UNIVERSITY OF SURREY

Solving the Ground State of a Quantum Particle in a Box using Artificial Neural Networks

by

Anthony Hills

A dissertation submitted in partial fulfillment for the degree of Physics (BSc Hons)

in the

Faculty of Engineering and Physical Sciences

Department of Physics

May 2018

Declaration of Authorship

I, ANTHONY HILLS, declare that this thesis titled, 'SOLVING THE GROUND STATE OF A QUANTUM PARTICLE IN A BOX USING ARTIFICIAL NEURAL NETWORKS' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Anthony Hills

Date: (21.05.18)

"Physics is becoming much too hard for physicists."

David Hilbert

UNIVERSITY OF SURREY

Abstract

Faculty of Engineering and Physical Sciences
Department of Physics

BSc Physics (Hons)

by Anthony Hills

The recent work of M. Troyer and G. Carleo have demonstrated the flexibility and high precision of using artificial neural networks to represent quantum states [1]. Although work has been done in solving for the ground-state of quantum wavefunctions that are functions of spin [1–3] using machine learning, little to none has been done in solving for wavefunctions that are functions of continuous input - namely spatial coordinates. In this regard, we make efforts to expand upon the work by M. Troyer and G. Carleo, by developing a method to solve for the ground state of a 1D quantum particle in a box using artificial neural networks.

This was an extremely ambitious project for such a short time frame. Current results demonstrate that we converge to non-physical ground-state wavefunctions that illustrate that more research needs to be done to improve upon our method and its implementation. Regardless, we discuss the method we have so far developed and made its implementation publicly available on GitHub [4].

Acknowledgements

I would like to thank my supervisor Dr Carlo Barbieri for supporting me with this project. In addition I would like to express my deepest respect to Carleo Giuseppe and Matthias Troyer for the work that they have done in making their method and code publicly available for solving the quantum many body-problem with artificial neural networks [5]. This has been extremely helpful in making efforts to expand upon their work.

Contents

Chapter 1

Introduction

At the heart of our theories for quantum mechanics, which describes how the Universe behaves at its most fundamental level, lies the wavefunction ψ. Possibly the most important concept, and hardest to understand - encoded in the value of ψ is all the information on a quantum state.

Built into the theory of quantum mechanics, the wavefunction itself is unmeasurable, and unobservable - and therefore remains one of the most elusive and difficult to grasp theoretical objects in modern physics.

At the microscopic scale, the world becomes fuzzy. Objects such as electrons and protons, that we classically think of as particles have challenged everything we know about our classical understanding of Physics. At the quantum scale, particles no longer have a well-defined position in space - and it can not be said in a very fundamental way exactly where a particle is. At these scales, particles have a non-zero probability of existing anywhere within infinite space. Properties such as position and momentum do not have defined values until they are observed. The probability of each possible observation is determined by the value of ψ. Quantum mechanics has been known to predict counterintuitive, and bizarre phenomena: such as particles in two places at once, particles tunneling through solid walls, and so on. For the above reasons, scientists are constantly seeking new and efficient ways to solve the Schrodinger equation.

Recent work by G. Carleo and M. Troyer has provided evidence that the wavefunction of a quantum system of spin particles can be represented extremely efficiently using an artificial neural network known as the Restricted Boltzmann Machine (RBM) [1]. Since research has been done to solve for wavefunctions that are functions of only spin, there currently exists a gap in the literature where little to no research has been done to solve for the wavefunction of quantum systems that are a functions of continuous input, such

as the spatial coordinates of a quantum particle, using machine learning. We therefore see it as of fundamental importance to make efforts to expand upon these methods to encompass a wider variety of quantum systems that are no longer a function of just binary spin, but instead are functions of continuous inputs.

The scope of this project is then to make a very first step to implement the above ideas based on a type of artificial neural network representing quantum systems described by continuous spatial coordinates - namely the 1D quantum particle in a box.

Chapter 2

Machine Learning Prerequisites

2.1 Why Neural Networks are so Powerful and Versatile

The recent innovations in machine learning have lead to artificial neural networks dominating the field of artificial intelligence. With such wide ranging applications, from self-driving cars [6] to generating music [7], neural networks have proven themselves not just powerful, but surprisingly versatile.

Machine learning has become increasingly pervasive in the technologies that underpin our modern society today. From Facebooks image recognition software, to Google Translate - all these technologies are based largely on the recent advances in machine learning. Most of these are becoming increasingly powerful in recent years mostly thanks to deep learning, one of the cutting edge advances in the field.

Before going into the details of machine learning, and how it can be applied it to perform scientific research - we discuss the large motivating forces behind the rapid progress in the field and how it fits within its larger mathematical context. We thus start from the very beginning; to the far-reaching problems of David Hilbert: arguably one of the earliest pioneers in the field of machine learning. Well known in the Physics community for his Hilbert space, he is celebrated for having published 23 of the hardest problems during his time [8]. At the beginning of the last century, he published these problems and among them was the 13th problem which goes as follows.

Consider finding the solution to a high order polynomial, such as of the 9th order:

$$x^9 + ax^6 + bx^3 + cx + 1 = 0 \tag{2.1}$$

It is well known from the works of Galois and others that for any polynomial which is of an order higher than 3, a solution that is in a simple algebraic form cannot be found [9]. The question that David Hilbert asked was whether the solution of this high dimensional function $x(a, b, c)$ could be expressed as a finite composition of much simpler functions.

To answer this question, we consider the work of Kolmogorov-Arnold, which is a large motivating hope driving the ongoing advancements in the field of machine learning. The theorem goes as follows.

Consider the scenario in which we have a high-dimensional function, which we assume to be continuous, and that depends on many variables. Through the work of Kolmogorov and Arnold, and the refinements that were developed by Sprechner in the 1960s [10], it can be shown that this function can be written as a finite composition of other functions:

$$F(x_1, x_2, \ldots, x_n) = \sum_{q=0}^{2n} \Phi(\sum_{p=1}^{n} \lambda_p \phi(x_p + \eta q) + q) \tag{2.2}$$

On the left hand side we have a complicated high-dimensional function of N variables - something that is typically hard to approximate.

However, this theorem states that very complicated high-dimensional functions can be easily approximated with just the composition of two, simpler, 1-dimensional functions. These functions are $\Phi(x)$ and $0 \leq \phi(x) \leq 1$, which are continuous, unknown, 1-dimensional functions that are to be determined when solving this problem. In addition, we have just a few parameters λ_q and η that are also to be determined. This theorem is particularly powerful, as it simplifies the complexity of high-dimensional functions.

This insight forms the basis for how machine learning applications work as in general, the problems they are trying to solve can be viewed as high-dimensional functions. For example, consider the scenario in which we have a picture that we want a machine to identify. The picture has been digitized, where the value of each pixel can be taken as $0 \leq X_i \leq 1$, and corresponds to the intensity of the light on that particular pixel. With this data, we want to devise a machine learning algorithm that outputs a probability that there is an object in the image that you know:

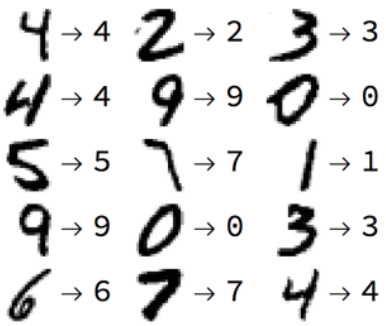

FIGURE 2.1: Handwritten digit recognition using machine learning [11]

For another example, consider the case with Google translate in which you take a high-dimensional input string in for example Mandarin, that outputs the same string in for English.

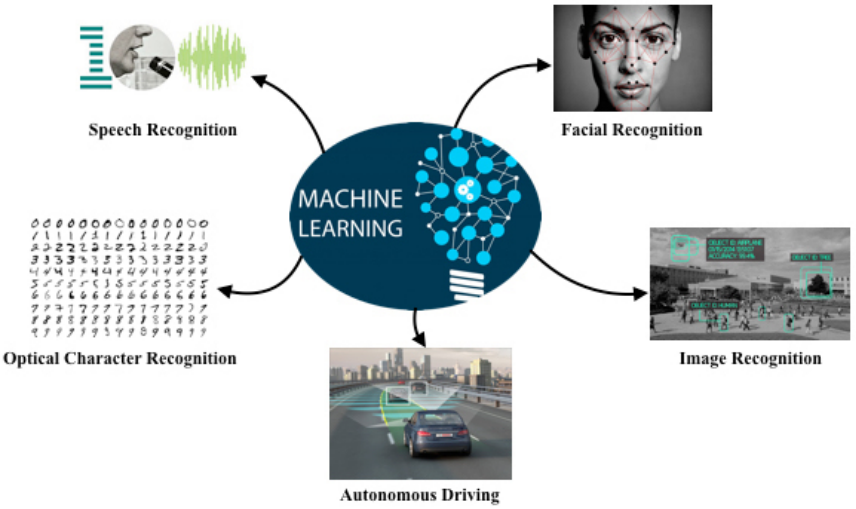

FIGURE 2.2: Example use cases for machine learning [12]

The goal of machine learning is to find the best high dimensional functions that realize some specific, likely complicated task - from driving a car to in our case solving Schrodingers equation, which will be discussed in much greater detail later.

However, in order to do so, we need some additional tools. It is not only enough to know this important theorem by Kolmogorov-Arnold (2.2) but it indeed provides an important hope. To reiterate this theorem, no matter how complicated a function is (e.g. driving a car), it can still be imagined that a close approximation of this function can be found in terms of much simpler 1-dimensional functions.

The idea of machine learning therefore is to find efficient representations of these unknown, complicated, high-dimensional functions in terms of much simpler, tractable 1-dimensional functions.

In practice, to express these high-dimensional functions in a tractactable form, machine learning practitioners commonly express them using machines that are known as Artificial Neural Networks (ANN).

2.2 Artificial Neural Networks

Artificial Neural Networks, simply put, are just another way of writing the theorems of Kolmogorov-Arnold, and are essentially high dimensional functions of some high dimensional input.

To illustrate this, the high-dimensional input $\vec{X}$ is a series of data points that are provided to the nodes (i.e. neurons) in the left-most, input layer of this diagram:

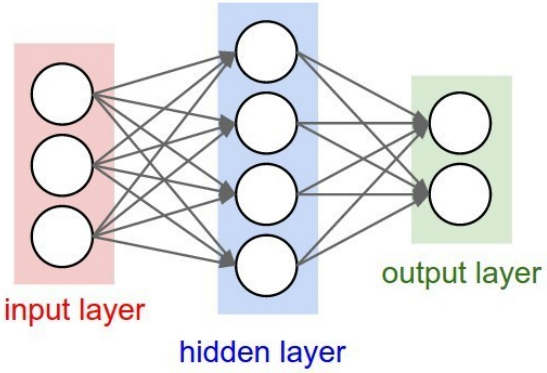

FIGURE 2.3: Artificial Neural Network [13]

A common feed-forward neural network [14] shown in figure 2.3 takes this input $\vec{X}$ and typically forms some linear combinations, transforming this input by passing it forward through initially randomly weighted connections in the structure of the network. By measuring the final output in the right most layer, we can measure a loss function: the error between the output of the neural network and the true output of what is desired (e.g. the correct classification of a person in an image). There exist many ways to minimize this error, but in practice the most common way to train this simple feed-forward neural network is known as backpropagation [15]. Simply put, after having done a forward pass, where we trial passing the input through the initial structure of the network until it has reached the output; we perform a backward pass, where we update the weighted connections between the nodes in the network accordingly. The idea is that,

just like in the human brain, where neurons that fire together, wire together [16], the more often that when two connected neurons produce a desired output - we keep their weighted connections accordingly - and weaken the strength of the connections between nodes that do not produce the desired output. We repeat this process of iteratively doing a forward pass, measuring the error, and then minimizing this error by doing a backward pass and updating the structure of the connections accordingly, until the error of the final output is within a desired accuracy.

The basic idea is that you start with an initial input (i.e. the intensity of pixels in an image), and these variables go through a series of layers where the information is transformed, and this eventually gives an output (i.e. a classification of the image provided).

