loc}>/N$')
plt.xlabel('iterations')
plt.plot(G_E,'r.',label = 'Actual $E_{g}$')
plt.legend()
plt.show()
```

For AFH model and Hubbard model the method remain same except the little changes in optimization and sampling which can be obtained from the calculations specified in Chapter-4.

References

[1] P. A. M. Dirac, Proc. Math. Phys. Eng. Sci. 1929, 123, 714.

[2] Rupp, Matthias. ”Machine learning for quantum mechanics in a nutshell”.Vol 115.International Journal of Quantum Chemistry. DIO:10.1002/qua.24954

[3] Ryczko, Kevin and Strubbe, David and Tamblyn, Isaac. (2018). ”Deep Learning and Density Functional Theory”.

[4] Mills, Kyle and Spanner, Michael and Tamblyn, Isaac.”Deep learning and the Schrodinger equation”.Phys. Rev. A. American Physical Society. Vol. 96 No. 4 (2017): DOI 10.1103/PhysRevA.96.042113

[5] Borysov, Stanislav S., B. M. Olsthoorn, Mustafa Emre Gedik, R. Matthias Geilhufe and Alexander V Balatsky. “Online search tool for graphical patterns in electronic band structures.” (2018).

[6] Hu, Wenjian and Singh, Rajiv R. P. and Scalettar, Richard T.”Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination”.Phys. Rev. E. American Physical Society. Vol. 95 No. 6 (2017):DIO 10.1103/PhysRevE.95.062122

[7] Carleo, Giuseppe and Troyer, Matthias. ”Solving the Quantum Many-Body Problem with Artificial Neural Networks”. Science. Vol. 355 (2017):DOI 10.1126/science.aag2302

[8] Saito ,Hiroki.”Solving the Bose–Hubbard Model with Machine Learning”.Journal of the Physical Society of Japan. Vol. 86 (2017):DIO 10.7566/JPSJ.86.093001

[9] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*.O’Reilly Media, Inc.(2016)ISBN: 9781449369880

[10] Bryant, Donald, ”Analysis Of Kolmogorov’s Superposition Theorem And Its Implementation In Applications With Low And High Dimensional Data.” (2008). Electronic Theses and Dissertations. 3689

[11] Giuseppe Carleo,” Machine learning methods for many body physics”. Lectures for the Advanced School on Quantum Science and Quantum technology (September 8-9, 2017). ICTP, Trieste, Italy

[12] Jin-Guo Liu, Shuo-Hui Li, and Lei Wang,”Lecture Note on Deep Learningand Quantum Many-BodyComputation”.Institute of Physics, Chinese Academy of Sciences, Beijing

[0].