The mathematical basis for this task to work is the theorem described by Kolmogorov-Arnold: the fact that we can approximate any high-dimensional function as a composition of simpler 1-dimensional functions.

By rephrasing this theorem using artificial neural networks, the full function can be expressed as a nested composition of non-linear, simpler, vector functions and the parameters of the neural network (w, and b).

$$F(\vec{x}; w, b) = \phi(A(B(C(\vec{x}))))$$

To illustrate this, consider the case of a the feed-forward neural network. Here, we take for example a linear combination of input, and transform it to define a new set of variables which define the first (hidden) layer of the network, as shown in figure 2.3:

$$y_j = \phi(\sum_i w_{ij} x_i + b_j) \tag{2.3}$$

Here ϕ is what are referred to as activation functions, which take inspiration from the biology of the brain, and often for example take the form of the logistic function:

$$\phi(x) = \frac{1}{1 + \exp(-x)} \tag{2.4}$$

We feed the variables from equation (2.3) into deeper layers:

$$z_k = \phi(\sum_j w'_{jk} y_j + b'_k) \tag{2.5}$$

until we reach the final layer, where (for example) we have a single output variable:

$$o = \phi(\sum_k w''_k z_k + b'') \tag{2.6}$$

Hence the full function of a neural network can be seen as a nested composition of non-linear vector functions:

$$F(\vec{x}; w, b) = \phi(A(B(C(\vec{x})))) \tag{2.7}$$

The role of machine learning, is almost always to learn the best set of parameters (in this case w_{ij} and b_j), of a machine to find compact approximations of these high-dimensional functions. There are many approaches in which we can converge to these best set of parameters, which in general lie in either three major different approaches.

2.3 Approaches to Learning

What we have discussed up to this point, are the motivations behind machine learning and a common type of machine: the artificial neural network. In this section, we discuss how we can train (i.e. teach) these machines to achieve certain goals.

In machine learning, there are three major approaches to teaching machines how to learn. These are *supervised*, *unsupervised*, and *reinforcement learning*.

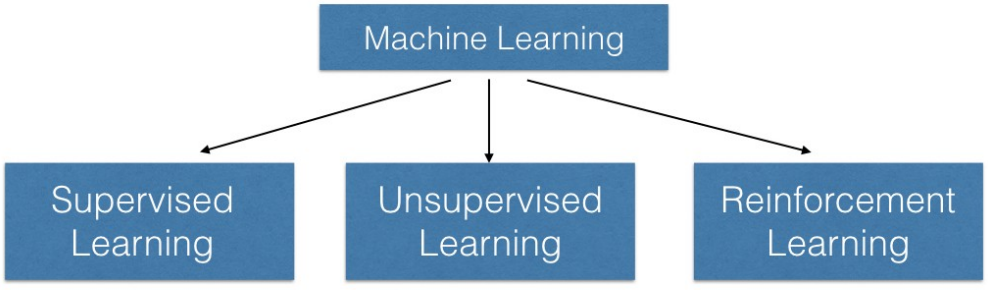

FIGURE 2.4: The three major approaches to teaching machines to learn [17]

In this project, we make use of the two latter approaches, unsupervised and reinforcement learning, to teach a model to solve for the ground state of a particle in a box, which will be discussed in greater detail later. Although some research has recently been done by

K. Mills et al. to solve for the ground state of a quantum particle in a box [18] using supervised learning, little to none has been done using unsupervised learning.

However, before going into detail about the method to solve for the ground state of a particle in a box using machine learning - it is important to understand the significance and key differences behind these classes of techniques that teach machines to learn.

Supervised learning involves supervising (i.e. teaching) a machine to be able to reproduce/ classify results by being shown many different examples of the correct, meticulously labeled results of what it is trying to understand.

For example, by showing a machine learning model a series of emails that have been labelled by a human as either spam or not spam, along with the features that compose the emails, we can teach the model to learn for itself a technique to understand whether new, unseen emails are either spam or not. This is an example of supervised learning, where the machine learning model is taught explicitly what its training data is (emails labeled as spam or not spam), and it learns using this labeled data to improve at a certain task (flagging unwanted emails as spam).

On the other hand, unsupervised learning is an entirely different approach to learning. More in line with what some may call true artificial intelligence, the idea is that the machine can be taught to learn to identify complex patterns and processes without a human providing guidance along away.

To illustrate the difference between supervised learning and unsupervised learning, consider the following example. An example of supervised learning would be to provide a machine learning model a series of labelled photographs with information of who exactly is in them. This data has been carefully labelled by humans, and this data is then fed into a machine to make its job easier: to separate new unseen images into unique albums for each person that was previously labeled, that contains images in which they now appear in.

To contrast this with an example of unsupervised learning, consider the case in which we have a series of photos of 10 different people in them. We dont have any information of which person appears in each photo, but regardless we want to devise a model that can learn to separate this dataset of unlabeled photos into 10 different piles, where each unique person appears.

The key difference is that without the need of a labelled training set, the machine is forced to learn on its own using unlabelled data - the type that is much more common in real world scenarios. Unsupervised learning commonly has more difficult algorithms

than supervised learning, since we know little to no information about the data or the outcomes that are to be expected.

Reinforcement learning on the other hand, is a type of learning method in which a machine makes decisions on what actions to take given a certain situation/ environment, so as to maximise a reward. In reinforcement learning, the machine is not explicitly told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them out by itself through trial and error. Some hallmark examples of reinforcement learning are the recent research projects by Google DeepMind, namely where they trained a machine to beat the highest ranked players in the world at the ancient, highly complex board game: Go [19]. Without human guidance or domain knowledge beyond game rules, the machine learned to master the game by iteratively playing the game against itself, learning to maximise its reward (winning the game) through trial and error.

The key difference therefore between these machine learning approaches, is the type of data we provide the machine to learn with. Supervised learning directly teaches the machine with labeled data, whereas on the other hand unsupervised and reinforcement learning forces the machine to learn using unlabeled data.

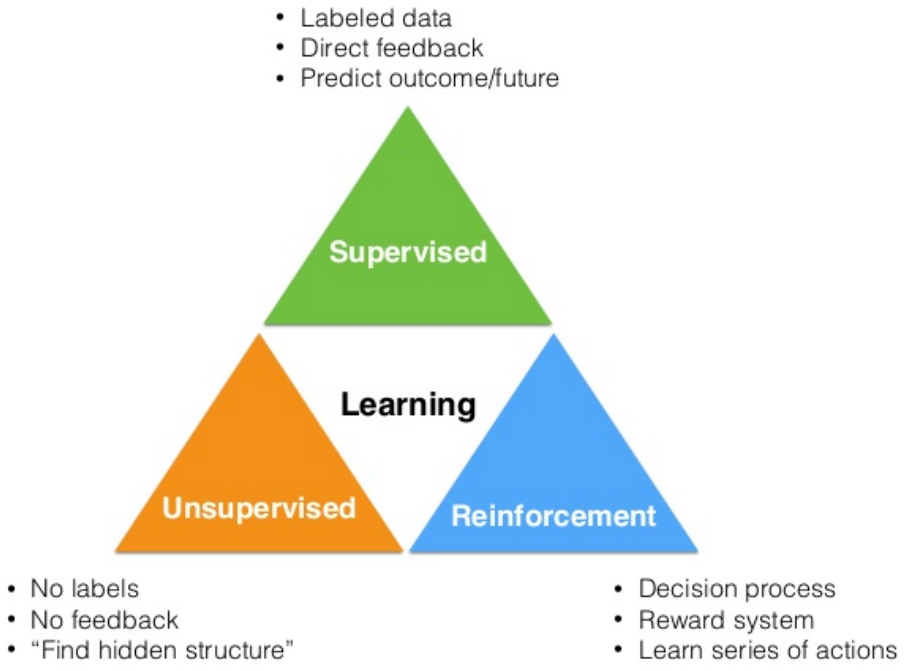

FIGURE 2.5: The distinguishing features between the three major approaches in teaching machines to learn [20].

2.4 Our Approach to Learning

For the problem we are trying to solve, finding the ground state for a particle in a box, the approach we have chosen to take is a combination of unsupervised learning with a reinforcement learning scheme. The machine we have chosen is a type of artificial neural network, known as the Restricted Boltzmann Machine, and the reinforcement learning scheme is based on the optimization objective to return the minimum energy measured for the trial wavefunctions used in quantum Variational Monte Carlo (VMC) simulations. The physics and machine learning of this will be discussed in greater detail later, but provides context for the following sections.

2.5 Kullback-Leibler Divergence

In the field of unsupervised learning, a common loss function that we seek to minimize is what is referred to as the Kullback-Leibler (KL) divergence. The KL divergence is a measure of the difference between two probability distributions, which is defined as:

$$D_{KL}(\Pi||F) = \sum_x \Pi(\vec{x}) \log \frac{\Pi(\vec{x})}{F(\vec{x};\vec{p})/Z(\vec{p})} \tag{2.8}$$

The key idea in this formula, is that the KL divergence attains a minimum when the two probability distributions, Π and F, are equal.

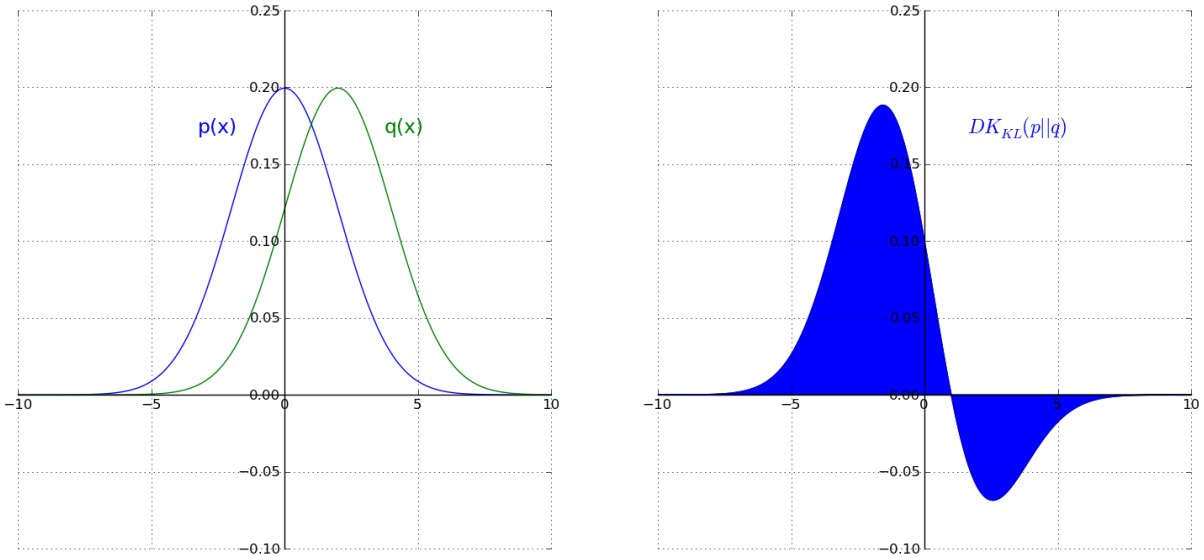

FIGURE 2.6: The KL divergence as the difference between two probability distributions [21].

The gradient of the KL divergence can be expressed exactly as the difference between two expectation values:

$$\partial_{p_k} D_{KL}(\Pi||F) = \sum_{\{\mathbf{x}\}} (-\partial_{p_k} \log F(\mathbf{x};\mathbf{p}) + \partial_{p_k} \log Z(\mathbf{p})) \tag{2.9}$$

$$= -\langle\langle \partial_{p_k} \log F(\mathbf{x};\mathbf{p})\rangle\rangle_{\Pi} + \frac{\sum_{\mathbf{x}} \partial_{p_k} \log F(\mathbf{x};\mathbf{p})}{Z(\mathbf{p})} \tag{2.10}$$

$$= -\langle\langle \partial_{p_k} \log F(\mathbf{x};\mathbf{p})\rangle\rangle_{\Pi} + \langle\langle \partial_{p_k} \log F(\mathbf{x};\mathbf{p})\rangle\rangle_{F}. \tag{2.11}$$

From a practical perspective, expectation values over Π can be easily computed using unlabeled data, without knowing explicitly the value of Π:

$$\langle\langle \partial_{p_k} \log F(\mathbf{x};\mathbf{p})\rangle\rangle \simeq \frac{1}{N_s} \sum_i \partial_{p_k} \log F(\mathbf{x_i};\mathbf{p}) \tag{2.12}$$

In the context of quantum mechanics, the wavefunction can be thought of as a probability distribution $|\psi|^2$. Later in this dissertation, we discuss a method in which we approximate this probability distribution $|\psi|^2$ using a neural network and improve upon this approximation by minimizing the KL divergence - learning an accurate approximation of the wavefunction. To do that, we measure the second expectation value in equation (2.11), using a procedure that generates samples from the neural network which will be described in section 2.8, and minimize it using an optimization algorithm known as Stochastic Gradient Descent.

2.6 Stochastic Gradient Descent

To minimize our loss function, the KL divergence, we make use of the optimization algorithm: stochastic gradient descent. This method borrows its roots from the classical Gradient Descent algorithm, a standard optimization algorithm in machine learning, which is an iterative approach in which we update the parameters of our machine (e.g. the strength of the connections W_{ij} between neurons in a neural network) to minimize a loss function (in this case, the KL divergence). In standard gradient descent, we update the parameters according to:

$$\mathbf{p}^{s+1} = \mathbf{p}^{(s)} - \eta G(\mathbf{p}^{(s)}) \tag{2.13}$$

where $G(\mathbf{p}^{(s)})$ is an approximation of the gradient of a function to be minimized, and η is a hyper-parameter known as the learning rate. Appropriate choice of the learning rate is important, such that we converge to a minimum efficiently, while not overshooting it. The general idea is that, a large learning rate allows the parameters in the machine to be updated more aggressively, descending towards the direction of a minimum with much larger jumps by following the slope of the curve of our loss function:

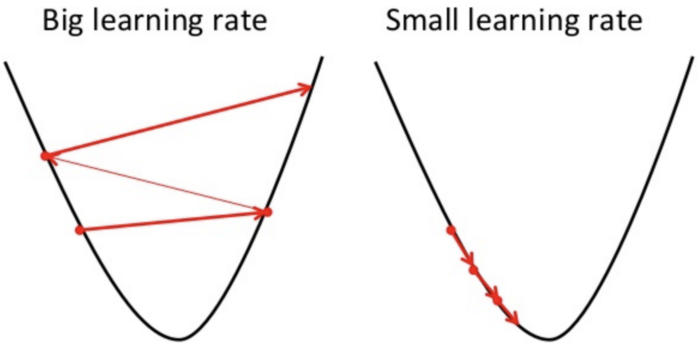

FIGURE 2.7: Converging to the minimum of a loss function using gradient descent by updating parameters according to equation (2.13) [22]

It should be noted that a learning rate that is too large will diverge from the global minimum, but a learning rate that is too small will take extremely long to converge, and is more likely to get stuck in local minima. Experimentation between choosing hyper-parameters such as the learning rate, and architecture of the neural network, is often needed, but there exist more sophisticated methods to determine the best possible combination of these such as grid-search and Bayesian optimization [23].

The standard Gradient Descent algorithm (GD) is typically not very efficient, but is fairly easy to understand. We start with a set of parameters, measure the gradient of our loss function, and update our parameters such that we descend towards the loss functions minimum.

Stochastic Gradient Descent (SGD) on the other hand, improves upon the setbacks of the standard gradient descent, offering a more efficient way of minimizing a loss function, while ensuring with greater probability that we converge to the global minimum - as it is less likely to get stuck in local minima.

The significant difference between the standard GD and the SGD is in how the parameter updates are computed. In standard GD, the G in equation (2.13) is just the standard gradient, i.e.:

$$G_{\mathrm{GD}}(p_1 \dots p_{N_p}) = \nabla_p \mathcal{L}(\mathbf{p}), \tag{2.14}$$

$$= \frac{1}{N_s} \sum_{i}^{N_s} \nabla_p |F(\mathbf{x_i}; W - y_i) \tag{2.15}$$

where $\mathcal{L}$ is our loss function to be minimized, and N_s is the size of our total dataset we are training our machine on.

In SGD, we instead use a *batch approximation*, where the sum runs on a much smaller subset of the dataset consisting of $N_b \ll N_s$ datapoints.

$$G_{\mathrm{SGD}}(p_1 \dots p_{N_p}) = \frac{1}{N_b} \sum_{i}^{N_b} \nabla_p |F(\mathbf{x_i}; \mathbf{p}) - y_i| \tag{2.16}$$

In the limit of large N_s, the true gradient can be thought of as an expectation value over the unknown probability distribution according to which the data $\vec{x}$ is distributed: $\Pi(\boldsymbol{x})$, such that:

$$G_{\mathrm{GD}}(p_1 \dots p_{N_p}) = \langle \langle \nabla_p |F(\mathbf{x}_i; boldp - y_i) \rangle \rangle_{\Pi} \tag{2.17}$$

If the data belonging to our batch is randomly drawn (with replacement) from the original dataset, then $G_{SGD}(\mathbf{p})$ can be interpreted as a noisy approximation of the true gradient.

If we assume that all the components of the gradient are subjected to the same amount of Gaussian noise of variance $\sigma \propto \frac{1}{\sqrt{N_b}}$:

$$G_k^{SGD}(p_1 \dots p_{N_p}) = \nabla_p \mathcal{L}(\mathbf{p}) + \mathrm{Normal}(\sigma) \tag{2.18}$$

We can then compare the standard GD parameter update rule in equation (2.13) with the discretized Langevin equation [24]:

$$p_k^{s+1} = p_k^s - \delta_t \partial_{p_k} \mathcal{L}(W) + \mathrm{Normal}(\sqrt{2\delta_t T}) \tag{2.19}$$

where δ_t is a small time step. This equation samples from the Boltzmann distribution:

$$\Pi_B(p_1 \dots p_M) = e^{-\frac{\mathcal{L}(\mathbf{p})}{T}} \tag{2.20}$$

which in the limit of $T \to 0$ converges to the minimum value (ground state) of the effective classical energy $\mathcal{L}(\mathbf{p})$. We therefore see that the variance of the gradient corresponds to the effective temperature, as:

$$\sigma^2 = var(\partial_{p_k}\mathcal{L}) = 2T/\delta_t\eta = \delta_t \tag{2.21}$$

Since we want to find the variational ground state, we require a scheme in which the temperature T is gradually decreased at each optimization step (i.e. $T_1 > T_2 > T_3 \dots$). The first thing we notice is that $\sigma^2 \simeq \frac{1}{N_s}$, decreases like the number of samples in the batch, therefore:

$$G_{\mathrm{GD}}(p_1 \dots p_{N_p}) = \nabla_p \mathcal{L}(\mathbf{p}), \tag{2.22}$$

$$\propto \frac{1}{N_s}\sum_{i}^{N_s} \nabla_p |F(\mathbf{x_i}; W - y_i) \tag{2.23}$$

A common way to reduce the temperature therefore is to reduce the learning rate η with the step number, or alternatively to increase the number of samples in the batch N_s with the iteration count (however this is used far less in comparison).

Having discussed a method that forms part of the solution to help minimize a loss function, the KL divergence, we can use it to learn the probability distributions behind sets of input.

In the next section we discuss how we can represent a wavefunction of a quantum system using a type of artificial neural network known as a Restricted Boltzmann Machine.

2.7 Restricted Boltzmann Machines

As the name suggests, Restricted Boltzmann Machines (RBMs) borrow their roots from Statistical Thermodynamics, and can be seen as a natural entrypoint to approach the field of machine learning coming from Physics.

In practice, RBMs can learn the probability distribution function F_{rbm} of a provided set of discrete, binary inputs $\vec{\sigma}$ - provided that an appropriate learning scheme is chosen. This is useful for when generating samples that distribute according to the probability distribution determined via the KullbackLeibler divergence discussed previously. The fact that we can have a machine that can learn the probability distributions behind sets

of input, can have interesting and far-ranging applications: from forging handwriting, to generating music of a specific genre and art-style unique to the artist of the input [7].

Mathematically, the idea of the RBM is to write a high-dimensional function, F_{rbm} , as the partition function of a classical object, of M hidden units $h_j = \pm 1$ connected to a high-dimensional input that must be binary valued $\sigma_i = \pm 1$:

$$F_{rbm}(\sigma_1, \sigma_2, \dots \sigma_N) = \sum_{\{h\}} \exp\Big[\sum_{ij} W_{ij}\sigma_i h_j + \sum_j h_j b_j + \sum_i \sigma_i a_i\Big] \tag{2.24}$$

In equation (2.24) above, we have the sum of an interaction matrix W_{ij} (i.e. the connection weights) which drives an interaction between the input variables σ_i and the hidden variables h_j in the network. In addition we have the sum over what are referred to as the bias terms: a_i for the input variables, and b_j for the hidden variables. In this function above, the parameters to be determined are the interaction matrix W_{ij} and the bias terms: a_i and b_j. Once these have been learned, the probability distribution F_{rbm} for any given input can be returned using equation (2.24).

In general we take a fixed set of M hidden variables, which take the values of $h_j = \pm 1$. Note that the hyper-parameter M can be tuned before the learning process begins, and in a sense corresponds to how smart the network is. One can imagine that the more artificial neurons M the network has, the more weighted connections W_{ij} there will be, and thus the more complex the network. The more complex the network, the more accurate approximations of the functions can be determined. Concretely, the results of G. Carleo and M. Troyer illustrate that the accuracy of a neural network increases with more artificial neurons M [1].

The machines are referred to as Restricted because there are no intra-layer connections, and only connections between different layers of hidden and visible neurons are allowed.

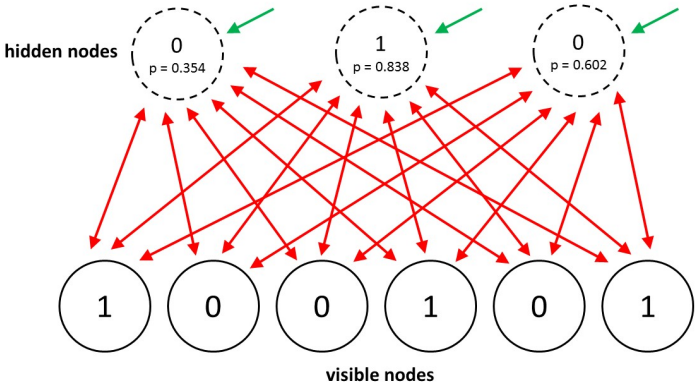

FIGURE 2.8: Restricted Boltzmann Machine [25]

In other words, we restrict the network such that there are no direct interactions between different input neurons σ_i and σ_j - and only connections between σi and h_j are allowed.

The key reason for restricting the connections is as follows. Since we do not have any interactions between the hidden variables h_j, we can easily perform the sum discussed in equation (2.24). Explicitly, we can factorise it in the form:

$$F_{rbm} = \sum_{\{h\}} \exp\Big[\sum_{ij} W_{ij}\sigma_i h_j + \sum_j h_j b_j + \sum_i \sigma_i a_i\Big] \tag{2.25}$$

$$= e^{\sum_i \sigma_i a_i} \times \sum_{\{h\}} \Pi_j \exp\Big[\sum_i W_{ij}\sigma_i h_j + h_j b_j \tag{2.26}$$

$$= e^{\sum_i \sigma_i a_i} \times \Pi_j \Big(\exp\Big[\sum_i W_{ij}\sigma_i + b_k\Big] + \exp\Big[-\sum_i W_{ij}\sigma_i - b_j\Big]\Big) \tag{2.27}$$

$$= e^{\sum_i \sigma_i a_i} \times \Pi_j 2\cosh\Big[\sum_i W_{ij}\sigma_i + b_j\Big]. \tag{2.28}$$

Where j is the index of the hidden neurons, and i is the index of the input neurons.

Thus, due to the restricted structure of the connections in the RBM, we are able to easily perform the summation described in equation (2.24) analytically, which is in practice on the Hilbert space of this variable, exponential.

2.8 Gibbs Sampling

To minimize the loss function of our RBM using SGD, the KL divergence, we need to measure the second expectation value of equation (2.11) discussed in section 2.5. To do so, we make use of the explicit form of the RBM described in equation (2.28) to generate samples that distribute according to its probability density: F_{RBM}. Doing so allows us to measure the derivatives that make up equation (2.11), which we then proceed to minimize using Stochastic Gradient Descent.

To generate samples that distribute according to the probability density of the RBM, F_{RBM} we make use of a sampling algorithm which is a special case of the more widely known Metropolis-Hastings algorithm [26]. The sampling algorithm that we use is known as Gibbs sampling.

The idea is that we have a probability density, in this case it has an explicit form described by equation (2.28), and in principle, we devise a Markov chain that generates a chain of samples [27].

By considering the case discussed by G. Carleo and M. Troyer [1] the input is a vector configuration of many-body spins that transitions into a sequence of other many-body spin configurations: $\vec{\sigma_1} \rightarrow \vec{\sigma_2}$ through some transition probability $T(\vec{\sigma_1} \rightarrow \vec{\sigma_2})$. This occurs as a stochastic process: given $\vec{\sigma_1}$ we generate stochastically a possible $\vec{\sigma_2}$ with some probability, and then once again given $\vec{\sigma_2}$ we advance the Markov chain even further, and so on. This provides a way to generate samples from a probability density provided that the so-called detailed balance condition is satisfied [28].

By illustrating this with the more commonly known Metropolis-Hastings algorithm (which can also be used but is less efficient to sample the relevant spin configurations in this case), we can show that these configurations of spins are distributed according to the original probability distribution, $\Pi(\vec{\sigma})$, if we accept the next configuration with a probability using the Metropolis-Hasting acceptance rule:

$$A(\sigma \rightarrow \sigma) = \min[1, \frac{\Pi(\vec{\sigma})}{\Pi(\sigma)} \frac{T(\sigma \rightarrow \sigma)}{T(\sigma \rightarrow \sigma)}] \tag{2.29}$$

To sample from this machine using the Metropolis algorithm, we pick at random one of the binary configurations of the input $\vec{\sigma} = \sigma_1, \sigma_2, \ldots, \sigma_N$ and flip it such that $\sigma \rightarrow \sigma$. By then computing the Metropolis-Hastings acceptance rule in equation (2.29), we let the machine decide whether or not to accept the new configuration of binary input, according to this probability in equation (2.29).

This is one possibility that has commonly been used to sample from the classical partition function of the Ising model [29], but the other possibility which is more popular in this field of Restricted Boltzmann Machines is instead a special case of Metropolis known as Gibbs sampling.

The idea of applying Gibbs sampling here is that we can sample from a joint space, that is an ensemble of *both* the input variables and the hidden variables in the Restricted Boltzmann machine: $\vec{X} = (\vec{\sigma}, \vec{h})$.

In particular, Gibbs sampling takes a transition probability, which drives $\vec{X}$ to a different configuration $\vec{X}$, where one or both the configurations of $\vec{h}$ and $\vec{\sigma}$ are changed. Typically this is achieved in one of two ways. The first type of sampling that can be done is where for example $\vec{\sigma}$ is changed $\vec{\sigma} \rightarrow \vec{\sigma}$ and $\vec{h}$ is left unchanged, leaving the new configuration $\vec{X} = (\vec{\sigma}, \vec{h})$. The other possibility is that we fix $\vec{\sigma}$ and sample only different configurations of $\vec{h}$.

In the first case where we fix $\vec{h}$ and sample only $\vec{\sigma}$, we make use of the transition probability:

$$T_\sigma(\vec{X} \to \vec{X}) = F(\sigma|h) = \frac{F(\sigma, h)}{\sum_\sigma F(\sigma, h)} \tag{2.30}$$

Where $F(\sigma, h)$ is the probability distribution returned by the Restricted Boltzmann Machine, as described in equation (2.28).

In the second case, where $\vec{\sigma}$ is fixed and we sample only $\vec{h}$ the transition probability is:

$$T_h(\vec{X} \to \vec{X}) = F(h|\sigma) = \frac{F(\sigma, h)}{\sum_h F(\sigma, h)} \tag{2.31}$$

In practice, we can compute exactly these transition probabilities above for the Restricted Boltzmann Machine allowing for the sampling of input configurations to be efficiently performed.

The significant advantage of this Gibbs sampling strategy is thus that at the end it is not necessary to perform an extra acceptance step as you would have needed in Metropolis sampling, but rather all moves that are being generated are accepted.

To illustrate this process even further, a possible step in Gibbs sampling would be: we freeze the hidden units $\vec{h}$ (i.e. we dont change them), and generate samples for all the possible inputs that can be had in the input layer, according to the probability distribution $F(\sigma|h)$ described in equation (2.30):

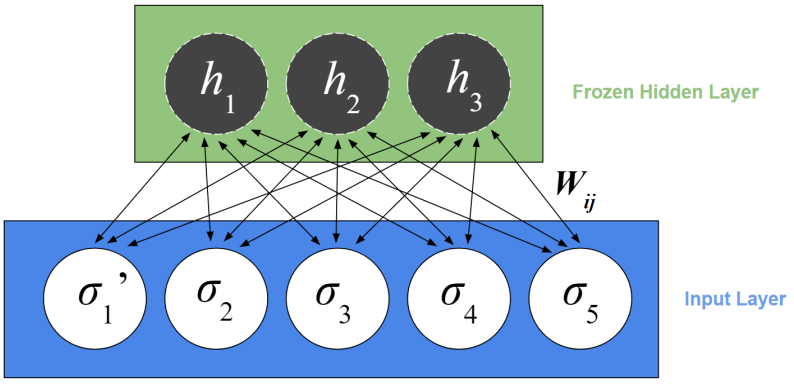

FIGURE 2.9: First possible Gibbs sampling step, where we sample only different configurations of input variables while fixing the hidden variables.

The other possible Gibbs sampling step would be we freeze the input units and sample all the hidden units according to $F(\sigma|h)$.

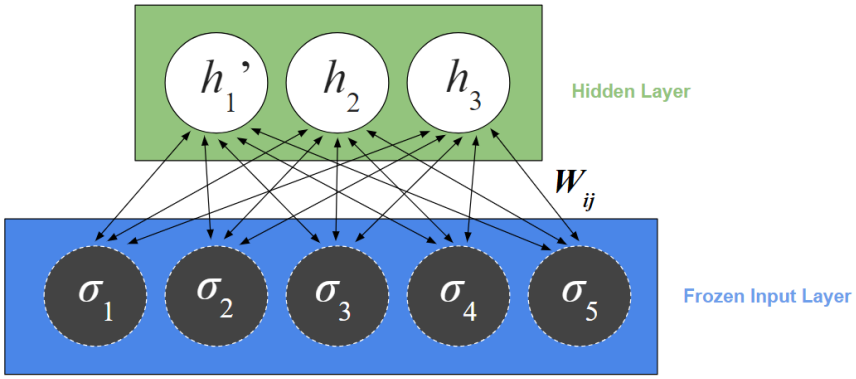

FIGURE 2.10: Other possible Gibbs sampling step is to fix the hidden units, and sample different configurations of the input units.

The Gibbs sampling strategy then continuously goes back and forth between these two possible steps.

At each step, we measure the expectation value over the approximate probability distribution of the Restricted Boltzmann Machine, which is the second term in the KL divergence in equation (2.11). Now that we know how to sample from this machine, we can thus calculate the derivatives to minimize the KL divergence: $D_k(\vec{\sigma};\vec{p}) = \frac{\partial}{\partial P_k}\log F_{RBM}(\vec{\sigma};\vec{p})$ discussed in section 2.5 .

To do so, in the case of the RBM the derivatives simply read:

$$D_{a_i}(\sigma) = \frac{1}{2}\sigma_i, \tag{2.32}$$

$$D_{b_j}(\sigma) = \frac{1}{2}\tanh(\theta_j), \tag{2.33}$$

$$D_{W_{ij}}(\sigma) = \frac{1}{2}\tanh(\theta_j)\sigma_i. \tag{2.34}$$

Hence, the gradient of the KL divergence is:

$$\partial_{p_k} D_{KL}(\Pi||F) = -\langle\langle D_k(\mathbf{x};\mathbf{p})\rangle\rangle_{\Pi} + \langle\langle D_k(\mathbf{x};\mathbf{p})\rangle\rangle_{F_{rbm}} \tag{2.35}$$

A popular strategy of computing this gradient is the so-called *contrastive divergence* method of Geoffrey Hinton [30], a leading figure in the field of machine learning.

In this case, one typically takes a very small batch size in the Gibbs sampling procedure (often as small as $N_s = 1$). This implies a large statistical error on the gradient, which however is strongly reduced by the fact that the starting configuration for the Gibbs sampling is not randomly chosen, but is instead one of the elements of the dataset.

This means that errors between the two terms in the gradient tend to cancel out due to correlated sampling.

Using this strategy of Gibbs sampling, we therefore have a method to minimize the KL divergence: the loss function of our RBM. By minimizing this loss function, the separation between two probability distributions, we now have a method us to arrive at an accurate approximation of the wavefunction approximated by the probability distribution of the RBM: $F_{RBM} = |\psi^2|$.

Chapter 3

Learning Quantum Mechanics

3.1 Solving Quantum Mechanics with Machine Learning

A central problem in quantum mechanics is solving the Schrodinger equation:

$$H|\psi_i\rangle = E_i|\psi_i\rangle \tag{3.1}$$

The problem that we consider is how can we rephrase this as an optimization problem that we can use machine learning to solve?

To rephrase this is an optimization problem, we make use of the technique: Quantum Variational Monte Carlo.

3.2 Variational Monte Carlo

Developed by pioneers in computational quantum physics such as Bill McMillan in the 1960s [31], quantum variational Monte Carlo has been used for decades and is nothing particularly new.

To rephrase the Schrodinger equation as an optimization problem, we transform the eigenvalue problem in (3.1) into a stochastic optimization problem. To do that, we begin with an alternative formulation of the Schrodinger equation, based on the variational principle. In particular, we consider the energy functional:

$$E[\psi] = \langle\psi|H|\psi\rangle \geq E_0 \tag{3.2}$$

where ψ is an arbitrary physical state, and E_0 is the exact ground-state energy of the Hamiltonian H. It is clear that from the variational theorem that one can find the exact ground-state wavefunction as the solution of the following optimization problem:

$$\psi_0 = \text{argmin}_\psi E[\psi] \tag{3.3}$$

However, for an arbitrary state ψ it is rarely possible to compute analytically the energy functional, since it involves integrals over high-dimensional space. To solve this problem, pioneers in the 1960s such as McMillian realized that the energy functional can be computed stochastically [32]. In particular, the Variational Monte Carlo method is based on the understanding that expectation values like (3.2) can be written as statistical averages over a suitable probability distribution.

3.3 Stochastic Estimates of Observables

If we assume that our Hilbert space is spanned by the quantum states $|\mathbf{x}\rangle$, then we can denote a sum over the Hilbert space with discrete sums. In particular we use the closure relation $\sum_x |\mathbf{x}\rangle\langle\mathbf{x}| = 1$.

By using the closure relation, we can rewrite a quantum expectation value of an arbitrary operator O as:

$$\frac{\langle\psi|O|\psi\rangle}{\langle\psi|\psi\rangle} = \frac{\sum_{x,x'}\langle\psi|\mathbf{x}\rangle\langle\mathbf{x}|O|\mathbf{x}'\rangle\langle\mathbf{x}'|\psi\rangle}{\sum_x\langle\psi|\mathbf{x}\rangle\langle\mathbf{x}|\psi\rangle} \tag{3.4}$$

$$= \frac{\sum_{x,x'}\psi^*(\mathbf{x})O_{xx'}\psi(\mathbf{x}')}{\sum_x|\psi(\mathbf{x})|^2} \tag{3.5}$$

If the operator O is diagonal in the computational basis, such that $O_{xx'} = \delta_{xx'}O(\mathbf{x})$, then:

$$\frac{\langle\psi|O|\psi\rangle}{\langle\psi|\psi\rangle} = \frac{\sum_x|\psi(\mathbf{x})|^2O(\mathbf{x})}{\sum_x|\psi(\mathbf{x})|^2} \tag{3.6}$$

$$\equiv \langle\langle O\rangle\rangle, \tag{3.7}$$

where $\langle\langle\ldots\rangle\rangle$ denotes a statistical expectation value over the probability distribution $\Pi(\mathbf{x}) = |\psi(\mathbf{x})|^2$. In other words, in this case quantum expectation values are exactly

equivalent to averaging over Hilbert-space states that are sampled according to the square-modulus of the wave-function.

In the case where the operator O is off-diagonal in the computational basis, then we can define an auxiliary diagonal operator, which is often referred to as the local operator.

$$O_{loc}(\mathbf{x}) = \sum_{x'} O_{xx'} \frac{\psi(\mathbf{x}')}{\mathbf{x}} \tag{3.8}$$

It is easily proven that:

$$\frac{\langle\psi|O|\psi\rangle}{\langle\psi|\psi\rangle} = \frac{\sum_x |\psi(\mathbf{x})|^2 O_{loc}(\mathbf{x})}{\sum_x |\psi(\mathbf{x})|^2} \tag{3.9}$$

$$\equiv \langle\langle O_{loc}\rangle\rangle. \tag{3.10}$$

Therefore, for any observable we can always compute expectation values over arbitrary wavefunctions as statistical averages.

For off-diagonal operators, the sum $\sum_{x'} O_{xx'} \frac{\psi(\mathbf{x}')}{\psi(\mathbf{x})}$ occupies a small section of the Hilbert space in which $\mathbf{x}'$ is such that $O_{xx'} \neq 0$. For a large majority of physical observables, and for a given $\mathbf{x}$, the number of elements $\mathbf{x}'$ that are connected by those matrix elements is polynomial in size, allowing for the summation to be performed systematically. This is contrasted by the summations in $\sum_x |\psi(\mathbf{x})|^2$, in which there are typically exponentially large permutations of $\mathbf{x}$ on which to perform the summation, which can therefore not be carried out by brute-force. The powerful idea behind Variational Monte Carlo therefore is to replace this sum over an intractably large number of possible states, with a statistical average over a large but finite set of states that are sampled according to a probability distribution $\Pi(\mathbf{x})$. Done this way, we therefore have a way to compute stochastically the expectation value of all properties of interest.

In the case in which the operator is the Hamiltonian, its expectation value can be computed using the estimator described in (3.10). The local estimator associated to the Hamiltonian is referred to as the local energy:

$$E_{loc}(\mathbf{x}) = \sum_{x'} H_{xx'} \frac{\psi(\mathbf{x}')}{\psi(\mathbf{x})} \tag{3.11}$$

3.4 Stochastic Variational Optimization

The ultimate goal here is to optimize the variational energy. In general, we assume that the wavefunction depends on some set of parameters $\mathbf{p} = p_1, \dots p_M$, where M can be intractably large. We have discussed before that the expectation value of the energy can be written as a statistical average of the form:

$$\langle H \rangle \simeq \langle \langle E_{loc} \rangle \rangle. \tag{3.12}$$

It can be easily shown that the gradient of the energy can be also written in the form of an expectation value of a stochastic variable. Specifically, we define:

$$D_k(\mathbf{x}) = \frac{\partial_{p_k} \psi(\mathbf{x})}{\psi(\mathbf{x})}, \tag{3.13}$$

Then:

$$\partial_{p_k} \langle H \rangle = \partial_{p_k} \frac{\sum_{x,x'} \psi^*(\mathbf{x}) H_{xx'} \psi(\mathbf{x}')}{\sum_x |\psi(\mathbf{x})|^2} \tag{3.14}$$

$$= \frac{\sum_{x,x'} \psi^*(\mathbf{x}) H_{xx'} D_k(\mathbf{x}') \psi(\mathbf{x}')}{\sum_x |\psi(\mathbf{x})|^2} + \frac{\sum_{x,x'} \psi^*(\mathbf{x}) D_k^*(\mathbf{x}) H_{xx'} \psi(\mathbf{x}')}{\sum_x |\psi(\mathbf{x})|^2} \tag{3.15}$$

$$- \frac{\sum_{x,x'} \psi^*(\mathbf{x}) H_{xx'} \psi(\mathbf{x}')}{\sum_x |\psi(\mathbf{x})|^2} \frac{\sum_x |\psi(\mathbf{x})|^2 (D_k(\mathbf{x})|^2)(D_k(\mathbf{x}) + D_k^*(\mathbf{x}))}{\sum_x |\psi(\mathbf{x})|^2} \tag{3.16}$$

$$= \frac{\sum_{x,x'} \frac{\psi^*(\mathbf{x})}{\psi^*(\mathbf{x}')} H_{xx'} D_k(\mathbf{x}') |\psi(\mathbf{x}')|^2 + \sum_{x,x'} |\psi(\mathbf{x})|^2 H_{xx'} D_k^*(\mathbf{x}') \frac{\psi(\mathbf{x}')}{\psi(\mathbf{x}))}}{\sum\limits_x |\psi(\mathbf{x})|^2} \tag{3.17}$$

$$\simeq \langle \langle E_{loc} D_k^* \rangle \rangle - \langle \langle E_{loc} \rangle \rangle \langle \langle D_k^* \rangle \rangle + cc. \tag{3.18}$$

Hence, we can efficiently write $\partial_{p,k} \langle H \rangle \simeq \langle \langle G_k \rangle \rangle$ with the energy-gradient estimator:

$$G_k(\mathbf{x}) = 2Re[(E_{loc}(\mathbf{x}) - \langle \langle E_{loc} \rangle \rangle D_k^*(\mathbf{x}))] \tag{3.19}$$

A striking feature of the energy and energy-gradient estimators is that they have what is known as the zero-variance property. This means that their statistical fluctuations are exactly zero when sampling from the exact ground-state wavefunction. To illustrate this, consider the example in which the wavefunction is real:

$$\begin{aligned}
\text{var}(E_{loc}) &= \langle E_{loc}^2 \rangle - \langle E_{loc} \rangle^2 && (3.20)\\
&= \sum_x \psi(\mathbf{x})^2 E_{loc}(\mathbf{x})^2 - \langle H \rangle^2 && (3.21)\\
&= \sum_x \sum_{x_1} H_{x,x_1} \psi(\mathbf{x_1}) \sum_{x_2} H_{x,x_2} \psi(\mathbf{x_2}) - \langle H \rangle^2 && (3.22)\\
&= \sum_{x_1} \psi(\mathbf{x}_1) \sum_x H_{x,x'} \sum_{x_2} H_{x,x_2} \psi(\mathbf{x_2}) - \langle H \rangle^2 && (3.23)\\
&= \langle H^2 \rangle - \langle H \rangle^2, && (3.24)
\end{aligned}$$

Therefore, the variance of the local energy is the energy variance. It is easy to show that if ψ is an eigenstate of H then $\langle H^2 \rangle = \langle H \rangle^2 = E_0^2$, and $var(E_{loc}) = 0$. In other words, their statistical fluctuations are precisely zero. This implies that the closer we get to the ground-state, the less fluctuations we have on the quantity we want to minimize: the energy.

To minimize our energy, we can use Stochastic Gradient Descent. Specifically, the SGD parameter update rule after each iteration is:

$$\partial_k^{s+1} = \partial_k^s - \eta \partial_{p_k} \langle H \rangle. \tag{3.25}$$

Rather than with the non-stochastic, deterministic Gradient Descent in (2.13), we now have stochastic averages of the gradient which are subjected to stochastic noise. Similarly to (2.19), we can define an effective Langevin temperature:

$$\begin{aligned}
T &= \frac{\eta \text{var}(\bar{G}_k)}{2} && (3.26)\\
&\propto \frac{\eta}{N_s} \text{var}(G_k) && (3.27)
\end{aligned}$$

An important difference that occurs due to this zero-variance property, is that it is no longer necessary to reduce the learning rate η when iterating, such that to not overshoot the global minimum. The main reason is that as we start to approach the exact ground state, then $var(G_k) \to 0$. Due to this, even a constant number of samples and a small fixed learning rate η are sufficient to converge to the ground state.

3.5 Using Restricted Boltzmann Machines to solve the Quantum Many-Body Problem: Ising model

Having discussed this method to obtain the best set of parameters in a Restricted Boltzmann Machine to accurately estimate an unknown probability distribution, we consider an example practical application in using this method to solve a difficult problem in quantum physics.

Specifically, we consider the recent work of G. Carleo and M. Troyer, where they have used this method as discussed to solve for the ground-state of the transverse-field Ising model in 1D, for a quantum many-body system of spin particles [1].

The particles in this system are arranged with periodic boundary conditions over a ring of L sites. The Hamiltonian for such a system is as follows:

$$H = -h\sum_i \sigma_i^x - J\sum_i \sigma_i^z\sigma_{i+1}^z \tag{3.28}$$

To improve efficiency, and since we know that in this case, the ground-state wavefunction is positive-definite, we consider the following form for the quantum state wavefunction:

$$\psi(\sigma_1^z, \sigma_2^z, \ldots, \sigma_N^z) = \sqrt{F_{RBM}(\sigma_1^z, \sigma_2^z, \ldots, \sigma_N^z)} \tag{3.29}$$

where F_{RBM} contains only real-valued parameters. An advantage of this formulation is that sampling from $F_{RBM}(\sigma_1^z, \sigma_2^z, \ldots, \sigma_N^z) = |\psi(\sigma_1^z, \sigma_2^z, \ldots, \sigma_N^z)|^2$ is more efficient, as it can be done using Gibbs sampling as previously discussed using unsupervised learning.

To compute the local energy for a given spin configuration, we make use of the following:

$$E_{loc}(\sigma) = \sum_\sigma H_{\sigma,\sigma}\frac{\psi(\sigma)}{\psi(\sigma)} \tag{3.30}$$

The sum in (3.30) runs over the $N+1$ configurations $\sigma(0) = \sigma$ and $\sigma(k) = \sigma_1^z \cdots - \sigma_k^z \ldots \sigma_N^z$, where $H_{\sigma,\sigma} = -J\sum_i \sigma_i^z\sigma_{i+1}^z$ and $H_{\sigma,\sigma(k>0)} = -h$. This sum can be efficiently calculated by pre-computing the values of what are referred to as effective angles θ_j. In particular:

$$\frac{\psi(\sigma(k))}{\psi(\sigma)} = e^{-2a_k\sigma_k} \times \Pi_j \frac{\cosh(\theta_j - 2\sigma_k W_{jk})}{\cosh(\theta_j)} \tag{3.31}$$

Where:

$$\theta_j(S) = b_j + \sum^{i} W_{ij}\sigma_i^z \tag{3.32}$$

To compute the variational derivatives, $D_k(\sigma) = \frac{\partial_{p_k}\psi(\sigma)}{\psi(\sigma)}$, we make use of the following:

$$D_{a_i}(\sigma) = \frac{1}{2}\sigma_i^z \tag{3.33}$$

$$D_{b_j}(\sigma) = \frac{1}{2}\tanh(\sigma_j) \tag{3.34}$$

$$D_{W_{ij}}(\sigma) = \frac{1}{2}\tanh(\sigma_j)\sigma_i^z \tag{3.35}$$

We minimize the gradient of the KL divergence by utilizing Gibbs sampling and stochastic gradient descent:

$$\partial_{p_k} D_{KL}(\Pi||F) = -\langle\langle D_k(\mathbf{x};\mathbf{p})\rangle\rangle_{\Pi} + \langle\langle D_k(\mathbf{x};\mathbf{p})\rangle\rangle_{F_{rbm}} \tag{3.36}$$

With this method we can evaluate accurate approximations of wavefunctions by learning the parameters of a Restricted Boltzmann Machine, such that our KL divergence is minimized. By utilizing variational Monte Carlo, we can continuously trial different wavefunctions returned by our RBM in a reinforcement learning scheme such that applying a Hamiltonian operator on them returns the minimum energy. The final trial wavefunctions that we converge to are thus the ground state wavefunctions, as applying the Hamiltonian on these returns the lowest (i.e. ground state) energy - as by the variational theorem.

By utilizing this method, G. Carleo and M. Troyer have demonstrated that they can solve the quantum many-body problem for the 1D Transverse Field Ising (TFI) and the 1D, and 2D Antiferromagnetic Heisenberg (AFH) models with extremely high accuracy:

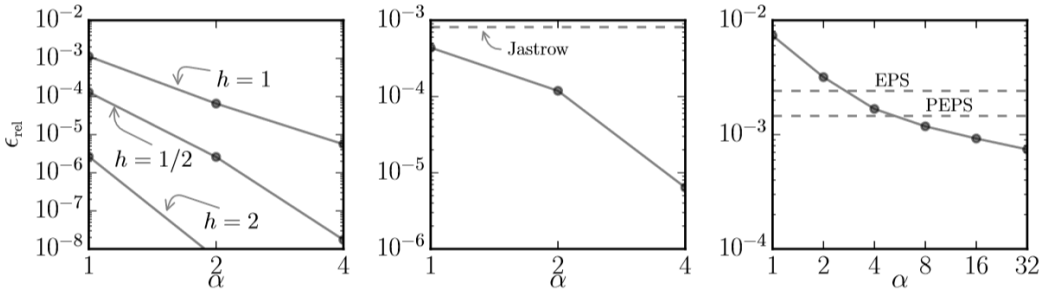

FIGURE 3.1: Relative error ϵ_{rel} of the ground state energy returned by the RBM compared to the exact value, for several test cases. Increased precision of the ground-state can be obtained by increasing the hidden unit density α. Accuracy for the 1D TFI model is shown in the left most panel with different values of field strength h, and for 80 spins chain with periodic boundary conditions (PBC). On the central panel is the accuracy for the 1D AFH model, for a 80 spins chain with PBC, compared to the Jastrow ansatz (horizontal dashed line). On the right most panel is the accuracy for the 2D AFH model on a 10×10 square lattice with PBC, compared to the precision obtained by EPS, and PEPS. Results in this figure were obtained from G. Carleo and M. Troyer's original research paper [1].

From figure 3.1 we observe that representing many-body quantum states with artificial neural networks can achieve MPS-grade accuracies in 1D, while it systematically improves the best known variational states for 2D finite lattice systems. Accuracy is demonstrated to increase by increasing the hidden unit density $\alpha = N_h/N_i$, where N_h and N_i are the number of hidden neurons and number of input neurons respectively.

Chapter 4

Finding the Ground State of a Quantum Particle in a Box

Having discussed the effectiveness of using artificial neural networks to approximate the ground-state for quantum many-body quantum systems that are functions of binary input (i.e. spin) - we now make efforts to expand upon this method to encompass a wider range of Hamiltonians. Namely, Hamiltonians that act on wavefunctions that are functions of continuous input, such as x-position, rather than binary valued spins. Specifically, we make an attempt to solve for the ground-state of the 1D quantum particle in a box using artificial neural networks.

The key obstacle in this approach, is that the Restricted Boltzmann Machine (which represents the wavefunction) can only take binary valued input. Previously, this was spin. However, we must now devise a method in which we can provide continuous input of real space, x-position, to the RBM to approximate the wavefunction of a 1D particle in a box. To do this, we have discretized the integration region of our box into a series of mesh-points (discretization points), and labeled each mesh-point with integer values that corresponds to its location along the integration region. We then convert these integer mesh-point locations into binary bytes - and provide the bits of these bytes as input to the RBM and adjust our Hamiltonian to reconvert these bytes into continuous x-position values along the discretized lattice. The method in which we attempt this is described in the next section, where we devise a way to write the variational wavefunctions on a discretized lattice.

4.1 Writing a Variational Wavefunction on a Discretized Lattice

Having discussed an example of using a Restricted Boltzmann Machine to solve for the ground state of Hamiltonians of binary spin, we devise a similar method in which we can apply these tools to solve for the ground state of a quantum particle in a box.

The central problem we are trying to solve here is the Schrodinger equation in 1D:

$$-\nabla^2\psi(x) + V(x)\psi(x) = E\psi(x) \tag{4.1}$$

To do this, we discretize the integration region of our particle in a box into a series of mesh points (i.e. discretization points) - and evaluate the amplitudes of the wavefunction, ψ, at each of these mesh-point locations as shown in figure 4.1.

We first assume that ψ is defined on an integration region of length L, where the potential energy is infinite outside of this region and hence $\psi = 0$ outside of the range L:

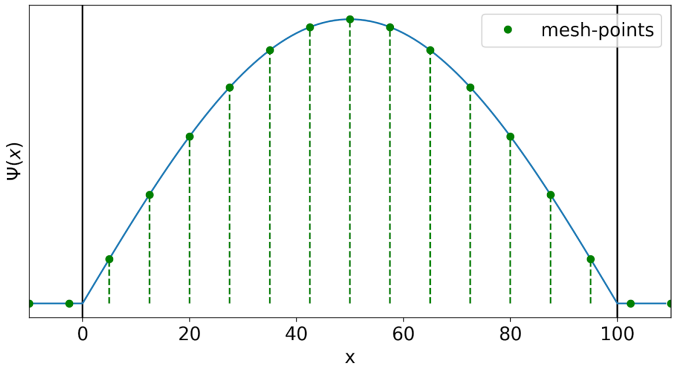

FIGURE 4.1: Wavefunction discretized into a series of mesh-points.

The eigenstate for a particle at a position $\bar{x}$, is given by:

$$|\bar{x}\rangle = \delta(x - \bar{x}) \tag{4.2}$$

The wavefunction can be expanded on this basis quantum state:

$$|\psi(x)\rangle = \int d\bar{x}\ C_{\bar{x}}^{\psi}\ |\bar{x}\rangle \tag{4.3}$$

Here the coefficients C_x^{ψ}, in fact, describe the wavefunction:

$$|\psi(x)\rangle = \int d\bar{x}\ C^{\psi}_{\bar{x}}\ \delta(x-\bar{x}) = C^{\psi}_{x} \tag{4.4}$$

In practice, we discretize the integration region L into 2^n points that are equally spaced from each other:

$$|\psi\rangle = \int d\bar{x}\ \overbrace{\psi(\bar{x})}^{C^{\psi}_{\bar{x}}}|\bar{x}\rangle \approx \sum_i C^{\psi}_i |x_i\rangle \tag{4.5}$$

where C^{ψ}_i are the coefficients of the expansion. Clearly, these are also the value of the wavefunction at the discretization points:

$$C^{\psi}_i = \psi(x_i) \tag{4.6}$$

$|x_i\rangle$ is our basis to expand the wavefunction ψ, and C^{ψ}_i are the coefficients of the expansion.

Note that any wavefunction of the basis can also be expanded on the $|x_i\rangle$ as well. Thus:

$$|x_j\rangle = \sum_i C^j_i |x_i\rangle \tag{4.7}$$

Since,

$$\langle x_k|x_j\rangle = \delta_{kj} \tag{4.8}$$

we therefore have that:

$$\langle x_i|x_j\rangle = \delta_{ij} = \langle x_i| \sum_k C^j_k |x_k\rangle = \sum_k C^j_k \underbrace{\langle x_i|x_k\rangle}_{\delta_{kj}} = C^j_i \tag{4.9}$$

Thus:

$$C^j_i = \delta_{ij} \tag{4.10}$$

By considering the work done by G. Carleo and M. Troyer for solving the quantum-many-body problem [1], they expand their quantum states for the Heisenberg and Ising

models into a basis for their Hilbert space. In their case, the states of the basis are each possible configurations of spins $\{\sigma_n\}$. The eigenvalues of equation (4.5) are:

$$|\psi\rangle = \sum_{\{\sigma\}} \psi(\{\sigma\})|\{\sigma\}\rangle \tag{4.11}$$

Where the basis states are:

$$|\{\sigma\}\rangle = |+,+,-,-,+,\dots\rangle, \tag{4.12}$$
$$= |+,-,+,+,-,\dots\rangle, \tag{4.13}$$
$$= \text{etc.} \tag{4.14}$$

Once again, the coefficient of the expansion are obtained by projecting the entire quantum state on each state of the basis:

$$\langle\{\sigma\}|\psi\rangle = \psi(\{\sigma\}) \tag{4.15}$$

This is the wavefunction that is approximated by the RBM neural network function.

It is important to note that $\psi(\{\sigma\})$ is a complex valued function of discretized variables, just like in equation (4.6).

The main problem to solve here is to map the spin configurations $\{\sigma\}$ into the $\{x_i\}$ basis. The mapping must be a 1-to-1 correspondence between each spin configuration and the discrete positions x_i:

$$X(\sigma) \to x_i \tag{4.16}$$
$$S(x_i) \to \{\sigma\} \tag{4.17}$$

Given that the spins have two possible values, we take these as a sequence of binary numbers (i.e. bytes) used to label each coordinate x_i:

$$i = \sum_{l=0}^{n=1} b(\sigma_l)2^l \tag{4.18}$$

Where the values of the bits, $b(\sigma_l)$, are either 0 or 1:

$$b_{\sigma_l} = \begin{cases} 0 & \text{if } \sigma_l = -1 \\ 1 & \text{if } \sigma_l = +1 \end{cases} \tag{4.19}$$

The coordinate points x_i are thus:

$$x_i = x_0 + i\Delta x \tag{4.20}$$

where:

$$\Delta x = \frac{L}{2^n - 1} \tag{4.21}$$

The VMC approach done by G. Carleo and M. Troyer samples possible spin configurations and calculates the local energy at each VMC sweep.

For a configuration $|\{\sigma\}\rangle$, the local energy is defined as:

$$E_{loc} = \frac{\langle\sigma|H|\psi\rangle}{\langle\sigma|\psi\rangle} \tag{4.22}$$

By inserting the identity and using equation (4.15), we have:

$$E_{loc} = \frac{\sum\limits_{\{\sigma'\}} \langle\sigma|H|\sigma'\rangle\langle\sigma|\psi\rangle}{\langle\sigma|\psi\rangle} = \frac{\sum\limits_{\sigma'}\langle\sigma|H|\sigma'\rangle \cdot \psi(\sigma')}{\psi(\sigma')} \tag{4.23}$$

Thus, G. Carleo and M. Troyer require the matrix elements of their Hamiltonians in the basis states of spin $|\{\sigma\}\rangle$.

Things are exactly the same for our particle in a box:

$$E_{loc} = \frac{\langle x_i|H|\psi\rangle}{\langle x_i|\psi\rangle} = \frac{\sum\limits_j \langle x_i|H|x_j\rangle\psi(x_j)}{\psi(x_i)} \tag{4.24}$$

Using the mapping defined in (4.18), we can solve for the energy of the particle in a box using the spin and RBM approach of G. Carleo and M. Troyer.

$$\langle\sigma_i|H|\sigma_j\rangle \to \langle X(\sigma_i)|H|X(\sigma_j)\rangle \tag{4.25}$$

$$\psi(\sigma_i) \to \psi(X(\sigma_i)) \tag{4.26}$$

By adapting the open source code of G. Carleo and M. Troyer [5], we note that their subroutines prestore the matrix elements of:

$$\langle\sigma|H|\sigma\rangle \tag{4.27}$$

By providing a state $|\sigma\rangle$, the state $|\sigma\rangle$ is expressed as a set of spin flips with respect to $|\sigma\rangle$. Then the matrix elements of equation (4.27) are used to calculate the local energy in equation (4.23).

To find the energy of our particle in a box, we evaluate the matrix elements of the Hamiltonian in the discretized coordinate space basis:

$$\langle x_i|\hat{H}|x_j\rangle = \langle x_i|-\nabla^2+\hat{V}|x_j\rangle \tag{4.28}$$

The double derivative of the wavefunction can be expanded on the basis $|x_i>$, exactly as for $|\psi>$ in equation (4.5). This gives:

$$|\nabla^2\psi\rangle = \int d\bar{x}\ \psi(\bar{x})|\bar{x}\rangle = \sum_i \psi(x_i)|x_i\rangle \tag{4.29}$$

where $\psi(x_i)$ needs to be calculated from the values of $\psi(x_i)$ at neighbouring discretization points:

$$\psi(x_i) = \frac{\psi(x_{i+1}) - 2\psi(x_i) + \psi(x_{i-1})}{(\Delta x)^2} \tag{4.30}$$

It should be noted that other approximation methods of the kinetic energy term exist, other than from equation (4.29) [33].

By applying equations (4.29) and (4.30) together, we can get the effect of the ∇^2 operator on the basis states.

$$\nabla^2|x_j\rangle = \sum_k (C^j_{k+1} - 2C^j_k + C^j_{k-1})\frac{1}{(\Delta x)^2}|x_k\rangle \tag{4.31}$$

Then, by applying equations (4.8) and (4.10), we get:

$$\langle x_i|\nabla^2|x_j\rangle = \frac{\delta_{j,i+1} - 2\delta_{i,j} + \delta_{j,i-1}}{(\Delta x)^2} = \begin{cases} \frac{1}{(\Delta x)^2} & \text{if } i = j \pm 1 \\ \frac{-2}{(\Delta x)^2} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.32}$$

To validate equation (4.32), consider applying it to a generic function $\psi(x)$

$$\langle x_i|\nabla^2|\psi\rangle = \sum_j \langle x_i|\nabla^2|x_j\rangle \cdot \langle x_j|\psi\rangle \tag{4.33}$$

$$= \sum_j \frac{\delta_{j,i+1} + \delta_{j,i-1} - 2\delta_{ij}}{(\Delta x)^2} \cdot \psi(x_j) \tag{4.34}$$

$$= \frac{\psi(x_{i+1}) - 2\psi(x_i) + \psi(x_{i-1})}{(\Delta x)^2} = \psi(x_i) \tag{4.35}$$

which confirms equations (4.29) and (4.30).

Finally, for the potential energy term, it is self-evident that:

$$\langle x_i|V(x)|x_j\rangle = V(x_i)\delta_{ij} \tag{4.36}$$

Hence, the final complete Hamiltonian operator for our discretization points is as follows:

$$\langle x_i|\hat{H}|x_j\rangle = \langle x_i|\nabla^2 + V(x)|x_j\rangle = \begin{cases} \frac{1}{(\Delta x)^2} & \text{if } i = j \pm 1 \\ \frac{-2}{(\Delta x)^2} + V(x_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.37}$$

4.2 Using Restricted Boltzmann Machines to Find the Ground State of a Particle in a Box

By providing the locations of the mesh points i as binary bytes as the input to the Restricted Boltzmann Machine, we find the probability distribution of this input by minimizing the KL divergence using Gibbs sampling, a form of unsupervised learning, and stochastic gradient descent as discussed in previous sections. Having learned the probability distribution F_{RBM} we can find thus an accurate representation of the wave-function as a function of the input x-position $\psi(x)$, as in our case, the wavefunction is positive definite and is therefore the square root of the probability distribution:

$$\psi(x) = \sqrt{F_{rbm}} \tag{4.38}$$

As an example scenario, consider the case in which the current input to the RBM is the binary byte representing the number 5: $\vec{\sigma} = (0, 0, 1, 0, 1)$:

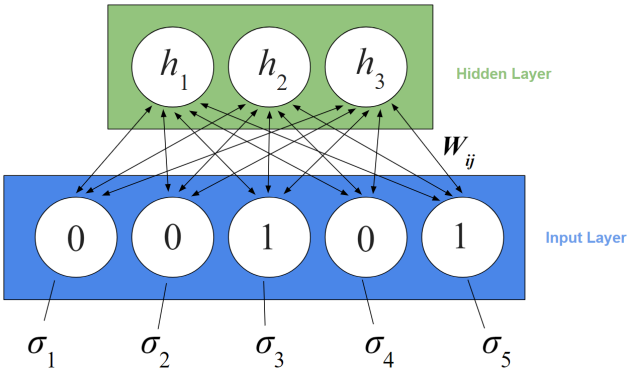

FIGURE 4.2: Example binary input to Restricted Boltzmann Machine.

Using this example input, the location of the mesh-point $i = 5$ is determined, using equation (4.18). The particles position in real space at this mesh-point can thus be determined using equation (4.20) as discussed in the previous section:

$$x = x_0 + i\Delta x = x_0 + (0, 0, 1, 0, 1)\Delta x = x_0 + 5\Delta x \tag{4.39}$$

By choosing 1024 meshpoints, we thus have $log_2(1024) = 10$ bits in our byte and hence the input $\vec{\sigma}$ is of size 10 (i.e. we thus have 10 input nodes in the network). Choosing an integration region of size $L = 120$, the mesh-points are equally spaced with a spacing of $\Delta x = \frac{L}{N_{mesh}} = \frac{120}{1024}$.

By choosing a large potential barrier of potential 1.e4 that is placed at both $x = 0$, and $x = 100$, we evaluate the wavefunctions at each of the mesh-points using the unsupervised learning method discussed previously, and minimize the measured energy to find the ground state using the reinforcement learning scheme based on variational Monte Carlo.

4.3 Validation of our Method to Solve the Variational Wavefunction on a Discretized Lattice

Using the code we have developed, which we have made publicly available on GitHub [4], we validate the method we have devised to solve for the variational wavefunction on

a discretized lattice.

The results of this method as discussed in the previous sections is as follows:

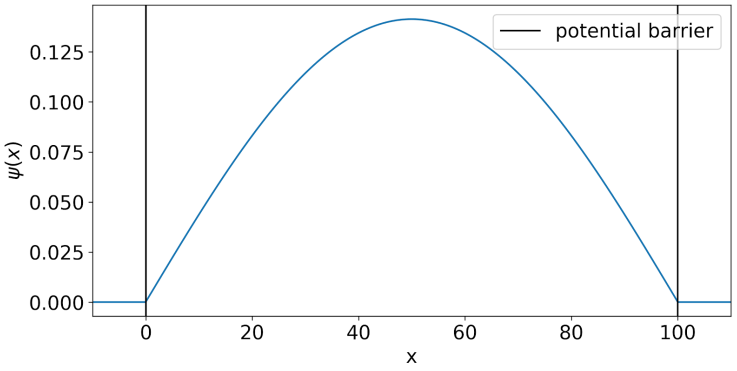

FIGURE 4.3: Ground state wavefunction calculated by discretizing the integration region into a series of 1024 mesh-points.

The mean squared error between the analytical value for the approximated wavefunction in this integration region using this method was 5.86e-9. The error between the approximate and analytical wavefunctions for the ground state were as follows:

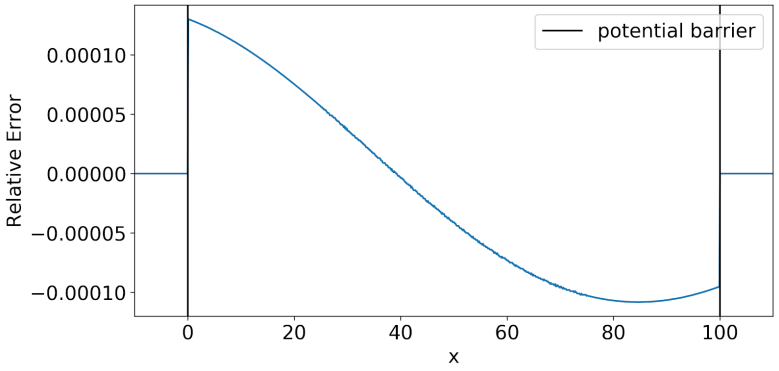

FIGURE 4.4: Relative error of the approximated ground state wavefunction compared to the analytical wavefunction as a function of location in the box.

Having shown that we can reproduce the analytical solutions with very good accuracy by discretizing the spatial coordinate, we utilize this method to find the ground state energy and wavefunction of a particle in a box using machine learning as discussed in the next section.

4.4 Results

Note: At this stage, the results are still a work in progress as this was a very ambitious project for such a short time frame.

In this section, we present the results so far of implementing the method we have devised to solve for the ground state wavefunction of a 1D quantum particle in a box using artificial neural networks.

By choosing a learning rate of $\eta = 0.0002$, 10 input nodes, 20 hidden nodes, and a batch size of $N_b = 100$ in the stochastic sampling of configurations of $\mathbf{x}$ in the variational method, the following results were obtained:

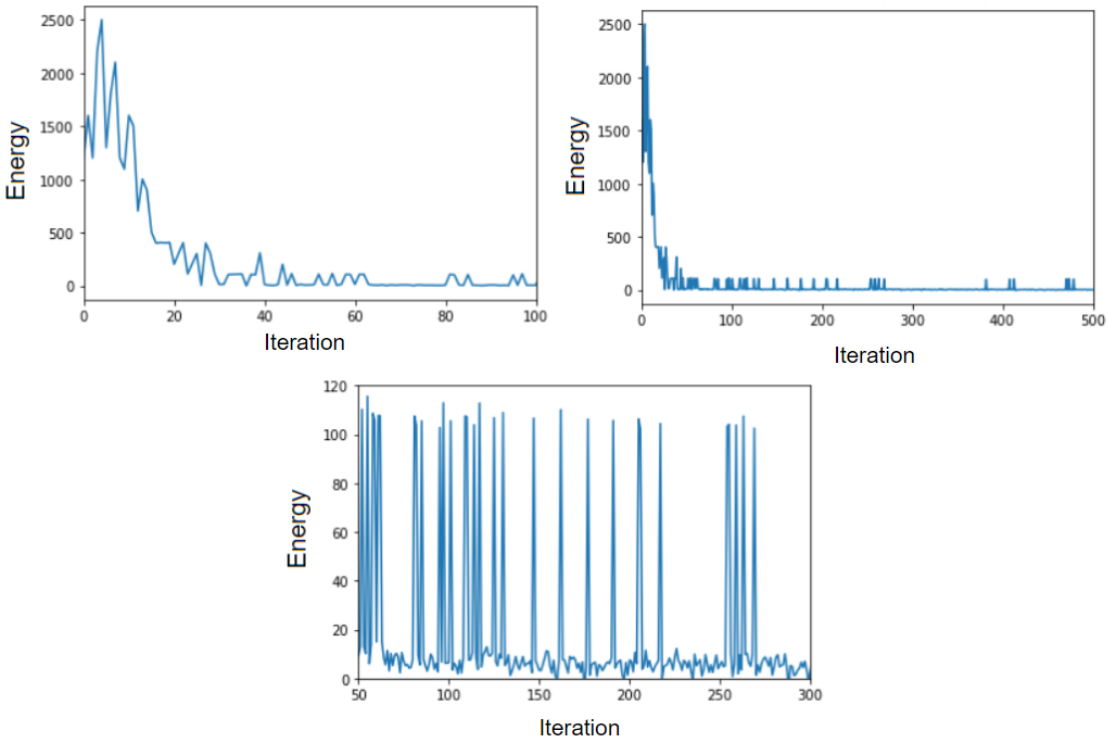

FIGURE 4.5: Reduction in energy due to reinforcement learning scheme based on variational Monte carlo. Stochastic estimates of energy gradient are used to converge towards a ground-state energy. Different limits on the axis are used to illustrate the minimization process.

By returning the wavefunction at each mesh-point location for the trained parameters of the RBM, using equation (2.28), where σ_i in this case are the values of the bits that form each byte that represent each mesh-point location along the discretization region - the RBM arrives at the following wavefucntions for the following number of iterations, and measured energies:

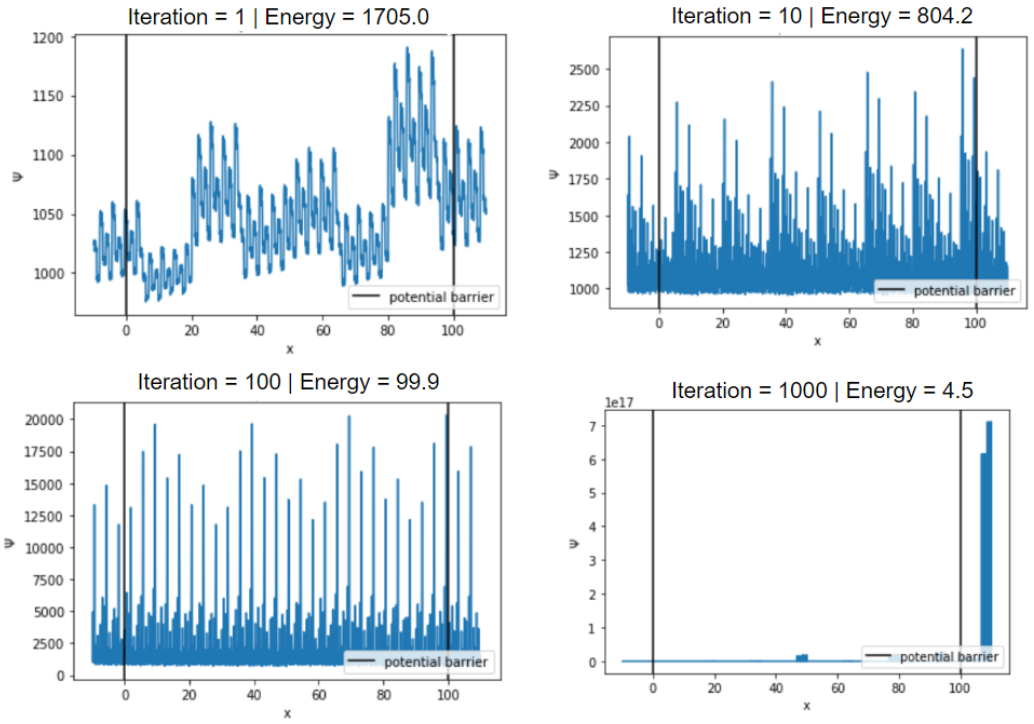

FIGURE 4.6: Reconstructed wavefunctions using trial wavefunctions returned by Restricted Boltzmann Machine at iteration 1, 10, 100, and 1000 from left to right respectively. These were measured to have energies of 1705, 804, 100, and 4.5 respectively.

It was found that the value of the learning rate had a significant effect on this methods effectiveness at reducing the energy of the system, as expected from the discussion in section 2.6. For a learning rate of $\eta = 0.2$, it was found that the RBM was unable to converge to a low energy - and instead diverged from the solution, getting stuck at a configuration being unable to reduce the energy.

FIGURE 4.7: Reduction in energy for a learning rate of $\eta = 0.2$

Increasing the number of hidden nodes did not appear to have an effect on the accuracy of the results - as the wavefunctions still do not resemble the quadratic ground-state

shape as expected. The wavefunctions were plotted by using the trained parameters of the Restricted Boltzmann Machine, and returning the wavefunction amplitude at that location in space:

FIGURE 4.8: Reconstructed wavefunctions and their measured energies for 10, 20, 40 hidden neurons.

4.5 Discussion

From figure 4.5, we observe how the energy is minimized with each variational sweep. We observe large spikes away from a minimum energy, and this is likely is due to the stochastic nature of the sampling and learning method, but possibly more importantly due to the way we have chosen to represent the x-position as a binary byte. This can lead to stochastic sampling of extreme values for the x-position, if a bit at the extreme edges of the of the byte are flipped off/on by chance. By chance, if an extreme value for x-position is sampled, then this can lead to extreme values for the wavefunction, and hence dramatic changes in the measured energy.

The results in figure 4.6 illustrate that the wavefunctions we converge to are non-physical, and do not resemble the exact ground-state wavefunction. Although it is often the case that increasing the number of hidden neurons in a neural network results in an approximation method that is more flexible, and higher in precision, as illustrated by the results by G. Carleo in figure 3.1, this does not appear to have much of an effect in this case. This is likely because the learning scheme we have used needs to be better adapted to suit using the continuous input of x-position that is provided to the Restricted Boltzmann Machine.

4.6 Conclusions

For the problem we have made efforts to solve: finding the ground state for a particle in a box, the approach we have chosen to take is a combination of unsupervised learning with a reinforcement learning scheme based on machine learning. The machine we have chosen is a type of artificial neural network, known as a Restricted Boltzmann Machine, and the reinforcement learning scheme was based on the optimization objective to return the minimum energy measured for the trial wavefunctions used in quantum Variational Monte Carlo (VMC) simulations.

To achieve this goal, we represented the variational parameters in the trial wavefunctions of the VMC sampling steps as the parameters of a Restricted Boltzmann Machine (RBM). Ideally, by training the RBM to find the minimum energy, the RBM should converge to a set of variational parameters that can accurately represent the ground state wavefunction.

At this stage, more research needs to be done to improve upon our method and its implementation. However, we have indeed made a first attempt to expand upon the recent methods developed by G. Carleo and M. Troyer, so that we can use Restricted Boltzmann Machines to solve for the ground-state wavefunctions that are functions of continuous input, rather than binary spin.

We have identified a gap in the literature, where Hamiltonians that act on wavefunctions of continuous input have yet to be solved using unsupervised learning. We have made a very first attempt to solve this problem, and hope that our efforts will be of use for those trying to solve this problem. For this reason we have decided to make our work publicly available on our GitHub repository [4], where our adapted code adapted from the work by G. Carleo and M. Troyer [5] is available for public use.

4.7 Further Work

A major obstacle in our approach was converting the continuous input of x-position into binary values so that it could be provided as input to the Restricted Boltzmann Machine. Doing so was problematic, and reduced the effectiveness of the Gibbs sampling strategy to find the best set of parameters for the ground-state wavefunction.

To improve upon this, it could be more suitable to use a different neural network architecture that can instead accept continuous input, so the step of converting the x-coordinates into binary bytes would no longer be required.

Promising architectures that can accept continuous input include the recent advancement in the neural networks known as Variational Autoencoders (VAEs). Not only would this be more suitable to accept the continuous input, but it has very recently been shown that VAEs can converge with higher accuracy, and are more flexible for learning the ground-state of systems that are typically much harder to approximate [34]. In addition work has been done to solve the Ising model with high precising using another class of neural networks known as Deep Convolutional Generative Adversarial Networks [35]. These could likely also be a promising alternative, as they can accept continuous making this more suitable than the Restricted Boltzmann Machine, to accept the continuous input of spatial coordinates for the 1D quantum particle in a box.

Appendix A

Access to our Code

The code we have adapted to reproduce these results, and implement the method discussed in this dissertation are made publicly available at https://github.com/INASIC/NQS.

Bibliography

[1] G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355:602–606, February 2017. doi: 10.1126/science.aag2302.

[2] X. Gao and L.-M. Duan. Efficient representation of quantum many-body states with deep neural networks. *Nature Communications*, 8:662, September 2017. doi: 10.1038/s41467-017-00705-2.

[3] Z. Cai and J. Liu. Approximating quantum many-body wave functions using artificial neural networks. *American Physical Society*, 97(3):035116, January 2018. doi: 10.1103/PhysRevB.97.035116.

[4] Anthony Hills. Adapted code that produces the results of this dissertation. URL https://github.com/INASIC/NQS.

[5] Giuseppe Carleo and Matthias Troyer. Open source code used to solve the quantum-many body problem with artificial neural networks. URL https://gitlab.com/nqs.

[6] Y. Tian, K. Pei, S. Jana, and B. Ray. DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars. *ArXiv e-prints*, August 2017.

[7] J.-P. Briot, G. Hadjeres, and F. Pachet. Deep Learning Techniques for Music Generation - A Survey. *ArXiv e-prints*, September 2017.

[8] David Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 1902.

[9] I. Stewart. Galois theory. 1989. doi: 10.1007/978-94-009-0839-0.

[10] D. A. Sprecher. On the structure of continuous functions of several variables. *American Physical Society*, 1965.

[11] Handwritten digit recognition. image used under fair use. URL https://www.wolfram.com/language/11/neural-networks/digit-classification.html?product=language.

[12] Example applications of machine learning. image used under fair use. URL https://www1.in.tum.de/lehrstuhl_1/people/people-archive/74-research/research-topics/896-machine-learning-applications.

[13] Architecture of an artificial neural network. image used under fair use. URL https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/.

[14] C. A. L. Bailer-Jones, R. Gupta, and H. P. Singh. An Introduction to Artificial Neural Networks. In R. Gupta, H. P. Singh, and C. A. L. Bailer-Jones, editors, *Automated Data Analysis in Astronomy*, page 51, 2002.

[15] Y. Chauvin and D. E. Rumelhart. *Backpropagation: Theory, architectures, and applications.* Hillsdale, NJ: Erlbaum., 1995.

[16] D. O. Hebb. *The organization of behavior: A neuropsychological theory.* New York: John Wiley and Sons, Inc., 1950. doi: 10.1002/sce.37303405110.

[17] The three major approaches to machine learning. image used under fair use. URL https://medium.com/@jmstone617/why-is-machine-learning-so-hot-50421f8c3c31.

[18] K. Mills, M. Spanner, and I. Tamblyn. Deep learning and the Schrodinger equation. *ArXiv e-prints*, February 2017.

[19] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017. URL http://dx.doi.org/10.1038/nature24270.

[20] The distinguishing features between the three major approaches in machine learning. image used under fair use. URL https://www.saagie.com/blog/machine-learning-pour-les-grand-meres.

[21] The kl divergence as the difference between probability distributions. image used under fair use. URL https://www.science-emergence.com/Articles/Divergence-de-Kullback-Leibler-avec-python-et-matplotlib/.

[22] The effect of the learning rate on the convergence of gradient descent. image used under fair use. URL https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0.

[23] E. Hazan, A. Klivans, and Y. Yuan. Hyperparameter Optimization: A Spectral Approach. *ArXiv e-prints*, June 2017.

[24] F. Langouche, D. Roekaerts, and E. Tirapegui. *General Langevin Equations and Functional Integration*, pages 295–318. Springer Netherlands, Dordrecht, 1981. ISBN 978-94-009-8368-7. doi: 10.1007/978-94-009-8368-7_17. URL https://doi.org/10.1007/978-94-009-8368-7_17.

[25] Architecture of restricted boltzmann machine. image used under fair use. URL https://towardsdatascience.com/deep-learning-meets-physics-restricted-boltzmann-machines-part-i-6df5c4918c15.

[26] Christian P. Robert and George Casella. *The Metropolis—Hastings Algorithm*, pages 267–320. Springer New York, New York, NY, 2004. ISBN 978-1-4757-4145-2. doi: 10.1007/978-1-4757-4145-2_7. URL https://doi.org/10.1007/978-1-4757-4145-2_7.

[27] N. Privault. *Understanding Markov Chains: Examples and Applications*. Springer Undergraduate Mathematics Series. Springer Singapore, 2013. ISBN 9789814451512. URL https://books.google.co.uk/books?id=e8bEBAAAQBAJ.

[28] J. R. Norris. *Markov Chains*. Cambridge University Press, Cambridge, UK, 1997.

[29] H. G. Katzgraber. Introduction to Monte Carlo Methods. *ArXiv e-prints*, May 2009.

[30] Geoffrey E. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*, pages 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_32. URL https://doi.org/10.1007/978-3-642-35289-8_32.

[31] W. L. McMillan. Ground state of liquid he^4. *Phys. Rev.*, 138:A442–A451, Apr 1965. doi: 10.1103/PhysRev.138.A442. URL https://link.aps.org/doi/10.1103/PhysRev.138.A442.

[32] M. Suzuki. *Quantum Monte Carlo Methods in Condensed Matter Physics*. World Scientific, 1993. ISBN 9789810236830. URL https://books.google.co.uk/books?id=OUsqJuE0nZ8C.

[33] M.M. Smirnov. *Second-order partial differential equations*. Noorhoff series of monographs and textbooks on pure and applied mathematics. Noordhoff, 1966. URL https://books.google.co.uk/books?id=KiO7AAAAIAAJ.

[34] A. Rocchetto, E. Grant, S. Strelchuk, G. Carleo, and S. Severini. Learning hard quantum distributions with variational autoencoders. *ArXiv e-prints*, October 2017.

[35] Z. Liu, S. P. Rodrigues, and W. Cai. Simulating the Ising Model with a Deep Convolutional Generative Adversarial Network. *ArXiv e-prints*, October 2017.